Package: psych (via r-universe)

September 30, 2026

Version 2.6.9

Date 2026-09-22

Title Procedures for Psychological, Psychometric, and Personality Research

Description A general purpose toolbox developed originally for personality, psychometric theory and experimental psychology. Functions are primarily for multivariate analysis and scale construction using factor analysis, principal component analysis, cluster analysis and reliability analysis, although others provide basic descriptive statistics. Item Response Theory is done using factor analysis of tetrachoric and polychoric correlations. Functions for analyzing data at multiple levels include within and between group statistics, including correlations and factor analysis. Validation and cross validation of scales developed using basic machine learning algorithms are provided, as are functions for simulating and testing particular item and test structures. Several functions serve as a useful front end for structural equation modeling. Graphical displays of path diagrams, including mediation models, factor analysis and structural equation models are created using basic graphics. Some of the functions are written to support a book on psychometric theory as well as publications in personality research. For more information, see the <https://personality-project.org/r/> web page.

License GPL (>= 2)

Imports mnormt,parallel,stats,graphics,grDevices,methods,lattice,nlme,GPArotation

Suggests psychTools, lavaan, lme4, Rcsdp, graph, knitr, Rgraphviz

LazyData yes

ByteCompile true

VignetteBuilder knitr

URL https://personality-project.org/r/psych/
https://personality-project.org/r/psych-manual.pdf

NeedsCompilation no

Author William Revelle [aut, cre] (ORCID: <https://orcid.org/0000-0003-4880-9610>)

Maintainer William Revelle <revelle@northwestern.edu>

Repository https://rforms.r-universe.dev

Date/Publication 2026-09-30 15:50:02 UTC

RemoteUrl https://github.com/cran/psych

RemoteRef HEAD

RemoteSha f0e9e6f5beeaf5206c2b846f45a94edabb294d76

Contents

00.psych . . . 6
alpha . . . 16
anova.psych . . . 22
AUC . . . 23
bassAckward . . . 26
Bechtoldt . . . 30
bestScales . . . 33
bfi . . . 38
bi.bars . . . 41
bigCor . . . 42
biPlot . . . 45
block.random . . . 48
bock . . . 49
cattell . . . 50
CFA . . . 51
circ.tests . . . 57
cluster.fit . . . 59
cluster.loadings . . . 60
cluster.plot . . . 62
cluster2keys . . . 63
cohen.d . . . 64
cohen.kappa . . . 70
comorbidity . . . 74
congruence . . . 75
cor.smooth . . . 77
cor.wt . . . 79
cor2dist . . . 81
corCi . . . 81
corFiml . . . 84
corPlot . . . 85
correct.cor . . . 89
cortest . . . 91
corTest . . . 94

cortest.bartlett . . . 97
cosinor . . . 99
cta . . . 103
densityBy . . . 107
describe . . . 109
describeBy . . . 113
diagram . . . 114
draw.tetra . . . 118
dummy.code . . . 119
Dwyer . . . 120
eigen.loadings . . . 121
ellipses . . . 122
error.bars . . . 124
error.bars.by . . . 127
error.crosses . . . 131
error.dots . . . 133
errorCircles . . . 136
esem . . . 138
fa . . . 143
fa.diagram . . . 157
fa.extension . . . 161
fa.lookup . . . 165
fa.multi . . . 168
fa.parallel . . . 172
fa.poly . . . 177
fa.random . . . 178
fa.sort . . . 184
faCor . . . 185
factor.congruence . . . 187
factor.fit . . . 190
factor.model . . . 191
factor.residuals . . . 192
factor.rotate . . . 193
factor.scores . . . 195
factor.stats . . . 197
factor2cluster . . . 200
faRotations . . . 201
fisherz . . . 204
fparse . . . 206
fsi . . . 207
Garcia . . . 209
geometric.mean . . . 211
glb.algebraic . . . 212
Gleser . . . 215
Gorsuch . . . 216
Harman . . . 217
harmonic.mean . . . 219
headTail . . . 220

ICC . . . 222
iclust . . . 225
ICLUST.cluster . . . 233
iclust.diagram . . . 235
ICLUST.graph . . . 237
ICLUST.rgraph . . . 240
ICLUST.sort . . . 242
interp.median . . . 244
irt.1p . . . 245
irt.fa . . . 247
irt.item.diff.rasch . . . 251
irt.responses . . . 253
kaiser . . . 254
KMO . . . 255
lmCor . . . 257
logistic . . . 265
lowerUpper . . . 266
make.keys . . . 268
manhattan . . . 270
mardia . . . 272
mat.sort . . . 274
matrix.addition . . . 275
mediate . . . 276
mixedCor . . . 281
mssd . . . 285
multi.hist . . . 286
multilevel.reliability . . . 288
omega . . . 293
omega.graph . . . 302
omegaStats . . . 305
outlier . . . 308
p.rep . . . 309
paired.r . . . 311
pairs.panels . . . 312
pairwiseCount . . . 315
parcels . . . 318
partial.r . . . 319
phi . . . 321
phi.demo . . . 323
phi2tetra . . . 324
Pinv . . . 325
plot.psych . . . 327
polar . . . 329
polychor.matrix . . . 331
predict.psych . . . 332
predicted.validity . . . 334
principal . . . 335
print.psych . . . 340

Promax . . . 341
psych.misc . . . 345
r.test . . . 349
rangeCorrection . . . 352
reliability . . . 354
rescale . . . 358
residuals.psych . . . 359
reverse.code . . . 360
RMSEA . . . 361
RV . . . 362
sat.act . . . 364
scaling.fits . . . 365
scatterHist . . . 366
schmid . . . 369
Schmid . . . 371
score.alpha . . . 373
score.multiple.choice . . . 374
scoreIrt . . . 376
scoreItems . . . 381
scoreOverlap . . . 388
scoreWtd . . . 392
scrub . . . 393
SD . . . 395
sim . . . 396
sim.anova . . . 402
sim.congeneric . . . 404
sim.hierarchical . . . 406
sim.irt . . . 409
sim.item . . . 414
sim.multilevel . . . 417
sim.omega . . . 420
sim.structure . . . 424
sim.VSS . . . 427
simulation.circ . . . 428
small.msq . . . 430
smc . . . 431
spider . . . 432
splitHalf . . . 435
statsBy . . . 440
structure.diagram . . . 445
structure.list . . . 449
superMatrix . . . 450
table2matrix . . . 452
Tal_Or . . . 453
test.irt . . . 454
test.psych . . . 455
testRetest . . . 457
tetrachoric . . . 460

thurstone 467
tr 469
Tucker 469
unidim 470
VSS 473
VSS.parallel 477
VSS.plot 478
VSS.scree 479
winsor 480
withinBetween 481
Yule 483

Index **486**

`00.psych` *A package for personality, psychometric, and psychological research*

Description

Overview of the psych package.

The psych package has been developed at Northwestern University to include functions most useful for personality and psychological research but is useful for anyone interested in basic scale construction and evaluation. Some of the functions (e.g., `read.file`, `read.clipboard` (in `psychTools`), `describe`, `describeBy`, `pairs.panels`, `error.bars` and `error.dots`) are useful for basic data entry and descriptive analyses. Use help(package="psych") or objects("package:psych") for a list of all functions. Two vignettes are included as part of the package. The intro vignette tells how to install psych and the overview vignette provides examples of using psych in many applications. In addition, there are a growing set of tutorials available on the `https://personality-project.org/r/` webpages.

A companion package, `psychTools`, includes larger data set examples and four more vignettes. `psychTools` also includes some basic file manipulation functions and utilities to convert RD files into LaTeX or HTML

Psychometric applications include routines `fa` for factor analysis using several diferent algorithms: minimum residual (fm="minres") (the default) , maximum likelihood (fm="mle"), minimum rank (fm=minrank), principal axes (fm="pa"), and weighted least squares (fm="wls") factor analysis as well as functions to do Schmid Leiman transformations (`schmid`) to transform a hierarchical factor structure into a bifactor solution.

A recent addition (in version 2.6.30 does Confirmatory Factor Analysis (`CFA`) using the non-iterative approach of Spearman/Guttman/Harmon as discussed by Dhaene and Rosseel (2025). This has been extended to do Confirmatory Bifactor Analysis (`CFA.bifactor`) and to do confirmatory Item Response analysis using (`irt.CFA`).

Although frequently confused for with Factor Analysis, Principal Components Analysis (`pca`) is also available. Please note that Principal Components Analysis is not Factor Analysis, for the former is a summary of the observed variables, while the latter models the observed variables in terms of (unobserved) latent variables. These are very different models and should not be confused.

Rotations may be done using factor or components transformations to a target matrix include the standard Promax transformation (Promax), a transformation to a cluster target, or to any simple target matrix (target.rot) as well as the ability to call many of the GPArotation functions (e.g., oblimin, quartimin, varimax, geomin, ...). Functions for determining the number of factors in a data matrix include Very Simple Structure (VSS) and Minimum Average Partial correlation (MAP). Parallel Analysis to compare the eigen values of modeled versus randpm data can be done using fa.parallel.

An alternative approach to dimension reduction and structural analysis is Item Cluster Analysis (ICLUST). This function is particularly appropriate for exploratory scale construction. The results are sometimes more useful than normal factor analysis. One of the statistics reported is coefficient Beta, an estimate of the worst split half reliability and thus of the size of a general factor.

There are a number of functions for finding various reliability coefficients (see Revelle and Condon, 2019). These include the traditional alpha which may be found for multiple scales and with more useful output by scoreItems, score.multiple.choice), beta (ICLUST and both of McDonald's omega coefficients (omega, omegaSem and omega.diagram) as well as Guttman's six estimates of internal consistency reliability (guttman) and the six measures of Intraclass correlation coefficients (ICC) discussed by Shrout and Fleiss are also available.

Many of these reliability statistics assume unidimensionality, which can be indexed using unidim. All of these statistics are reported for multiple scales using the reliability function.

Basic multilevel analyses may be done by statsBy and multilevel.reliability.

The scoreItems, and score.multiple.choice functions may be used to form single or multiple scales from sets of dichotomous, multilevel, or multiple choice items by specifying scoring keys. scoreOverlap correct interscale correlations for overlapping items, so that it is possible to examine hierarchical or nested structures.

Scales can be formed that best predict (after cross validation) particular criteria using bestScales using unit weighted or correlation weights. This procedure, also called the BISCUIT algorithm (Best Items Scales that are Cross validated, Unit weighted, Informative, and Transparent) is a simple alternative to more complicated supervised machine learning algorithms.

Additional functions make for more convenient descriptions of item characteristics include 1 and 2 parameter Item Response measures. The tetrachoric, polychoric, and irt.fa, irt.CFA functions are used to find 2 parameter descriptions of item functioning. scoreIrt, scoreIrt.1pl and scoreIrt.2pl do basic IRT based scoring.

A number of procedures have been developed as part of the Synthetic Aperture Personality Assessment (SAPA https://www.sapa-project.org/) project. These routines facilitate forming and analyzing composite scales equivalent to using the raw data but doing so by adding within and between cluster/scale item correlations. These functions include extracting clusters from factor loading matrices (factor2cluster), synthetically forming clusters from correlation matrices (cluster.cor), and finding multiple ((lmCor) and partial ((partial.r) correlations from correlation matrices.

If forming empirical scales, or testing out multiple regressions, it is important to cross validate the results. crossValidation will do this on a different data set.

lmCor and mediate meet the desire to do regressions and mediation analysis from either raw data (perhaps with missing values) or from correlation matrices. If raw data are provided, these functions can also do moderation analyses. lmCor matches the output of the lm function for complete data, but will work also on correlation or covariance matrices. The lmCor output includes graphical displays of the regression path coefficients.

Functions to generate simulated data with particular structures include sim.circ (for circumplex structures), sim.item (for general structures) and sim.congeneric (for a specific demonstration of congeneric measurement). The functions sim.congeneric and sim.hierarchical can be used to create data sets with particular structural properties. A more general form for all of these is sim.structural for generating general structural models. These are discussed in more detail in the vignette (psych_for_sem).

Functions to apply various standard statistical tests include p.rep and its variants for testing the probability of replication, r.con for the confidence intervals of a correlation, and r.test to test single, paired, or sets of correlations.

In order to study diurnal or circadian variations in mood, it is helpful to use circular statistics. Functions to find the circular mean (circadian.mean), circular (phasic) correlations (circadian.cor) and the correlation between linear variables and circular variables (circadian.linear.cor) supplement a function to find the best fitting phase angle (cosinor) for measures taken with a fixed period (e.g., 24 hours).

A dynamic model of personality and motivation (the Cues-Tendency-Actions model) is include as cta.

A number of useful helper functions allow for data input (read.file), and data manipulation cs and dfOrder,

The most recent development version of the package is always available for download as a *source* file from the repository at the PMC lab:

install.packages("psych", repos = "https://personality-project.org/r/", type="source").

This will provide the most recent version for PCs and Macs.

Details

Two vignettes (intro.pdf and scoring.pdf) are useful introductions to the package. They may be found as vignettes in R or may be downloaded from https://personality-project.org/r/psych/intro.pdf https://personality-project.org/r/psych/overview.pdf and https://personality-project.org/r/psych/psych_for_sem.pdf. In addition, there are a number of "HowTo"s available at https://personality-project.org/r/

The more important functions in the package are for the analysis of multivariate data, with an emphasis upon those functions useful in scale construction of item composites. However, there are a number of very useful functions for basic data manipulation including read.file, read.clipboard, describe, pairs.panels, error.bars and error.dots) which are useful for basic data entry and descriptive analyses.

When given a set of items from a personality inventory, one goal is to combine these into higher level item composites. This leads to several questions:

1) What are the basic properties of the data? describe reports basic summary statistics (mean, sd, median, mad, range, minimum, maximum, skew, kurtosis, standard error) for vectors, columns of matrices, or data.frames. describeBy provides descriptive statistics, organized by one or more grouping variables. statsBy provides even more detail for data structured by groups including within and between correlation matrices, ICCs for group differences, as well as basic descriptive statistics organized by group.

pairs.panels shows scatter plot matrices (SPLOMs) as well as histograms and the Pearson correlation for scales or items. error.bars will plot variable means with associated confidence intervals.

errorCircles will plot confidence intervals for both the x and y coordinates. corr.test will find the significance values for a matrix of correlations. error.dots creates a dot chart with confidence intervals.

2) What is the most appropriate number of item composites to form? After finding either standard Pearson correlations, or finding tetrachoric or polychoric correlations, the dimensionality of the correlation matrix may be examined. The number of factors/components problem is a standard question of factor analysis, cluster analysis, or principal components analysis. Unfortunately, there is no agreed upon answer. The Very Simple Structure (VSS) set of procedures has been proposed as on answer to the question of the optimal number of factors. Other procedures (VSS.scree, VSS.parallel, fa.parallel, and MAP) also address this question. nfactors combine several of these approaches into one convenient function. Unfortunately, there is no best answer to the problem.

3) What are the best composites to form? Although this may be answered using principal components (principal, aka pca), principal axis (factor.pa) or minimum residual (factor.minres) factor analysis (all part of the fa function) and to show the results graphically (fa.diagram), it is sometimes more useful to address this question using cluster analytic techniques. Previous versions of ICLUST (e.g., Revelle, 1979) have been shown to be particularly successful at forming maximally consistent and independent item composites. Graphical output from ICLUST.graph uses the Graphviz dot language and allows one to write files suitable for Graphviz. If Rgraphviz is available, these graphs can be done in R.

Graphical organizations of cluster and factor analysis output can be done using cluster.plot which plots items by cluster/factor loadings and assigns items to that dimension with the highest loading.

4) How well does a particular item composite reflect a single construct? This is a question of reliability and general factor saturation. Multiple solutions for this problem result in (Cronbach's) alpha (alpha, scoreItems), (Revelle's) Beta (ICLUST), and (McDonald's) omega (both omega hierarchical and omega total). Additional reliability estimates may be found in the guttman and reliability functions.

How well the item composites reflect a single dimension can be estimated using unidim.

This can also be examined by applying irt.fa Item Response Theory techniques using factor analysis of the tetrachoric or polychoric correlation matrices and converting the results into the standard two parameter parameterization of item difficulty and item discrimination. Information functions for the items suggest where they are most effective.

5) For some applications, data matrices are synthetically combined from sampling different items for different people. So called Synthetic Aperture Personality Assessement (SAPA) techniques allow the formation of large correlation or covariance matrices even though no one person has taken all of the items. To analyze such data sets, it is easy to form item composites based upon the covariance matrix of the items, rather than original data set. These matrices may then be analyzed using a number of functions (e.g., cluster.cor, fa, ICLUST, pca, mat.regress, and factor2cluster.

6) More typically, one has a raw data set to analyze. alpha will report several reliablity estimates as well as item-whole correlations for items forming a single scale, score.items will score data sets on multiple scales, reporting the scale scores, item-scale and scale-scale correlations, as well as coefficient alpha, alpha-1 and G6+. Using a 'keys' matrix (created by make.keys or by hand), scales can have overlapping or independent items. score.multiple.choice scores multiple choice items or converts multiple choice items to dichtomous (0/1) format for other functions.

If the scales have overlapping items, then scoreOverlap will give similar statistics, but correcting for the item overlap.

7) The reliability function combines the output from several different ways to estimate reliability including omega and splitHalf.

8) In addition to classical test theory (CTT) based scores of either totals or averages, 1 and 2 parameter IRT based scores may be found with scoreIrt.1pl, scoreIrt.2pl or more generally scoreIrt. Although highly correlated with CTT estimates, these scores take advantage of different item difficulties and are particularly appropriate for the problem of missing data.

9) If the data has a multilevel structure (e.g, items nested within time nested within subjects) the multilevel.reliability aka mlr function will estimate generalizability coefficients for data over subjects, subjects over time, etc. mlPlot will provide plots for each subject of items over time. mlArrange takes the conventional wide output format and converts it to the long format necessary for some multilevel functions. Other functions useful for multilevel data include statsBy and faBy.

An additional set of functions generate simulated data to meet certain structural properties. sim.anova produces data simulating a 3 way analysis of variance (ANOVA) or linear model with or with out repeated measures. sim.item creates simple structure data, sim.circ will produce circumplex structured data, sim.dichot produces circumplex or simple structured data for dichotomous items. These item structures are useful for understanding the effects of skew, differential item endorsement on factor and cluster analytic soutions. sim.structural will produce correlation matrices and data matrices to match general structural models. (See the vignette).

When examining personality items, some people like to discuss them as representing items in a two dimensional space with a circumplex structure. Tests of circumplex fit circ.tests have been developed. When representing items in a circumplex, it is convenient to view them in polar coordinates.

Additional functions for testing the difference between two independent or dependent correlation r.test, to find the phi or Yule coefficients from a two by table, or to find the confidence interval of a correlation coefficient.

Many data sets are included either in psych or in psychTools. E.g., bfi represents 25 personality items thought to represent five factors of personality, ability has 14 multiple choice iq items. sat.act has data on self reported test scores by age and gender. galton Galton's data set of the heights of parents and their children. peas recreates the original Galton data set of the genetics of sweet peas. heights and cubits provide even more Galton data, vegetables provides the Guilford preference matrix of vegetables. cities provides airline miles between 11 US cities (demo data for multidimensional scaling).

Partial Index (to see the entire index, see the link at the bottom of every help page)

psych A package for personality, psychometric, and psychological research.

Useful data entry and descriptive statistics

read.file	search for, find, and read from file
read.clipboard	shortcut for reading from the clipboard
read.clipboard.csv	shortcut for reading comma delimited files from clipboard
read.clipboard.lower	shortcut for reading lower triangular matrices from the clipboard
read.clipboard.upper	shortcut for reading upper triangular matrices from the clipboard

describe	Basic descriptive statistics useful for psychometrics
describe.by	Find summary statistics by groups
statsBy	Find summary statistics by a grouping variable, including within and between correlation matrices.
mlArrange	Change multilevel data from wide to long format
headtail	combines the head and tail functions for showing data sets
bigCor	Find large correlation matrices rapidly by partitioning the data
pairs.panels	SPLOM and correlations for a data matrix
corr.test	Correlations, sample sizes, and p values for a data matrix
corPlot	graphically show the size of correlations in a correlation matrix
multi.hist	Histograms and densities of multiple variables arranged in matrix form
skew	Calculate skew for a vector, each column of a matrix, or data.frame
kurtosi	Calculate kurtosis for a vector, each column of a matrix or dataframe
geometric.mean	Find the geometric mean of a vector or columns of a data.frame
harmonic.mean	Find the harmonic mean of a vector or columns of a data.frame
error.bars	Plot means and error bars
error.bars.by	Plot means and error bars for separate groups
error.crosses	Two way error bars
interp.median	Find the interpolated median, quartiles, or general quantiles.
rescale	Rescale data to specified mean and standard deviation
table2df	Convert a two dimensional table of counts to a matrix or data frame

Data reduction through cluster and factor analysis

CFA	Confirmatory factor analysis.
CFA.bifactor	Bifactor solutions with confirmatory factor model.
fa	Combined function for principal axis, minimum residual, weighted least squares, and maximum likelihood factor analysis
factor.pa	Do a principal Axis factor analysis (deprecated)
factor.minres	Do a minimum residual factor analysis (deprecated)
factor.wls	Do a weighted least squares factor analysis (deprecated)
fa.graph	Show the results of a factor analysis or principal components analysis graphically
fa.diagram	Show the results of a factor analysis without using Rgraphviz
fa.sort	Sort a factor or principal components output
fa.extension	Apply the Dwyer extension for factor loadingss
principal	Do an eigen value decomposition to find the principal components of a matrix
fa.parallel	Scree test and Parallel analysis
fa.parallel.poly	Scree test and Parallel analysis for polychoric matrices
factor.scores	Estimate factor scores given a data matrix and factor loadings
guttman	8 different measures of reliability (6 from Guttman (1945)
`irt.fa`	Apply factor analysis to dichotomous/or polytomous items items to get IRT parameters
`iclust`	Apply the ICLUST algorithm
ICLUST.diagram	The base R graphics output function called by `iclust`
ICLUST.graph	Graph the output from ICLUST using the dot language
ICLUST.rgraph	Graph the output from ICLUST using rgraphviz
kaiser	Apply kaiser normalization before rotating
reliability	A wrapper function to find alpha, omega, split half. etc.

polychoric	Find the polychoric correlations for items and find item thresholds
poly.mat	Find the polychoric correlations for items (uses J. Fox's hetcor)
omega	Calculate the omega estimate of factor saturation (requires the GPArotation package)
omega.graph	Draw a hierarchical or Schmid Leiman orthogonalized solution (uses Rgraphviz)
partial.r	Partial variables from a correlation matrix
predict	Predict factor/component scores for new data
schmid	Apply the Schmid Leiman transformation to a correlation matrix
scoreItems	Combine items into multiple scales and find alpha
score.multiple.choice	Combine items into multiple scales and find alpha and basic scale statistics
scoreOverlap	Find item and scale statistics (similar to score.items) but correct for item overlap
lmCor	Find Cohen's set correlation between two sets of variables (see also lmCor for the latest version)
smc	Find the Squared Multiple Correlation (used for initial communality estimates)
tetrachoric	Find tetrachoric correlations and item thresholds
polyserial	Find polyserial and biserial correlations for item validity studies
mixed.cor	Form a correlation matrix from continuous, polytomous, and dichotomous items
unidim	How well does a unidimensional model fit the data?
VSS	Apply the Very Simple Structure criterion to determine the appropriate number of factors.
VSS.parallel	Do a parallel analysis to determine the number of factors for a random matrix
VSS.plot	Plot VSS output
VSS.scree	Show the scree plot of the factor/principal components
MAP	Apply the Velicer Minimum Absolute Partial criterion for number of factors

Functions for reliability analysis (some are listed above as well).

alpha	Find coefficient alpha and Guttman Lambda 6 for a scale (see also score.items)
guttman	8 different measures of reliability (6 from Guttman (1945)
splitHalf	Find all possible split halfs and the respective reliability values.
omega	Calculate the omega estimates of reliability (requires the GPArotation package)
omegaSem	Calculate the omega estimates of reliability using a Confirmatory model (requires the sem package)
ICC	Intraclass correlation coefficients
score.items	Combine items into multiple scales and find alpha
glb.algebraic	The greates lower bound found by an algebraic solution (requires Rcsdp). Written by Andreas Moeltner
reliability	Multiple estimates of reliability in one function

Procedures particularly useful for Synthetic Aperture Personality Assessment

alpha	Find coefficient alpha and Guttman Lambda 6 for a scale (see also score.items)
bestScales	A bootstrap aggregation function for choosing most predictive unit weighted items
make.keys	Create the keys file for score.items or cluster.cor
correct.cor	Correct a correlation matrix for unreliability
count.pairwise	Count the number of complete cases when doing pair wise correlations
cluster.cor	find correlations of composite variables from larger matrix
cluster.loadings	find correlations of items with composite variables from a larger matrix
eigen.loadings	Find the loadings when doing an eigen value decomposition
fa	Do a minimal residual or principal axis factor analysis and estimate factor scores

fa.extension	Extend a factor analysis to a set of new variables
factor.pa	Do a Principal Axis factor analysis and estimate factor scores
factor2cluster	extract cluster definitions from factor loadings
factor.congruence	Factor congruence coefficient
factor.fit	How well does a factor model fit a correlation matrix
factor.model	Reproduce a correlation matrix based upon the factor model
factor.residuals	Fit = data - model
factor.rotate	“hand rotate" factors
guttman	8 different measures of reliability
lmCor	standardized multiple regression from raw or correlation matrix input Formerly called setCor
mat.regress	standardized multiple regression from raw or correlation matrix input
polyserial	polyserial and biserial correlations with massive missing data
tetrachoric	Find tetrachoric correlations and item thresholds
polychoric	Find polychoric correlations and item thresholds
mixedCor	Find correlations for continuous, polytomous, and dichotomous data at the same time.

Functions for generating simulated data sets

sim	The basic simulation functions
sim.anova	Generate 3 independent variables and 1 or more dependent variables for demonstrating ANOVA and lm designs
sim.circ	Generate a two dimensional circumplex item structure
sim.item	Generate a two dimensional simple structure with particular item characteristics
sim.congeneric	Generate a one factor congeneric reliability structure
sim.minor	Simulate nfact major and nvar/2 minor factors
sim.structural	Generate a multifactorial structural model
sim.irt	Generate data for a 1, 2, 3 or 4 parameter logistic model
sim.VSS	Generate simulated data for the factor model
phi.demo	Create artificial data matrices for teaching purposes
sim.hierarchical	Generate simulated correlation matrices with hierarchical or any structure
sim.spherical	Generate three dimensional spherical data (generalization of circumplex to 3 space)

Graphical functions (require Rgraphviz) – deprecated

structure.graph	Draw a sem or regression graph
fa.graph	Draw the factor structure from a factor or principal components analysis
omega.graph	Draw the factor structure from an omega analysis(either with or without the Schmid Leiman transformation)
ICLUST.graph	Draw the tree diagram from ICLUST

Graphical functions that do not require Rgraphviz

diagram	A general set of diagram functions.
structure.diagram	Draw a sem or regression graph
fa.diagram	Draw the factor structure from a factor or principal components analysis
omega.diagram	Draw the factor structure from an omega analysis(either with or without the Schmid Leiman transformatic
ICLUST.diagram	Draw the tree diagram from ICLUST
plot.psych	A call to plot various types of output (e.g. from irt.fa, fa, omega, iclust
cor.plot	A heat map display of correlations
scatterHist	Bivariate scatter plot and histograms
spider	Spider and radar plots (circular displays of correlations)

Circular statistics (for circadian data analysis)

circadian.cor	Find the correlation with e.g., mood and time of day
circadian.linear.cor	Correlate a circular value with a linear value
circadian.mean	Find the circular mean of each column of a a data set
cosinor	Find the best fitting phase angle for a circular data set

Miscellaneous functions (Some are in psychTools)

comorbidity	Convert base rate and comorbity to phi, Yule and tetrachoric
df2latex	Convert a data.frame or matrix to a LaTeX table
rd2html	Convert a Rd file to HTML (Useful when preparing documentation)
dummy.code	Convert categorical data to dummy codes
fisherz	Apply the Fisher r to z transform
fisherz2r	Apply the Fisher z to r transform
ICC	Intraclass correlation coefficients
cortest.mat	Test for equality of two matrices (see also cortest.normal, cortest.jennrich)
cortest.bartlett	Test whether a matrix is an identity matrix
paired.r	Test for the difference of two paired or two independent correlations
r.con	Confidence intervals for correlation coefficients
r.test	Test of significance of r, differences between rs.
p.rep	The probability of replication given a p, r, t, or F
phi	Find the phi coefficient of correlation from a 2 x 2 table
phi.demo	Demonstrate the problem of phi coefficients with varying cut points
phi2poly	Given a phi coefficient, what is the polychoric correlation
phi2poly.matrix	Given a phi coefficient, what is the polychoric correlation (works on matrices)
polar	Convert 2 dimensional factor loadings to polar coordinates.
scaling.fits	Compares alternative scaling solutions and gives goodness of fits
scrub	Basic data cleaning
tetrachor	Finds tetrachoric correlations
thurstone	Thurstone Case V scaling
tr	Find the trace of a square matrix
wkappa	weighted and unweighted versions of Cohen's kappa
Yule	Find the Yule Q coefficient of correlation

Yule.inv	What is the two by two table that produces a Yule Q with set marginals?
Yule2phi	What is the phi coefficient corresponding to a Yule Q with set marginals?
Yule2tetra	Convert one or a matrix of Yule coefficients to tetrachoric coefficients.

Functions that are under development and not recommended for casual use

irt.item.diff.rasch	IRT estimate of item difficulty with assumption that theta = 0
irt.person.rasch	Item Response Theory estimates of theta (ability) using a Rasch like model

Data sets included in the psych or psychTools package

bfi	represents 25 personality items thought to represent five factors of personality
Thurstone	8 different data sets with a bifactor structure
cities	The airline distances between 11 cities (used to demonstrate MDS)
epi.bfi	13 personality scales
iqitems	14 multiple choice iq items
msq	75 mood items
sat.act	Self reported ACT and SAT Verbal and Quantitative scores by age and gender
Tucker	Correlation matrix from Tucker
galton	Galton's data set of the heights of parents and their children
heights	Galton's data set of the relationship between height and forearm (cubit) length
cubits	Galton's data table of height and forearm length
peas	Galton's data set of the diameters of 700 parent and offspring sweet peas
vegetables	Guilford's preference matrix of vegetables (used for thurstone)

A debugging function that may also be used as a demonstration of psych.

test.psych	Run a test of the major functions on 5 different data sets. Primarily for development purposes. Although the output can be used as a demo of the various functions.

Note

Development versions (source code) of this package are maintained at the repository https://personality-project.org/r/ along with further documentation. Specify that you are downloading a source package.

Some functions require other packages. Specifically, omega and schmid require the GPArotation package, ICLUST.rgraph and fa.graph require Rgraphviz but have alternatives using the diagram functions. i.e.:

function	requires

omega	GPArotation
schmid	GPArotation
ICLUST.rgraph	Rgraphviz
fa.graph	Rgraphviz
structure.graph	Rgraphviz
glb.algebraic	Rcsdp

Author(s)

William Revelle

References

A general guide to personality theory and research may be found at the personality-project https://personality-project.org/. See also the short guide to R at https://personality-project.org/r/. In addition, see

Revelle, W. (in preparation) An Introduction to Psychometric Theory with applications in R. Springer. at https://personality-project.org/r/book/

Revelle, W. and Condon, D.M. (2019) Reliability from alpha to omega: A tutorial. Psychological Assessment, 31, 12, 1395-1411. doi:10.1037/pas0000754

Examples

```
#See the separate man pages and the complete index.
#to test most of the psych package run the following
#test.psych()
```

---

alpha *Find two estimates of reliability: Cronbach's alpha and Guttman's Lambda 6.*

Description

Internal consistency measures of reliability range from ω_h to α to ω_t. This function reports two estimates: Cronbach's coefficient α and Guttman's λ_6. Also reported are item - whole correlations, α if an item is omitted, and item means and standard deviations.

Usage

```
alpha(x, keys=NULL,cumulative=FALSE, title=NULL, max=10,na.rm = TRUE,
   check.keys=FALSE,n.iter=1,delete=TRUE,use="pairwise",warnings=TRUE,
   n.obs=NULL,impute=NULL, discrete=TRUE
   )
alpha.ci(alpha,n.obs,n.var=NULL,p.val=.05,digits=2) #returns an invisible object
alpha2r(alpha, n.var)
r2alpha(r, n.var)
```

Arguments

`x`	A data.frame or matrix of data, or a covariance or correlation matrix
`keys`	If some items are to be reversed keyed, then either specify the direction of all items or just a vector of which items to reverse
`title`	Any text string to identify this run
`cumulative`	should means reflect the sum of items or the mean of the items. The default value is means.
`max`	the number of categories/item to consider if reporting category frequencies. Defaults to 10, passed to `link{response.frequencies}`
`na.rm`	The default is to remove missing values and find pairwise correlations
`check.keys`	if TRUE, then find the first principal component and reverse key items with negative loadings. Give a warning if this happens.
`n.iter`	Number of iterations if bootstrapped confidence intervals are desired
`delete`	Delete items with no variance and issue a warning
`use`	Options to pass to the cor function: "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". The default is "pairwise"
`warnings`	By default print a warning and a message that items were reversed. Suppress the message if warnings = FALSE
`alpha`	The value to use for confidence intervals
`n.obs`	If using correlation matrices as input, by specify the number of observations, we can find confidence intervals
`impute`	How should we impute missing data? Not at all, medians, or means
`discrete`	If TRUE, then report frequencies by categories.
`n.var`	Number of items in the scale (to find r.bar)
`p.val`	width of confidence interval (pval/2 to 1-p.val/2)
`digits`	How many digits to use for alpha.ci
`r`	The average r to use in r2alpha

Details

Alpha is one of several estimates of the internal consistency reliability of a test.

Surprisingly, more than a century after Spearman (1904) introduced the concept of reliability to psychologists, there are still multiple approaches for measuring it. Although very popular, Cronbach's α (1951) underestimates the reliability of a test and over estimates the first factor saturation.

α (Cronbach, 1951) is the same as Guttman's $\lambda 3$ (Guttman, 1945) and may be found by

$$\lambda_3 = \frac{n}{n-1}\Big(1 - \frac{tr(\vec{V})_x}{V_x}\Big) = \frac{n}{n-1}\frac{V_x - tr(\vec{V}_x)}{V_x} = \alpha$$

Perhaps because it is so easy to calculate and is available in most commercial programs, alpha is without doubt the most frequently reported measure of internal consistency reliability. Alpha is the mean of all possible spit half reliabilities (corrected for test length). For a unifactorial test, it is a

reasonable estimate of the first factor saturation, although if the test has any microstructure (i.e., if it is "lumpy") coefficients β (Revelle, 1979; see ICLUST) and ω_h (see omega) are more appropriate estimates of the general factor saturation. ω_t (see omega) is a better estimate of the reliability of the total test.

Guttman's Lambda 6 (G6) considers the amount of variance in each item that can be accounted for the linear regression of all of the other items (the squared multiple correlation or smc), or more precisely, the variance of the errors, e_j^2, and is

$$\lambda_6 = 1 - \frac{\sum e_j^2}{V_x} = 1 - \frac{\sum(1 - r_{smc}^2)}{V_x}.$$

The squared multiple correlation is a lower bound for the item communality and as the number of items increases, becomes a better estimate.

G6 is also sensitive to lumpyness in the test and should not be taken as a measure of unifactorial structure. For lumpy tests, it will be greater than alpha. For tests with equal item loadings, alpha > G6, but if the loadings are unequal or if there is a general factor, G6 > alpha. alpha is a generalization of an earlier estimate of reliability for tests with dichotomous items developed by Kuder and Richardson, known as KR20, and a shortcut approximation, KR21. (See Revelle, in prep).

Alpha and G6 are both positive functions of the number of items in a test as well as the average intercorrelation of the items in the test. When calculated from the item variances and total test variance, as is done here, raw alpha is sensitive to differences in the item variances. Standardized alpha is based upon the correlations rather than the covariances.

A useful index of the quality of the test that is linear with the number of items and the average correlation is the Signal/Noise ratio where

$$s/n = \frac{n\bar{r}}{1 - \bar{r}}$$

(Cronbach and Gleser, 1964; Revelle and Condon (2019)).

More complete reliability analyses of a single scale can be done using the omega function which finds ω_h and ω_t based upon a hierarchical factor analysis.

Alternative functions score.items and cluster.cor will also score multiple scales and report more useful statistics. "Standardized" alpha is calculated from the inter-item correlations and will differ from raw alpha.

Four alternative item-whole correlations are reported, three are conventional, one unique. raw.r is the correlation of the item with the entire scale, not correcting for item overlap. std.r is the correlation of the item with the entire scale, if each item were standardized. r.drop is the correlation of the item with the scale composed of the remaining items. Although each of these are conventional statistics, they have the disadvantage that a) item overlap inflates the first and b) the scale is different for each item when an item is dropped. Thus, the fourth alternative, r.cor, corrects for the item overlap by subtracting the item variance but then replaces this with the best estimate of common variance, the smc. This is similar to a suggestion by Cureton (1966).

If some items are to be reversed keyed then they can be specified by either item name or by item location. (Look at the 3rd and 4th examples.) Automatic reversal can also be done, and this is based upon the sign of the loadings on the first principal component (Example 5). This requires the check.keys option to be TRUE. Previous versions defaulted to have check.keys=TRUE, but some users complained that this made it too easy to find alpha without realizing that some items had been

reversed (even though a warning was issued!). Thus, I have set the default to be check.keys=FALSE with a warning that some items need to be reversed (if this is the case). To suppress these warnings, set warnings=FALSE.

Scores are based upon the simple averages (or totals) of the items scored. Thus, if some items are missing, the scores reflect just the items answered. This is particularly problematic if using total scores (with the cumulative=TRUE option). To impute missing data using either means or medians, use the `scoreItems` function. Reversed items are subtracted from the maximum + minimum item response for all the items.

When using raw data, standard errors for the raw alpha are calculated using equation 2 and 3 from Duhhachek and Iacobucci (2004). This is problematic because some simulations suggest these values are too small. It is probably better to use bootstrapped values.

`alpha.ci` finds confidence intervals using the Feldt et al. (1987) procedure. This procedure does not consider the internal structure of the test the way that the Duhachek and Iacobucci (2004) procedure does. That is, the latter considers the variance of the covariances, while the Feldt procedure is based upon just the mean covariance. In March, 2022, alpha.ci was finally fixed to follow the Feldt procedure. The confidence intervals reported by alpha use both the Feld and the Duhaceck and Iabocucci precedures. Note that these match for large values of N, but differ for smaller values.

Because both of these procedures use normal theory, if you really care about confidence intervals, using the boot option (n.iter > 1) is recommended.

Bootstrapped resamples are found if n.iter > 1. These are returned as the boot object. They may be plotted or described. The 2.5% and 97.5% values are returned in the boot.ci object.

Value

`total`	a list containing
`raw_alpha`	alpha based upon the covariances
`std.alpha`	The standarized alpha based upon the correlations
`G6(smc)`	Guttman's Lambda 6 reliability
`average_r`	The average interitem correlation
`median_r`	The median interitem correlation
`mean`	For data matrices, the mean of the scale formed by averaging or summing the items (depending upon the cumulative option)
`sd`	For data matrices, the standard deviation of the total score
`alpha.drop`	A data frame with all of the above for the case of each item being removed one by one.
`item.stats`	A data frame including
`n`	number of complete cases for the item
`raw.r`	The correlation of each item with the total score, not corrected for item overlap.
`std.r`	The correlation of each item with the total score (not corrected for item overlap) if the items were all standardized
`r.cor`	Item whole correlation corrected for item overlap and scale reliability
`r.drop`	Item whole correlation for this item against the scale without this item
`mean`	for data matrices, the mean of each item

sd	For data matrices, the standard deviation of each item
response.freq	For data matrices, the frequency of each item response (if less than 20) May be suppressed by specifying discrete=FALSE.
scores	Scores are by default simply the average response for all items that a participant took. If cumulative=TRUE, then these are sum scores. Note, this is dangerous if there are lots of missing values.
boot.ci	The lower, median, and upper ranges of the 95% confidence interval based upon the bootstrap.
boot	a 6 column by n.iter matrix of boot strapped resampled values
Unidim	An index of unidimensionality
Fit	The fit of the off diagonal matrix

Note

By default, items that correlate negatively with the overall scale will be reverse coded. This option may be turned off by setting check.keys = FALSE. If items are reversed, then each item is subtracted from the minimum item response + maximum item response where min and max are taken over all items. Thus, if the items intentionally differ in range, the scores will be off by a constant. See scoreItems for a solution.

Two item level statistics are worth comparing: the mean interitem r and the median interitem r. If these differ very much, that is a sign that the scale is not particularly homogeneous.

Variables without variance do not contribute to reliability but do contribute to total score. They are dropped with a warning that they had no variance and were thus dropped. However the scores found still include these values in the calculations.

If the data have been preprocessed by the dplyr package, a strange error can occur. alpha expects either data.frames or matrix input. data.frames returned by dplyr have had three extra classes added to them which causes alpha to break. The solution is merely to change the class of the input to "data.frame".

Two experimental measures of Goodness of Fit are returned in the output: Unidim and Fit. They are not printed or displayed, but are available for analysis. The first is an index of how well the modeled average correlations actually reproduce the original correlation matrix. The second is how well the modeled correlations reproduce the off diagonal elements of the matrix. Both are indices of squared residuals compared to the squared original correlations. These two measures are under development and might well be modified or dropped in subsequent versions.

Author(s)

William Revelle

References

Cronbach, L.J. (1951) Coefficient alpha and the internal strucuture of tests. Psychometrika, 16, 297-334.

Cureton, E. (1966). Corrected item-test correlations. Psychometrika, 31(1):93-96.

Cronbach, L.J. and Gleser G.C. (1964)The signal/noise ratio in the comparison of reliability coefficients. Educational and Psychological Measurement, 24 (3) 467-480.

Duhachek, A. and Iacobucci, D. (2004). Alpha's standard error (ase): An accurate and precise confidence interval estimate. Journal of Applied Psychology, 89(5):792-808.

Feldt, L. S., Woodruff, D. J., & Salih, F. A. (1987). Statistical inference for coefficient alpha. Applied Psychological Measurement (11) 93-103.

Guttman, L. (1945). A basis for analyzing test-retest reliability. Psychometrika, 10 (4), 255-282.

Revelle, W. (in preparation) An introduction to psychometric theory with applications in R. Springer. (Available online at https://personality-project.org/r/book/).

Revelle, W. Hierarchical Cluster Analysis and the Internal Structure of Tests. Multivariate Behavioral Research, 1979, 14, 57-74.

Revelle, W. and Condon, D.M. (2019) Reliability from alpha to omega: A tutorial. Psychological Assessment, 31, 12, 1395-1411. https://doi.org/10.1037/pas0000754. https://osf.io/preprints/psyarxiv/2y3w9 Preprint available from PsyArxiv

Revelle, W. and Condon, D.M. (2018) Reliability. In Irwing, P., Booth, T. and Hughes, D. (Eds). the Wiley-Blackwell Handbook of Psychometric Testing: A multidisciplinary reference on survey, scale, and test development.

Revelle, W. and Zinbarg, R. E. (2009) Coefficients alpha, beta, omega and the glb: comments on Sijtsma. Psychometrika, 74 (1) 1145-154.

See Also

omega, ICLUST, guttman, scoreItems, cluster.cor

Examples

```
set.seed(42) #keep the same starting values
#four congeneric measures
r4 <- sim.congeneric()
alpha(r4)
#nine hierarchical measures -- should actually use omega
r9 <- sim.hierarchical()
alpha(r9)

# examples of two independent factors that produce reasonable alphas
#this is a case where alpha is a poor indicator of unidimensionality

two.f <- sim.item(8,categorical=TRUE)$item
#specify which items to reverse key by name
 alpha(two.f,keys=c("V3","V4","V5","V6"))
 cov.two <- cov(two.f)
 alpha(cov.two,check.keys=TRUE)
 #automatic reversal base upon first component
alpha(two.f,check.keys=TRUE)    #note that the median is much less than the average R
#this suggests (correctly) that the 1 factor model is probably wrong
#an example with discrete item responses  -- show the frequencies
items <- sim.congeneric(N=500,short=FALSE,low=-2,high=2,
        categorical=TRUE) #500 responses to 4 discrete items with 5 categories
a4 <- alpha(items$observed)  #item response analysis of congeneric measures
a4
#summary just gives Alpha
```

```
summary(a4)

alpha2r(alpha = .74,n.var=4)

#because alpha.ci returns an invisible object, you need to print it
print(alpha.ci(.74, 100,p.val=.05,n.var=4))
```

---

anova.psych *Model comparison for regression, mediation, cluster and factor analysis*

Description

When doing regressions from the data or from a correlation matrix using setCor or doing a mediation analysis using link{mediate}, it is useful to compare alternative models. Since these are both regression models, the appropriate test is an Analysis of Variance. Similar tests, using Chi Square may be done for factor analytic models.

Usage

```
## S3 method for class 'psych'
anova(object,...)
```

Arguments

object	An object from setCor, mediate, omega, fa, or iclust.
...	More objects of the same type may be supplied here

Details

setCor returns the SE.residual and degrees of freedom. These are converted to SSR and then an analysis of variance is used to compare two (or more) models. For omega or fa the change in the ML chisquare statistic as a function of change in df is reported.

Value

An ANOVA table comparing the models.

Note

The code has been adapted from the anova.lm function in stats and the anova.sem by John Fox.

Author(s)

Wiliam Revelle

See Also

setCor, mediate, omega, fa, iclust

Examples

```
if(require("psychTools")) {
m1 <- lmCor(reaction ~ import, data = Tal_Or,std=FALSE)
m2 <- lmCor(reaction ~ import+pmi, data = Tal_Or,std=FALSE)
m3 <- lmCor(reaction ~ import+pmi + cond, data = Tal_Or,std=FALSE)
anova(m1,m2,m3)
}


#Several interesting test cases are taken from analyses of the Spengler data set
#Although the sample sizes are actually very large in the first wave,  I use the
#sample sizes from the last wave
#This data set is actually in psychTools but is copied here until we can update psychTools
#We set the n.iter to be 50 instead of the default value of 5,000
if(require("psychTools")) {

 mod1 <- mediate(Income.50 ~ IQ + Parental+ (Ed.11) ,data=Spengler,
    n.obs = 1952, n.iter=50)
 mod2 <- mediate(Income.50 ~ IQ + Parental+ (Ed.11)  + (Income.11)
  ,data=Spengler,n.obs = 1952, n.iter=50)

#Now, compare these models
anova(mod1,mod2)
}

f3 <- fa(Thurstone,3,n.obs=213)  #we need to specifiy the n.obs for the test to work
f2 <- fa(Thurstone,2, n.obs=213)
anova(f2,f3)
```

AUC *Decision Theory measures of specificity, sensitivity, and d prime*

Description

In many fields, decisions and outcomes are categorical even though the underlying phenomenon are probably continuous. E.g. students are accepted to graduate school or not, they finish or not. X-Rays are diagnosed as patients having cancer or not. Outcomes of such decisions are usually labeled as Valid Positives, Valid Negatives, False Positives and False Negatives. In hypothesis testing, False Positives are known as Type I errors, while False Negatives are Type II errors. The relationship between these four cells depends upon the correlation between the decision rule and the outcome as well as the level of evidence needed for a decision (the criterion). Signal Detection Theory and Decision Theory have a number of related measures of performance (accuracy = VP + VN), Sensitivity (VP/(VP + FN)), Specificity (1 - FP), d prime (d'), and the area under the Response Operating Characteristic Curve (AUC). More generally, these are examples of correlations based upon dichotomous data. AUC addresses some of these questions.

Usage

```
AUC(t=NULL,BR=NULL,SR=NULL,Phi=NULL,VP=NULL,labels=NULL,plot="b",zero=TRUE,correct=.5,
     col=c("blue","red"))
```

Arguments

`t`	a 4 x 1 vector or a 2 x2 table of TP, FP, FN, TN values (see below) May be counts or proportions.
`BR`	Base Rate of successful outcomes or actual symptom (if t is not specified)
`SR`	Selection Rate for candidates or diagnoses (if t is not specified)
`Phi`	The Phi correlation coefficient between the predictor and the outcome variable (if t is not specified)
`VP`	The number of Valid Positives (selected applicants who succeed; correct diagnoses).(if t and Phi are not specified)
`labels`	Names of variables 1 and 2
`plot`	"b" (both), "d" (decision theory), "a" (auc), or "n" neither
`zero`	If True, then the noise distribution is centered at zero
`correct`	Cell values of 0 are replaced with correct. (See `tetrachoric` for a discussion of why this is needed.)
`col`	The color choice for the VP and FP, defaults to =c("blue","red") but could be c("grey","black") if we want to avoid colors

Details

The problem of making binary decisions about the state of the world is ubiquitous. We see this in Null Hypothesis Significance Testing (NHST), medical diagnoses, and selection for occupations. Variously known as NHST, Signal Detection Theory, clinical Assessment, or college admissions, all of these domains share the same two x two decision task.

Although the underlying phenomena are probably continuous, a typical decision or diagnostic situation makes dichotomous decisions: Accept or Reject, correctly identified, incorrectly identified. In Signal Detection Theory, the world has two states: Noise versus Signal + Noise. The decision is whether there is a signal or not.

In diagnoses, it is whether to diagnose an illness or not given some noisy signal (e.g., an X-Ray, a set of diagnostic tests).

In college admissions, we accept some students and reject others. Four-Five years later we observe who "succeeds" or graduates.

All of these decisions lead to four cells based upon a two x two categorization. Given the true state of the world is Positive or Negative, and a rater assigns positive or negative ratings, then the resulting two by two table has True (Valid) Positives and True (Valid) Negatives on the diagonal and False Positives and False Negatives off the diagonal.

When expressed as percentages of the total, then Base Rates (BR) depend upon the state of the world, but Selection Ratios (SR) are under the control of the person making the decision and affect the number of False Positives and the number of Valid Positives.

Given a two x two table of counts or percentages

	Decide +	Decide -	
True +	Valid Positive	False Negative	Base Rate %

True -	False Positive	Valid Negative	1- Base Rate
	Selection ratio	1 - Selection ratio	(Total N)

Unfortunately, although this way of categorizing the data is typical in assessment (e.g., Wiggins 1973), and everything is expressed as percentages of the total, in some decision papers, VP are expressed as the ratio of VP to total positive decisions (e.g., Wickens, 1984). This requires dividing through by the column totals (and represented as VP* and FP* in the table below).

The relationships implied by these data can be summarized as a `phi` or `tetrachoric` correlation between the raters and the world, or as a decision process with several alternative measures. If we make the assumption that the two dimensions are continuous and were artificially dichotomised, then the `tetrachoric` correlation is an estimate of the continuous correlation between these two latent dimensions. If we think of the data as truly representing two states e.g., vaccinated or not vaccinanated, dead or alive, then the `phi` coefficient is more appropriate.

Sensitivity, Specificity, Accuracy, Area Under the Curve, and d' (d prime). These measures may be defined as

Measure	Definition
Sensitivity	VP/(VP+ FN)
Specificity	VN/(FP + VN)
Accuracy	VP + VN
VP*	VP/(VP + FP)
FP*	(FP/(VP + FP
d'	z(VP*) - z(FP*)
d'	sqrt(2) z(AUC)
beta	prob(X/S)/(prob(X/N))

Although only one point is found, we can form a graphical display of VP versus FP as a smooth curve as a function of the decision criterion. The smooth curve assumes normality whereas the other merely are the two line segments between the points (0,0), (FP,VP), (1,1). The resulting correlation between the inferred continuous state of the world and the dichotomous decision process is a biserial correlation.

When using table input, the values can be counts and thus greater than 1 or merely probabilities which should add up to 1. Base Rates and Selection Ratios are proportions and thus less than 1.

Alternative ways of describing AUC of prediction are discussed in `cohen.d` and include Common Language Effect Size `d2CL` and related ways of discussing effect sizes.

Value

`phi`	Phi coefficient of the two by two table
`tetra`	Tetrachoric (latent) coefficient inferred from the two by two table
`r.bis`	Biserial correlation of continuous state of world with decision
`observed`	The observed input (as a check)
`probabilities`	Observed values/ total number of observations

conditional	prob / rowSums(prob)
Accuracy	percentage of True Positives + True Negatives
Sensitivity	VP/(VP + FN)
Specificity	VN/(FP + VN)
d.prime	difference of True Positives versus True Negatives
beta	ratio of ordinates at the decision point

Author(s)

William Revelle

References

Metz, C.E. (1978) Basic principles of ROC analysis. Seminars in Nuclear Medicine, 8, 283-298.

Wiggins, Jerry S. (1973) Personality and Prediction: Principles of Personality Assessment. Addison-Wesley.

Wickens, Christopher D. (1984) Engineering Psychology and Human Performance. Merrill.

See Also

phi, phi2tetra ,Yule, Yule.inv Yule2phi, tetrachoric and polychoric, comorbidity

Examples

```
AUC(c(30,20,20,30))  #specify the table input
AUC(c(140,60,100,900)) #Metz example with colors
AUC(c(140,60,100,900),col=c("grey","black"))  #Metz example 1 no colors
AUC(c(80,120,40, 960)) #Metz example 2  Note how the accuracies are the same but d's differ
AUC(c(49,40,79,336)) #Wiggins p 249
AUC(BR=.05,SR=.254,Phi = .317) #Wiggins 251 extreme Base Rates
```

---

bassAckward — *The Bass-Ackward factoring algorithm discussed by Goldberg*

Description

Goldberg (2006) described a hierarchical factor structure organization from the "top down". The original idea was to do successive factor analyses from 1 to nf factors organized by factor score correlations from one level to the next. Waller (2007) discussed a simple way of doing this for components without finding the scores. Using the factor correlations (from Gorsuch) to organize factors hierarchically results may be organized at many different levels. The algorithm may be applied to principal components (pca) or to true factor analysis.

Usage

```
bassAckward(r, nfactors = 1, fm = "minres", rotate = "oblimin", scores = "tenBerge",
   adjust=TRUE, plot=TRUE,cut=.3, use = "pairwise", cor = "cor", weight = NULL,
   correct = 0.5,...)
bassAckward.diagram(x,digits=2,cut = .3,labels=NULL,marg=c(1.5,.5,1.0,.5),
   main="BassAckward",items=TRUE,sort=TRUE,lr=TRUE,curves=FALSE,organize=TRUE,
   values=FALSE,...)
```

Arguments

r	A correlation matrix or a data matrix suitable for factoring
nfactors	Factors from 1 to nfactors will be extracted. If nfactors is a a vector, then just the number of factors specified in the vector will be extracted. (See examples).
fm	Factor method. The default is 'minres' factoring. Although to be consistent with the original Goldberg article, we can also do principal components (fm ="pca").
rotate	What type of rotation to apply. The default for factors is oblimin. Unlike the normal call to pca where the default is varimax, in bassAckward the default for pca is oblimin.
scores	What factor scoring algorithm should be used. The default is "tenBerge", other possibilities include "regression", or "bartlett"
adjust	If using any other scoring proceure that "tenBerge" should we adjust the correlations for the lack of factor score fit?
plot	By default draw a bassAckward diagram
use	How to treat missing data. Use='pairwise" finds pairwise complete correlations.
cor	What kind of correlation to find. The default is Pearson.
weight	Should cases be weighted? Default, no.
correct	If finding tetrachoric or polychoric correlations, what correction should be applied to empty cells (defaults to .5)
x	The object returned by bassAckward
digits	Number of digits to display on each path
cut	Values greater than the abs(cut) will be displayed in a path diagram.
labels	Labels may be taken from the output of the bassAckward function or can be specified as a list.
marg	Margins are set to be slightly bigger than normal to allow for a cleaner diagram
main	The main title for the figure
items	if TRUE, show the items associated with the factors
sort	if TRUE, sort the items by factor loadings
lr	Should the graphic be drawn left to right or top to bottom
curves	Should we show the correlations between factors at the same level
organize	Rename and sort the factors at two lowest levels for a more pleasing figure
values	If TRUE, then show the percent variance accounted for by this factor.
...	Other graphic parameters (e.g., cex)

Details

This is essentially a wrapper to the fa and pca combined with the faCor functions. They are called repeatedly and then the weights from the resulting solutions are used to find the factor/component correlations.

Although the default is do all factor solutions from 1 to the nfactors, this can be simplified by specifying just some of the factor solutions. Thus, for the 135 items of the spi, it is more reasonable to ask for 3,5, and 27 item solutions.

The function bassAckward.diagram may be called using the diagram function or may be called directly.

The output from bassAckward.diagram is the sorted factor structure suitable for using fa.lookup.

Although not particularly pretty, it is possible to do Schmid-Leiman rotations at each level. Specify the rotation as rotate="schmid".

Value

Call	Echo the call
fm	Echos the factor method used
fa	A list of the factor loadings at each level
bass.ack	A list of the factor correlations at each level
summary	The factors at each level
sumnames	Summary of the factor names
labels	Factor labels including items for each level
r	The correlation matrix analyzed
Phi	The factor correlations at each level
fa	The factor analysis loadings at each level, now includes Phi values
fa.vac	The variance accounted for by each factor at each level

Note

Goldberg calculated factor/component scores and then correlated these. Waller suggests just looking at the unrotated components and then examining the correlations when rotating different numbers of components. I do not follow the Waller procedure, but rather find successive factors and then find factor/component correlations following Gorsuch.

It is important to note that the BassAckward solution is not a hierarchical solution in the standard meaning. The factors are not factors of factors as is found in a hierarchical model (e.g. sim.hierarchical or omega, but is merely way of organising solutions with a different number of factors. In each case, unlike omega the factors are of the original variables, not the lower level factors. Thus, detailed statistics for any level of the hierarchy may be found by doing a factoring with that particular number of factors.

To find basic statistics for the multiple factorings, examine the fa object. For more detail just do the factoring at that level.

To see the items associated with the lowest level factors, use fa.lookup. For the items associated with other levels, use fa.lookup specifying the level. (See examples.)

Author(s)

William Revelle

References

Goldberg, L.R. (2006) Doing it all Bass-Ackwards: The development of hierarchical factor structures from the top down. Journal of Research in Personality, 40, 4, 347-358.

Gorsuch, Richard, (1983) Factor Analysis. Lawrence Erlebaum Associates.

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at https://personality-project.org/r/book/

Waller, N. (2007), A general method for computing hierarchical component structures by Goldberg's Bass-Ackwards method, Journal of Research in Personality, 41, 4, 745-752, DOI: 10.1016/j.jrp.2006.08.005

See Also

fa, pca, omega and iclust for alternative hierarchical solutions. link{fa.lookup} to show the items in the lowest level of the solution.

Examples

```
bassAckward(Thurstone,4,main="Thurstone data set")
f.labels <- list(level1=cs(Approach,Avoid),level2=cs(PosAffect,NegAffect,Constraint),
level3 = cs(Extraversion,Agreeableness,Neuroticism,Conscientiousness,Openness))

ba <- bassAckward(bfi[1:25],c(2,3,5),labels=f.labels,
        main="bfi data set from psychTools", values=TRUE)
print(ba,short=FALSE)
#show the items associated with the 5 level solution
fa.lookup(ba,dictionary=bfi.dictionary)
#now show the items associated with the 3 level solution
fa.lookup(ba$fa[[2]],dictionary=bfi.dictionary)
# compare the 3 factor solution to what get by extracting 3 factors directly
f3 <- fa(bfi[1:25],3)
f3$loadings - ba$fa[[2]]$loadings   # these are the same
 #do pca instead of factors  just summarize, don't plot
summary(bassAckward(bfi[1:25],c(1,3,5,7),fm="pca",
       main="Components",plot=FALSE))
##not run, but useful example

f.labels <- list(level1 = cs(Neg,Pos,Constrain),level2 = cs(Extra,Neuro,Cons,Open,Agree),
level3 = cs(attnseeking,sociability,impulsivity,
   charisma,sensationseek,emotexpr,humor,anxiety,
   emotstab,irritability,wellbeing,industry,order,author,honesty,perfect,easygoing,
   selfcontrol,conservatism,creativity,introspect,art,
   intellect,conform,adaptability,compassion,trust))
if(require("psychTools") ) {
sp5 <- bassAckward(psychTools::spi[11:145], c(3,5,27),labels=f.labels,
           main="spi data set from psychTools")}
```

Bechtoldt *Seven data sets showing a bifactor solution.*

Description

Holzinger-Swineford (1937) introduced the bifactor model of a general factor and uncorrelated group factors. The Holzinger data sets are original 14 * 14 matrix from their paper as well as a 9 *9 matrix used as an example by Joreskog. The Thurstone correlation matrix is a 9 * 9 matrix of correlations of ability items. The Reise data set is 16 * 16 correlation matrix of mental health items. The Bechtholdt data sets are both 17 x 17 correlation matrices of ability tests.

Usage

```
data(Thurstone)
data(Thurstone.33)
data(Thurstone.33G)
data(Thurstone.9)
data(Holzinger)
data(Holzinger.9)
data(Bechtoldt)
data(Bechtoldt.1)
data(Bechtoldt.2)
data(Reise)
```

Details

Holzinger and Swineford (1937) introduced the bifactor model (one general factor and several group factors) for mental abilities. This is a nice demonstration data set of a hierarchical factor structure that can be analyzed using the `omega` function or using sem. The bifactor model is typically used in measures of cognitive ability.

There are several ways to analyze such data. One is to use the `omega` function to do a hierarchical factoring using the Schmid-Leiman transformation. This can then be done as an exploratory and then as a confirmatory model using `omegaSem`. Another way is to do a regular factor analysis and use either a `bifactor` or `biquartimin` rotation. These latter two functions implement the Jennrich and Bentler (2011) bifactor and biquartimin transformations. The `bifactor` rotation suffers from the problem of local minima (Mansolf and Reise, 2016) and thus a mixture of exploratory and confirmatory analysis might be preferred.

The 14 variables are ordered to reflect 3 spatial tests, 3 mental speed tests, 4 motor speed tests, and 4 verbal tests. The sample size is 355.

Another data set from Holzinger (Holzinger.9) represents 9 cognitive abilities (Holzinger, 1939) and is used as an example by Karl Joreskog (2003) for factor analysis by the MINRES algorithm and also appears in the LISREL manual as example NPV.KM. This data set represents the scores from the Grant White middle school for 9 tests: "t01_visperc" "t02_cubes" "t04_lozenges" "t06_paracomp" "t07_sentcomp" "t09_wordmean" "t10_addition" "t12_countdot" and "t13_sccaps" and as variables x1 ... x9 (for the Grant-White school) in the lavaan package.

Another classic data set is the 9 variable Thurstone problem which is discussed in detail by R. P. McDonald (1985, 1999) and and is used as example in the sem package as well as in the PROC CALIS manual for SAS. These nine tests were grouped by Thurstone and Thurstone, 1941 (based on other data) into three factors: Verbal Comprehension, Word Fluency, and Reasoning. The original data came from Thurstone and Thurstone (1941) but were reanalyzed by Bechthold (1961) who broke the data set into two. McDonald, in turn, selected these nine variables from the larger set of 17 found in Bechtoldt.2. The sample size is 213.

Another set of 9 cognitive variables attributed to Thurstone (1933) is the data set of 4,175 male students reported by Carl Brigham of Princeton to the College Entrance Examination Board. (Brigham, 1932, Table VIII p 352.) This set does not show a clear bifactor solution but is included as a demonstration of the differences between a maximimum likelihood factor analysis solution versus a principal axis factor solution. On page 352, we are given reliability estimates of .86, .73, .83, .97,.94, .85, .92, .92, and .90. These are based upon sampling 743 boys.

A parallel set (also from Brigham) is the correlation matrix of the same 9 variables for 2899 girls. (Thurstone.33G)

Tucker (1958) uses 9 variables from Thurstone and Thurstone (1941) for his example of `interbattery` factor analysis.

More recent applications of the bifactor model are to the measurement of psychological status. The Reise data set is a correlation matrix based upon >35,000 observations to the Consumer Assessment of Health Care Provideers and Systems survey instrument. Reise, Morizot, and Hays (2007) describe a bifactor solution based upon 1,000 cases.

The five factors from Reise et al. reflect Getting care quickly (1-3), Doctor communicates well (4-7), Courteous and helpful staff (8,9), Getting needed care (10-13), and Health plan customer service (14-16).

The two Bechtoldt data sets are two samples from Thurstone and Thurstone (1941). They include 17 variables, 9 of which were used by McDonald to form the Thurstone data set. The sample sizes are 212 and 213 respectively. The six proposed factors reflect memory, verbal, words, space, number and reasoning with three markers for all expect the rote memory factor. 9 variables from this set appear in the Thurstone data set.

Two more data sets with similar structures are found in the `Harman` data set. This includes the another 9 variables (with 696 subjects) from Holzinger used by Harman `link{Harman.Holzinger}` as well as 8 affective variables from `link{burt}`.

Another data set that is worth examining for tests of bifactor structure is the holzinger.swineford data set which includes the original data from Holzinger and Swineford (1939) supplied by Keith Widaman. This is in the psychTools package.

- Bechtoldt.1: 17 x 17 correlation matrix of ability tests, N = 212.
- Bechtoldt.2: 17 x 17 correlation matrix of ability tests, N = 213.
- Holzinger: 14 x 14 correlation matrix of ability tests, N = 355
- Holzinger.9: 9 x 9 correlation matrix of ability tests, N = 145
- Reise: 16 x 16 correlation matrix of health satisfaction items. N = 35,000
- Thurstone: 9 x 9 correlation matrix of ability tests, N = 213
- Thurstone.33: Another 9 x 9 correlation matrix of ability tests, N=4175
- Thurstone:9: And yet another 9 x 9 correlation matrix of ability tests, N =710

Note

Note that these are tests, not items. Thus, it was possible to find the reliabilities of each test.

For the Holzinger 14 tests these were found from 1- t2 where t = c(.332, .517, .360, .382, .354,.249, .444, .393, .455, .424, .393, .487, .534, .382) (page 53) and thus the reliabilities were 0.890, 0.733, 0.870, 0.854, 0.875, 0.938, 0.803, 0.846, 0.793, 0.820, 0.846, 0.763, 0.715, 0.854.

For the Holzinger.9 tests, the reliabilities for the Grant-White tests were: .76, .57, .94, .65, .75, .87, .95, .84 and .89 (Keith Widaman, personal communication, 2020),

Source

Holzinger: Holzinger and Swineford (1937)
Reise: Steve Reise (personal communication)
sem help page (for Thurstone) Brigham (for Thurstone.33)

References

Bechtoldt, Harold, (1961). An empirical study of the factor analysis stability hypothesis. Psychometrika, 26, 405-432.

Brigham, Carl C. (1932) A study of errors. College Entrance Examination Board.

Holzinger, Karl and Swineford, Frances (1937) The Bi-factor method. Psychometrika, 2, 41-54

Holzinger, K., & Swineford, F. (1939). A study in factor analysis: The stability of a bifactor solution. Supplementary Educational Monograph, no. 48. Chicago: University of Chicago Press.

McDonald, Roderick P. (1999) Test theory: A unified treatment. L. Erlbaum Associates. Mahwah, N.J.

Mansolf, Maxwell and Reise, Steven P. (2016) Exploratory Bifactor Analysis: The Schmid-Leiman Orthogonalization and Jennrich-Bentler Analytic Rotations, Multivariate Behavioral Research, 51:5, 698-717, DOI: 10.1080/00273171.2016.1215898

Reise, Steven and Morizot, Julien and Hays, Ron (2007) The role of the bifactor model in resolving dimensionality issues in health outcomes measures. Quality of Life Research. 16, 19-31.

Thurstone, Louis Leon (1933) The theory of multiple factors. Edwards Brothers, Inc. Ann Arbor.

Thurstone, Louis Leon and Thurstone, Thelma (Gwinn). (1941) Factorial studies of intelligence. The University of Chicago Press. Chicago, Il.

Tucker, Ledyard (1958) An inter-battery method of factor analysis, Psychometrika, 23, 111-136.

Examples

```
if(!require(GPArotation)) {message("I am sorry, to run omega requires GPArotation")
        } else {
#holz <- omega(Holzinger,4, title = "14 ability tests from Holzinger-Swineford")
#bf <- omega(Reise,5,title="16 health items from Reise")
#omega(Reise,5,labels=colnames(Reise),title="16 health items from Reise")
thur.om <- omega(Thurstone,title="9 variables from Thurstone") #compare with
thur.bf   <- fa(Thurstone,3,rotate="biquartimin")
factor.congruence(thur.om,thur.bf)
}
```

bestScales *A bootstrap aggregation function for choosing most predictive unit weighted items*

Description

bestScales forms scales from the items/scales most correlated with a particular criterion and then cross validates on a hold out sample using unit weighted scales. This may be repeated n.iter times using either basic bootstrap aggregation (bagging) techniques or K-fold cross validation. Thus, the technique is known as BISCUIT (Best Items Scales that are Cross validated, Unit weighted, Informative, and Transparent). Given a dictionary of item content, bestScales will sort by criteria correlations and display the item content. Options for bagging (bootstrap aggregation) are included. An alternative to unit weighting is to weight items by their zero order correlations (cross validated) with the criteria. This weighted version is called BISCWIT and is an optional output.

Usage

```
bestScales(x,criteria,min.item=NULL,max.item=NULL, delta = 0,
           cut=.1, n.item =10, wtd.cut=0, wtd.n=10,
          n.iter =1, folds=1, p.keyed=.9,
          overlap=FALSE, dictionary=NULL, check=TRUE,labels=NULL,
           impute="none",log.p=FALSE,digits=2,seed=NULL)
bestItems(x,criteria=1,cut=.1, n.item=10, abs=TRUE,
   dictionary=NULL,check=FALSE,digits=2,use="pairwise",method="pearson")
```

Arguments

x	A data matrix or data frame depending upon the function.
criteria	Which variables (by name or location) should be the empirical target for bestScales and bestItems. May be a separate object.
min.item	Find unit weighted and correlation weighted scales from min.item to max.item
max.item	These are all summarized in the final.multi.valid object
delta	Return items where the predicted r + delta * se of r < max value
cut	Return all values in abs(x[,c1]) > cut.
wtd.cut	When finding the weighted scales, use all items with zero order correlations > wtd.cut
wtd.n	When finding the weighted scales, use the wtd.n items that are > than wtd.cut
abs	if TRUE, sort by absolute value in bestItems
dictionary	a data.frame with rownames corresponding to rownames in the f$loadings matrix or colnames of the data matrix or correlation matrix, and entries (may be multiple columns) of item content.
check	if TRUE, delete items with no variance
labels	labels for criteria if desired

n.item	How many items make up an empirical scale, or (bestItems, show the best n.items)
overlap	Are the correlations with other criteria fair game for bestScales
impute	When finding the best scales, and thus the correlations with the criteria, how should we handle missing data? The default is to drop missing items. (That is to say, to use pairwise complete correlations.)
n.iter	How many times to perform a bootstrap estimate. Replicate the best item function n.iter times, sampling roughly 1-1/e of the cases each time, and validating on the remaining 1/e of the cases for each iteration.
folds	If folds > 1, this is k-folds validation. Note, set n.iter > 1 to do bootstrap aggregation, or set folds > 1 to do k-folds.
p.keyed	The proportion of replications needed to include items in the final best keys.
log.p	Select items based upon the log of the probability of the correlations. This will only have an effect if the number of pairwise cases differs drastically from pair to pair.
digits	round to digits when showing output.
use	How to handle missing data. Defaults to "pairwise"
method	Which correlation to find. Defaults to "pearson"
seed	Set a random seed (if NULL then create one)

Details

There are a number of procedures that can be used for predicting criteria from a set of predictors. The generic term for this is "machine learning" or "statistical learning". The basic logic of these procedures is to find a set of items that best predict a criteria according to some fit statistic and then cross validate these items numerous times. "lasso" regression (least absolute shrinkage and selection) is one such example. bestScales differs from these procedures by unit weighting items chosen from their zero order correlations with the criteria rather than weighting the partial correlations ala regression. This is an admittedly simple procedure that takes into account the well known finding (Wilks, 1938; Wainer, 1976; Dawes, 1979; Waller, 2008) that although regression weights are optimal for any particular data set, unit weights are almost as good (fungible) and more robust across sample variation.

Following some suggestions, we have added the ability to find scales where items are weighted by their zero order correlations with the criteria. This is effectively a comprimise between unit weighting and regression weights (where the weights are the zero order correlations times the inverse of the correlation matrix). This weighted version may be thought of as BISCWIT in contrast to the unit weighted version BISCUIT.

To be comparable to other ML algorithms, we now consider multiple solutions (for number of items >= min.item to max.item). The final scale consists of the number items which maximize the validity or at least are within delta * standard error of r of the maximum.

Thus, bestScales will find up to n.items per criterion that have absolute correlations with a criterion greater than cut. If the overlap option is FALSE (default) the other criteria are not used. This is an example of “dust bowl empiricism" in that there is no latent construct being measured, just those items that most correlate with a set of criteria. The empirically identified items are then formed into scales (ignoring concepts of internal consistency) which are then correlated with the criteria.

Clearly, bestScales is capitalizing on chance associations. Thus, we should validate the empirical scales by deriving them on a fraction of the total number of subjects, and cross validating on the remaining subjects. (This is known both as K-fold cross validation and bagging. Both may be done). If folds > 1, then a k-fold cross validation is done. This removes 1/k (a fold) from the sample for the derivation sample and validates on that remaining fold. This is done k-folds times. Traditional cross validation would thus be a k-fold with k =2. More modern applications seem to prefer k =10 to have 90% derivation sample and a 10% cross validation sample.

The alternative, known as 'bagging' is to do a bootstrap sample (which because it is sampling with replacement will typically extract 1- 1/e = 63.2% of the sample) for the derivation sample (the bag) and then validate it on the remaining 1/e of the sample (the out of bag). This is done n.iter times. This should be repeated multiple times (n.iter > 1, e.g. n.iter=1000) to get stable cross validations.

One can compare the validity of these two approaches by trying each. The average predictability of the n.iter samples are shown as are the average validity of the cross validations. This can only be done if x is a data matrix/ data.frame, not a correlation matrix. For very large data sets (e.g., those from SAPA) these scales seem very stable.

bestScales is effectively a straight forward application of 'bagging' (bootstrap aggregation) and machine learning as well as k-fold validation.

The criteria can be the colnames of elements of x, or can be a separate data.frame.

bestItems and lookup are simple helper functions to summarize correlation matrices or factor loading matrices. bestItems will sort the specified column (criteria) of x on the basis of the (absolute) value of the column. The return as a default is just the rowname of the variable with those absolute values > cut. If there is a dictionary of item content and item names, then include the contents as a two column matrix with rownames corresponding to the item name and then as many fields as desired for item content. (See the example dictionary bfi.dictionary).

The derived model can be further validated against yet another hold out sample using the predict.psych function if given the best scale object and the new data set.

Value

bestScales returns the correlation of the empirically constructed scale with each criteria and the items used in the scale. If a dictionary is specified, it also returns a list (value) that shows the item content. Also returns the keys.list so that scales can be found using cluster.cor or scoreItems. If using replications (bagging or kfold) then it also returns the best.keys, a list suitable for scoring.

There are actually four keys lists reported.

best.keys are all the items used to form unit weighted scales with the restriction of n.item.

weights may be used in the scoreWtd function to find scales based upon the raw correlation weights.

If the min.item and max.item options are used, then two more sets of weights are provided.

optimal.keys are a subset of the best.keys, taking just those items that increase the cross validation values up to the delta * se of the maximum. This is a slightly more parsimonious set.

optimal.weights is analogous to optimal keys, but for supplies weights for just those items that are used to predict cross validation values up to delta * se of the maximum.

The best.keys object is a list of items (with keying information) that may be used in subsequent analyses. These "best.keys" are formed into scale scores for the "final.valid" object which reports how well the best.keys work on the entire sample. This is, of course, not cross validated. Further cross validation can be done using the predict.psych function.

`scores` Are the unit weighted scores from the original items

`best.keys` A key list of those items that were used in the unit weighting.

`wtd.scores` Are the zero-order correlation based scores.

`weights` the scoring weights used

`final.multi.valid`
An object with the unit weighted and correlation weighted correlations from low.step to high.step

The print and summary output list a number of summary statistics for each criteria. This is given for the default case (number of items fixed) and then if requested, the optimal values chosen from min.item to max.item:

The default statistics:

derivation mean Mean correlation of fixed length scale with the criteria, derivation sample

derivation.sd The standard deviation of these estimates

validation.m The mean cross validated correlations with the criteria

validation.sd The standard deviations of these estimates

final.valid The correlation of the pooled models with all the subjects

final.wtd The correlation of the pooled weighted model with all subjects

N.wtd Number of items used in the final weighted model

The optimal number of items statistics:

n The mean number of items meeting the criteria

unit The mean derivation predictive valididy

n.wtd the mean number of items used in the wtd scales

wtd The mean derivation wtd correlaton

valid.n the mean number of items in the cross validation sample

valid.unit The mean cross validated unit weighted correlations

valid.wtd.n The mean number of items used in the cross validated correlated weighed scale

valid.wtd The mean cross validated weighted correlation with criteria

n.final The optimal number of items on the final cross validation sample

n.wtd.final The optimal number of weighted items on the final cross validation.

derviation.mean

`bestItems` returns a sorted list of factor loadings or correlations with the labels as provided in the dictionary. If given raw data, then the correlations with the criteria variables are found first according to "use" and "method".

The stats object can be used to create `error.dots` plots to show the mean estimate and the standard error of the estimates. See the examples.

The resulting best scales can be cross validated on a different hold out sample using `crossValidation`. See the last example.

Note

Although bestScales was designed to form the best unit weighted scales, for large data sets, there seems to be some additional information in weighting by the average zero-order correlation.

To create a dictionary, create an object with row names as the item numbers, and the columns as the item content. See the link{bfi.dictionary} as an example.

This is a very computationally intensive function which can be speeded up considerably by using multiple cores and using the parallel package. The number of cores to use when doing bestScales may be specified using the options command. The greatest step in speed is going from 1 core to 2. This is about a 50

In July 2026 the option of presetting a "random seed" when doing multiple folds was added to allow complete replicability of solutions.

Note

Although empirical scale construction is appealing, it has the basic problem of capitalizing on chance. Thus, be careful of over interpreting the results unless working with large samples. Iteration and bootstrapping aggregation (bagging) gives information on the stability of the solutions.

It is also important to realize that the variables that are most related to the criterion might be because of other, trivial reasons (e.g. height predicts gender)

Author(s)

William Revelle

References

Dawes, R.M. (1979) The robust beauty of improper linear models in decision making, American Psychologist, 34, 571-582.

Elleman, L. G., McDougald, S. K., Condon, D. M., & Revelle, W. 2020 . That takes the BISCUIT: A comparative study of predictive accuracy and parsimony of four statistical learning techniques in personality data, with data missingness conditions. European Journal of Psychological Assessment. 36. (6) 948-958/ (Open source copy available at https://personality-project.org/revelle/publications/elleman.20.pdf)

Revelle, W. (in preparation) An introduction to psychometric theory with applications in R. Springer. (Available online at https://personality-project.org/r/book/).

Wainer, H. (1979) Estimating coefficients in linear models: It don't make no nevermind. Psychological Buletin, 83, 213-217.

Waller, N.G. (2008), Fungible weights in multiple regression. Psychometrica, 73, 691-703.

Wilks, S. S. (1938), Weighting systems for linear functions of correlated variables when there is no dependent variable. Psychometrika. 3. 23-40.

See Also

fa, iclust,principal, error.dots

Examples

```
#This is an example of 'bagging' (bootstrap aggregation)
#not run in order to pass the timing tests for Debian at CRAN
#bestboot <- bestScales(bfi,criteria=cs(gender,age,education),
# n.iter=10,dictionary=bfi.dictionary[1:3])
#bestboot
#compare with 10 fold cross validation
#don't test for purposes of speed in installation to pass the debian CRAN test
tenfold <- bestScales(bfi,criteria=cs(gender,age,education),fold=10,
           dictionary= bfi.dictionary[1:3])
tenfold

#Then, to display the results graphically
#Note that we scale the two graphs with the same x.lim values

 error.dots(tenfold,eyes=TRUE,pch=16,xlim=c(0,.4))
 # error.dots(bestboot,add=TRUE,xlim=c(0,.4))
#do this again, but this time display the scale fits from 1 to 15 Items
# tenfold <- bestScales(bfi,criteria=cs(gender,age,education),fold=10,
# dictionary= bfi.dictionary[1:3],min.item=1,max.item=15)
# matplot(tenfold$multi.validities$wtd.deriv,typ="b",
# xlab="Number of Items",main="Fit as a function of number of items")   #the wtd weights
# matpoints(tenfold$multi.validities$unit.deriv,typ="b",lwd=2) #the unit weights

#an example of using crossValidation with bestScales
set.seed(42)
ss <- sample(1:2800,1400)
model <- bestScales(bfi[ss,],criteria=cs(gender,education,age))
original.fit <- crossValidation(model,bfi[ss,]) #the derivation set
cross.fit <- crossValidation(model,bfi[-ss,])  #the cross validation set
summary(original.fit)
summary(cross.fit)
```

---

bfi *25 Personality items representing 5 factors*

Description

25 personality self report items taken from the International Personality Item Pool (ipip.ori.org) were included as part of the Synthetic Aperture Personality Assessment (SAPA) web based personality assessment project. The data from 2800 subjects are included here as a demonstration set for scale construction, factor analysis, and Item Response Theory analysis. Three additional demographic variables (sex, education, and age) are also included.

Usage

```
data(bfi)
data(bfi.dictionary)
```

Format

A data frame with 2800 observations on the following 28 variables. (The q numbers are the SAPA item numbers).

`A1` Am indifferent to the feelings of others. (q_146)

`A2` Inquire about others' well-being. (q_1162)

`A3` Know how to comfort others. (q_1206)

`A4` Love children. (q_1364)

`A5` Make people feel at ease. (q_1419)

`C1` Am exacting in my work. (q_124)

`C2` Continue until everything is perfect. (q_530)

`C3` Do things according to a plan. (q_619)

`C4` Do things in a half-way manner. (q_626)

`C5` Waste my time. (q_1949)

`E1` Don't talk a lot. (q_712)

`E2` Find it difficult to approach others. (q_901)

`E3` Know how to captivate people. (q_1205)

`E4` Make friends easily. (q_1410)

`E5` Take charge. (q_1768)

`N1` Get angry easily. (q_952)

`N2` Get irritated easily. (q_974)

`N3` Have frequent mood swings. (q_1099

`N4` Often feel blue. (q_1479)

`N5` Panic easily. (q_1505)

`O1` Am full of ideas. (q_128)

`O2` Avoid difficult reading material.(q_316)

`O3` Carry the conversation to a higher level. (q_492)

`O4` Spend time reflecting on things. (q_1738)

`O5` Will not probe deeply into a subject. (q_1964)

`gender` Males = 1, Females =2

`education` 1 = HS, 2 = finished HS, 3 = some college, 4 = college graduate 5 = graduate degree

`age` age in years

Details

The first 25 items are organized by five putative factors: Agreeableness, Conscientiousness, Extraversion, Neuroticism, and Openness. The scoring key is created using `make.keys`, the scores are found using `score.items`.

These five factors are a useful example of using `irt.fa` to do Item Response Theory based latent factor analysis of the `polychoric` correlation matrix. The endorsement plots for each item, as well as the item information functions reveal that the items differ in their quality.

The item data were collected using a 6 point response scale: 1 Very Inaccurate 2 Moderately Inaccurate 3 Slightly Inaccurate 4 Slightly Accurate 5 Moderately Accurate 6 Very Accurate

as part of the Synthetic Apeture Personality Assessment (SAPA https://www.sapa-project.org/) project. To see an example of the data collection technique, visit https://www.SAPA-project.org/ or the International Cognitive Ability Resource at https://icar-project.org. The items given were sampled from the International Personality Item Pool of Lewis Goldberg using the sampling technique of SAPA. This is a sample data set taken from the much larger SAPA data bank.

Note

The bfi data set and items should not be confused with the BFI (Big Five Inventory) of Oliver John and colleagues (John, O. P., Donahue, E. M., & Kentle, R. L. (1991). The Big Five Inventory–Versions 4a and 54. Berkeley, CA: University of California,Berkeley, Institute of Personality and Social Research.)

Source

The items are from the ipip (Goldberg, 1999). The data are from the SAPA project (Revelle, Wilt and Rosenthal, 2010) , collected Spring, 2010 (https://www.sapa-project.org/).

References

Goldberg, L.R. (1999) A broad-bandwidth, public domain, personality inventory measuring the lower-level facets of several five-factor models. In Mervielde, I. and Deary, I. and De Fruyt, F. and Ostendorf, F. (eds) Personality psychology in Europe. 7. Tilburg University Press. Tilburg, The Netherlands.

Revelle, W., Wilt, J., and Rosenthal, A. (2010) Individual Differences in Cognition: New Methods for examining the Personality-Cognition Link In Gruszka, A. and Matthews, G. and Szymura, B. (Eds.) Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control, Springer.

Revelle, W, Condon, D.M., Wilt, J., French, J.A., Brown, A., and Elleman, L.G. (2016) Web and phone based data collection using planned missing designs. In Fielding, N.G., Lee, R.M. and Blank, G. (Eds). SAGE Handbook of Online Research Methods (2nd Ed), Sage Publcations.

See Also

bi.bars to show the data by age and gender, irt.fa for item factor analysis applying the irt model.

Examples

```
data(bfi)
psych::describe(bfi)
# create the bfi.keys (actually already saved in the data file)
 bfi.keys <-
  list(agree=c("-A1","A2","A3","A4","A5"),conscientious=c("C1","C2","C3","-C4","-C5"),
extraversion=c("-E1","-E2","E3","E4","E5"),neuroticism=c("N1","N2","N3","N4","N5"),
openness = c("O1","-O2","O3","O4","-O5"))

 scores <- psych::scoreItems(bfi.keys,bfi,min=1,max=6) #specify the minimum and maximum values
```

```
scores
#show the use of the keys.lookup with a dictionary
psych::keys.lookup(bfi.keys,bfi.dictionary[,1:4])
```

---

bi.bars *Draw pairs of bargraphs based on two groups*

Description

When showing e.g., age or education distributions for two groups, it is convenient to plot them back to back. bi.bars will do so.

Usage

```
bi.bars(x,var=NULL,grp=NULL,horiz,color,label=NULL,zero=FALSE,xlab,ylab,...)
```

Arguments

x	The data frame or matrix from which we specify the data
var	The variable to plot
grp	a grouping variable.
horiz	horizontal (default) or vertical bars
color	colors for the two groups – defaults to blue and red
label	If specified, labels for the dependent axis
zero	If TRUE, subtract the minimum value to make the numbers range from 0 to max -min. This is useful if showing heights
xlab	xaxis label if appropriate
ylab	y axis label otherwise
...	Further parameters to pass to the graphing program

Details

A trivial, if useful, function to draw back to back histograms/barplots. One for each group.

Value

a graphic

Author(s)

William Revelle

See Also

describe, describeBy and statsBy for descriptive statistics and error.bars error.bars.by and densityBy violinBy for graphic displays

Examples

```
#data(bfi)
bi.bars(bfi,"age","gender" ,ylab="Age",main="Age by males and females")
 bi.bars(bfi,"education","gender",xlab="Education",
     main="Education by gender",horiz=FALSE)
```

---

bigCor *Find large correlation matrices by stitching together smaller ones found more rapidly*

Description

When analyzing many subjects (ie. 100,000 or more) with many variables (i.e. 1000 or more) core R can take a long time and sometime exceed memory limits (i.e. with 600K subjects and 6K variables). bigCor runs (in parallel if multicores are available) by breaking the variables into subsets (of size=size), finding all subset correlations, and then stitches the resulting matrices into one large matrix. Noticeable improvements in speed compared to cor.

Usage

```
bigCor(x, size = NULL, use = "pairwise",cor="pearson",correct=.5)
```

Arguments

x	A data set of numeric variables
size	What should the size of the subsets be? Defaults to NCOL (x)/20
use	The standard correlation option. "pairwise" allows for missing data
cor	Defaults to Pearson correlations, alteratives are polychoric and spearman
correct	Correction for continuity for polychoric correlations. (see polychoric)

Details

The data are divided into subsets of size=size. Correlations are then found for each subset and pairs of subsets.

Time is roughly linear with the number of cases and increases by the square of the number of variables. The benefit of more cores is noticeable. It seems as if with 4 cores, we should use sizes to split it into 8 or 12 sets. Otherwise we don't actually use all cores efficiently.

There is some overhead in using multicores. So for smaller problems (e.g. the 4,000 cases of the 145 items of the psychTools::spi data set, the timings are roughly .14 seconds for bigCor (default size) and .10 for normal cor. For small problems, this actually gets worse as we use more cores. The cross over point seems to be at roughly 5K subjects. (updated these timings to recognize the M1 Max chip. An increase of 4x in speed! They had been .44 and .36.)

The basic loop loops over the subsets. When the size is a integer subset of the number of variables and is a multiple of the number of cores, the multiple cores will be used more. Notice the benefit

of 660/80 versus 660/100. But this breaks down if we try 660/165. Further notice the benefit when using a smaller subset (55) which led to the 4 cores being used more.

The following timings are included to help users tinker with parameters:

Timings (in seconds) for various problems with 645K subjects on an 8 core Mac Book Pro with a 2.4 GHZ Intell core i9.

options(mc.cores=4) (Because we have 8 we can work at the same time as we test this.)

First test it with 644,495 subjects and 1/10 of the number of possible variables. Then test it for somewhat fewer variables.

Variables	size	2 cores	4 cores	compared to normal cor function
660	100	430	434	430
660	80	600	348	notice the improvement with 8ths
660	165		666	(Stitching seems to have been very slow)
660	55		303	Even better if we break it into 12ths!
500	100		332	322 secs
480	120	408	365	315 Better to change the size
480	60	358	206	This leads to 8 splits

We also test it with fewer subjects. Time is roughly linear with number of subjects.

Variables	size	2 cores	4 cores	compared to normal cor function Further comparisons with fewer subjects (100K)
480	60	57	31	47 with normal cor. Note the effect of n subjects!
200	50	19.9	13.6	27.13
100	25	4.6	3.5	5.85

One last comparison, 10,000 subjects, showing the effect of getting the proper size value. You can tune on these smaller sets of subjects before trying large problems.

Variables	size	2 cores	4 cores	compared to normal cor function
480	120	5.2	5.1	4.51
480	60	2.9	2.88	4.51
480	30	2.65	2.691	
480	20	2.73	2.77	
480	10	2.82	2.97	too many splits?
200	50	2.18	1.39	2.47 for normal cor (1.44 with 8 cores 2.99 with 1 core)
200	25	1.2	1.17	2.47 for normal cor
(1.16 with 8 cores, 1.17 with 1 core)				
100	25	.64	.52	.56

Timings updated in 2/23 using a MacBook Pro with M1 max chip 10,000 subjects 953 variables suggests that a very small size (e.g. 20) is probably optimal

Variables	size	2 cores	4 cores	8 cores	compared to normal cor function
953	20	7.92	4.55	2.88	11.04
953	30	7.98	4.88	3.15	11.04
953	40	8.22	5.14	3.63	11.16
953	60	8.51	5.59	3.93	11.16
953	80	8.31	5.59	4.14	11.16
953	120	8.33	6.22	4.75	11.16

Timings updated in 11/25 using a MacBook Pro with M4 max chip. 1,000 or 10,000 subjects, 6625 variables

n var	n.obs	size	2 cores	4 cores	8 cores	12 cores
6625	1000	default	9.0	5.3	3.6	2.8
6625	1000	20	11	6.4	4.1	3.2
6625	1000	40	8.3	5.0	3.3	2.6
6625	1000	80	7.6	4.5	3.2	2.5
6625	1000	120	6.6	4.6	3.1	2.7
6625	1000	160	7.2	4.5	3.4	2.8
6625	10000	default	63.9	34.4	24.8	20.7
6625	10000	20	70.2	37.8	25.2	21.6
6625	10000	40	62.5	33.5	25.5	19.2
6625	10000	80	54.1	31.6	24.9	19.4
6625	10000	120	57.3	31.3	24.2	22.5
6625	10000	160	56.5	33.9	24.7	22.1

Moral, get a fast chip and use as many cores as you can.

Does not seem to work with data.tables

Value

The correlation matrix

Author(s)

William Revelle

References

Examples of large data sets with massively missing data are taken from the SAPA project. e.g.,

William Revelle, Elizabeth M. Dworak, and David M. Condon (2021) Exploring the persome: The power of the item in understanding personality structure. Personality and Individual Differences, 169, doi:10.1016/j.paid.2020.109905

David Condon (2018)The SAPA Personality Inventory: an empirically-derived, hierarchically-organized self-report personality assessment model. PsyArXiv /sc4p9/ doi:10.31234/osf.io/sc4p9

See Also

pairwiseCountBig which will do the same, but find the count of observations per cell.

Examples

```
R <- bigCor(bfi,10)
#compare the results with
r.bfi <- cor(bfi,use="pairwise")
all.equal(R,r.bfi)
```

---

biPlot — *Draw biplots of factor or component scores by factor or component loadings*

Description

Extends the biplot function to the output of fa, fa.poly or principal. Will plot factor scores and factor loadings in the same graph. If the number of factors > 2, then all pairs of factors are plotted. Factor score histograms are plotted on the diagonal. The input is the resulting object from fa, principal, or fa.poly with the scores=TRUE option. Points may be colored according to other criteria.

Usage

```
biPlot(x, labels=NULL,cex=c(.75,1),main="BiPlot from fa",
hist.col="cyan",xlim.s=c(-3,3),ylim.s=c(-3,3),xlim.f=c(-1,1),ylim.f=c(-1,1),
maxpoints=100,adjust=1.2,col,pos, arrow.len = 0.1,pch=16,choose=NULL,
cuts=1,cutl=.0,group=NULL,smoother=FALSE,vars=TRUE,...)

#replaces the deprecated function
## S3 method for class 'psych'
biplot(x, labels=NULL,cex=c(.75,1),main="Biplot from fa",
hist.col="cyan",xlim.s=c(-3,3),ylim.s=c(-3,3),xlim.f=c(-1,1),ylim.f=c(-1,1),
maxpoints=100,adjust=1.2,col,pos, arrow.len = 0.1,pch=16,choose=NULL,
cuts=1,cutl=.0,group=NULL,smoother=FALSE,vars=TRUE,...) #deprecated
```

Arguments

x	The output from fa, fa.poly or principal with the scores=TRUE option
labels	if NULL, draw the points with the plot character (pch) specified. To identify the data points, specify labels= 1:n where n is the number of observations, or labels =rownames(data) where data was the data set analyzed by the factor analysis.
cex	A vector of plot sizes of the data labels and of the factor labels

main	A main title for a two factor biplot
hist.col	If plotting more than two factors, the color of the histogram of the factor scores
xlim.s	x limits of the scores. Defaults to plus/minus three sigma
ylim.s	y limits of the scores.Defaults to plus/minus three sigma
xlim.f	x limits of the factor loadings.Defaults to plus/minus 1.0
ylim.f	y limits of the factor loadings.Defaults to plus/minus 1.0
maxpoints	When plotting 3 (or more) dimensions, at what size should we switch from plotting "o" to plotting "."
adjust	an adjustment factor in the histogram
col	a vector of colors for the data points and for the factor loading labels
pos	If plotting labels, what position should they be in? 1=below, 2=left, 3 top, 4 right. If missing, then the assumption is that labels should be printed instead of data points.
arrow.len	the length of the arrow head
pch	The plotting character to use. pch=16 gives reasonable size dots. pch="." gives tiny points. If adding colors, use pch between 21 and 25. (see examples).
choose	Plot just the specified factors
cuts	Do not label cases with abs(factor scores) < cuts) (Actually, the distance of the x and y scores from 0)
cutl	Do not label variables with communalities in the two space < cutl
group	A vector of a grouping variable for the scores. Show a different color and symbol for each group.
smoother	If TRUE then do a smooth scatter plot (which shows the density rather than the data points). Only useful for large data sets.
vars	If TRUE, draw arrows for the variables, and plot the scores. If FALSE, then draw arrows for the scores and plot the variables.
...	more options for graphics

Details

Uses the generic biplot function to take the output of a factor analysis `fa`, `fa.poly` or principal components analysis `principal` and plot the factor/component scores along with the factor/component loadings.

This is an extension of the generic biplot function to allow more control over plotting points in a two space and also to plot three or more factors (two at time).

This will work for objects produced by `fa`, `fa.poly` or `principal` if they applied to the original data matrix. If however, one has a correlation matrix (e.g., based upon the output from `tetrachoric` or `polychoric`), and has done either `fa` or `principal` on the correlations, then obviously, we can not do a biplot.

However, both of those functions produce a weights matrix, which, in combination with the original data can be used to find the scores by using `factor.scores`. Since biplot.psych is looking for two elements of the x object: x$loadings and x$scores, you can create the appropriate object to plot, or add it to the factor object See the third and fourth examples.

In order to just plot the loadings, use `fa.plot`. Or, if we want to show the loadings as vectors, use pch = "".

Author(s)

William Revelle

See Also

fa, fa.poly, principal, fa.plot, pairs.panels

Examples

```
#the standard example
data(USArrests)
fa2 <- fa(USArrests,2,scores=TRUE)
biPlot(fa2,labels=rownames(USArrests))

# plot the 3 factor solution
#data(bfi)
fa3 <- fa(bfi[1:200,1:15],3,scores=TRUE)
biPlot(fa3)
#just plot factors 1 and 3 from that solution
biPlot(fa3,choose=c(1,3))

#
fa2 <- fa(bfi[16:25],2)  #factor analysis
fa2$scores <- fa2$scores[1:100,]  #just take the first 100
#now plot with different colors and shapes for males and females
biPlot(fa2,pch=c(24,21)[bfi[1:100,"gender"]],
group =bfi[1:100,"gender"],
   main="Biplot of Openness and Neuroticism by gender")

## An example from the correlation matrix
r <- cor(bfi[1:200,1:10], use="pairwise") #find the correlations
f2 <- fa(r,2)
#biPlot(f2)  #this throws an error    (not run)

#f2 does not have scores, but we can find them
f2$scores <- factor.scores(bfi[1:200,1:10],f2)

biPlot(f2,main="biplot from correlation matrix and factor scores")


#or create a new object with the scores
 #find the correlations for all subjects
r <- cor(bfi[1:10], use="pairwise")
f2 <- fa(r,2)
x <- list()
#find the scores for just the first 200 subjects
x$scores <- factor.scores(bfi[1:200,1:10],f2)
x$loadings <- f2$loadings
class(x) <- c('psych','fa')
biPlot(x,main="biPplot from correlation matrix combined with factor scores")
```

block.random *Create a block randomized structure for n independent variables*

Description

Random assignment of n subjects with an equal number in all of N conditions may done by block randomization, where the block size is the number of experimental conditions. The number of Independent Variables and the number of levels in each IV are specified as input. The output is a the block randomized design.

Usage

```
block.random(n, ncond = NULL)
```

Arguments

n	The number of subjects to randomize. Must be a multiple of the number of experimental conditions
ncond	The number of conditions for each IV. Defaults to 2 levels for one IV. If more than one IV, specify as a vector. If names are provided, they are used, otherwise the IVs are labeled as IV1 ... IVn

Value

blocks	A matrix of subject numbers, block number, and randomized levels for each IV

Note

Prepared for a course on Research Methods in Psychology https://personality-project.org/courses/205/205.syllabus.html

Author(s)

William Revelle

Examples

```
br <- block.random(n=24,c(2,3))
pairs.panels(br)
br <- block.random(96,c(time=4,drug=3,sex=2))
pairs.panels(br)
```

bock *Bock and Liberman (1970) data set of 1000 observations of the LSAT*

Description

An example data set used by McDonald (1999) as well as other discussions of Item Response Theory makes use of a data table on 10 items (two sets of 5) from the Law School Admissions Test (LSAT). Included in this data set is the original table as well as the reponses for 1000 subjects on the first set (Figure Classification) and second set (Debate).

Usage

```
data(bock)
```

Format

A data frame with 32 observations on the following 8 variables.

`index` 32 response patterns

`Q1` Responses to item 1

`Q2` Responses to item 2

`Q3` Responses to item 3

`Q4` Responses to item 4

`Q5` Responses to item 5

`Ob6` count of observations for the section 6 test

`Ob7` count of observations for the section 7 test

Two other data sets are derived from the bock dataset. These are converted using the `table2df` function.

lsat6 reponses to 5 items for 1000 subjects on section 6

lsat7 reponses to 5 items for 1000 subjects on section 7

Details

The lsat6 data set is analyzed in the ltm package as well as by McDonald (1999). lsat7 is another 1000 subjects on part 7 of the LSAT. Both sets are described by Bock and Lieberman (1970). Both sets are useful examples of testing out IRT procedures and showing the use of `tetrachoric` correlations and item factor analysis using the `irt.fa` function.

Source

R. Darrell Bock and M. Lieberman (1970). Fitting a response model for dichotomously scored items. Psychometrika, 35(2):179-197.

References

R.P. McDonald. Test theory: A unified treatment. L. Erlbaum Associates, Mahwah, N.J., 1999.

Examples

```
data(bock)
responses <- table2df(bock.table[,2:6],count=bock.table[,7],
        labs= paste("lsat6.",1:5,sep=""))
describe(responses)
## maybe str(bock.table) ; plot(bock.table) ...
```

`cattell` *12 cognitive variables from Cattell (1963)*

Description

Rindskopf and Rose (1988) use this data set to demonstrate confirmatory second order factor models. It is a nice example data set to explore hierarchical structure and alternative factor solutions. It contains measures of fluid and crystallized intelligence.

Usage

```
data("cattell")
```

Format

A correlation matrix of the following 12 variables from 277 7th and 8th graders

Verbal A verbal ability test from Thurstone

Verbal2 A verbal ability test from Thurstone

Space1 A Spatial ability test from Thurstone

Space2 A Spatial ability test from Thurstone

Reason1 A reasoning test from Thurstone

Reason2 A reasoning test from Thurstone

Number1 A Numerical ability test from Thurstone

Number2 A Numerical ability test from Thurstone

IPATSer A "culture fair" series from the IPAT

IPATCLAS A "culture fair" classification test from the IPAT

IPATMatr A "culture fair" matrix reasoning test from the IPAT

IPATTop A "culture fair" topology test from the IPAT

Details

Cattell (1963) reported on 8 cognitive variables from Thurstone and four from the Institute for Personality Assessment Test (IPAT). Rindskopf and Rose (1988) use this data set as an example of second order factor analysis. It is thus a nice set for examining alternative solutions such as bifactor rotation, omega hierarchical, as well as esem and interbattery factor analysis.

Source

David Rindskopf and Tedd Rose, (1988) Some Theory and Applications of Confirmatory Second-Order Factor Analysis, Multivariate Behavioral Research, 23, 51-67.

References

Cattell, R. B. (1963).Theory of fluid and crystallized intelligence: A critical experiment. Journal of Educational Psychology, 54, 1-22.

David Rindskopf and Tedd Rose, (1988) Some Theory and Applications of Confirmatory Second-Order Factor Analysis, Multivariate Behavioral Research, 23, 51-67.

Examples

```
data(cattell)
corPlot(cattell,numbers=TRUE,upper=FALSE,diag=FALSE,
             main="12 cognitive variables from Cattell (1963)",xlas=2)
```

CFA *Confirmatory Factor Analysis Without Iteration*

Description

Factor analysis is a procedure for identifying latent variables thought to account for the correlations or covariances between observed variables. There are two approaches to factor analysis: Exploratory Factor Analysis (e.g., EFA using the fa function) and Confirmatory Factor Analysis (CFA). Perhaps the best way to do Confirmatory Factor Analysis is with the laavan package's cfa function. CFA in psych is a simple and more limited version for those who want to stay within the psych package and take advantage of various psych package options. CFA uses the direct approach (the multiple group method) using the Spearman/Guttman approach as discussed by Dhaene and Rosseel, 2025.

Usage

```
CFA(model=NULL,x=NULL, all=FALSE, cor = "cor", use ="pairwise", n.obs = NA,
 orthog=FALSE,  weight=NULL,correct=0, method="regression",
 missing=FALSE,impute="none",Grice=FALSE,n.iter=100)

CFA.bifactor(model=NULL,x=NULL,all=FALSE,g=FALSE, cor="cor", use="pairwise", n.obs=NA,
  orthog=FALSE, weight=NULL, correct=0, method="regression",
   missing=FALSE,impute="none",
```

```
  Grice=FALSE ,
  n.iter=1)

irt.CFA(model=NULL,x=NULL,all=TRUE,n.obs=NA, orthog=FALSE, weight=NULL, correct=0,
    method="regression",
 missing=FALSE,impute="none",
 Grice=FALSE,
 plot=TRUE,
 n.iter=1,
 scores=FALSE)
```

Arguments

model	If specified, the model can either be in lavaan syntax or a keys list such as those used in scoreItems or scoreOverlap. See examples. If the model is missing, then the analysis will be a for one factor model of all of the data.
x	A data matrix or correlation /covariance matrix. (If missing, then assumed to be the values of the first object.)
all	if TRUE, then do the analysis for all the variables in the r matrix. If FALSE, then select just those variables in the r matrix defined by the model.
g	if TRUE, find a hierarchical (higher level) model.
cor	How to find the correlations: "cor" is Pearson", "cov" is covariance, "tet" uses tetrachoric, "poly" uses polychoric, "mixed" uses mixedCor for a mixture of tetrachorics, polychorics, Pearsons, biserials, and polyserials, Yuleb is Yulebonett, Yuleq and YuleY are the obvious Yule coefficients as appropriate
n.obs	Number of observations if given a correlation/covariance matrix, defaults to 100
orthog	Should the factors be allowed to correlate (orthog=FALSE) or forced to be orthogonal (orthog=TRUE)
use	How to treat missing data, use="pairwise" is the default". See cor for other options.
weight	If not NULL, a vector of length n.obs that contains weights for each observation. The NULL case is equivalent to all cases being weighted 1.
correct	correction value for 0 values in tetrachoric correlation. See the discussion in tetrachoric for alternative values.
method	Correlations are by default found using Pearson. Alternative methods for the correlation may be Spearman or Kendall.
missing	if r is a data matrix, and missing=TRUE, then impute missing values using either the median or the mean. Specifying impute="none" will not impute data points and thus will have some missing data in the factor scores.
impute	"median" or "mean" values are used to replace missing values
Grice	If TRUE, use the Grice method for factor indeterminacy.
plot	should irt.CFA display the plots (defaults to TRUE)
n.iter	Number of bootstrap resamples of the CFA solution. Defaults to 100.
scores	If true, and given raw data, find irt based scores for the CFA based factors.

Details

Most EFA and CFA functions use maximum likelihood functions to estimate the coefficients. However, as Maccallum et al. (2007), and Dhaene and Rosseel (2024) point out, ML approaches are not necessarily optimal for finite (e.g., small) samples. Maccallum et al. (2007) discuss why ML fails on some problems that minres procedures do not.

Confirmatory factor analysis may be done without iteration (and thus not using Maximum Likelihood procedures) by using some very old techniques. The algorithm follows that of Dhaene and Rosseel (2024) using the "Spearman" Multiple Group Method to estimate the communalities. This method was introduced by Guttman (1952) and is discussed by Harman (1967).

`CFA` follows the Spearman approach for communalities discussed by Dhaene and Rosseel (2024) and described as the "Multiple Group Method". I use the upper case name (CFA) to avoid conflicts with lavaan's cfa function. Following Harman (1967) (Chapter 7, p 115-117) the communality of each variable is estimated by the ratio of the sum of all the correlations to the sum of squared correlations with that variable. The square root of the communality is the factor loading.

Guttman (1952) points out that a weighting matrix of -1, 0, and 1 times is essentially a regression model where the use of differential weights doesn't make much difference.

`CFA.bifactor` first does a CFA on all of the variables, and then does another CFA using the model matrix or keys list on the residual correlation matrix. The results are in relatively close agreement with those from lavaan, but are not identical.

To do a "S-1" solution (Eid, 2017; Li and Savalei, 2025) just specify a model with not all variables defined as group factors.

Options for `CFA.bifactor` include solving the correlations as simple bi-factor model, or as a hierarchical/higher level model using the g=TRUE option. Graphical output in the examples shows the difference of the two approaches.

Guttman and Harman's original method seem to be restricted to positive manifolds and finds the communalities based upon the the correlations.

$h_i^2 = \frac{(\Sigma r_i)^2 - \Sigma r_i^2}{2(\Sigma r_{i<j} - \Sigma r_i)} h_i^2 = (\Sigma r_i)^2 - \Sigma r_i^2 / 2(\Sigma r_{i<j} - \Sigma r_i)$

CFA estimates communalities using the absolute values of r when finding the sums. This allows applying the method to personality data sets such as the `bfi` or sapa data sets as well as mood data as found in the msqR data sets (sapa and msqR are in psychTools).

If the g parameter is set to true, a hierarchical or second order solution is found by first factoring all the variables for a one factor (g) model and then factoring the residualized matrix using the model based factors. This is shown in the test.hi example.

`irt.CFA` does a confirmatory analysis (CFA) on the polychoric/tetrachoric correlations and then reports the results in terms of an `irt.fa` (IRT) model. See `irt.fa` for a discussion of the equivalence of a 2PL IRT model and a (non-linear) factor analysis of items using polychoric or tetrachoric correlations.

By default, the `CFA` will do n.iter=100 bootstrap resamplings to provide confidence intervals for the solutions. Because `CFA.bifactor` uses `CFA` to find the general factor and then the residual group factors, confidence intervals are also reported.

The loadings and goodness of fits are based upon the median bootstrapped values.

Value

loadings	Factor (Structure) Loadings
Pattern	Factor Pattern coefficients
Phi	Factor correlations
communalities	As estimated using the Spearman/Guttman procedure
dof	Degrees of freedom is the number of original corrlations - number of loadings - number of between factor correlations
stats	as found by fa.stats
scores	Factor scores.
...	Many other statistics as reported by fa
Call	echoes the call to the function

Note

The examples include a number of comparisons with the output of the lavaan package. These are not run, but can be examined after loading lavaan. The general observation is that the results are very similar, but not identical. The loadings are identical for the 9 variable Thurstone problem, but differ slightly for the 24 Holzinger problem. lavaan has several ways of estimating coefficients. The ULS results match CFA most closely.

Further note that cross loadings are not allowed.

Author(s)

William Revelle

References

Sara Dhaene and Yves Rosseel, 2024, An Evaluation of non-iterative estimators in confirmatory factor analysis. Structural Equation Modeling (31) 1 1-13. doi: 10.1080/10705511.2023.2187285

Michael Eid, Christian Geiser, Tobias Koch and Moritz Heene (2017) Anomalous Results in G-Factor Models: Explanations and Alternatives. Psychological Methods, 22, 541-562

Guttman. L. (1952) Multiple group methods for common-factor analysis: their basis, computation, and interpretation. Psychometrika, 17, (2) 209-222.

H.H. Harman (1967) Modern Factor Analysis. University of Chicago Press.

Li, Sijia and Savalei, Victoria (2025), Evaluating Statistical Fit of Confirmatory Bifactor Models: Updated Recommendations and a Review of Current Practice. Psychological Methods. doi.org/10.1037/met0000730

MacCallum, Robert C. and Browne, Michael W. and Cai, Li (2007) Factor analysis models as approximations. In Cudeck, Robert and MacCallum, Robert C. (Eds). Factor analysis at 100: Historical developments and future directions. Lawrence Erlbaum Associates Publishers.

See Also

fa for exploratory analysis and more discussion of factor analysis in general. omegaStats to allow quick comparisons with other functions. irt.CFA calculates and displays the CFA analysis of tetra/polychoric correlations in the IRT metric.

Examples

```
 #test set from Harman Table 7.1 P 116
har5 <- structure(c(1, 0.485, 0.4, 0.397, 0.295, 0.485, 1, 0.397, 0.397,
0.247, 0.4, 0.397, 1, 0.335, 0.275, 0.397, 0.397, 0.335, 1, 0.195,
0.295, 0.247, 0.275, 0.195, 1), dim = c(5L, 5L), dimnames = list(
    c("V1", "V2", "V3", "V4", "V5"), c("V1", "V2", "V3", "V4",
    "V5")))
#in all of these examples, I am setting n.iter to be very small in order to pass CRAN test
CFA(har5,n.iter=5)  #The Harman example.  Note that the model not necessary for the 1 factor case.

CFA(Harman_5,n.iter=5)  #the Harman example of a Heywood case

v9 <- sim.hierarchical()  #Create a 3 correlated factor model using default values
model <- 'F1=~ V1 + V2 + V3
          F2=~ V4 + V5 + V6
          F3 =~ V7 +V8 + V9'
CFA(model,v9,n.iter=5)


model9 <- 'F1 =~ .9*V1 + .8*V2 + .7*V3
           F2 =~ .8*V4 + .7*V5 +.6*V6
           F3 =~ .7*V7 + .6*V8 +.5*V9
           F1 ~ .6*F2 + .5*F3
           F2 ~  .4*F3'
#An alternative way to create 3 correlated factors
#note that CFA drops the coefficients, the model is for generating the data
 #lavaan does not drop coefficients
v9s <- sim(model9,n=500)
 test <- CFA(model,v9s$observed , n.iter=5)  #do a cfa using Lavaan syntax
 test.bi <- CFA.bifactor(model9,v9,n.iter=5)
 test.hi <- CFA.bifactor(model9,v9,g=TRUE,n.iter=5)

#graphic displays make the output more understandable.
diagram(test)   #show three correlated factors
diagram(test.bi) #show the bifactor solution
diagram(test.hi) #show the hierarchical/higher order solution

#this next example requires psychTools  not run
#for a four factor model using keys

#CFA(psychTools::ability.keys[-1],psychTools::ability,  cor="tet")

CFA(bfi.keys,bfi, n.iter=5)   # a five factor model of the bfi items
#irt.CFA(bfi.keys,bfi[1:25])

colnames(Thurstone) <- rownames(Thurstone) <- paste0("x",1:9  ) #to match lavaan syntax
model <- HS.model <- ' visual  =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9 '
c3 <- CFA(model,Thurstone,n.obs=213,n.iter=5)  #compare with the lavaan solution
           #which has a smaller chi^2
```

```
c3  #show the result
diagram(c3)  #graphically display the result

c3.hi <- CFA.bifactor(model,Thurstone,n.obs=213,n.iter=5)

c5 <- irt.CFA(bfi.keys,bfi[1:25],n.iter=5)

#do not run the next examples, they require lavaan
#They compare lavaan cfa solutions to CFA

if(FALSE) {
#
#The next examples require lavaan and are thus not run
library(lavaan)
#The basic lavaan example
fit <- cfa(model,sample.cov=Thurstone,sample.nobs=213,std.lv=TRUE, estimator="ML")
factor.congruence(fit,c3)  #identical loadings to 2 decimals
round(fit@Model@GLIST$lambda-c3$loadings,4)
#add the g factor
HS.model <- ' general =~  x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
              visual  =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9 '
g.fit <- cfa(HS.model,sample.cov=Thurstone,sample.nobs=213,std.lv=TRUE,orthogonal=TRUE)
fa.congruence(g.fit,c3.hi) #identical congruence to 2 decimals
round(g.fit@Model@GLIST$lambda-c3.hi$loadings,2)  #loadings with ULS are identicla

#All 24 variables from Harman
harman24 <- psychTools::holzinger.raw[157:301,8:31]
colnames(harman24) <- paste0("v",1:24)
 mod.24<-'
 g=~v1+v2+v3+v4+v5+v6+v7+v8+v9+v10+v11+v12+v13+v14+v15+v16+v17+v18+v19+v20+v21+v22+v23+v24
  spatial =~ v1 + v2 + v3 + v4
                verbal=~ v5 + v6 + v7 + v8 + v9
                perceptual =~ v10 + v11 + v12 + v13
                recognition =~ v14+v15 + v16  + v17
                memory =~ v18 + v19 + v20
               '
#the lavaan solution (requires lavaan)
lav.har.uls <- cfa(mod.24, data=harman24,std.lv=TRUE,std.ov=TRUE, orthogonal=TRUE,
           estimator="ULS")

lav.har.ml <-cfa(mod.24, data=harman24,std.lv=TRUE,std.ov=TRUE,orthogonal=TRUE)

 model.har24.5  <- 'spatial =~ v1 + v2 + v3 + v4
                verbal=~ v5 + v6 + v7 + v8 + v9
                perceptual =~ v10 + v11 + v12 + v13
                recognition =~ v14+v15 + v16  + v17
                memory =~ v18 + v19 + v20'
 cfa.har24 <- CFA(model.har24.5,harman24)
 cfa.har.bi <- CFA.bifactor(model.har24.5,harman24)

 factor.congruence(list(lav.har.uls,lav.har.ml,cfa.har.bi)) #g is very good f1-4 very good
```

```
round(lav.har@Model@GLIST$lambda-cfa.har.bi$loadings,2)   #not the same
    }
```

---

circ.tests *Apply four tests of circumplex versus simple structure*

Description

Rotations of factor analysis and principal components analysis solutions typically try to represent correlation matrices as simple structured. An alternative structure, appealing to some, is a circumplex structure where the variables are uniformly spaced on the perimeter of a circle in a two dimensional space. Generating these data is straightforward, and is useful for exploring alternative solutions to affect and personality structure.

Usage

```
circ.tests(loads, loading = TRUE, sorting = TRUE)
```

Arguments

loads	A matrix of loadings loads here
loading	Are these loadings or a correlation matrix loading
sorting	Should the variables be sorted sorting

Details

"A common model for representing psychological data is simple structure (Thurstone, 1947). According to one common interpretation, data are simple structured when items or scales have nonzero factor loadings on one and only one factor (Revelle & Rocklin, 1979). Despite the commonplace application of simple structure, some psychological models are defined by a lack of simple structure. Circumplexes (Guttman, 1954) are one kind of model in which simple structure is lacking.

"A number of elementary requirements can be teased out of the idea of circumplex structure. First, circumplex structure implies minimally that variables are interrelated; random noise does not a circumplex make. Second, circumplex structure implies that the domain in question is optimally represented by two and only two dimensions. Third, circumplex structure implies that variables do not group or clump along the two axes, as in simple structure, but rather that there are always interstitial variables between any orthogonal pair of axes (Saucier, 1992). In the ideal case, this quality will be reflected in equal spacing of variables along the circumference of the circle (Gurtman, 1994; Wiggins, Steiger, & Gaelick, 1981). Fourth, circumplex structure implies that variables have a constant radius from the center of the circle, which implies that all variables have equal communality on the two circumplex dimensions (Fisher, 1997; Gurtman, 1994). Fifth, circumplex structure implies that all rotations are equally good representations of the domain (Conte & Plutchik, 1981; Larsen & Diener, 1992). (Acton and Revelle, 2004)

Acton and Revelle reviewed the effectiveness of 10 tests of circumplex structure and found that four did a particularly good job of discriminating circumplex structure from simple structure, or

circumplexes from ellipsoidal structures. Unfortunately, their work was done in Pascal and is not easily available. Here we release R code to do the four most useful tests:

1 The Gap test of equal spacing

2 Fisher's test of equality of axes

3 A test of indifference to Rotation

4 A test of equal Variance of squared factor loadings across arbitrary rotations.

To interpret the values of these various tests, it is useful to compare the particular solution to simulated solutions representing pure cases of circumplex and simple structure. See the example output from `circ.simulation` and compare these plots with the results of the circ.test.

Value

A list of four items is returned. These are the gap, fisher, rotation and variance test results.

`gaps`	gap.test
`fisher`	fisher.test
`RT`	rotation.test
`VT`	variance.test

Note

Of the 10 criterion discussed in Acton and Revelle (2004), these tests operationalize the four most useful.

Author(s)

William Revelle

References

Acton, G. S. and Revelle, W. (2004) Evaluation of Ten Psychometric Criteria for Circumplex Structure. Methods of Psychological Research Online, Vol. 9, No. 1 https://personality-project.org/revelle/publications/acton.revelle.mpr110_10.pdf

See Also

To understand the results of the circ.tests it it best to compare it to simulated values. Thus, see `circ.simulation`, `sim.circ`

Examples

```
circ.data <- circ.sim(24,500)
circ.fa <- fa(circ.data,2)
plot(circ.fa,title="Circumplex Structure")
ct <- circ.tests(circ.fa)
#compare with non-circumplex data
simp.data <- item.sim(24,500)
simp.fa <- fa(simp.data,2)
```

```
plot(simp.fa,title="Simple Structure")
st <- circ.tests(simp.fa)
res <- rbind(ct[1:4],st[1:4])
rownames(res) <- c("circumplex","Simple")
print(res,digits=2)
```

`cluster.fit` *cluster Fit: fit of the cluster model to a correlation matrix*

Description

How well does the cluster model found by `ICLUST` fit the original correlation matrix? A similar algorithm `factor.fit` is found in `VSS`. This function is internal to ICLUST but has more general use as well.

In general, the cluster model is a Very Simple Structure model of complexity one. That is, every item is assumed to represent only one factor/cluster. Cluster fit is an analysis of how well this model reproduces a correlation matrix. Two measures of fit are given: cluster fit and factor fit. Cluster fit assumes that variables that define different clusters are orthogonal. Factor fit takes the loadings generated by a cluster model, finds the cluster loadings on all clusters, and measures the degree of fit of this somewhat more complicated model. Because the cluster loadings are similar to, but not identical to factor loadings, the factor fits found here and by `factor.fit` will be similar.

Usage

```
cluster.fit(original, load, clusters, diagonal = FALSE)
```

Arguments

`original`	The original correlation matrix being fit
`load`	Cluster loadings – that is, the correlation of individual items with the clusters, corrected for item overlap
`clusters`	The cluster structure
`diagonal`	Should we fit the diagonal as well?

Details

The cluster model is similar to the factor model: R is fitted by C'C. Where C <- Cluster definition matrix x the loading matrix. How well does this model approximate the original correlation matrix and how does this compare to a factor model?

The fit statistic is a comparison of the original (squared) correlations to the residual correlations. Fit = 1 - r*2/r2 where r* is the residual correlation of data - model and model = C'C.

Value

`clusterfit`	The cluster model is a reduced form of the factor loading matrix. That is, it is the product of the elements of the cluster matrix * the loading matrix.
`factorfit`	How well does the complete loading matrix reproduce the correlation matrix?

Author(s)

Maintainer: William Revelle <revelle@northwestern.edu>

References

https://personality-project.org/r/r.ICLUST.html

See Also

VSS, ICLUST, factor2cluster, cluster.cor, factor.fit

Examples

```
r.mat<- Harman74.cor$cov
iq.clus <- ICLUST(r.mat,nclusters =2)
fit <- cluster.fit(r.mat,iq.clus$loadings,iq.clus$clusters)
fit
```

---

cluster.loadings *Find item by cluster correlations, corrected for overlap and reliability*

Description

Given a n x n correlation matrix and a n x c matrix of -1,0,1 cluster weights for those n items on c clusters, find the correlation of each item with each cluster. If the item is part of the cluster, correct for item overlap. Part of the ICLUST set of functions, but useful for many item analysis problems.

Usage

```
cluster.loadings(keys, r.mat, correct = TRUE,SMC=TRUE)
```

Arguments

keys	Cluster keys: a matrix of -1,0,1 cluster weights
r.mat	A correlation matrix
correct	Correct for reliability
SMC	Use the squared multiple correlation as a communality estimate, otherwise use the greatest correlation for each variable

Details

Given a set of items to be scored as (perhaps overlapping) clusters and the intercorrelation matrix of the items, find the clusters and then the correlations of each item with each cluster. Correct for item overlap by replacing the item variance with its average within cluster inter-item correlation.

Although part of ICLUST, this may be used in any SAPA (https://www.sapa-project.org/) application where we are interested in item-whole correlations of items and composite scales.

For information about SAPA see Revelle et al, 2010, 2016. For information about SAPA based measures of ability, see https://icar-project.org.

These loadings are particularly interpretable when sorted by absolute magnitude for each cluster (see ICLUST.sort).

Value

loadings	A matrix of item-cluster correlations (loadings)
cor	Correlation matrix of the clusters
corrected	Correlation matrix of the clusters, raw correlations below the diagonal, alpha on diagonal, corrected for reliability above the diagonal
sd	Cluster standard deviations
alpha	alpha reliabilities of the clusters
G6	G6* Modified estimated of Guttman Lambda 6
count	Number of items in the cluster

Note

Although part of ICLUST, this may be used in any SAPA application where we are interested in item- whole correlations of items and composite scales.

Author(s)

Maintainer: William Revelle <revelle@northwestern.edu>

References

ICLUST: https://personality-project.org/r/r.ICLUST.html

Revelle, W., Wilt, J., and Rosenthal, A. (2010) Individual Differences in Cognition: New Methods for examining the Personality-Cognition Link In Gruszka, A. and Matthews, G. and Szymura, B. (Eds.) Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control, Springer.

Revelle, W, Condon, D.M., Wilt, J., French, J.A., Brown, A., and Elleman, L.G. (2016) Web and phone based data collection using planned missing designs. In Fielding, N.G., Lee, R.M. and Blank, G. (Eds). SAGE Handbook of Online Research Methods (2nd Ed), Sage Publcations.

See Also

ICLUST, factor2cluster, cluster.cor

Examples

```
r.mat<- Harman74.cor$cov
clusters <- matrix(c(1,1,1,rep(0,24),1,1,1,1,rep(0,17)),ncol=2)
cluster.loadings(clusters,r.mat)
```

`cluster.plot` *Plot factor/cluster loadings and assign items to clusters by their highest loading.*

Description

Cluster analysis and factor analysis are procedures for grouping items in terms of a smaller number of (latent) factors or (observed) clusters. Graphical presentations of clusters typically show tree structures, although they can be represented in terms of item by cluster correlations.

Cluster.plot plots items by their cluster loadings (taken, e.g., from `ICLUST`) or factor loadings (taken, eg., from `fa`). Cluster membership may be assigned apriori or may be determined in terms of the highest (absolute) cluster loading for each item.

If the input is an object of class "kmeans", then the cluster centers are plotted.

Usage

```
cluster.plot(ic.results, cluster = NULL, cut = 0, labels=NULL,
          title = "Cluster plot",pch=18,pos,show.points=TRUE,choose=NULL,...)
fa.plot(ic.results, cluster = NULL, cut = 0, labels=NULL,title,
   jiggle=FALSE,amount=.02,pch=18,pos,show.points=TRUE,choose=NULL,main=NULL,...)
factor.plot(ic.results, cluster = NULL, cut = 0, labels=NULL,title,jiggle=FALSE,
                  amount=.02,pch=18,pos,show.points=TRUE,...)  #deprecated
```

Arguments

`ic.results`	A factor analysis or cluster analysis output including the loadings, or a matrix of item by cluster correlations. Or the output from a kmeans cluster analysis.
`cluster`	A vector of cluster membership
`cut`	Assign items to clusters if the absolute loadings are > cut
`labels`	If row.names exist they will be added to the plot, or, if they don't, labels can be specified. If labels =NULL, and there are no row names, then variables are labeled by row number.)
`title`	Any title
`jiggle`	When plotting with factor loadings that are almost identical, it is sometimes useful to "jiggle" the points by jittering them. The default is to not jiggle.
`amount`	if jiggle=TRUE, then how much should the points be jittered?

pch	factor and clusters are shown with different pch values, starting at pch+1
pos	Position of the text for labels for two dimensional plots. 1=below, 2 = left, 3 = above, 4= right
show.points	When adding labels to the points, should we show the points as well as the labels. For many points, better to not show them, just the labels.
choose	Specify the factor/clusters to plot
main	Any title – redundant with title
...	Further options to plot

Details

Results of either a factor analysis or cluster analysis are plotted. Each item is assigned to its highest loading factor, and then identified by variable name as well as cluster (by color). The cluster assignments can be specified to override the automatic clustering by loading. Both of these functions may be called directly or by calling the generic plot function. (see example).

Value

Graphical output is presented.

Author(s)

William Revelle

See Also

ICLUST, ICLUST.graph, fa.graph, plot.psych

Examples

```
circ.data <- circ.sim(24,500)
circ.fa <- fa(circ.data,2)
plot(circ.fa,cut=.5)
f5 <- fa(bfi[1:25],5)
plot(f5,labels=colnames(bfi)[1:25],show.points=FALSE)
plot(f5,labels=colnames(bfi)[1:25],show.points=FALSE,choose=c(1,2,4))
```

---

cluster2keys *Convert a cluster vector (from e.g., kmeans) to a keys matrix suitable for scoring item clusters.*

Description

The output of the kmeans clustering function produces a vector of cluster membership. The score.items and cluster.cor functions require a matrix of keys. cluster2keys does this.

May also be used to take the output of an ICLUST analysis and find a keys matrix. (By doing a call to the factor2cluster function.

Usage

```
cluster2keys(c)
```

Arguments

c	A vector of cluster assignments or an object of class “kmeans" that contains a vector of clusters.

Details

Note that because kmeans will not reverse score items, the clusters defined by kmeans will not necessarily match those of ICLUST with the same number of clusters extracted.

Value

keys	A matrix of keys suitable for score.items or cluster.cor

Author(s)

William Revelle

See Also

cluster.cor,score.items, factor2cluster, make.keys

Examples

```
test.data <- Harman74.cor$cov
kc <- kmeans(test.data,4)
keys <- cluster2keys(kc)
keys  #these match those found by ICLUST
cluster.cor(keys,test.data)
```

---

cohen.d *Find Cohen d and confidence intervals*

Description

Given a data.frame or matrix, find the standardized mean difference (Cohen’s d) and confidence intervals for each variable depending upon a grouping variable. Convert the d statistic to the r equivalent, report the student’s t statistic and associated p values, and return statistics for both values of the grouping variable. The Mahalanobis distance between the centroids of the two groups in the space defined by all the variables ia also found. Confidence intervals for Cohen d for one group (difference from 0) may also be found. Several measures of the distributional overlap (e.g. OVL, OVL2, etc.) are available.

Usage

```
cohen.d(x, group,alpha=.05,std=TRUE,sort=NULL,dictionary=NULL,MD=TRUE,data=NULL)
cohenPooled(cd=list(), cohen="TRUE")
d.robust(x,group,trim=.2)
cohen.d.ci(d,n=NULL,n2=NULL,n1=NULL,alpha=.05)
d.ci(d,n=NULL,n2=NULL,n1=NULL,alpha=.05)
cohen.d.by(x,group,group2,alpha=.05,MD=TRUE)
d2r(d)
r2d(rho)
d2t(d,n=NULL,n2=NULL,n1=NULL)
t2d(t,n=NULL,n2=NULL,n1=NULL)
m2t(m1,m2,s1,s2,n1=NULL,n2=NULL,n=NULL,pooled=TRUE)  #returns d invisibily
m2d(m1,m2,s1,s2,n1=NULL,n2=NULL,n=NULL,pooled=TRUE)
d2OVL(d)  #Percent overlap for 1 distribtion
d2OVL2(d)  #Percent overlap joint distribution
d2CL(d)   #Common language effect size
d2U3(d)   #Proportion in higher group exceedding median of lower group
cd.validity(d, keys, abs=TRUE)
```

Arguments

x	A data frame or matrix (can be specified in formula mode)
group	Some dichotomous grouping variable (may be specified using formula input (see example))
group2	Apply cohen.d for each of the subgroups defined by group2 (may be specified by formula as well)
data	If using formula mode and specifying a particular variable (see example)
cd	A list of objects returned from multiple runs of cohen.d to be pooled
cohen	By default, pool cohen.d's. if FALSE pool hedges.g
d	An effect size
keys	A list of scoring keys (similar to scoreItems)
abs	When finding average cd validities, should we take absolute values (TRUE)
trim	The amount of trimming used in finding the means and sds in d.robust
n	Total sample size (of groups 1 and 2)
n1	Sample size of group 1 (if only one group)
n2	Sample size of group 2
pooled	Pool the two variances
t	Student's t statistic
alpha	1-alpha is the width of the confidence interval
std	Find the correlation rather covariance matrix
rho	A correlation to be converted to an effect size
m1	Mean of group 1

m2	Mean of group 2
s1	Standard deviation of group 1
s2	Standard deviation of group 2
sort	Should we sort (and if so, in which direction), the results of cohen.d? Directions are "decreasing" or "increasing". If TRUE, sorts in a decreasing order.
dictionary	What are the items being described?
MD	Find Mahalanobis distance in cohen.d.

Details

There are many ways of reporting how two groups differ. Cohen's d statistic is just the differences of means expressed in terms of the pooled within group standard deviation. This is insensitive to sample size. r is the a universal measure of effect size that is a simple function of d, but is bounded -1 to 1. The t statistic is merely d * sqrt(n)/2 and thus reflects sample size.

$$d = \frac{M2 - M1}{Sp}$$

where Sp is the pooled standard deviation.

$$Sp = \sqrt{\frac{(n1-1)*s1^2 + (n2-1)*s2^2}{N}}$$

Cohens d uses N as the divisor for the pooled sums of squares. Hedges g uses N-2.

Confidence intervals for Cohen's d are found by converting the d to a t, finding the confidence intervals for t, and then converting those back to ds. This take advantage of the uniroot function and the non-centrality parameter of the t distribution.

The results of cohen.d may be displayed using the error.dots function. This will include the labels provided in the dictionary.

In the case of finding the confidence interval (using cohen.d.ci for a comparison against 0 (the one sample case), specify n1. This will yield a d = t/sqrt(n1) whereas in the case of the difference between two samples, d = 2*t/sqrt(n) (for equal sample sizes n = n1+ n2) or d = t/sqrt(1/n1 + 1/n2) for the case of unequal sample sizes.

Since we find d and then convert this to t, using d2t, the question is how to pool the variances. Until 7/14/21 I was using the total n to estimate the t and thus the p values. In response to a query (see news), I switched to using the actual sample size ns (n1 and n2) and then finding t based upon the hedges g value. This produces t values as reported by t.test with the var.equal = TRUE option.

It is probably useful to comment that the various confidence intervals reported are based upon normal theory and should be interpreted cautiously.

cohen.d.by will find Cohen's d for groups for each subset of the data defined by group2. The summary of the output produces a simplified listing of the d values for each variable for each group. May be called directly from cohen.d by using formula input and specifying two grouping variables.

d.robust follows Algina et al. 2005) to find trimmed means (trim =.2) and Winsorize variances (trim =.2). Supposedly, this provides a more robust estimate of effect sizes.

m2t reports Student's t.test for two groups given their means, standard deviations, and sample size. This is convenient when checking statistics where those estimates are provided, but the raw data are

not available. By default, it gives the pooled estimate of variance, but if pooled is FALSE, it applies Welch's correction.

The Mahalanobis Distance combines the individual ds and weight them by their unique contribution: $D = \sqrt{d' R^{-1} d}$. By default, `cohen.d` will find the Mahalanobis distance between the two groups (if there is more than one DV.) This requires finding the correlation of all of the DVs and can fail if that matrix is not invertible because some pairs do not exist. Thus, setting MD=FALSE will prevent the Mahalanobis calculation.

In May, 2026, I added the partial d (wt.d) which is d controlling for the other variables. This is a generalization of the logic for finding betas in multiple regression. See the examples.

Marco del Giudice (2019) has a very helpful paper discussing how to interpret d and Md in terms of various overlap coefficients. These may be found by the use of the `d2OVL` (percent overlap for 1 distribution), `d2OVL2` percent overlap of joint distributions, `d2CL` (the common language effect size), and `d2U3` (proportion in higher group exceeding median of the lower group).

$$OVL = 2\phi(-d/2)$$

is the proportion of overlap (and gets smaller the larger the d). where Phi is the cumulative density function of the normal distribution.

$$OVL_2 = \frac{OVL}{2 - OVL}$$

The proportion of individuals in one group above the median of the other group is U3

$$U_3 = \phi_d$$

.

The Common Language Effect size

$$CL = \phi(d * \sqrt{2})$$

These last two get larger with (abs (d)). For graphic displays of Cohen's d and Mahalanobis D, see the `scatterHist` examples, or the example from the psychTools::GERAS data set.

Value

`d`	Cohen's d statistic, including the upper and lower confidence levels
`hedges.g`	Hedge's g statistic, including the upper and lower confidence levels
`M.dist`	Mahalanobis distance between the two groups
`t`	Student's t statistic
`r`	The point biserial r equivalent of d
`n`	sample size used for each analysis
`p`	The probability of abs(t)>0
`descriptive`	The descriptive statistics for each group. This is useful to show means and then the d values.
`OVL`	etc. some of the measures of overlap discussed by DelGiudice, 2009

Note

Cohen and Hedges differ in they way they calculate the pooled within group standard deviation. I find the treatment by McGrath and Meyer to be most helpful in understanding the differences.

Author(s)

William Revelle

References

Cohen, Jackob (1988) Statistical Power Analysis for the Behavioral Sciences. 2nd Edition, Lawrence Erlbaum Associates.

Algina, James and Keselman, H. J. and Penfield, Randall D. (2005) An Alternative to Cohen's Standardized Mean Difference Effect Size: A Robust Parameter and Confidence Interval in the Two Independent Groups Case. Psychological Methods. 10, 317-328.

Goulet-Pelletier, Jean-Christophe and Cousineau, Denis.(2018) A review of effect sizes and their confidence intervals, Part I: The Cohen's d family. The Quantitative Methods for Psychology, 14, 242-265.

Marco Del Giudice (2019) Measuring Sex Differences and Similarities, (in VanderLaan and Wong (ed.) Gender and sexuality development: Contemporary theory and research.)

McGrath, Robert E and Meyer, Gregory J. (2006) When effect sizes disagree: the case of r and d. Psychological methods, 11, 4, 386-401.

See Also

`describeBy`, `describe` and `plot` or `error.dots` to display the results. `scatterHist` to show d and MD for pairs of variables. (See in particular the use of `scatterHist` on psychTools::GERAS daa set.)

Examples

```
cohen.d(sat.act,"gender")
#robust version
round(d.robust(sat.act,"gender")$robust.d,2)

#formula input is nicer
cohen.d(sat.act ~ gender) #formula input version
#if we want to report the group means, we grab the data in descriptive
cd <- cohen.d(sat.act ~ gender)
cd.df <- data.frame(d = cd$cohen.d[,"effect"], male = cd$descriptive$mean[1,cd$order[-1]],
      female = cd$descriptive$mean[2, cd$order[-1]])
#report cohen.d by another group
cd <- cohen.d.by(sat.act,"gender","education")
cohen.d(SATV + SATQ ~ gender, data=sat.act) #just choose two variables
summary(cd)  #summarize the output

#now do it for subsets and then pool them
cd1 <- cohen.d(sat.act[1:300,],group= "gender")
cd2 <- cohen.d(sat.act[301:600,],group="gender")
```

```
cd.pooled <- cohenPooled(list(cd1,cd2))
#graph them using error.dots
error.dots(stats=cd.pooled)
#or show them as two sets together using plot
plot(cd1,sort=FALSE)
plot(cd2,sort=FALSE,add=TRUE,col="red",lwd=2)



#formula version combines these functions
cd <- cohen.d(sat.act ~ gender + education)  #find d by gender for each level of education
summary(cd)

if(require(psychTools)) {
mf.pers <- cohen.d(M.pers + F.pers ~ gender,data=GERAS.scales)
mf.pers.cog <- cohen.d(M.pers + F.pers + M.cog + F.cog ~ gender,data=GERAS.scales)
mf.pers.cog.act  <- cohen.d(M.pers + F.pers + M.cog + F.cog  + M.act  + F.act ~ gender,
     data=GERAS.scales)
mf.pers.cog.act.MF  <- cohen.d(M.pers + F.pers + M.cog + F.cog  + M.act  + F.act  + M + F ~ gender,
      data=GERAS.scales)
mf.pers.cog.act.MF.all  <- cohen.d(M.pers + F.pers + M.cog + F.cog  + M.act  + F.act  + M + F
     + MF.all~ gender,data=GERAS.scales)

MF.all =cohen.d(MF.all~gender,data=GERAS.scales)
#summarize
summary.df <- data.frame(Scale_name=cs(person, person.cognitive, person.cognitive.activity,
       All_MF_scales,MF.all),
      N_subscales =c(2,4,6,8,1) , N_items =c(20,34,50,50,50),D = round(c(mf.pers$M.dist,
     mf.pers.cog$M.dist, mf.pers.cog.act$M.dist, mf.pers.cog.act.MF$M.dist,
      MF.all$M.dist) ,2))
}

#now show several examples of confidence intervals
#one group (d vs 0)
#consider the t from the cushny data set
t2d( -4.0621,n1=10)
d.ci(-1.284549,n1=10)  #the confidence interval of the effect of drug on sleep
#two groups
d.ci(.62,n=64)  #equal group size
d.ci(.62,n1=35,n2=29) #unequal group size
#several examples of d and t from data
m2d(52.58,-70.65,49.9,47.5) #Terman and Miles 1936

#graphically show the various overlap statistics
curve(d2OVL2(x),0,3,xlab="d",ylab="",lty="dashed",
       main="Four representations of effect size (d) ")
curve(d2OVL(x),0,3,xlab="d",add=TRUE,)
curve(d2CL(x),0,3,add=TRUE)
curve(d2U3(x), add=TRUE,lty="dotted")
text(1,.37,"OVL2")
text(2,.37,"OVL")
text(1,.88,"U3")
text(2, .88,"CL")
```

cohen.kappa *Find Cohen's kappa and weighted kappa coefficients for correlation of two raters*

Description

Cohen's kappa (Cohen, 1960) and weighted kappa (Cohen, 1968) may be used to find the agreement of two raters when using nominal scores. Light's kappa is just the average cohen.kappa if using more than 2 raters.

weighted.kappa is (probability of observed matches - probability of expected matches)/(1 - probability of expected matches). Kappa just considers the matches on the main diagonal. Weighted kappa considers off diagonal elements as well.

Usage

```
cohen.kappa(x, w=NULL,n.obs=NULL,alpha=.05,levels=NULL,w.exp=2)
wkappa(x, w = NULL)    #deprectated
```

Arguments

x	Either a two by n data with categorical values from 1 to p or a p x p table. If a data array, a table will be found.
w	A p x p matrix of weights. If not specified, they are set to be 0 (on the diagonal) and (distance from diagonal) off the diagonal)^2.
n.obs	Number of observations (if input is a square matrix.
alpha	Probability level for confidence intervals
levels	Specify the levels if some levels of x or y are completely missing. See Examples
w.exp	Exponent to apply to weights matrix – see examples

Details

When cateogorical judgments are made with two cateories, a measure of relationship is the phi coefficient. However, some categorical judgments are made using more than two outcomes. For example, two diagnosticians might be asked to categorize patients three ways (e.g., Personality disorder, Neurosis, Psychosis) or to categorize the stages of a disease. Just as base rates affect observed cell frequencies in a two by two table, they need to be considered in the n-way table (Cohen, 1960).

Kappa considers the matches on the main diagonal. A penalty function (weight) may be applied to the off diagonal matches. If the weights increase by the square of the distance from the diagonal, weighted kappa is similar to an Intra Class Correlation (ICC).

Derivations of weighted kappa are sometimes expressed in terms of similarities, and sometimes in terms of dissimilarities. In the latter case, the weights on the diagonal are 1 and the weights off the diagonal are less than one. In this case, if the weights are 1 - squared distance from the diagonal / k, then the result is similar to the ICC (for any positive k).

cohen.kappa may use either similarity weighting (diagonal = 0) or dissimilarity weighting (diagonal = 1) in order to match various published examples.

The input may be a two column data.frame or matrix with columns representing the two judges and rows the subjects being rated. Alternatively, the input may be a square n x n matrix of counts or proportion of matches. If proportions are used, it is necessary to specify the number of observations (n.obs) in order to correctly find the confidence intervals.

The confidence intervals are based upon the variance estimates discussed by Fleiss, Cohen, and Everitt who corrected the formulae of Cohen (1968) and Blashfield.

Some data sets will include data with numeric categories with some category values missing completely. In the sense that kappa is a measure of category relationship, this should not matter. But when finding weighted kappa, the number of categories weighted will be less than the number of categories potentially in the data. This can be remedied by specifying the levels parameter. This is a vector of the levels potentially in the data (even if some are missing). See the examples.

If there are more than 2 raters, then the average of all raters is known as Light's kappa. (Conger, 1980).

Following a request from Julius Pfadt, the weights matrix may be formed by squared distances (the default) or linear distances (w.exp=1)

Value

`kappa`	Unweighted kappa
`weighted.kappa`	The default weights are quadratric.
`var.kappa`	Variance of kappa
`var.weighted`	Variance of weighted kappa
`n.obs`	number of observations
`weight`	The weights used in the estimation of weighted kappa
`confid`	The alpha/2 confidence intervals for unweighted and weighted kappa
`plevel`	The alpha level used in determining the confidence limits

Note

As is true of many R functions, there are alternatives in other packages. The Kappa function in the vcd package estimates unweighted and weighted kappa and reports the variance of the estimate. The input is a square matrix. The ckappa and wkappa functions in the psy package take raw data matrices. The kappam.light function from the irr package finds Light's average kappa.

To avoid confusion with Kappa (from vcd) or the kappa function from base, the function was originally named wkappa. With additional features modified from psy::ckappa to allow input with a different number of categories, the function has been renamed cohen.kappa.

Unfortunately, to make it more confusing, the weights described by Cohen are a function of the reciprocals of those discucssed by Fleiss and Cohen. The cohen.kappa function uses the appropriate formula for Cohen or Fleiss-Cohen weights.

There are some cases where the large sample size approximation of Fleiss et al. will produce confidence intervals exceeding +/- 1. Clearly, for these cases, the upper (or lower for negative

values) should be set to 1. Boot strap resampling shows the problem is that the values are not symmetric. See the last (unrun) example.

It is also possible to have more than 2 raters. In this case, cohen.kappa is reported for all pairs of raters (e.g. R1 and R2, R1 and R3, ... R3 and R4). To see the confidence intervals for these cohen.kappas, use the print command with the all=TRUE option. (See the exmaple of multiple raters.)

Author(s)

William Revelle

References

Banerjee, M., Capozzoli, M., McSweeney, L and Sinha, D. (1999) Beyond Kappa: A review of interrater agreement measures The Canadian Journal of Statistics / La Revue Canadienne de Statistique, 27, 3-23

Cohen, J. (1960). A coefficient of agreement for nominal scales. Educational and Psychological Measurement, 20 37-46

Cohen, J. (1968). Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. Psychological Bulletin, 70, 213-220.

Conger, A. J. (1980) Integration and generalization of kappas for multiple raters, Psychological Bulletin,, 88, 322-328.

Fleiss, J. L., Cohen, J. and Everitt, B.S. (1969) Large sample standard errors of kappa and weighted kappa. Psychological Bulletin, 72, 332-327.

Light, R. J. (12971) Measures of response agreement for qualitative data: Some generalizations and alternatives, Psychological Bulletin, 76, 365-377.

Zwick, R. (1988) Another look at interrater agreement. Psychological Bulletin, 103, 374 - 378.

Examples

```
#rating data (with thanks to Tim Bates)
rater1 = c(1,2,3,4,5,6,7,8,9) # rater one's ratings
rater2 = c(1,3,1,6,1,5,5,6,7) # rater two's ratings
cohen.kappa(x=cbind(rater1,rater2))

#data matrix taken from Cohen
cohen <- matrix(c(
0.44, 0.07, 0.09,
0.05, 0.20, 0.05,
0.01, 0.03, 0.06),ncol=3,byrow=TRUE)

#cohen.weights  weight differences
cohen.weights <- matrix(c(
0,1,3,
1,0,6,
3,6,0),ncol=3)
```

```
cohen.kappa(cohen,cohen.weights,n.obs=200)
#cohen reports .492 and .348

#another set of weights
#what if the weights are non-symmetric
wc <- matrix(c(
0,1,4,
1,0,6,
2,2,0),ncol=3,byrow=TRUE)
cohen.kappa(cohen,wc)
#Cohen reports kw = .353

cohen.kappa(cohen,n.obs=200)  #this uses the squared weights

fleiss.cohen <- 1 - cohen.weights/9
cohen.kappa(cohen,fleiss.cohen,n.obs=200)

#however, Fleiss, Cohen and Everitt weight similarities
fleiss <- matrix(c(
106, 10,4,
22,28, 10,
2, 12,  6),ncol=3,byrow=TRUE)

#Fleiss weights the similarities
weights <- matrix(c(
 1.0000, 0.0000, 0.4444,
 0.0000, 1.0000, 0.6667,
 0.4444, 0.6667, 1.0000),ncol=3)

 cohen.kappa(fleiss,weights,n.obs=200)

 #another example is comparing the scores of two sets of twins
 #data may be a 2 column matrix
 #compare weighted and unweighted
 #also look at the ICC for this data set.
 twins <- matrix(c(
    1, 2,
    2, 3,
    3, 4,
    5, 6,
    6, 7), ncol=2,byrow=TRUE)
  cohen.kappa(twins)

#data may be explicitly categorical
x <- c("red","yellow","blue","red")
y <- c("red",  "blue", "blue" ,"red")
xy.df <- data.frame(x,y)
ck <- cohen.kappa(xy.df)
ck
ck$agree

#Example for specifying levels
#The problem of missing categories (from Amy Finnegan)
```

```
#We need to specify all the categories possible using the levels option
numbers <- data.frame(rater1=c(6,3,7,8,7),
                      rater2=c(6,1,8,5,10))
cohen.kappa(numbers)  #compare with the next analysis
cohen.kappa(numbers,levels=1:10)  #specify the number of levels
              #   these leads to slightly higher weighted kappa

#finally, input can be a data.frame of ratings from more than two raters
ratings <- matrix(rep(1:5,4),ncol=4)
ratings[1,2] <- ratings[2,3] <- ratings[3,4] <- NA
ratings[2,1] <- ratings[3,2] <- ratings[4,3] <- 1
ck <- cohen.kappa(ratings)
ck  #just show the raw and weighted kappas
print(ck, all=TRUE)  #show the confidence intervals as well


 #In the case of confidence intervals being artificially truncated to +/- 1, it is
 #helpful to compare the results of a boot strap resample
 #ck.boot <-function(x,s=1:nrow(x)) {cohen.kappa(x[s,])$kappa}
 #library(boot)
 #ckb <- boot(x,ck.boot,R=1000)
 #hist(ckb$t)
```

`comorbidity` *Convert base rates of two diagnoses and their comorbidity into phi, Yule, and tetrachorics*

Description

In medicine and clinical psychology, diagnoses tend to be categorical (someone is depressed or not, someone has an anxiety disorder or not). Cooccurrence of both of these symptoms is called comorbidity. Diagnostic categories vary in their degree of comorbidity with other diagnostic categories. From the point of view of correlation, comorbidity is just a name applied to one cell in a four fold table. It is thus possible to analyze comorbidity rates by considering the probability of the separate diagnoses and the probability of the joint diagnosis. This gives the two by two table needed for a phi, Yule, or tetrachoric correlation.

Usage

```
comorbidity(d1, d2, com, labels = NULL)
```

Arguments

`d1`	Proportion of diagnostic category 1
`d2`	Proportion of diganostic category 2
`com`	Proportion of comorbidity (diagnostic category 1 and 2)
`labels`	Names of categories 1 and 2

Value

twobytwo	The two by two table implied by the input
phi	Phi coefficient of the two by two table
Yule	Yule coefficient of the two by two table
tetra	Tetrachoric coefficient of the two by two table

Author(s)

William Revelle

See Also

phi, phi2tetra ,Yule, Yule.inv Yule2phi, tetrachoric and polychoric, as well as AUC for graphical displays

Examples

```
comorbidity(.2,.15,.1,c("Anxiety","Depression"))
```

---

congruence *Matrix and profile congruences and distances*

Description

The congruence coefficient two matrices is just the cross product of their respective values divided by the square root of their sums of squares. If the columns are zero centered, this is just the correlation. If the columns are centered around the scale neutral point, this is Cohen's profile correlation. A set of distances (city block, euclidean, Minkowski) may be found by the distance function.

Usage

```
congruence(x,y=NULL)
cohen.profile(x,y=NULL ,M=NULL)
distance(x,y=NULL,r=2)
```

Arguments

x	A matrix of factor loadings or a list of matrices of factor loadings
y	A second matrix of factor loadings (if x is a list, then y may be empty)
M	The midpoint of the items which should be used for reflection. NULL if to be found from the data.
r	The exponent for the generalized distance. (r=1 is city block, r=2 is euclidian, r=100 or larger emphasize the largest distance)

Details

Congruences are the cosines of pairs of vectors defined by a matrix and based at the origin. Thus, for values that differ only by a scaler the congruence will be 1.

For two matrices, F1 and F2, the measure of congruence, phi, is

$$\phi = \frac{\sum F_1 F_2}{\sqrt{\sum(F_1^2)\sum(F_2^2)}}.$$

It is an interesting exercise to compare congruences with the correlations of the respective values. Congruences are based upon the raw cross products, while correlations are based upon centered cross products. That is, correlations of items are cosines of the vectors based at the mean loading for each column.

$$\phi = \frac{\sum(F_1 - a)(F_2 - b)}{\sqrt{\sum((F_1 - a)^2)\sum((F_2 - b)^2)}}.$$

.

For congruence coefficients, a = b= 0. For correlations a=mean F1, b= mean F2.

Input may either be one or two matrices.

For congruences of factor or component loading matrices, use `factor.congruence`.

Normally, all factor loading matrices should be complete (have no missing loadings). In the case where some loadings are missing, if the use option is specified, then variables with missing loadings are dropped.

If the data are zero centered, this is the correlation, if the data are centered around the scale midpoint (M), this is Cohen's Similarity coefficient. See examples. If M is not specified, it is found as the midpoint of the items in x and y.

`cohen.profile` applies the `congruence` function to data centered around M. M may be specified, or found from the data. The last example is taken from Cohen (1969).

`distance` finds the generalized distance as a function of r. City block (r=1), Euclidean (r=2) or weighted towards maximimum (r >2).

Value

A matrix of congruences or distances.

Author(s)

William Revelle

References

Burt, Cyril (1948) Methods of factor-analysis with and without successive approximation. British Journal of Educational Psychology, 7 (2) 172-195.

Burt, Cyril (1948) The factorial study of temperamental traits. British Journal of Statistical Psychology, 1(3) 178-203.

Cohen, Jacob (1969), rc: A profile similarity coefficient invariant over variable reflection. Psychological Bulletin, 71 (4) 281-284.

See Also

factor.congruence.

Examples

```
#cohen's example
# a and b have reversed one item around the midpoint
co <-  data.frame(ira=c(2,6,5,6,4),
       jim=c(1,3,5,4,4),
      a=c(5,6,5,6,4),b=c(6,3,5,4,4))

lowerMat(congruence(co-3.5)) # congruence around the midpoint is insensitive to reflection
lowerCor(co)   #the correlation is not
lowerMat(congruence(scale(co,scale=FALSE))) #zero centered congruence is  r
cohen.profile(co)
```

---

cor.smooth — *Smooth a non-positive definite correlation matrix to make it positive definite*

Description

Factor analysis requires positive definite correlation matrices. Unfortunately, with pairwise deletion of missing data or if using tetrachoric or polychoric correlations, not all correlation matrices are positive definite. cor.smooth does a eigenvector (principal components) smoothing. Negative eigen values are replaced with 100 * eig.tol, the matrix is reproduced and forced to a correlation matrix using cov2cor.

Usage

```
cor.smooth(x,eig.tol=10^-12)
cor.smoother(x,cut=.01)
```

Arguments

x	A correlation matrix or a raw data matrix.
eig.tol	the minimum acceptable eigenvalue
.	
cut	Report all abs(residuals) > cut

Details

The smoothing is done by eigen value decomposition. eigen values < eig.tol are changed to 100 * eig.tol. The positive eigen values are rescaled to sum to the number of items. The matrix is recomputed (eigen.vectors %*% diag(eigen.values) %*% t(eigen.vectors) and forced to a correlation matrix using cov2cor. (See Bock, Gibbons and Muraki, 1988 and Wothke, 1993).

This does not implement the Knol and ten Berge (1989) solution, nor do nearcor and posdefify in sfmsmisc, not does nearPD in Matrix. As Martin Maechler puts it in the posdedify function, "there are more sophisticated algorithms to solve this and related problems."

cor.smoother examines all of nvar minors of rank nvar-1 by systematically dropping one variable at a time and finding the eigen value decomposition. It reports those variables, which, when dropped, produce a positive definite matrix. It also reports the number of negative eigenvalues when each variable is dropped. Finally, it compares the original correlation matrix to the smoothed correlation matrix and reports those items with absolute deviations great than cut. These are all hints as to what might be wrong with a correlation matrix.

Value

The smoothed matrix with a warning reporting that smoothing was necessary (if smoothing was in fact necessary).

Author(s)

William Revelle

References

R. Darrell Bock, Robert Gibbons and Eiji Muraki (1988) Full-Information Item Factor Analysis. Applied Psychological Measurement, 12 (3), 261-280.

Werner Wothke (1993), Nonpositive definite matrices in structural modeling. In Kenneth A. Bollen and J. Scott Long (Editors),Testing structural equation models, Sage Publications, Newbury Park.

D.L. Knol and JMF ten Berge (1989) Least squares approximation of an improper correlation matrix by a proper one. Psychometrika, 54, 53-61.

See Also

`tetrachoric`, `polychoric`, `fa` and `irt.fa`, and the `burt` data set.

See also nearcor and posdefify in the sfsmisc package and nearPD in the Matrix package.

Examples

```
if(require(psychTools)) {

burt <- psychTools::burt
bs <- cor.smooth(psychTools::burt)  #burt data set is not positive definite
plot(burt[lower.tri(burt)],bs[lower.tri(bs)],ylab="smoothed values",xlab="original values")
abline(0,1,lty="dashed")

round(burt - bs,3)
fa(burt,2) #this throws a warning that the matrix yields an improper solution
#Smoothing first throws a warning that the matrix was improper,
#but produces a better solution
fa(cor.smooth(burt),2)
}

#this next example is a correlation matrix from DeLeuw used as an example
```

```
#in Knol and ten Berge.
#the example is also used in the nearcor documentation
 cat("pr is the example matrix used in Knol DL, ten Berge (1989)\n")
 pr <- matrix(c(1,     0.477, 0.644, 0.478, 0.651, 0.826,
  0.477, 1,     0.516, 0.233, 0.682, 0.75,
  0.644, 0.516, 1,     0.599, 0.581, 0.742,
  0.478, 0.233, 0.599, 1,     0.741, 0.8,
  0.651, 0.682, 0.581, 0.741, 1,     0.798,
  0.826, 0.75,  0.742, 0.8,   0.798, 1),
       nrow = 6, ncol = 6)

sm <- cor.smooth(pr)
resid <- pr - sm
# several goodness of fit tests
# from Knol and ten Berge
tr(resid %*% t(resid)) /2

# from nearPD
sum(resid^2)/2
```

`cor.wt` *The sample size weighted correlation may be used in correlating aggregated data*

Description

If using aggregated data, the correlation of the means does not reflect the sample size used for each mean. cov.wt in RCore does this and returns a covariance matrix or the correlation matrix. The cor.wt function weights by sample size or by standard errors and by default return correlations.

Usage

```
cor.wt(data,vars=NULL, w=NULL,sds=NULL, cor=TRUE)
```

Arguments

`data`	A matrix or data frame
`vars`	Variables to analyze
`w`	A set of weights (e.g., the sample sizes)
`sds`	Standard deviations of the samples (used if weighting by standard errors)
`cor`	Report correlations (the default) or covariances

Details

A weighted correlation is just $r_{ij} = \frac{\sum(wt_k(x_{ik}-x_{jk})}{\sqrt{wt_{ik}\sum(x_{ik}^2)wt_j k\sum(x_{jk}^2)}}$ where x_{ik} is a deviation from the weighted mean.

The weighted correlation is appropriate for correlating aggregated data, where individual data points might reflect the means of a number of observations. In this case, each point is weighted by its sample size (or alternatively, by the standard error). If the weights are all equal, the correlation is just a normal Pearson correlation.

Used when finding correlations of group means found using `statsBy`.

Value

`cor`	The weighted correlation
`xwt`	The data as weighted deviations from the weighted mean
`wt`	The weights used (calculated from the sample sizes).
`mean`	The weighted means
`xc`	Unweighted, centered deviation scores from the weighted mean
`xs`	Deviation scores weighted by the standard error of each sample mean

Note

A generalization of `cov.wt` in core R

Author(s)

William Revelle

See Also

See Also as `cov.wt`, `statsBy`

Examples

```
means.by.age <- statsBy(sat.act,"age")
wt.cors <- cor.wt(means.by.age)
lowerMat(wt.cors$r)  #show the weighted correlations
unwt <- lowerCor(means.by.age$mean)
mixed <- lowerUpper(unwt,wt.cors$r)  #combine both results
cor.plot(mixed,TRUE,main="weighted versus unweighted correlations")
diff <- lowerUpper(unwt,wt.cors$r,TRUE)
cor.plot(diff,TRUE,main="differences of weighted versus unweighted correlations")
```

cor2dist — *Convert correlations to distances (necessary to do multidimensional scaling of correlation data)*

Description

A minor helper function to convert correlations (ranging from -1 to 1) to distances (ranging from 0 to 2). $d = \sqrt{(2(1-r))}$.

Usage

```
cor2dist(x)
```

Arguments

x	If square, then assumed to be a correlation matrix, otherwise the correlations are found first.

Value

dist: a square matrix of distances.

Note

For an example of doing multidimensional scaling on data that are normally factored, see Revelle (in prep)

Author(s)

William Revelle

References

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at https://personality-project.org/r/book/

corCi — *Bootstrapped and normal confidence intervals for raw and composite correlations*

Description

Although normal theory provides confidence intervals for correlations, this is particularly problematic with Synthetic Aperture Personality Assessment (SAPA) data where the individual items are Massively Missing at Random. Bootstrapped confidence intervals are found for Pearson, Spearman, Kendall, tetrachoric, or polychoric correlations and for scales made from those correlations. If given a correlation matrix and sample size(s), normal theory confidence intervals are provided.

Usage

```
corCi(x, keys = NULL, n.iter = 100,  p = 0.05,overlap = FALSE,
 poly = FALSE, method = "pearson", plot=TRUE,minlength=5,n=NULL,...)

cor.ci(x, keys = NULL, n.iter = 100,  p = 0.05,overlap = FALSE,
 poly = FALSE, method = "pearson", plot=TRUE,minlength=5,n=NULL,...)
```

Arguments

x	The raw data, or a correlation matrix if not doing bootstrapping
keys	If NULL, then the confidence intervals of the raw correlations are found. Otherwise, composite scales are formed from the keys applied to the correlation matrix (in a logic similar to cluster.cor but without the bells and whistles) and the confidence of those composite scales intercorrelations.
n.iter	The number of iterations to bootstrap over. This will be very slow if using tetrachoric/or polychoric correlations.
p	The upper and lower confidence region will include 1-p of the distribution.
overlap	If true, the correlation between overlapping scales is corrected for item overlap.
poly	if FALSE, then find the correlations using the method specified (defaults to Pearson). If TRUE, the polychoric correlations will be found (slowly). Because the polychoric function uses multicores (if available), and corCi does as well, the number of cores used is options("mc.cores")^2.
method	"pearson","spearman", "kendall"
plot	Show the correlation plot with correlations scaled by the probability values. To show the matrix in terms of the confidence intervals, use cor.plot.upperLowerCi.
minlength	What is the minlength to use in abbreviations of the cis? Defaults to 5
n	If finding confidence intervals from a correlation matrix, specify the n
...	Other parameters for axis (e.g., cex.axis to change the font size, srt to rotate the numbers in the plot)

Details

If given a correlation matrix, then confidence intervals are found based upon the sample sizes using the conventional r2z fisher transformation (fisherz and the normal distribution.

If given raw data, correlations are found. If keys are specified (the normal case), then composite scales based upon the correlations are found and reported. This is the same procedure as done using cluster.cor or scoreItems.

Then (with raw data) the data are recreated n.iter times by sampling subjects (rows) with replacement and the correlations (and composite scales) are found again (and again and again). Mean and standard deviations of these values are calculated based upon the Fisher Z transform of the correlations. Summary statistics include the original correlations and their confidence intervals. For those who want the complete set of replications, those are available as an object in the resulting output.

Although particularly useful for SAPA (https://www.sapa-project.org/) type data where we have lots of missing data, this will work for any normal data set as well.

Although the correlations are shown automatically as a cor.plot, it is possible to show the upper and lower confidence intervals by using cor.plot.upperLowerCi. This will also return, invisibly, a matrix for printing with the lower and upper bounds of the correlations shown below and above the diagonal (see the first example).

Value

rho	The original (composite) correlation matrix.
means	Mean (of Fisher transformed) correlation retransformed back to the r units
sds	Standard deviation of Fisher transformed correlations
ci	Mean +/- alpha/2 of the z scores as well as the alpha/2 and 1-alpha/2 quantiles. These are labeled as lower.emp(ircal), lower.norm(al), upper.norm and upper.emp.
replicates	The observed replication values so one can do one's own estimates

Author(s)

William Revelle

References

For SAPA type data, see Revelle, W., Wilt, J., and Rosenthal, A. (2010) Personality and Cognition: The Personality-Cognition Link. In Gruszka, A. and Matthews, G. and Szymura, B. (Eds.) Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control, Springer.

See Also

make.keys, cluster.cor, and scoreItems for forming synthetic correlation matrices from composites of item correlations. See scoreOverlap for correcting for item overlap in scales. See also corr.test for standard significance testing of correlation matrices. See also lowerCor for finding and printing correlation matrices, as well as lowerMat for displaying them. Also see cor.plot.upperLowerCi for displaying the confidence intervals graphically.

Examples

```
#find confidence intervals of a correlation matrix with specified sample size
ci <- corCi(Thurstone[1:6,1:6],n=213)
ci  #show them
R <- cor.plot.upperLowerCi(ci)  #show them graphically
R #show them as a matrix


#confidence intervals by bootstrapping requires raw data
corCi(bfi[1:200,1:10])  # just the first 10 variables
#The keys have overlapping scales
keys <- list(agree=c("-A1","A2","A3","A4","A5"), conscientious= c("C1",
  "C2","C3","-C4","-C5"),extraversion=c("-E1","-E2","E3","E4","E5"), neuroticism=
```

```
  c("N1", "N2", "N3","N4","N5"), openness = c("O1","-O2","O3","O4","-O5"),
 alpha=c("-A1","A2","A3","A4","A5","C1","C2","C3","-C4","-C5","N1","N2","N3","N4","N5"),
beta = c("-E1","-E2","E3","E4","E5","O1","-O2","O3","O4","-O5") )


#do not correct for item overlap
rci <-  corCi(bfi[1:200,],keys,n.iter=10,main="correlation with overlapping scales")
#also shows the graphic -note the overlap
#correct for overlap
rci <-  cor.ci(bfi[1:200,],keys,overlap=TRUE, n.iter=10,main="Correct for overlap")
#show the confidence intervals
ci <- cor.plot.upperLowerCi(rci)  #to show the upper and lower confidence intervals
ci   #print the confidence intervals in matrix form
```

`corFiml` *Find a Full Information Maximum Likelihood (FIML) correlation or covariance matrix from a data matrix with missing data*

Description

Makes use of functions adapted from the lavaan package to find FIML covariance/correlation matrices. FIML can be much slower than the normal pairwise deletion option of cor, but provides slightly more precise estimates.

Usage

```
corFiml(x, covar = FALSE,show=FALSE)
```

Arguments

`x`	A data.frame or data matrix
`covar`	By default, just return the correlation matrix. If covar is TRUE, return a list containing the covariance matrix and the ML fit function.
`show`	If show=TRUE, then just show the patterns of missingness, but don't do the FIML. Useful for understanding the process of fiml.

Details

In the presence of missing data, Full Information Maximum Likelihood (FIML) is an alternative to simply using the pairwise correlations. The implementation in the lavaan package for structural equation modeling has been adapted for the simpler case of just finding the correlations or covariances.

The pairwise solution for any pair of variables is insensitive to other variables included in the matrix. On the other hand, the ML solution depends upon the entire set of items being correlated. This will lead to slightly different solutions for different subsets of variables.

The basic FIML algorithm is to find the pairwise ML solution for covariances and means for every pattern of missingness and then to weight the solution by the size of every unique pattern of missingness.

Value

cor	The correlation matrix found using FIML
cov	The covariance matrix found using FIML
fx	The ML fit function

Note

The functions used in lavaan are not exported and so have been copied (and simplified) to the psych package.

Author(s)

Wiliam Revelle

See Also

To use the resulting correlations, see fa. To see the pairwise pattern of missingness, see count.pairwise.

Examples

```
rML <- corFiml(bfi[20:27])
rpw <- cor(bfi[20:27],use="pairwise")
round(rML - rpw,3)
mp <- corFiml(bfi[20:27],show=TRUE)
mp
```

---

corPlot *Create an image plot for a correlation or factor matrix*

Description

Correlation matrices may be shown graphically by using the image function to emphasize structure. This is a particularly useful tool for showing the structure of correlation matrices with a clear structure. Partially meant for the pedagogical value of the graphic for teaching or discussing factor analysis and other multivariate techniques. The sort option uses iclust to sort the matrix before plotting.

Usage

```
corPlot(r,numbers=TRUE,colors=TRUE,n=51,main=NULL,zlim=c(-1,1),
 show.legend=TRUE, labels=NULL,n.legend=10,keep.par=TRUE,select=NULL, pval=NULL,
 digits=2, trailing=TRUE, cuts=c(.001,.01),scale=TRUE,cex,MAR,upper=TRUE,diag=TRUE,
 symmetric=TRUE,stars=FALSE, adjust="holm",xaxis=1, xlas=0, ylas=2,ysrt=0,xsrt=0,
   gr=NULL, alpha=.75,  min.length=NULL,sort=FALSE,n.obs=NULL, ...)

corPlotUpperLowerCi(R,numbers=TRUE,cuts=c(.001,.01,.05),select=NULL,
    main="Upper and lower confidence intervals of correlations",adjust=FALSE,...)
```

```
 cor.plot(r,numbers=TRUE,colors=TRUE,n=51,main=NULL,zlim=c(-1,1),
show.legend=TRUE, labels=NULL,n.legend=10,keep.par=TRUE,select=NULL, pval=NULL,
 digits=2, trailing=TRUE, cuts=c(.001,.01),scale=TRUE,cex,MAR,upper=TRUE,diag=TRUE,
 symmetric=TRUE,stars=FALSE, adjust="holm",xaxis=1, xlas=0, ylas=2,ysrt=0,xsrt=0,
   gr=NULL, alpha=.75, min.length=NULL, sort=FALSE, n.obs=NULL,...)    #deprecated

cor.plot.upperLowerCi(R,numbers=TRUE,cuts=c(.001,.01,.05),select=NULL,
    main="Upper and lower confidence intervals of correlations",adjust=FALSE,...)
      #deprecated
```

Arguments

r	A correlation matrix or the output of fa, principal or omega, or a raw data matrix.
R	The object returned from cor.ci
numbers	Display the numeric value of the correlations. (As of September, 2019) Defaults to TRUE.
colors	Defaults to TRUE and colors use colors from the colorRampPalette from red through white to blue, but colors=FALSE will use a grey scale
n	The number of levels of shading to use. Defaults to 51
main	A title. Defaults to "correlation plot"
zlim	The range of values to color – defaults to -1 to 1. If specified as NULL, then defaults to min and max observed correlation.
show.legend	A legend (key) to the colors is shown on the right hand side
labels	if NULL, use column and row names, otherwise use labels
n.legend	How many categories should be labelled in the legend?
sort	If true, then sort the variables using the iclust algorithm
keep.par	restore the graphic parameters when exiting
pval	scale the numbers by their pvals, categorizing them based upon the values of cuts
digits	Round off to digits. Defaults to 2.
trailing	Show trailing zeros.
cuts	Scale the numbers by the categories defined by pval < cuts
scale	Should the size of the numbers be scaled by the significance level?
select	Select the subset of variables to plot
cex	Character size. Should be reduced a bit for large numbers of variables.
MAR	Allows for adjustment .of the margins if using really long labels or big fonts
upper	Should the upper off diagonal matrix be drawn, or left blank?
diag	Should we show the diagonal?
symmetric	By default, if given a non-symmetric matrix, we find the correlations using pair.wise complete and then show them. If wanting to display a non-symmetric matrix, then specify that symmetric is FALSE

stars	For those people who like to show the 'significance' of correlations by using magic astricks, set stars=TRUE
n.obs	If you want to show "stars" for symmetric input matrices (i.e. correlations), specify the number of observations
adjust	If showing significance, should we adjust for multiple tests? The default is to show zero order probabilities below the diagonal and adjust these using the 'holm' correction above the diagonal. Use adjust = "none" if no adjustment is desired. adjust is also used in corPlotUpperLowerCI to show the nominal alpha confidence intervals (adjust =FALSE) or the Bonferonni adjusted confidence intervals (adjust=TRUE).
xlas	Orientation of the x axis labels (1 = horizontal, 0, parallel to axis, 2 perpendicular to axis)
ylas	Orientation of the y axis labels (1 = horizontal, 0, parallel to axis, 2 perpendicular to axis)
ysrt	Rotation of y labels in degrees
xsrt	Rotation of x labels in degrees
xaxis	By default, draw this below the figure. If xaxis=3, then it wil be drawn above the figure
gr	A color gradient: e.g., gr <- colorRampPalette(c("#B52127", "white", "#2171B5")) will produce slightly more pleasing (to some) colors. See next to last example.
alpha	The degree of transparency (0 = completely, 1= not). Default value of .75 makes somewhat moreor pleasing plots when using numbers.
min.length	If not NULL, then the maximum number of characters to use in row/column labels
...	Other parameters for axis (e.g., cex.axis to change the font size, srt to rotate the numbers in the plot)

Details

When summarizing the correlations of large data bases or when teaching about factor analysis or cluster analysis, it is useful to graphically display the structure of correlation matrices. This is a simple graphical display using the image function.

The difference between mat.plot with a regular image plot is that the primary diagonal goes from the top left to the lower right. zlim defines how to treat the range of possible values. -1 to 1 and the color choice is more reasonable. Setting it as c(0,1) will lead to negative correlations treated as zero. This is advantageous when showing general factor structures, because it makes the 0 white.

There is an interesting case when plotting correlations corrected for attenuation. Some of these might exceed 1. In this case, either set zlim = NULL (to use the observed maximum and minimum values) or all values above 1 will be given a slightly darker shade than 1, but do not differ.

The default shows a legend for the color coding on the right hand side of the figure.

Inspired, in part, by a paper by S. Dray (2008) on the number of components problem.

Modified following suggestions by David Condon and Josh Wilt to use a more meaningful color choice ranging from dark red (-1) through white (0) to dark blue (1). Further modified to allow for color choices using the gr option (suggested by Lorien Elleman). Further modified to include the

numerical value of the correlation. (Inspired by the corrplot package). These values may be scaled according the the probability values found in cor.ci or corTest.

Unless specified, the font size is dynamically scaled to have a cex = 10/max(nrow(r),ncol(r). This can produce fairly small fonts for large problems. The font size of the labels may be adjusted using cex.axis which defaults to one.

By default cor.ci calls corPlotUpperLowerCi and scales the correlations based upon "significance" values. The correlations plotted are the upper and lower confidence boundaries. To show the correlations themselves, call corPlot directly.

If using the output of corTest, the upper off diagonal will be scaled by the corrected probability, the lower off diagonal the scaling is the uncorrected probabilities.

If given raw data or correlation matrix, corPlotUpperLowerCi will automatically call corTest or cor.ci.

If using the output of corTest or cor.ci as input to corPlotUpperLowerCi, the upper off diagonal will be the upper bounds and the lower off diagonal the lower bounds of the confidence intervals. If adjust=TRUE, these will use the Holm or Bonferroni adjusted values (depending upon corTest).

To compare the elements of two correlation matrices, corPlot the results from lowerUpper.

To do multiple corPlot on the same plot, specify that show.legend=FALSE and keep.par=FALSE. See the last examples.

Care should be taken when selecting rows and columns from a non-symmetric matrix (e.g., the corrected correlations from scoreItems or scoreOverlap).

To show a factor loading matrix (or any non-symmetric matrix), set symmetric=FALSE. Otherwise the input will be treated as raw data and correlations will be found.

The sort option will sort the matrix using the output from iclust. To sort the matrix use another order, use mat.sort first. To find correlations other than Pearson, plot the output from e.g., mixed.cor.

Author(s)

William Revelle

References

Dray, Stephane (2008) On the number of principal components: A test of dimensionality based on measurements of similarity between matrices. Computational Statistics & Data Analysis. 52, 4, 2228-2237.

See Also

fa, mat.sort, cor.ci, corTest lowerUpper.

Examples

```
corPlot(Thurstone,main="9 cognitive variables from Thurstone")
#just blue implies positive manifold
#select just some variables to plot
corPlot(Thurstone, zlim=c(0,1),main="9 cognitive variables from Thurstone",select=c(1:3,7:9))
#now show a non-symmetric plot
```

```
corPlot(Thurstone[4:9,1:3], zlim=c(0,1),main="9 cognitive variables
 from Thurstone",numbers=TRUE,symmetric=FALSE)

#Two ways of including stars to show significance
#From the raw data
corPlot(sat.act,numbers=TRUE,stars=TRUE)
#from a correlation matrix with pvals
cp <- corTest(sat.act)  #find the correlations and pvals
r<- cp$r
p <- cp$p
corPlot(r,numbers=TRUE,diag=FALSE,stars=TRUE, pval = p,main="Correlation plot
 with Holm corrected 'significance'")

#now red means less than .5
corPlot(mat.sort(Thurstone),TRUE,zlim=c(0,1),
       main="9 cognitive variables from Thurstone (sorted by factor loading) ")
simp <- sim.circ(24)
corPlot(cor(simp),main="24 variables in a circumplex")

#scale by raw and adjusted probabilities
rs <- corTest(sat.act[1:200,] ) #find the probabilities of the correlations
corPlot(r=rs$r,numbers=TRUE,pval=rs$p,main="Correlations scaled by probability values")
 #Show the upper and lower confidence intervals
corPlotUpperLowerCi(R=rs,numbers=TRUE)

#now do this again, but with lighter colors
gr <- colorRampPalette(c("#B52127", "white", "#2171B5"))
corPlot(r=rs$r,numbers=TRUE,pval=rs$p,main="Correlations scaled by probability values",gr=gr)

corPlotUpperLowerCi(R=rs,numbers=TRUE,gr=gr)



if(require(psychTools)) {
#do multiple plots
#Also show the xaxis option
op <- par(mfrow=c(2,2))
corPlot(psychTools::ability,show.legend=FALSE,keep.par=FALSE,upper=FALSE)
f4 <- fa(psychTools::ability,4)
corPlot(f4,show.legend=FALSE,keep.par=FALSE,numbers=TRUE,xlas=3)
om <- omega(psychTools::ability,4)
corPlot(om,show.legend=FALSE,keep.par=FALSE,numbers=TRUE,xaxis=3)
par(op)


corPlotUpperLowerCi(rs,adjust=TRUE,main="Holm adjusted confidence intervals",gr=gr)
}
```

`correct.cor` *Find dis-attenuated correlations given correlations and reliabilities*

Description

Given a raw correlation matrix and a vector of reliabilities, report the disattenuated correlations above the diagonal.

Usage

```
correct.cor(x, y)
```

Arguments

x	A raw correlation matrix
y	Vector of reliabilities

Details

Disattenuated correlations may be thought of as correlations between the latent variables measured by a set of observed variables. That is, what would the correlation be between two (unreliable) variables be if both variables were measured perfectly reliably.

This function is mainly used if importing correlations and reliabilities from somewhere else. If the raw data are available, use `score.items`, or `cluster.loadings` or `cluster.cor`.

Examples of the output of this function are seen in `cluster.loadings` and `cluster.cor`

Value

Raw correlations below the diagonal, reliabilities on the diagonal, disattenuated above the diagonal.

Author(s)

Maintainer: William Revelle <revelle@northwestern.edu>

References

Revelle, W. (in preparation) An Introduction to Psychometric Theory with applications in R. Springer. at https://personality-project.org/r/book/

See Also

`cluster.loadings` and `cluster.cor`

Examples

```
# attitude from the datasets package
#example 1 is a rather clunky way of doing things

a1 <- attitude[,c(1:3)]
a2 <- attitude[,c(4:7)]
x1 <- rowSums(a1)  #find the sum of the first 3 attitudes
x2 <- rowSums(a2)   #find the sum of the last 4 attitudes
alpha1 <- alpha(a1)
```

```
alpha2 <- alpha(a2)
x <- matrix(c(x1,x2),ncol=2)
x.cor <- cor(x)
alpha <- c(alpha1$total$raw_alpha,alpha2$total$raw_alpha)
round(correct.cor(x.cor,alpha),2)
#
#much better - although uses standardized alpha
clusters <- matrix(c(rep(1,3),rep(0,7),rep(1,4)),ncol=2)
cluster.loadings(clusters,cor(attitude))
# or
clusters <- matrix(c(rep(1,3),rep(0,7),rep(1,4)),ncol=2)
cluster.cor(clusters,cor(attitude))
#
#best
keys <- make.keys(attitude,list(first=1:3,second=4:7))
scores <- scoreItems(keys,attitude)
scores$corrected

#However, to do the more general case of correcting correlations for reliabilty
#corrected <- cor2cov(x.cor,1/alpha)
#diag(corrected) <- 1
```

---

cortest *Chi square tests of whether a single matrix is an identity matrix, or a pair of matrices are equal.*

Description

Steiger (1980) pointed out that the sum of the squared elements of a correlation matrix, or the Fisher z score equivalents, is distributed as chi square under the null hypothesis that the values are zero (i.e., elements of the identity matrix). This is particularly useful for examining whether correlations in a single matrix differ from zero or for comparing two matrices. Jennrich (1970) also examined tests of differences between matrices.

Usage

```
cortest(R1,R2=NULL,n1=NULL,n2 = NULL, fisher = TRUE,cor=TRUE,method="pearson",
           use ="pairwise") #same as cortest.normal this does the steiger test
cortest.normal(R1, R2 = NULL, n1 = NULL, n2 = NULL, fisher = TRUE)    #the steiger test

cortest.jennrich(R1,R2,n1=NULL, n2=NULL)  #the Jennrich test
cortest.mat(R1,R2=NULL,n1=NULL,n2 = NULL) #an alternative test
```

Arguments

R1	A correlation matrix. (If R1 is not rectangular, and cor=TRUE, the correlations are found).
R2	A correlation matrix. If R2 is not rectangular, and cor=TRUE, the correlations are found. If R2 is NULL, then the test is just whether R1 is an identity matrix.
n1	Sample size of R1
n2	Sample size of R2
fisher	Fisher z transform the correlations?
cor	By default, if the input matrices are not symmetric, they are converted to correlation matrices. That is, they are treated as if they were the raw data. If cor=FALSE, then the input matrices are taken to be correlation matrices.
method	Which type of correlation to find ("pearson", "spearman","kendall")
use	How to handle missing data (defaults to pairwise)

Details

There are several ways to test if a matrix is the identity matrix. The most well known is the chi square test of Bartlett (1951) and Box (1949). cortest.bartlett. A very straightforward test, discussed by Steiger (1980) is to find the sum of the squared correlations or the sum of the squared Fisher transformed correlations. Under the null hypothesis that all the correlations are equal to 0, this sum is distributed as chi square. This is implemented in cortest and cortest.normal and will give different values than cortest.bartlett.

For comparing two matrices, the null hypothesis is that they are equal and thus the differences are 0. The test, from Steiger, is based on the sum of squared (fisher z transformed or not) residuals. The sum of these squared residuals * the sample size (-3) should be distributed as chi square.

Yet another test, is the Jennrich(1970) test of the equality of two matrices. This compares the differences between two matrices to the averages of two matrices using a chi square test. This is implemented in cortest.jennrich.

Yet another option cortest.mat is to compare the two matrices using an approach analogous to that used in evaluating the adequacy of a factor model. In factor analysis, the maximum likelihood fit statistic is
$f = log(trace((FF' + U2)^{-1}R) - log(|(FF' + U2)^{-1}R|) - n.items.$

This in turn is converted to a chi square

$\chi^2 = (n.obs - 1 - (2 * p + 5)/6 - (2 * factors)/3)) * f$ (see fa.)

That is, the model (M = FF' + U2) is compared to the original correlation matrix (R) by a function of $M^{-1}R$. By analogy, in the case of two matrices, A and B, cortest.mat finds the chi squares associated with $A^{-1}B$ and AB^{-1}. The sum of these two χ^2 will also be a χ^2 but with twice the degrees of freedom.

Value

chi2	The chi square statistic
df	Degrees of freedom for the Chi Square
prob	The probability of observing the Chi Square under the null hypothesis.

SRMR	The square root of the mean squared residual.
RMSEA	The Root Mean Square Error of Approximation is derived from chi square and asymptotically tends towards the SRMR.

Note

For the case of two matrices versus the difference of the correlations the results will differ if the fisher r to z transform is used.

Note

Both the cortest.jennrich and cortest.normal are probably overly stringent. The ChiSquare values for pairs of random samples from the same population are larger than would be expected. This is a good test for rejecting the null of no differences.

Author(s)

William Revelle

References

Steiger, James H. (1980) Testing pattern hypotheses on correlation matrices: alternative statistics and some empirical results. Multivariate Behavioral Research, 15, 335-352.

Jennrich, Robert I. (1970) An Asymptotic χ^2 Test for the Equality of Two Correlation Matrices. Journal of the American Statistical Association, 65, 904-912.

Maydeu-Olivares, Alberto (2017) Assessing the Size of Model Misfit in Structural Equation Models, Psychometrika, 82, 533–558.

See Also

cortest.bartlett corr.test

Examples

```
set.seed(42)
x <- matrix(rnorm(1000),ncol=10)
cortest.normal(x)  #just test if this matrix is an identity
#now create two correlation matrices that should be equal
x <- sim.congeneric(loads =c(.9,.8,.7,.6,.5),N=1000,short=FALSE)
y <- sim.congeneric(loads =c(.9,.8,.7,.6,.5),N=1000,short=FALSE)

cortest(x$r,y$r,n1=1000,n2=1000) #The Steiger test
cortest.jennrich(x$r,y$r,n1=100,n2=1000) # The Jennrich test

cortest.mat(x$r,y$r,n1=1000,n2=1000)   #twice the degrees of freedom as the Jennrich

#create a new matrix that is different
z <- sim.congeneric(loads=c(.8,.8,.7,.7, .6), N= 1000, short=FALSE)
cortest(x$r,z$r,n1=1000)  #these should be different
```

```
#compare the results for forming one matrix of differences versus
# testing the two matrices.
dif=x$r - z$r
cortest(dif,n1=1000)  #versus

cortest(dif,n1=1000,fisher=FALSE)
cortest(x$r,z$r,n1=1000 , fisher=FALSE)  #these should be the same
```

---

corTest *Find the correlations, sample sizes, and probability values between elements of a matrix or data.frame.*

Description

Although the cor function finds the correlations for a matrix, it does not report probability values. cor.test does, but for only one pair of variables at a time. corr.test uses cor to find the correlations for either complete or pairwise data and reports the sample sizes and probability values as well. For symmetric matrices, raw probabilites are reported below the diagonal and correlations adjusted for multiple comparisons above the diagonal. In the case of different x and ys, the default is to adjust the probabilities for multiple tests. Both corr.test and corr.p return raw and adjusted confidence intervals for each correlation. corInfo returns the Fisher information of the correlation matrix.

Usage

```
corTest(x, y = NULL, use = "pairwise",method="pearson",adjust="holm",
    alpha=.05,ci=TRUE,minlength=5,normal=TRUE)
corr.test(x, y = NULL, use = "pairwise",method="pearson",adjust="holm",
    alpha=.05,ci=TRUE,minlength=5,normal=TRUE)
corr.p(r,n,adjust="holm",alpha=.05,minlength=5,ci=TRUE)
corInfo(x,n=NULL,show=FALSE,fisher=FALSE)
```

Arguments

x	A matrix or dataframe
y	A second matrix or dataframe with the same number of rows as x
use	use="pairwise" is the default value and will do pairwise deletion of cases. use="complete" will select just complete cases.
method	method="pearson" is the default value. The alternatives to be passed to cor are "spearman" and "kendall". These last two are much slower, particularly for big data sets.
adjust	What adjustment for multiple tests should be used? ("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"). See p.adjust for details about why to use "holm" rather than "bonferroni").

alpha	alpha level of confidence intervals
r	A correlation matrix
n	Number of observations if using corr.p. May be either a matrix (as returned from corr.test, or a scaler. Set to n - np if finding the significance of partial correlations. (See below).
ci	By default, confidence intervals are found. However, this leads to a noticable slowdown of speed, particularly for large problems. So, for just the rs, ts and ps, set ci=FALSE
minlength	What is the minimum length for abbreviations. Defaults to 5.
normal	By default, probabilities for method="spearman" and method="kendall" are found by normal theory. If normal=="FALSE", then repetitive calls are made to cor.test. This is much slower, but gives more accurate p values. exact is set to be FALSE which means that exact p values for small samples are not found given the problem of ties.
show	if TRUE, then show the lower correlation matrix for corInfo.
fisher	Adjust correlations to fisher z

Details

corr.test uses the cor function to find the correlations, and then applies a t-test to the individual correlations using the formula

$$t = \frac{r * \sqrt{(n-2)}}{\sqrt{(1-r^2)}}$$

$$se = \sqrt{(\frac{1-r^2}{n-2})}$$

The t and Standard Errors are returned as objects in the result, but are not normally displayed. Confidence intervals are found and printed if using the print(short=FALSE) option. These are found by using the fisher z transform of the correlation and then taking the range r +/- qnorm(alpha/2) * se and the standard error of the z transforms is

$$se = \sqrt{(\frac{1}{n-3})}$$

. These values are then back transformed to be in correlation units. They are returned in the ci object.

Note that in the case of method=="kendall" since these are the normal theory confidence intervals they are slightly too big.

The probability values may be adjusted using the Holm (or other) correction. If the matrix is symmetric (no y data), then the original p values are reported below the diagonal and the adjusted above the diagonal. Otherwise, all probabilities are adjusted (unless adjust="none"). This is made explicit in the output. Confidence intervals are shown for raw and adjusted probabilities in the ci object.

For those who like the conventional use of "magic asterisks" to show (stars) to represent conventional levels of significance, the object stars is returned (but not shown)). See the examples.

corr.p may be applied to the results of partial.r if n is set to n - s (where s is the number of variables partialed out) Fisher, 1924.

corInfo finds the Fisher information of each correlation and then sums across the correlations. Fisher Information is just $n * (1 + r^2)/(1 - r^2)^2)$.

Value

r	The matrix of correlations
n	Number of cases per correlation
t	value of t-test for each correlation
p	two tailed probability of t for each correlation. For symmetric matrices, p values adjusted for multiple tests are reported above the diagonal.
se	standard error of the correlation
ci	the alpha/2 lower and upper values.
ci2	ci but with the adjusted pvalues as well. This was added after tests showed we were breaking some packages that were calling the ci object without bothering to check for its dimensions.
ci.adj	These are the adjusted ((Holm or Bonferroni) confidence intervals. If asking to not adjust, the Holm adjustments for the confidence intervals are shown anyway, but the probability values are not adjusted and the appropriate confidence intervals are shown in the ci object.
stars	For those people who like to show magic asterisks denoting "statistical significance" the stars object flags those correlation values that are unlikely given normal theory. See the last example for how to print these neatly.

Note

For very large matrices (> 200 x 200), there is a noticeable speed improvement if confidence intervals are not found.

That adjusted confidence intervals are shown even when asking for no adjustment might be confusing. If you don't want adjusted intervals, just use the ci object. The adjusted values are given in the ci.adj object.

See Also

cor.test for tests of a single correlation, Hmisc::rcorr for an equivalant function, r.test to test the difference between correlations, and cortest.mat to test for equality of two correlation matrices.

Also see cor.ci for bootstrapped confidence intervals of Pearson, Spearman, Kendall, tetrachoric or polychoric correlations. In addition cor.ci will find bootstrapped estimates of composite scales based upon a set of correlations (ala cluster.cor).

In particular, see p.adjust for a discussion of p values associated with multiple tests.

Other useful functions related to finding and displaying correlations include link{corPlot} to graphically display the correlation matrix, and lowerCor for finding the correlations and then displaying the lower off diagonal using the lowerMat function. lowerUpper to compare two correlation matrices. Also see pairs.panels to show the correlations and scatter plots.

Examples

```
ct  <- corTest(attitude)
#ct <- corr.test(attitude)  #find the correlations and give the probabilities
ct #show the results


cts <- corr.test(attitude[1:3],attitude[4:6]) #reports all values corrected for multiple tests

#corr.test(sat.act[1:3],sat.act[4:6],adjust="none")  #don't adjust the probabilities

#take correlations and show the probabilities as well as the confidence intervals
print(corr.p(cts$r,n=30),short=FALSE)

#don't adjust the probabilities
print(corr.test(sat.act[1:3],sat.act[4:6],adjust="none"),short=FALSE)

#print out the stars object without showing quotes
print(corr.test(attitude)$stars,quote=FALSE)  #note that the adjusted ps are given as well


kendall.r <- corr.test(bfi[1:40,4:6], method="kendall", normal=FALSE)
#compare with
cor.test(x=bfi[1:40,4],y=bfi[1:40,6],method="kendall", exact=FALSE)
print(kendall.r,digits=6)

corInfo(attitude)
```

`cortest.bartlett` *Bartlett's test that a correlation matrix is an identity matrix*

Description

Bartlett (1951) proposed that -ln(det(R)*(N-1 - (2p+5)/6) was distributed as chi square if R were an identity matrix. A useful test that residuals correlations are all zero. Contrast to the Kaiser-Meyer-Olkin test.

Usage

```
cortest.bartlett(R, n = NULL,diag=TRUE)
```

Arguments

R	A correlation matrix. (If R is not square, correlations are found and a warning is issued.
n	Sample size (if not specified, 100 is assumed).
diag	Will replace the diagonal of the matrix with 1s to make it a correlation matrix.

Details

More useful for pedagogical purposes than actual applications. The Bartlett test is asymptotically chi square distributed.

Note that if applied to residuals from factor analysis (fa) or principal components analysis (principal) that the diagonal must be replaced with 1s. This is done automatically if diag=TRUE. (See examples.)

An Alternative way of testing whether a correlation matrix is factorable (i.e., the correlations differ from 0) is the Kaiser-Meyer-Olkin KMO test of factorial adequacy.

Value

chisq	Assymptotically chisquare
p.value	Of chi square
df	The degrees of freedom

Author(s)

William Revelle

References

Bartlett, M. S., (1951), The Effect of Standardization on a chi square Approximation in Factor Analysis, Biometrika, 38, 337-344.

See Also

cortest.mat, cortest.normal, cortest.jennrich

Examples

```
set.seed(42)
x <- matrix(rnorm(1000),ncol=10)
r <- cor(x)
cortest.bartlett(r)      #random data don't differ from an identity matrix
#data(bfi)
cortest.bartlett(bfi[1:200,1:10])    #not an identity matrix
f3 <- fa(Thurstone,3)
f3r <- f3$resid
cortest.bartlett(f3r,n=213,diag=FALSE)  #incorrect

cortest.bartlett(f3r,n=213,diag=TRUE)  #correct (by default)
```

cosinor *Functions for analysis of circadian or diurnal data*

Description

Circadian data are periodic with a phase of 24 hours. These functions find the best fitting phase angle (cosinor), the circular mean, circular correlation with circadian data, and the linear by circular correlation. These functions are useful for demonstrating diurnal variation and simple ways of analyzing the data.

Usage

```
cosinor(angle,x=NULL,code=NULL,data=NULL,hours=TRUE,period=24,
            plot=FALSE,opti=FALSE,na.rm=TRUE)
cosinor.plot(angle,x=NULL,data = NULL, IDloc=NULL, ID=NULL,hours=TRUE, period=24,
 na.rm=TRUE,ylim=NULL,ylab="observed",xlab="Time (double plotted)",
 main="Cosine fit",add=FALSE,multi=FALSE,typ="l",...)
 cosinor.period(angle,x=NULL,code=NULL,data=NULL,hours=TRUE,period=seq(23,26,1),
            plot=FALSE,opti=FALSE,na.rm=TRUE)
circadian.phase(angle,x=NULL,code=NULL,data=NULL,hours=TRUE,period=24,
            plot=FALSE,opti=FALSE,na.rm=TRUE)
circadian.mean(angle,data=NULL, hours=TRUE,na.rm=TRUE)
circadian.sd(angle,data=NULL,hours=TRUE,na.rm=TRUE)
circadian.stats(angle,data=NULL,hours=TRUE,na.rm=TRUE)
circadian.F(angle,group,data=NULL,hours=TRUE,na.rm=TRUE)
circadian.reliability(angle,x=NULL,code=NULL,data = NULL,min=16,
         oddeven=FALSE, hours=TRUE,period=24,plot=FALSE,opti=FALSE,na.rm=TRUE)
circular.mean(angle,na.rm=TRUE) #angles in radians
circadian.cor(angle,data=NULL,hours=TRUE,na.rm=TRUE)  #angles in radians
circular.cor(angle,na.rm=TRUE) #angles in radians
circadian.linear.cor(angle,x=NULL,data=NULL,hours=TRUE)
```

Arguments

angle	A data frame or matrix of observed values with the time of day as the first value (unless specified in code) angle can be specified either as hours or as radians)
code	A subject identification variable
data	A matrix or data frame of data. If specified, then angle and code are variable names (or locations). See examples.
group	If doing comparisons by groups, specify the group code
.	
min	The minimum number of observations per subject to use when finding split half reliabilities.
oddeven	Reliabilities are based upon odd and even items (TRUE) or first vs. last half (FALSE). Default is first and last half.

period	Although time of day is assumed to have a 24 hour rhythm, other rhythms may be fit. If calling cosinor.period, a range may be specified.
IDloc	Which column number is the ID field
ID	What specific subject number should be plotted for one variable
plot	if TRUE, then plot the first variable (angle)
opti	opti=TRUE: iterative optimization (slow) or opti=FALSE: linear fitting (fast)
hours	If TRUE, measures are in 24 hours to the day, otherwise, radians
x	A set of external variables to correlate with the phase angles
na.rm	Should missing data be removed?
ylim	Specify the range of the y axis if the defaults don't work
ylab	The label of the yaxis
xlab	Labels for the x axis
main	the title of the graphic
add	If doing multiple (spagetti) plots, set add = TRUE for the second and beyond plots
multi	If doing multiple (spagetti) plots, set multi=TRUE for the first and subsequent plots
typ	Pass the line type to graphics
...	any other graphic parameters to pass

Details

When data represent angles (such as the hours of peak alertness or peak tension during the day), we need to apply circular statistics rather than the more normal linear statistics (see Jammalamadaka (2006) for a very clear set of examples of circular statistics). The generalization of the mean to circular data is to convert each angle into a vector, average the x and y coordinates, and convert the result back to an angle. A statistic that represents the compactness of the observations is R which is the (normalized) vector length found by adding all of the observations together. This will achieve a maximum value (1) when all the phase angles are the same and a minimum (0) if the phase angles are distributed uniformly around the clock.

The generalization of Pearson correlation to circular statistics is straight forward and is implemented in cor.circular in the circular package and in `circadian.cor` here. Just as the Pearson r is a ratio of covariance to the square root of the product of two variances, so is the circular correlation. The circular covariance of two circular vectors is defined as the average product of the sines of the deviations from the circular mean. The variance is thus the average squared sine of the angular deviations from the circular mean. Circular statistics are used for data that vary over a period (e.g., one day) or over directions (e.g., wind direction or bird flight). Jammalamadaka and Lund (2006) give a very good example of the use of circular statistics in calculating wind speed and direction.

The code from CircStats and circular was adapted to allow for analysis of data from various studies of mood over the day. Those two packages do not seem to handle missing data, nor do they take matrix input, but rather emphasize single vectors.

The cosinor function will either iteratively fit cosines of the angle to the observed data (opti=TRUE) or use the circular by linear regression to estimate the best fitting phase angle. If cos.t <- cos(time)

and sin.t = sin(time) (expressed in hours), then beta.c and beta.s may be found by regression and the phase is $sign(beta.c) * acos(beta.c/\sqrt{(}beta.c^2 + beta.s^2)) * 12/pi$

Simulations (see examples) suggest that with incomplete times, perhaps the optimization procedure yields slightly better fits with the correct phase than does the linear model, but the differences are very small. In the presence of noisey data, these advantages seem to reverse. The recommendation thus seems to be to use the linear model approach (the default). The fit statistic reported for cosinor is the correlation of the data with the model [cos(time - acrophase)].

The `circadian.reliability` function splits the data for each subject into a first and second half (by default, or into odd and even items) and then finds the best fitting phase for each half. These are then correlated (using `circadian.cor`) and this correlation is then adjusted for test length using the conventional Spearman-Brown formula. Returned as object in the output are the statistics for the first and second part, as well as an ANOVA to compare the two halves.

`circular.mean` and `circular.cor` are just `circadian.mean` and `circadian.cor` but with input given in radians rather than hours.

The `circadian.linear.cor` function will correlate a set of circular variables with a set of linear variables. The first (angle) variables are circular, the second (x) set of variables are linear.

The `circadian.F` will compare 2 or more groups in terms of their mean position. This is adapted from the equivalent function in the circular pacakge. This is clearly a more powerful test the more each group is compact around its mean (large values of R).

Value

`phase`	The phase angle that best fits the data (expressed in hours if hours=TRUE).
`fit`	Value of the correlation of the fit. This is just the correlation of the data with the phase adjusted cosine.
`mean.angle`	A vector of mean angles
`n`, `mean`, `sd`	The appropriate circular statistic.
`correl`	A matrix of circular correlations or linear by circular correlations
`R`	R is the vector length (0-1) of the mean vector when finding circadian statistics using `circadian.stats`
`z`, `p`	z is the number of observations x R^2. p is the probability of a z.
`phase.rel`	The reliability of the phase measures. This is the circular correlation between the two halves adjusted using the Spearman-Brown correction.
`fit.rel`	The split half reliability of the fit statistic.
`split.F`	Do the two halves differ from each other? One would hope not.
`group1`, `group2`	The statistics from each half
`splits`	The individual data from each half.

Note

These functions have been adapted from the circular package to allow for ease of use with circadian data, particularly for data sets with missing data and multiple variables of interest.

Author(s)

William Revelle

References

See circular statistics Jammalamadaka, Sreenivasa and Lund, Ulric (2006),The effect of wind direction on ozone levels: a case study, Environmental and Ecological Statistics, 13, 287-298.

See Also

See the circular and CircStats packages.

Examples

```
time <- seq(1:24) #create a 24 hour time
pure <- matrix(time,24,18)
colnames(pure) <- paste0("H",1:18)
pure <- data.frame(time,cos((pure - col(pure))*pi/12)*3 + 3)
    #18 different phases but scaled to 0-6  match mood data
matplot(pure[-1],type="l",main="Pure circadian arousal rhythms",
    xlab="time of day",ylab="Arousal")
op <- par(mfrow=c(2,2))
 cosinor.plot(1,3,pure)
 cosinor.plot(1,5,pure)
 cosinor.plot(1,8,pure)
 cosinor.plot(1,12,pure)

p <- cosinor(pure) #find the acrophases (should match the input)

#now, test finding the acrophases for  different subjects on 3 variables
#They should be the first 3, second 3, etc. acrophases of pure
pp <- matrix(NA,nrow=6*24,ncol=4)
pure <- as.matrix(pure)
pp[,1] <- rep(pure[,1],6)
pp[1:24,2:4] <- pure[1:24,2:4]
pp[25:48,2:4] <- pure[1:24,5:7] *2   #to test different variances
pp[49:72,2:4] <- pure[1:24,8:10] *3
pp[73:96,2:4] <- pure[1:24,11:13]
pp[97:120,2:4] <- pure[1:24,14:16]
pp[121:144,2:4] <- pure[1:24,17:19]
pure.df <- data.frame(ID = rep(1:6,each=24),pp)
colnames(pure.df) <- c("ID","Time",paste0("V",1:3))
cosinor("Time",3:5,"ID",pure.df)

op <- par(mfrow=c(2,2))
 cosinor.plot(2,3,pure.df,IDloc=1,ID="1")
 cosinor.plot(2,3,pure.df,IDloc=1,ID="2")
 cosinor.plot(2,3,pure.df,IDloc=1,ID="3")
 cosinor.plot(2,3,pure.df,IDloc=1,ID="4")

 #now, show those in one panel as spagetti plots
op <- par(mfrow=c(1,1))
```

```
cosinor.plot(2,3,pure.df,IDloc=1,ID="1",multi=TRUE,ylim=c(0,20),ylab="Modeled")
 cosinor.plot(2,3,pure.df,IDloc=1,ID="2",multi=TRUE,add=TRUE,lty="dotdash")
 cosinor.plot(2,3,pure.df,IDloc=1,ID="3",multi=TRUE,add=TRUE,lty="dashed")
 cosinor.plot(2,3,pure.df,IDloc=1,ID="4",multi=TRUE,add=TRUE,lty="dotted")

set.seed(42)   #what else?
noisy <- pure
noisy[,2:19]<- noisy[,2:19] + rnorm(24*18,0,.2)

n <- cosinor(time,noisy) #add a bit of noise

small.pure <- pure[c(8,11,14,17,20,23),]
small.noisy <- noisy[c(8,11,14,17,20,23),]
small.time <- c(8,11,14,17,20,23)

cosinor.plot(1,3,small.pure,multi=TRUE)
cosinor.plot(1,3,small.noisy,multi=TRUE,add=TRUE,lty="dashed")


# sp <- cosinor(small.pure)
# spo <- cosinor(small.pure,opti=TRUE) #iterative fit
# sn <- cosinor(small.noisy) #linear
# sno <- cosinor(small.noisy,opti=TRUE) #iterative
# sum.df <- data.frame(pure=p,noisy = n, small=sp,small.noise = sn,
#         small.opt=spo,small.noise.opt=sno)
# round(sum.df,2)
# round(circadian.cor(sum.df[,c(1,3,5,7,9,11)]),2)  #compare alternatives
#
# #now, lets form three "subjects" and show how the grouping variable works
# mixed.df <- rbind(small.pure,small.noisy,noisy)
# mixed.df <- data.frame(ID=c(rep(1,6),rep(2,6),rep(3,24)),
#           time=c(rep(c(8,11,14,17,20,23),2),1:24),mixed.df)
# group.df <- cosinor(angle="time",x=2:20,code="ID",data=mixed.df)
# round(group.df,2)  #compare these values to the sp,sn,and n values done separately
```

---

cta *Simulate the C(ues) T(endency) A(ction) model of motivation*

Description

Dynamic motivational models such as the Dynamics of Action (Atkinson and Birch, 1970, Revelle, 1986) may be reparameterized as a simple pair of differential (matrix) equations (Revelle, 1986, 2008). This function simulates the dynamic aspects of the CTA. The CTA model is discussed in detail in Revelle and Condon (2015). The basic duynamic model is that tendencies and the resulting actions show inertia.

Usage

```
cta(n=3,t=5000,
     cues = NULL,
      act=NULL,
      inhibit=NULL,expect = NULL,
      consume = NULL,tendency = NULL,tstrength=NULL,
      type="both",fast=2 ,compare=FALSE,
      learn=FALSE,reward=NULL,
      period =NULL)
cta.15(n = 3, t = 5000, cues = NULL,stim=NULL, act = NULL, inhibit = NULL, consume = NULL,
   ten = NULL,  type = "both", fast = 2)
```

Arguments

n	number of actions to simuate
t	length of time to simulate
cues	a vector of cue strengths
stim	A vector of the environmental stimuli
act	matrix of associations between cues and action tendencies
inhibit	inhibition matrix
consume	Consummation matrix
ten	Initial values of action tendencies
type	show actions, tendencies, both, or state diagrams
fast	display every fast time (skips
expect	A matrix of expectations
tendency	starting values of tendencies
tstrength	a vector of starting value of tendencies
compare	Allows a two x two graph to compare two plots
learn	Allow the system to learn (self reinforce) over time
reward	The strength of the reward for doing an action.
period	Introduce periodicity to the cues

Details

A very thorough discussion of the CTA model is available from Revelle (2008). An application of the model is discussed in Revelle and Condon (2015).

cta.15 is the version used to produce the figures and analysis in Revelle and Condon (2015). cta is the most recent version and includes a learning function developed in collaboration with Luke Smillie at the University of Melbourne.

As of June, 2026 the period option was introduced to simulate periodic input (e.g. seasonal effects if simulating weather). Weather/climate is a nice analogy of personality states and traits. Traits (climate) is what you expect, states (weather) is what you get. Weather (and states) are dynamic functions showing inertia.

The dynamics of action (Atkinson and Birch, 1970) was a model of how instigating forces elicited action tendencies which in turn elicited actions. The basic concept was that action tendencies had inertia. That is, a wish (action tendency) would persist until satisfied and would not change without an instigating force. The consummatory strength of doing an action was thought in turn to reduce the action tendency. Forces could either be instigating or inhibitory (leading to "negaction").

Perhaps the simplest example is the action tendency (T) to eat a pizza. The instigating forces (F) to eat the pizza include the smell and look of the pizza, and once eating it, the flavor and texture. However, if eating the pizza, there is also a consummatory force (C) which was thought to reflect both the strength (gusto) of eating the pizza as well as some constant consummatory value of the activity (c). If not eating the pizza, but in a pizza parlor, the smells and visual cues combine to increase the tendency to eat the pizza. Once eating it, however, the consummatory effect is no longer zero, and the change in action tendency will be a function of both the instigating forces and the consummatory forces. These will achieve a balance when instigating forces are equal to the consummatory forces. The asymptotic strength of eating the pizza reflects this balance and does not require a “set point" or “comparator".

To avoid the problems of instigating and consummatory lags and the need for a decision mechanism, it is possible to reparameterize the original DOA model in terms of action tendencies and actions (Revelle, 1986). Rather than specifying inertia for action tendencies and a choice rule of always expressing the dominant action tendency, it is useful to distinguish between action tendencies (t) and the actions (a) themselves and to have actions as well as tendencies having inertial properties. By separating tendencies from actions, and giving them both inertial properties, we avoid the necessity of a lag parameter, and by making the decision rule one of mutual inhibition, the process is perhaps easier to understand. In an environment which affords cues for action (c), cues enhance action tendencies (t) which in turn strengthen actions (a). This leads to two differential equations, one describing the growth and decay of action tendencies (t), the other of the actions themselves (a).

$$dt = Sc - Ca$$

and

$$da = Et - Ia$$

. (See Revelle and Condon (2015) for an extensive discussion of this model.)

`cta` simulates this model, with the addition of a learning parameter such that activities strengthen the connection between cues and tendencies. The learning part of the cta model is still under development. `cta.15` represents the state of the cta model as described in the Revelle and Condon (2015) article.

If the environmental cues are not constant but rather vary systematically over time (think seasons for the weather) some interesting things occur.

Value

graphical output unless type="none"

`cues`	echo back the cue input
`inhibition`	echo back the inhibitory matrix
`time`	time spent in each activity
`frequency`	Frequency of each activity

tendencies	average tendency strengths
actions	average action strength
summary.tend	all of the tendencies over time – to show correlations
summary.act	All of the actions over time

Author(s)

William Revelle

References

Atkinson, John W. and Birch, David (1970) The dynamics of action. John Wiley, New York, N.Y.

Revelle, William (1986) Motivation and efficiency of cognitive performance in Brown, Donald R. and Veroff, Joe (ed). Frontiers of Motivational Psychology: Essays in honor of J. W. Atkinson. Springer. (Available as a pdf at https://personality-project.org/revelle/publications/dynamicsofmotivation.pdf.)

Revelle, W. (2008) Cues, Tendencies and Actions. The Dynamics of Action revisted. https://personality-project.org/revelle/publications/cta.pdf

Revelle, W. and Condon, D. (2015) A model for personality at three levels. Journal of Research in Personality, <doi:10.1016/j.jrp.2014.12.006>

Ashley D. Brown and Revelle, W. (2021) Modeling the dynamics of action. In Measuring and Modeling Persons and Situations. <doi:10.1016/B978-0-12-819200-9.00012-0 >

Examples

```
#not run
#cta()   #default values, running over time
#cta(type="state") #default values, in a state space  of tendency 1 versus tendency 2
#these next are examples without graphic output
#not run
#two introverts
#c2i <- c(.95,1.05)
#cta(n=2,t=10000,cues=c2i,type="none")
#two extraverts
#c2e <- c(3.95,4.05)
#cta(n=2,t=10000,cues=c2e,type="none")
#three introverts
#c3i <-  c(.95,1,1.05)
#cta(3,t=10000,cues=c3i,type="none")
#three extraverts
#c3i <- c(3.95,4, 4.05)
#cta(3,10000,c3i,type="none")
#mixed
#c3 <- c(1,2.5,4)
#cta(3,10000,c3,type="none")
#state.space <-cta(n=4,t=10000,cues=c(4.1,4.0,1.1,1.0))
#show the state.space diagram
#pairs.panels(state.space$tendency.time[4001:10000,])  #allows some burn in
#scatterHist(state.space$tendency.time[4001:10000,1:2],pch=".",ellipse=FALSE,smooth=FALSE)
```

densityBy — *Create a 'violin plot' or density plot of the distribution of a set of variables*

Description

Among the many ways to describe a data set, one is a density plot for each value of a grouping variable and another is violin plot of multiple variables. A density plot shows the density for different groups to show effect sizes. A violin plot is similar to a box plot but shows the actual distribution. Median and 25th and 75th percentile lines are added to the display. If a grouping variable is specified, violinBy will draw violin plots for each variable and for each group. Data points may be drawn as well in what is known as a "raincloud plot".

Usage

```
violin(x,data=NULL, var=NULL, grp=NULL, grp.name=NULL, xlab=NULL, ylab=NULL,
 main="Density plot", vertical=TRUE, dots=FALSE, rain=FALSE, jitter=.05, alpha=1,
 errors=FALSE, eyes=TRUE, adjust=1, restrict=TRUE, xlim=NULL, add=FALSE,
 col=NULL, pch=20, scale=NULL,...)

violinBy(x, var=NULL, grp=NULL, data=NULL, grp.name=NULL, xlab=NULL, ylab=NULL,
 main="Density plot", vertical=TRUE, dots=FALSE, rain=FALSE, jitter=.05, alpha= 1,
 errors=FALSE, eyes=TRUE, adjust=1, restrict=TRUE, xlim=NULL, add=FALSE,
 col=NULL, pch=20, scale=NULL,
 ylim=NULL, xlas=0,ylas=2,order=NULL,...)

densityBy(x, var=NULL, grp=NULL,data=NULL, freq=FALSE, col=c("blue","red","black"),
 alpha=.5, adjust=1, ylim=NULL, xlim=NULL, xlab="Variable", ylab="Density",
     main="Density Plot",legend=NULL)
```

Arguments

x	A matrix or data.frame (can be expressed in formula input)
var	The variable(s) to display
grp	The grouping variable(s)
data	The name of the data object if using formula input
grp.name	If the grouping variable is specified, then what names should be give to the group? Defaults to 1:ngrp
ylab	The y label
xlab	The x label
main	Figure title
vertical	If TRUE, plot the violins vertically, otherwise, horizontonally
dots	if TRUE, add a stripchart with the data points

rain	If TRUE, draw a half violin with rain drops
jitter	If doing a stripchart, then jitter the points this much
errors	If TRUE, add error bars or cats eyes to the violins
eyes	if TRUE and errors=TRUE, then draw cats eyes
alpha	A degree of transparency (0=transparent ... 1 not transparent)
adjust	Allows smoothing of density histograms when plotting variables like height
freq	if TRUE, then plot frequencies (n * density)
restrict	Restrict the density to the observed max and min of the data
xlim	if not specified, will be .5 beyond the number of variables
ylim	If not specified, determined by the data
add	Allows overplotting
col	Allows for specification of colours. The default for 2 groups is blue and red, for more group levels, rainbows.
pch	The plot character for the mean is by default a small filled circle. To not show the mean, use pch=NA
scale	If NULL, scale the widths by the square root of sample size, otherwise scale by the value supplied.
legend	If not NULL, draw a legend at c(topleft,topright,top,left,right)
xlas	Which direction should labels appear on the x-axis
ylas	Which direction should labels appear on the y-axis?
order	If TRUE, sort the plots by the descending order of the mean values
...	Other graphic parameters

Details

Describe the data using a violin plot. Change alpha to modify the shading. The grp variable may be used to draw separate violin plots for each of multiple groups.

For relatively smallish data sets (< 500-1000), it is informative to also show the actual data points. This done with the dots=TRUE option. The jitter value is arbitrarily set to .05, but making it larger (say .1 or .2) will display more points.

Perhaps even prettier, is draw "raincloud" plots (half violins with rain drops)

Value

The density (y axis) by value (x axis) of the data (for densityBy) or a violin plot for each variable (perhaps broken down by groups)

Note

Formula input added July 12, 2020

Author(s)

William Revelle

See Also

describe, describeBy and statsBy for descriptive statistics and error.bars, error.bars.by and bi.bars, histBy and scatterHist for other graphic displays

Examples

```
violin(bfi[4:8])
violin(SATV + SATQ ~ gender, data=sat.act, grp.name =cs(MV,FV,MQ,FQ)) #formula input
violinBy(bfi,var=4:7,grp ="gender",grp.name=c("M","F"))
#rain does not work for multiple DVS, just up to 2 IVs
violinBy(SATV  ~ education,data =sat.act, rain=TRUE, pch=".", vertical=FALSE)  #rain

densityBy(SATV  ~ gender,data =sat.act,legend=1)  #formula input
```

---

describe *Basic descriptive statistics useful for psychometrics*

Description

There are many summary statistics available in R; this function provides the ones most useful for scale construction and item analysis in classic psychometrics. Range is most useful for the first pass in a data set, to check for coding errors. Parallelizes if multiple cores are available.

Usage

```
describe(x, na.rm = TRUE, interp=FALSE,skew = TRUE, ranges = TRUE,trim=.1,
         type=3,check=TRUE,fast=NULL,quant=NULL,IQR=FALSE,omit=FALSE,data=NULL,
              size=50)
describeData(x,head=4,tail=4)
describeFast(x)
```

Arguments

x	A data frame or matrix
na.rm	The default is to delete missing data. na.rm=FALSE will delete the case.
interp	Should the median be standard or interpolated
skew	Should the skew and kurtosis be calculated?
ranges	Should the range be calculated?
trim	trim=.1 – trim means by dropping the top and bottom trim fraction
type	Which estimate of skew and kurtosis should be used? (See details.)
check	Should we check for non-numeric variables? Slower but helpful.

fast	if TRUE, will do n, means, sds, min, max, ranges for an improvement in speed. If NULL, will switch to fast mode for large (ncol * nrow > 10^7) problems, otherwise defaults to fast = FALSE
quant	if not NULL, will find the specified quantiles (e.g. quant=c(.25,.75) will find the 25th and 75th percentiles)
IQR	If TRUE, show the interquartile range
omit	Do not convert non-numerical variables to numeric, omit them instead
head	show the first 1:head cases for each variable in describeData
tail	Show the last nobs-tail cases for each variable in describeData
data	Allows formula input for specific grouping variables
size	For very big problems (in terms of nvar) set size to some reasonable value.

Details

In basic data analysis it is vital to get basic descriptive statistics. Procedures such as summary and Hmisc::describe do so. The describe function in the psych package is meant to produce the most frequently requested stats in psychometric and psychology studies, and to produce them in an easy to read data.frame. If a grouping variable is called for in formula mode, it will also call describeBy to the processing. The results from describe can be used in graphics functions (e.g., error.crosses).

The range statistics (min, max, range) are most useful for data checking to detect coding errors, and should be found in early analyses of the data.

Although describe will work on data frames as well as matrices, it is important to realize that for data frames, descriptive statistics will be reported only for those variables where this makes sense (i.e., not for alphanumeric data).

If the check option is TRUE, variables that are categorical or logical are converted to numeric and then described. These variables are marked with an * in the row name. This is somewhat slower. Note that in the case of categories or factors, the numerical ordering is not necessarily the one expected. For instance, if education is coded "high school", "some college" , "finished college", then the default coding will lead to these as values of 2, 3, 1. Thus, statistics for those variables marked with * should be interpreted cautiously (if at all).

In a typical study, one might read the data in from the clipboard (read.clipboard), show the splom plot of the correlations (pairs.panels), and then describe the data.

na.rm=FALSE is equivalent to describe(na.omit(x))

When finding the skew and the kurtosis, there are three different options available. These match the choices available in skewness and kurtosis found in the e1071 package (see Joanes and Gill (1998) for the advantages of each one).

If we define $m_r = [\sum(X - mx)^r]/n$ then

Type 1 finds skewness and kurtosis by $g_1 = m_3/(m_2)^{3/2}$ and $g_2 = m_4/(m_2)^2 - 3$.

Type 2 is $G1 = g1 * \sqrt{n * (n-1)}/(n-2)$ and $G2 = (n-1) * [(n+1)g2 + 6]/((n-2)(n-3))$.

Type 3 is $b1 = [(n-1)/n]^{3/2} m_3/m_2^{3/2}$ and $b2 = [(n-1)/n]^{3/2} m_4/m_2^2)$.

The additional helper function describeData just scans the data array and reports on whether the data are all numerical, logical/factorial, or categorical. This is a useful check to run if trying to get descriptive statistics on very large data sets where to improve the speed, the check option is FALSE.

An even faster overview of the data is `describeFast` which reports the number of total cases, number of complete cases, number of numeric variables and the number which are factors.

The fast=TRUE option will lead to a speed up of about 50% for larger problems by not finding all of the statistics (see NOTE)

To describe the data for different groups, see `describeBy` or specify the grouping variable(s) in formula mode (see the examples).

Value

A data.frame of the relevant statistics:
item name
item number
number of valid cases
mean
standard deviation
trimmed mean (with trim defaulting to .1)
median (standard or interpolated
mad: median absolute deviation (from the median).
minimum
maximum
skew
kurtosis
standard error

Note

For very large data sets that are data.frames, `describe` can be rather slow. Converting the data to a matrix first is recommended. However, if the data are of different types, (factors or logical), this is not possible. If the data set includes columns of character data, it is also not possible. Thus, a quick pass with `describeData` is recommended. Even faster is a quick pass with `describeFast` which just counts number of observations per variable and reports the type of data (numerical, factor, logical).

For the greatest speed, at the cost of losing information, do not ask for ranges or for skew and turn off check. This is done automatically if the fast option is TRUE or for large data sets.

Note that by default, fast=NULL. But if the number of cases x number of variables exceeds (ncol * nrow > 10^7), fast will be set to TRUE. This will provide just n, mean, sd, min, max, range, and standard errors. To get all of the statistics (but at a cost of greater time) set fast=FALSE.

The problem seems to be a memory limitation in that the time taken is an accelerating function of nvars * nobs. Thus, for a largish problem (72,000 cases with 1680 variables) which might take 330 seconds, doing it as two sets of 840 variable cuts the time down to 80 seconds.

The object returned is a data frame with the normal precision of R. However, to control the number of digits displayed, you can set digits in a print command, rather than losing precision at the descriptive stats level. See the last two examples. One just sets the number of digits, one gives uses signif to make 'prettier' output where all numbers are displayed to the same number of digits.

The MAD (median absolute deviation from the median) is calculated using the mad function from the stats package in Core-R. Note that by default, the MAD is adjusted by a scaling factor (1.4826) that will give the expectation of the MAD to be the same as the standard deviation for normal data.

An interesting problem with describe is that a function with the same name is in the Hmisc package. HMisc is loaded by qqgraph which in turn is loaded by SemPlot. So even if not directly loading HMisc, if you load SemPlot after loading psych, describe will not work, but the reverse order for loading should work.

Author(s)

https://personality-project.org/revelle.html

Maintainer: William Revelle <revelle@northwestern.edu>

References

Joanes, D.N. and Gill, C.A (1998). Comparing measures of sample skewness and kurtosis. The Statistician, 47, 183-189.

See Also

describeBy, skew, kurtosi interp.median, read.clipboard. Then, for graphic output, see error.crosses, pairs.panels, error.bars, error.bars.by and densityBy, or violinBy

Examples

```
data(sat.act)
describe(sat.act)
describe(sat.act ~ gender) #formula mode option calls describeBy for the entire data frame
describe(SATV + SATQ ~ gender, data=sat.act) #formula mode specifies just two variables

describe(sat.act,skew=FALSE)
describe(sat.act,IQR=TRUE) #show the interquartile Range
describe(sat.act,quant=c(.1,.25,.5,.75,.90) ) #find the 10th, 25th, 50th,
                   #75th and 90th percentiles



describeData(sat.act) #the fast version just  gives counts and head and tail

print(describeFast(sat.act),short=FALSE)  #even faster is just counts  (just less information)

#now show how to adjust the displayed number of digits
 des <- describe(sat.act)  #find the descriptive statistics.  Keep the original accuracy
 des  #show the normal output, which is rounded to 2 decimals
 print(des,digits=3)  #show the output, but round to 3 (trailing) digits
 print(des, signif=3) #round all numbers to the 3 significant digits
```

describeBy *Basic summary statistics by group*

Description

Report basic summary statistics by a grouping variable. Useful if the grouping variable is some experimental variable and data are to be aggregated for plotting. Partly a wrapper for by and describe

Usage

```
describeBy(x, group=NULL,mat=FALSE,type=3,digits=15,data,...)
describe.by(x, group=NULL,mat=FALSE,type=3,...)  # deprecated
```

Arguments

x	a data.frame or matrix. See note for statsBy.
group	a grouping variable or a list of grouping variables. (may be ignored if calling using the formula mode.)
mat	provide a matrix output rather than a list
type	Which type of skew and kurtosis should be found
digits	When giving matrix output, how many digits should be reported?
data	Needed if using formula input
...	parameters to be passed to describe

Details

To get descriptive statistics for several different grouping variables, make sure that group is a list. In the case of matrix output with multiple grouping variables, the grouping variable values are added to the output.

As of July, 2020, the grouping variable(s) may be specified in formula mode (see the examples).

The type parameter specifies which version of skew and kurtosis should be found. See describe for more details.

An alternative function (statsBy) returns a list of means, n, and standard deviations for each group. This is particularly useful if finding weighted correlations of group means using cor.wt. More importantly, it does a proper within and between group decomposition of the correlation.

cohen.d will work for two groups. It converts the data into mean differences and pools the within group standard deviations. Returns cohen.d statistic as well as the multivariate generalization (Mahalanobis D).

Value

A data.frame of the relevant statistics broken down by group:
item name
item number
number of valid cases
mean
standard deviation
median
mad: median absolute deviation (from the median)
minimum
maximum
skew
standard error

Author(s)

William Revelle

See Also

describe, statsBy, densityBy and violinBy, cohen.d, cohen.d.by, and cohen.d.ci as well as error.bars and error.bars.by for other graphical displays.

Examples

```
data(sat.act)
describeBy(sat.act,sat.act$gender) #just one grouping variable
describeBy(sat.act ~ gender)   #describe the entire set formula input
describeBy(SATV + SATQ ~ gender,data =sat.act)  #specify the data set if using formula
#describeBy(sat.act,list(sat.act$gender,sat.act$education))  #two grouping variables
describeBy(sat.act ~ gender +  education) #two grouping variables
des.mat <- describeBy(age ~ education,mat=TRUE,data = sat.act) #matrix (data.frame) output
des.mat <- describeBy(age ~ education + gender, data=sat.act,
               mat=TRUE,digits=2)  #matrix output  rounded to 2 decimals
```

---

diagram *Helper functions for drawing path model diagrams*

Description

Path models are used to describe structural equation models or cluster analytic output. These functions provide the primitives for drawing path models. Used as a substitute for some of the functionality of Rgraphviz.

Usage

```
diagram(fit,...)
dia.rect(x, y = NULL, labels = NULL,  cex = 1,  xlim = c(0, 1), ylim = c(0, 1),
    draw=TRUE, ...)
dia.ellipse(x, y = NULL, labels = NULL, cex=1,e.size=.05, xlim=c(0,1),
       ylim=c(0,1),draw=TRUE,  ...)
dia.triangle(x, y = NULL, labels =NULL,  cex = 1, xlim=c(0,1),ylim=c(0,1),...)
dia.ellipse1(x,y,e.size=.05,xlim=c(0,1),ylim=c(0,1),draw=TRUE,...)
dia.shape(x, y = NULL, labels = NULL, cex = 1,
         e.size=.05, xlim=c(0,1), ylim=c(0,1), shape=1, ...)
dia.arrow(from,to,labels=NULL,scale=1,cex=1,adj=2,both=FALSE,pos=NULL,l.cex,
        gap.size,draw=TRUE,col="black",lty="solid",...)
dia.curve(from,to,labels=NULL,scale=1,...)
dia.curved.arrow(from,to,labels=NULL,scale=1,both=FALSE,dir=NULL,draw=TRUE,...)
dia.self(location,labels=NULL,scale=.8,side=2,draw=TRUE,...)
dia.cone(x=0, y=-2, theta=45, arrow=TRUE,curves=TRUE,add=FALSE,labels=NULL,
      xlim = c(-1, 1), ylim=c(-1,1),... )
multi.self(self.list,...)
multi.arrow(arrows.list,...)
multi.curved.arrow(curved.list,l.cex=NULL,...)
multi.rect(rect.list,...)
```

Arguments

fit	The results from a factor analysis fa, components analysis principal, omega reliability analysis, omega, cluster analysis iclust, topdown (bassAckward) bassAckward or confirmatory factor analysis, cfa, or structural equation model,sem, using the lavaan package.
x	x coordinate of a rectangle or ellipse
y	y coordinate of a rectangle or ellipse
e.size	The size of the ellipse (scaled by the number of variables
labels	Text to insert in rectangle, ellipse, or arrow
cex	adjust the text size
col	line color (normal meaning for plot figures)
lty	line type
l.cex	Adjust the text size in arrows, defaults to cex which in turn defaults to 1
gap.size	Tweak the gap in an arrow to be allow the label to be in a gap
adj	Where to put the label along the arrows (values are then divided by 4)
both	Should the arrows have arrow heads on both ends?
scale	modifies size of rectangle and ellipse as well as the curvature of curves. (For curvature, positive numbers are concave down and to the left
from	arrows and curves go from
to	arrows and curves go to
location	where is the rectangle?

shape	Which shape to draw
xlim	default ranges
ylim	default ranges
draw	Draw the text box
side	Which side of boxes should errors appear
theta	Angle in degrees of vectors
arrow	draw arrows for edges in dia.cone
add	if TRUE, plot on previous plot
curves	if TRUE, draw curves between arrows in dia.cone
pos	The position of the text in . Follows the text positions of 1, 2, 3, 4 or NULL
dir	Should the direction of the curve be calculated dynamically, or set as "up" or "left"
...	Most graphic parameters may be passed here
self.list	list saved from dia.self
arrows.list	lst saved from dia.arrow
curved.list	list saved from dia.curved.arrow
rect.list	list saved from dia.rect

Details

The diagram function calls fa.diagram, omega.diagram, ICLUST.diagram, lavaan.diagram or bassAckward.diagram depending upon the class of the fit input. See those functions for particular parameter values.

The remaining functions are the graphic primitives used by fa.diagram, structure.diagram, omega.diagram, ICLUST.diagram and het.diagram

They create rectangles, ellipses or triangles surrounding text, connect them to straight or curved arrows, and can draw an arrow from and to the same rectangle.

To speed up the plotting, dia.rect and dia.arrow can suppress the actual drawing and return the locations and values to plot. These values can then be directly called by text or rect with matrix input. This leads to an impressive increase in speed when doing many variables.

The functions multi.rect, multi.self, multi.arrow and multi.curved.arrow will take the saved output from the appropriate primitives and then draw them all at once.

Each shape (ellipse, rectangle or triangle) has a left, right, top and bottom and center coordinate that may be used to connect the arrows.

Curves are double-headed arrows. By default they go from one location to another and curve either left or right (if going up or down) or up or down (going left to right). The direction of the curve may be set by dir="up" for left right curvature.

The helper functions were developed to get around the infelicities associated with trying to install Rgraphviz and graphviz.

These functions form the core of fa.diagram,het.diagram.

Better documentation will be added as these functions get improved. Currently the helper functions are just a work around for Rgraphviz.

dia.cone draws a cone with (optionally) arrows as sides and centers to show the problem of factor indeterminacy.

Value

Graphic output

Author(s)

William Revelle

See Also

The diagram functions that use the dia functions: fa.diagram, structure.diagram, omega.diagram, and ICLUST.diagram.

Examples

```
#first, show the primitives
xlim=c(-2,10)
ylim=c(0,10)
plot(NA,xlim=xlim,ylim=ylim,main="Demonstration of diagram functions",axes=FALSE,xlab="",ylab="")
ul <- dia.rect(1,9,labels="upper left",xlim=xlim,ylim=ylim)
ml <- dia.rect(1,6,"middle left",xlim=xlim,ylim=ylim)
ll <- dia.rect(1,3,labels="lower left",xlim=xlim,ylim=ylim)
bl <- dia.rect(1,1,"bottom left",xlim=xlim,ylim=ylim)
lr <- dia.ellipse(7,3,"lower right",xlim=xlim,ylim=ylim,e.size=.07)
ur <- dia.ellipse(7,9,"upper right",xlim=xlim,ylim=ylim,e.size=.07)
mr <- dia.ellipse(7,6,"middle right",xlim=xlim,ylim=ylim,e.size=.07)
lm <- dia.triangle(4,1,"Lower Middle",xlim=xlim,ylim=ylim)
br <- dia.rect(9,1,"bottom right",xlim=xlim,ylim=ylim)
dia.curve(from=ul$left,to=bl$left,"double headed",scale=-1)

dia.arrow(from=lr,to=ul,labels="right to left")
dia.arrow(from=ul,to=ur,labels="left to right")
dia.curved.arrow(from=lr,to=ll,labels ="right to left")
dia.curved.arrow(to=ur,from=ul,labels ="left to right")
dia.curve(ll$top,ul$bottom,"right")  #for rectangles, specify where to point

dia.curve(ll$top,ul$bottom,"left",scale=-1)  #for rectangles, specify where to point
dia.curve(mr,ur,"up")  #but for ellipses, you may just point to it.
dia.curve(mr,lr,"down")
dia.curve(mr,ur,"up")
dia.curved.arrow(mr,ur,"up")  #but for ellipses, you may just point to it.
dia.curved.arrow(mr,lr,"down")  #but for ellipses, you may just point to it.

dia.curved.arrow(ur$right,mr$right,"3")
dia.curve(ml,mr,"across")
dia.curve(ur$right,lr$right,"top down",scale =2)
dia.curved.arrow(br$top,lr$right,"up")
dia.curved.arrow(bl,br,"left to right")
dia.curved.arrow(br,bl,"right to left",scale=-1)
dia.arrow(bl,ll$bottom)
dia.curved.arrow(ml,ll$right)
dia.curved.arrow(mr,lr$top)
```

```
#now, put them together in a factor analysis diagram
v9 <- sim.hierarchical()
f3 <- fa(v9,3,rotate="cluster")
fa.diagram(f3,error=TRUE,side=3)
```

---

draw.tetra *Draw a correlation ellipse and two normal curves to demonstrate tetrachoric correlation*

Description

A graphic of a correlation ellipse divided into 4 regions based upon x and y cutpoints on two normal distributions. This is also an example of using the layout function. Draw a bivariate density plot to show how tetrachorics work.

Usage

```
draw.tetra(r, t1, t2,shade=TRUE)
draw.cor(r=.5,expand=10,theta=30,phi=30,N=101,nbcol=30,box=TRUE,
main="Bivariate density  rho = ",cuts=NULL,all=TRUE,ellipses=TRUE,ze=.15)
```

Arguments

r	the underlying Pearson correlation defines the shape of the ellipse
t1	X is cut at tau
t2	Y is cut at Tau
shade	shade the diagram (default is TRUE)
expand	The relative height of the z axis
theta	The angle to rotate the x-y plane
phi	The angle above the plane to view the graph
N	The grid resolution
nbcol	The color resolution
box	Draw the axes
main	The main title
cuts	Should the graphic show cuts (e.g., cuts=c(0,0))
all	Show all four parts of the tetrachoric
ellipses	Draw a correlation ellipse
ze	height of the ellipse if requested

Details

A graphic demonstration of the tetrachoric correlation. Used for teaching purposes. The default values are for a correlation of .5 with cuts at 1 and 1. Any other values are possible. The code is also a demonstration of how to use the layout function for complex graphics using base graphics.

Author(s)

William Revelle

See Also

tetrachoric to find tetrachoric correlations, irt.fa and fa.poly to use them in factor analyses, scatter.hist to show correlations and histograms.

Examples

```
#if(require(mvtnorm)) {
#draw.tetra(.5,1,1)
#draw.tetra(.8,2,1)} else {print("draw.tetra requires the mvtnorm package")
#draw.cor(.5,cuts=c(0,0))}

draw.tetra(.5,1,1)
draw.tetra(.8,2,1)
draw.cor(.5,cuts=c(0,0))
```

---

dummy.code *Create dummy coded variables*

Description

Given a variable x with n distinct values, create n new dummy coded variables coded 0/1 for presence (1) or absence (0) of each variable. A typical application would be to create dummy coded college majors from a vector of college majors. Can also combine categories by group. By default, NA values of x are returned as NA (added 10/20/17)

Usage

```
dummy.code(x,group=NULL,na.rm=TRUE,top=NULL,min=NULL)
```

Arguments

x	A vector to be transformed into dummy codes
group	A vector of categories to be coded as 1, all others coded as 0.
na.rm	If TRUE, return NA for all codes with NA in x
top	If specified, then just dummy code the top values, and make the rest NA
min	If specified, then dummy code all values >= min

Details

When coding demographic information, it is typical to create one variable with multiple categorical values (e.g., ethnicity, college major, occupation). dummy.code will convert these categories into n distinct dummy coded variables.

If there are many possible values (e.g., country in the SAPA data set) then specifying top will assign dummy codes to just a subset of the data.

If using dummy coded variables as predictors, remember to use n-1 variables.

If group is specified, then all values of x that are in group are given the value of 1, otherwise, 0. (Useful for combining a range of science majors into STEM or not. The example forms a dummy code of any smoking at all.)

Value

A matrix of dummy coded variables

Author(s)

William Revelle

Examples

```
new <- dummy.code(sat.act$education)
new.sat <- data.frame(new,sat.act)
round(cor(new.sat,use="pairwise"),2)
#dum.smoke <- dummy.code(spi$smoke,group=2:9)
#table(dum.smoke,spi$smoke)
#dum.age <- dummy.code(round(spi$age/5)*5,top=5)  #the most frequent five year blocks
```

Dwyer — *8 cognitive variables used by Dwyer for an example.*

Description

Dwyer (1937) introduced a technique for factor extension and used 8 cognitive variables from Thurstone. This is the example data set used in his paper.

Usage

```
data(Dwyer)
```

Format

The format is: num [1:8, 1:8] 1 0.58 -0.28 0.01 0.36 0.38 0.61 0.15 0.58 1 ... - attr(*, "dimnames")=List of 2 ..$: chr [1:8] "V1" "V2" "V3" "V4" ... ..$: chr [1:8] "V1" "V2" "V3" "V4" ...

Source

Data matrix retyped from the original publication.

References

Dwyer, Paul S. (1937), The determination of the factor loadings of a given test from the known factor loadings of other tests. Psychometrika, 3, 173-178

Examples

```
data(Dwyer)
Ro <- Dwyer[1:7,1:7]
Roe <- Dwyer[1:7,8]
fo <- fa(Ro,2,rotate="none")
fa.extension(Roe,fo)
```

---

eigen.loadings — *Convert eigen vectors and eigen values to the more normal (for psychologists) component loadings*

Description

The default procedures for principal component returns values not immediately equivalent to the loadings from a factor analysis. eigen.loadings translates them into the more typical metric of eigen vectors multiplied by the squareroot of the eigenvalues. This lets us find pseudo factor loadings if we have used princomp or eigen.
If we use `principal` to do our principal components analysis, then we do not need this routine.

Usage

```
eigen.loadings(x)
```

Arguments

x	the output from eigen or a list of class princomp derived from princomp

Value

A matrix of Principal Component loadings more typical for what is expected in psychometrics. That is, they are scaled by the square root of the eigenvalues.

Note

Useful for SAPA analyses

Author(s)

< revelle@northwestern.edu >
https://personality-project.org/revelle.html

Examples

```
x <- eigen(Harman74.cor$cov)
x$vectors[1:8,1:4]  #as they appear from eigen
y <- princomp(covmat=Harman74.cor$cov)
y$loadings[1:8,1:4] #as they appear from princomp
eigen.loadings(x)[1:8,1:4] # rescaled by the eigen values
z <- pca(Harman74.cor$cov,4,rotate="none")
z$loadings[1:8,1:4]  #as they appear in pca
```

`ellipses` *Plot data and 1 and 2 sigma correlation ellipses*

Description

For teaching correlation, it is useful to draw ellipses around the mean to reflect the correlation. This variation of the ellipse function from John Fox's car package does so. Input may be either two vectors or a matrix or data.frame. In the latter cases, if the number of variables >2, then the ellipses are done in the `pairs.panels` function. Ellipses may be added to existing plots. The minkowski function is included as a generalized ellipse.

Usage

```
ellipses(x, y = NULL, add = FALSE, smooth=TRUE, lm=FALSE,data=TRUE, n = 2,
 span=2/3, iter=3, col = "red", xlab =NULL,ylab= NULL,size=c(1,2), ...)
minkowski(r=2,add=FALSE,main=NULL,xl=1,yl=1)
```

Arguments

`x`	a vector,matrix, or data.frame
`y`	Optional second vector
`add`	Should a new plot be created, or should it be added to?
`smooth`	smooth = TRUE -> draw a loess fit
`lm`	lm=TRUE -> draw the linear fit
`data`	data=TRUE implies draw the data points
`n`	Should 1 or 2 ellipses be drawn
`span`	averaging window parameter for the lowess fit
`iter`	iteration parameter for lowess
`col`	color of ellipses (default is red
`xlab`	label for the x axis
`ylab`	label for the y axis
`size`	The size of ellipses in sd units (defaults to 1 and 2)
`...`	Other parameters for plotting
`r`	r=1 draws a city block, r=2 is a Euclidean circle, r > 2 tends towards a square
`main`	title to use when drawing Minkowski circles
`xl`	stretch the x axis
`yl`	stretch the y axis

Details

Ellipse dimensions are calculated from the correlation between the x and y variables and are scaled as sqrt(1+r) and sqrt(1-r). They are then scaled as size[1] and size[2] standard deviation units. To scale for 95 and 99 percent confidence use c(1.64,2.32)

Value

A single plot (for 2 vectors or data frames with fewer than 3 variables. Otherwise a call is made to pairs.panels.

Note

Adapted from John Fox's ellipse and data.ellipse functions.

Author(s)

William Revelle

References

Galton, Francis (1888), Co-relations and their measurement. Proceedings of the Royal Society. London Series, 45, 135-145.

See Also

pairs.panels

Examples

```
if(require(psychTools)) {
data(psychTools::galton)
galton <- psychTools::galton
ellipses(galton,lm=TRUE)

ellipses(galton$parent,galton$child,xlab="Mid Parent Height",
                  ylab="Child Height") #input are two vectors
}
data(sat.act)
ellipses(sat.act)  #shows the pairs.panels ellipses
minkowski(2,main="Minkowski circles")
minkowski(1,TRUE)
minkowski(4,TRUE)
```

error.bars *Plot means and confidence intervals*

Description

One of the many functions in R to plot means and confidence intervals. Can be done using barplots if desired. Can also be combined with such functions as boxplot or violin to summarize distributions. Means and standard errors are calculated from the raw data using describe. Alternatively, plots of means +/- one standard deviation may be drawn.

Usage

```
error.bars(x,stats=NULL,data=NULL,group=NULL, ylab = "Dependent Variable",
 xlab="Independent Variable", main=NULL,eyes=TRUE, ylim = NULL, xlim=NULL,
 alpha=.05, sd=FALSE, labels = NULL, pos = NULL,  arrow.len = 0.05,
 arrow.col="black",add = FALSE,bars=FALSE, within=FALSE,
 col=c("black","blue","red"), density=-10,...)

error.bars.tab(t,way="columns",raw=FALSE,col=c('blue','red'),...)
```

Arguments

x	A data frame or matrix of raw data OR, a formula of the form DV ~ IV. If formula input is specified, error.bars.by is called.
t	A table of frequencies
stats	Alternatively, a data.frame of descriptive stats from (e.g., describe). if specified, the means, sd, n and perhaps of se of a data set to be plotted
data	If using formula input, specify the object where the data may found
group	If not null, then do error.bars.by syntax
ylab	y label
xlab	x label
main	title for figure
ylim	if specified, the limits for the plot, otherwise based upon the data
xlim	if specified, the x limits for the plot, otherwise c(.5,nvar + .5)
eyes	should 'cats eyes' plots be drawn
alpha	alpha level of confidence interval – defaults to 95% confidence interval
sd	if TRUE, draw one standard deviation instead of standard errors at the alpha level
labels	X axis label
pos	where to place text: below, left, above, right
arrow.len	How long should the top of the error bars be?
arrow.col	What color should the error bars be?

add	add=FALSE, new plot, add=TRUE, just points and error bars
bars	bars=TRUE will draw a bar graph if you really want to do that
within	should the error variance of a variable be corrected by 1-SMC?
col	color(s) of the catseyes. Defaults to blue.
density	If negative, solid colors, if positive, how many lines to draw
way	Percentages are based upon the row totals (default) column totals, or grand total of the data Table
raw	If raw is FALSE, display the graphs in terms of probability, raw TRUE displays the data in terms of raw counts
...	other parameters to pass to the plot function, e.g., typ="b" to draw lines, lty="dashed" to draw dashed lines

Details

Drawing the mean +/- a confidence interval is a frequently used function when reporting experimental results. By default, the confidence interval is 1.96 standard errors of the t-distribution.

If within=TRUE, the error bars are corrected for the correlation with the other variables by reducing the variance by a factor of (1-smc). This allows for comparisons between variables.

The error bars are normally calculated from the data using the `describe` function. If, alternatively, a matrix of statistics is provided with column headings of values, means, and se, then those values will be used for the plot (using the stats option). If n is included in the matrix of statistics, then the distribution is drawn for a t distribution for n-1 df. If n is omitted (NULL) or is NA, then the distribution will be a normal distribution.

If sd is TRUE, then the error bars will represent one standard deviation from the mean rather than be a function of alpha and the standard errors.

See the last two examples for the case of plotting data with statistics from another function.

Alternatively, `error.bars.tab` will take tabulated data and convert to either row, column or overall percentages, and then plot these as percentages with the equivalent standard error (based upon sqrt(pq/N)).

In August, 2018, the functionality of `error.bars` and `error.bars.by` were combined so that if groups are specified, then the error bars are done by group. Furthermore, if the x variable is a formula of the form DV ~ IV, then `error.bars.by` is called to do the plotting.

Value

Graphic output showing the means + x

These confidence regions are based upon normal theory and do not take into account any skew in the variables. More accurate confidence intervals could be found by resampling.

The error.bars.tab function will return (invisibly) the cell means and standard errors.

Author(s)

William Revelle

See Also

error.crosses for two way error bars, error.bars.by for error bars for different groups as well as error.dots.

The error.bars.by is useful for showing the results of one or two way ANOVAs in that it will display means and CIs for one or more DVs for one or two IVs.

In addition, as pointed out by Jim Lemon on the R-help news group, error bars or confidence intervals may be drawn using

function	package
bar.err	(agricolae)
plotCI	(gplots)
xYplot	(Hmisc)
dispersion	(plotrix)
plotCI	(plotrix)

For advice why not to draw bar graphs with error bars, see the page at biostat.mc.vanderbilt.edu/wiki/Main/DynamitePlots.

Examples

```
set.seed(42)
x <- matrix(rnorm(1000),ncol=20)
boxplot(x,notch=TRUE,main="Notched boxplot with error bars")
error.bars(x,add=TRUE)
abline(h=0)

#show 50% confidence regions and color each variable separately
error.bars(attitude,alpha=.5,
   main="50 percent confidence limits",col=rainbow(ncol(attitude)) )

error.bars(attitude,bar=TRUE)  #show the use of bar graphs


#combine with a strip chart and boxplot
stripchart(attitude,vertical=TRUE,method="jitter",jitter=.1,pch=19,
           main="Stripchart with 95 percent confidence limits")
boxplot(attitude,add=TRUE)
error.bars(attitude,add=TRUE,arrow.len=.2)

#use statistics from somewhere else
#by specifying n, we are using the t distribution for confidences
#The first example allows the variables to be spaced along the x axis
my.stats <- data.frame(values=c(1,2,8),mean=c(10,12,18),se=c(2,3,5),n=c(5,10,20))
 error.bars(stats=my.stats,type="b",main="data with confidence intervals")
#don't connect the groups
my.stats <- data.frame(values=c(1,2,8),mean=c(10,12,18),se=c(2,3,5),n=c(5,10,20))
      error.bars(stats=my.stats,main="data with confidence intervals")
#by not specifying value, the groups are equally spaced
my.stats <- data.frame(mean=c(10,12,18),se=c(2,3,5),n=c(5,10,20))
rownames(my.stats) <- c("First", "Second","Third")
```

```
error.bars(stats=my.stats,xlab="Condition",ylab="Score")


#Consider the case where we get stats from describe
temp <- describe(attitude)
error.bars(stats=temp)

#show these do not differ from the other way by overlaying the two
error.bars(attitude,add=TRUE,col="red")

#n is omitted
#the error distribution is a normal distribution
my.stats <- data.frame(mean=c(2,4,8),se=c(2,1,2))
rownames(my.stats) <- c("First", "Second","Third")
error.bars(stats=my.stats,xlab="Condition",ylab="Score")
#n is specified
#compare this with small n which shows larger confidence regions
my.stats <- data.frame(mean=c(2,4,8),se=c(2,1,2),n=c(10,10,3))
rownames(my.stats) <- c("First", "Second","Third")
error.bars(stats=my.stats,xlab="Condition",ylab="Score")


#example of arrest rates (as percentage of condition)
arrest <- data.frame(Control=c(14,21),Treated =c(3,23))
rownames(arrest) <- c("Arrested","Not Arrested")
error.bars.tab(arrest,ylab="Probability of Arrest",xlab="Control vs Treatment",
main="Probability of Arrest varies by treatment")

#EXAMPLE from Anderson and Revelle, 1983
 an.rev.df <- data.frame(means = c(88.88,77.24,74.90,88.72), se=c(4.66,4.43,6.82,7.34),
      n =c(16,18,14,21), Imp =c(1,1,2,2),Drug=c(1,2,1,2))
# not run because of some weird error in R CMD check
#error.bars.by(means ~ Imp + Drug,stats=an.rev.df,density=30,v.labels=cs(Placebo,Drug),
# ylab="Lines Read",xlab="Drug condition",main ="Anderson and Revelle,1983")


#Show the raw  rates
error.bars.tab(arrest,raw=TRUE,ylab="Number Arrested",xlab="Control vs Treatment",
main="Count of Arrest varies by treatment")

#If a grouping variable is specified, the function calls error.bars.by
#Use error.bars.by to have more control over the output.
#Show  how to use grouping variables
error.bars(SATV + SATQ ~ gender, data=sat.act) #one grouping variable, formula input
error.bars(SATV + SATQ ~ education + gender,data=sat.act)#two  grouping variables
```

`error.bars.by` *Plot means and confidence intervals for multiple groups*

Description

One of the many functions in R to plot means and confidence intervals. Meant mainly for demonstration purposes for showing the probabilty of replication from multiple samples. Can also be combined with such functions as boxplot to summarize distributions. Means and standard errors for each group are calculated using `describeBy`.

Usage

```
error.bars.by(x,group,data=NULL, by.var=FALSE,x.cat=TRUE,ylab =NULL,xlab=NULL, main=NULL,
 ylim= NULL, xlim=NULL,
 eyes=TRUE, alpha=.05,sd=FALSE,labels=NULL,v.labels=NULL,v2.labels=NULL,
 add.labels=NULL,pos=NULL, arrow.len=.05, min.size=1,add=FALSE,
 bars=FALSE,within=FALSE,
 colors=c("black","blue","red"), lty,lines=TRUE,
 legend=0,pch=16,density=-10,stats=NULL,...)
```

Arguments

`x`	A data frame or matrix
`group`	A grouping variable
`data`	If using formula input, the data file must be specified
`by.var`	A different line for each group (default) or each variable
`x.cat`	Is the grouping variable categorical (TRUE) or continuous (FALSE
`ylab`	y label
`xlab`	x label
`main`	title for figure
`ylim`	if specified, the y limits for the plot, otherwise based upon the data
`xlim`	if specified, the x limits for the plot, otherwise based upon the data
`eyes`	Should 'cats eyes' be drawn'
`alpha`	alpha level of confidence interval. Default is 1- alpha =95% confidence interval
`sd`	sd=TRUE will plot Standard Deviations instead of standard errors
`labels`	X axis label
`v.labels`	For a bar plot legend, these are the variable labels, for a line plot, the labels of the grouping variable.
`v2.labels`	the names for the 2nd grouping variable, if there is one
`add.labels`	if !NULL, then add the v2.labels to the left/right of the lines (add.labels="right")
`pos`	where to place text: below, left, above, right
`arrow.len`	How long should the top of the error bars be?
`min.size`	Draw error bars for groups > min.size
`add`	add=FALSE, new plot, add=TRUE, just points and error bars
`bars`	Draw a barplot with error bars rather than a simple plot of the means
`within`	Should the s.e. be corrected by the correlation with the other variables?

`colors`	groups will be plotted in different colors (mod n.groups). See the note for how to make them transparent.
`lty`	line type may be specified in the case of not plotting by variables
`lines`	By default, when plotting different groups, connect the groups with a line of type = lty. If lines is FALSE, then do not connect the groups
`legend`	Where should the legend be drawn: 0 (do not draw it), 1= lower right corner, 2 = bottom, 3 ... 8 continue clockwise, 9 is the center. Negative values will reverse the groups in the legend.
`pch`	The first plot symbol to use. Subsequent groups are pch + group
`density`	How many lines/inch should fill the cats eyes. If missing, non-transparent colors are used. If negative, transparent colors are used. May be a vector for different values.
`stats`	if specified, the means, sd, n and perhaps of se of a data set to be plotted
`...`	other parameters to pass to the plot function e.g., lty="dashed" to draw dashed lines

Details

Drawing the mean +/- a confidence interval is a frequently used function when reporting experimental results. By default, the confidence interval is 1.96 standard errors (adjusted for the t-distribution).

Improved/modified in August, 2018 to allow formula input (see examples) as well as to more properly handle multiple groups.

Following a request for better labeling of the grouping variables, the v.lab option is implemented for line graphs as well as bar graphs. Note that if using multiple grouping variables, the labels are for the variable with the most levels (which should be the first one.)

This function was originally just a wrapper for `error.bars` but has been written to allow groups to be organized either as the x axis or as separate lines.

If desired, a barplot with error bars can be shown. Many find this type of plot to be uninformative (e.g., https://biostat.mc.vanderbilt.edu/DynamitePlots) and recommend the more standard dot plot.

Note in particular, if choosing to draw barplots, the starting value is 0.0 and setting the ylim parameter can lead to some awkward results if 0 is not included in the ylim range. Did you really mean to draw a bar plot in this case?

For up to three groups, the colors are by default "black", "blue" and "red". For more than 3 groups, they are by default rainbow colors with an alpha factor (transparency) of .5.

To make colors semitransparent, set the density to a negative number. See the last example.

Value

Graphic output showing the means + x% confidence intervals for each group. For ci=1.96, and normal data, this will be the 95% confidence region. For ci=1, the 68% confidence region.

These confidence regions are based upon normal theory and do not take into account any skew in the variables. More accurate confidence intervals could be found by resampling.

The results of describeBy are reported invisibly.

See Also

See Also as error.crosses, histBy, scatterHist, error.bars and error.dots

Examples

```
data(sat.act)
#The generic plot of variables by group
error.bars.by( SATV + SATQ ~ gender,data=sat.act)  #formula input
error.bars.by( SATV + SATQ ~ gender,data=sat.act,v.lab=cs(male,female)) #labels
error.bars.by(SATV + SATQ ~ education + gender, data =sat.act) #see below
error.bars.by(sat.act[1:4],sat.act$gender,legend=7) #specification of variables
error.bars.by(sat.act[1:4],sat.act$gender,legend=7,labels=cs(male,female))

#a bar plot
error.bars.by(sat.act[5:6],sat.act$gender,bars=TRUE,labels=c("male","female"),
    main="SAT V and SAT Q by gender",ylim=c(0,800),colors=c("red","blue"),
    legend=5,v.labels=c("SATV","SATQ"))  #draw a barplot
#a bar plot of SAT by age -- not recommended, see the next plot
error.bars.by(SATV + SATQ ~ education,data=sat.act,bars=TRUE,xlab="Education",
   main="95 percent confidence limits of Sat V and Sat Q", ylim=c(0,800),
   v.labels=c("SATV","SATQ"),colors=c("red","blue") )
#a better graph uses points not bars
#use formulat input
  #plot SAT V and SAT Q by education
error.bars.by(SATV + SATQ ~ education,data=sat.act,TRUE, xlab="Education",
    legend=5,labels=colnames(sat.act[5:6]),ylim=c(525,700),
     main="self reported SAT scores by education",
     v.lab =c("HS","in coll", "< 16", "BA/BS", "in Grad", "Grad/Prof"))
#make the cats eyes semi-transparent by specifying a negative density

error.bars.by(SATV + SATQ ~ education,data=sat.act, xlab="Education",
    legend=5,labels=c("SATV","SATQ"),ylim=c(525,700),
     main="self reported SAT scores by education",density=-10,
     v.lab =c("HS","in coll", "< 16", "BA/BS", "in Grad", "Grad/Prof"))

#use labels to specify the 2nd grouping variable, v.lab to specify the first
error.bars.by(SATV  ~ education  + gender,data=sat.act, xlab="Education",
    legend=5,labels=cs(male,female),ylim=c(525,700),
     main="self reported SAT scores by education",density=-10,
     v.lab =c("HS","in coll", "< 16", "BA/BS", "in Grad", "Grad/Prof"),
     colors=c("red","blue"))


#now for a more complicated examples using 25 big 5 items scored into 5 scales
#and showing age trends by decade
#this shows how to convert many levels of a grouping variable (age) into more manageable levels.
data(bfi)   #The Big 5 data
#first create the keys
 keys.list <- list(Agree=c(-1,2:5),Conscientious=c(6:8,-9,-10),
      Extraversion=c(-11,-12,13:15),Neuroticism=c(16:20),Openness = c(21,-22,23,24,-25))
 keys <- make.keys(bfi,keys.list)
 #then create the scores for those older than 10 and less than 80
```

```
 bfis <- subset(bfi,((bfi$age > 10) & (bfi$age < 80)))

 scores <- scoreItems(keys,bfis,min=1,max=6) #set the right limits for item reversals
 #now draw the results by age
#specify the particular colors to use
 error.bars.by(scores$scores,round(bfis$age/10)*10,by.var=TRUE,
      main="BFI age trends",legend=3,labels=colnames(scores$scores),
        xlab="Age",ylab="Mean item score",
        colors=cs(green,yellow,black,red,blue),
        v.labels =cs(10-14,15-24,25-34,35-44,45-54,55-64,65-74))
#show transparency
 error.bars.by(scores$scores,round(bfis$age/10)*10,by.var=TRUE,
      main="BFI age trends",legend=3,labels=colnames(scores$scores),
        xlab="Age",ylab="Mean item score", density=-10,
        colors=cs(green,yellow,black,red,blue),
        v.labels =cs(10-14,15-24,25-34,35-44,45-54,55-64,65-74))
```

`error.crosses` *Plot x and y error bars*

Description

Given two vectors of data (X and Y), plot the means and show standard errors in both X and Y directions.

Usage

```
error.crosses(x,y,labels=NULL,main=NULL,xlim=NULL,ylim= NULL,
xlab=NULL,ylab=NULL,pos=NULL,offset=1,arrow.len=.2,alpha=.05,sd=FALSE,add=FALSE,
colors=NULL,col.arrows=NULL,col.text=NULL,...)
```

Arguments

`x`	A vector of data or summary statistics (from Describe)
`y`	A second vector of data or summary statistics (also from Describe)
`labels`	the names of each pair – defaults to rownames of x
`main`	The title for the graph
`xlim`	xlim values if desired– defaults to min and max mean(x) +/- 2 se
`ylim`	ylim values if desired – defaults to min and max mean(y) +/- 2 se
`xlab`	label for x axis – grouping variable 1
`ylab`	label for y axis – grouping variable 2
`pos`	Labels are located where with respect to the mean?
`offset`	Labels are then offset from this location
`arrow.len`	Arrow length
`alpha`	alpha level of error bars

sd	if sd is TRUE, then draw means +/- 1 sd)
add	if TRUE, overlay the values with a prior plot
colors	What color(s) should be used for the plot character? Defaults to black
col.arrows	What color(s) should be used for the arrows – defaults to colors
col.text	What color(s) should be used for the text – defaults to colors
...	Other parameters for plot

Details

For an example of two way error bars describing the effects of mood manipulations upon positive and negative affect, see https://personality-project.org/revelle/publications/happy-sad-appendix/FIG.A-6.pdf

The second example shows how error crosses can be done for multiple variables where the grouping variable is found dynamically. The errorCircles example shows how to do this in one step.

Author(s)

William Revelle
<revelle@northwestern.edu>

See Also

To draw error bars for single variables error.bars, or by groups error.bars.by, or to find descriptive statistics describe or descriptive statistics by a grouping variable describeBy and statsBy.

A much improved version is now called errorCircles.

Examples

```
#just draw one pair of variables
desc <- describe(attitude)
x <- desc[1,]
y <- desc[2,]
error.crosses(x,y,xlab=rownames(x),ylab=rownames(y))

#now for a bit more complicated plotting
data(bfi)
desc <- describeBy(bfi[1:25],bfi$gender) #select a high and low group
error.crosses(desc$'1',desc$'2',ylab="female scores",
   xlab="male scores",main="BFI scores by gender")
 abline(a=0,b=1)

#do it from summary statistics  (using standard errors)
g1.stats <- data.frame(n=c(10,20,30),mean=c(10,12,18),se=c(2,3,5))
g2.stats <- data.frame(n=c(15,20,25),mean=c(6,14,15),se =c(1,2,3))
error.crosses(g1.stats,g2.stats)

#Or, if you prefer to draw +/- 1 sd. instead of 95% confidence
g1.stats <- data.frame(n=c(10,20,30),mean=c(10,12,18),sd=c(2,3,5))
```

```
g2.stats <- data.frame(n=c(15,20,25),mean=c(6,14,15),sd =c(1,2,3))
error.crosses(g1.stats,g2.stats,sd=TRUE)

#and seem even fancy plotting: This is taken from a study of mood
#four films were given (sad, horror, neutral, happy)
#with a pre and post test
if(require(psychTools)){

data(psychTools::affect)
colors <- c("black","red","green","blue")
films <- c("Sad","Horror","Neutral","Happy")
affect.mat <- describeBy(psychTools::affect[10:17],psychTools::affect$Film,mat=TRUE)
 error.crosses(affect.mat[c(1:4,17:20),],affect.mat[c(5:8,21:24),],
    labels=films[affect.mat$group1],xlab="Energetic Arousal",
     ylab="Tense Arousal",colors =
     colors[affect.mat$group1],pch=16,cex=2)
}
```

`error.dots` *Show a dot.chart with error bars for different groups or variables*

Description

Yet one more of the graphical ways of showing data with error bars for different groups. A dot.chart with error bars for different groups or variables is found using from `describe`, `describeBy`, `statsBy`, `corCi`, `corr.test` or data from `bestScales`.

Usage

```
error.dots(x=NULL, var = NULL, se = NULL, group = NULL, sd = FALSE, effect=NULL,
stats=NULL, head = 12, tail = 12, sort = TRUE, decreasing = TRUE, main = NULL,
 alpha = 0.05, eyes = FALSE,items=FALSE, min.n = NULL, max.labels = 40, labels = NULL,
 label.width=NULL, select=NULL,
 groups = NULL, gdata = NULL, cex =  par("cex"),  pt.cex = cex, pch = 21, gpch = 21,
 bg = par("bg"), fg=par("fg"), color = par("fg"), gcolor = par("fg"),
 lcolor = "gray", xlab = NULL, ylab = NULL, xlim = NULL,add=FALSE,order=NULL, ...)
```

Arguments

`x`	A data frame or matrix of raw data, or the resulting object from `describe`, `describeBy`, `statsBy`, `bestScales`, `corCi`, `corr.test`, or `cohen.d`
`var`	The variable to show (particularly if doing describeBy or StatsBy plots).
`se`	Source of a standard error
`group`	A grouping variable, if desired. Will group the data on group for one variable (var)
`sd`	if FALSE, confidence intervals in terms of standard errors, otherwise draw one standard deviation

effect	Should the data be compared to a specified group (with mean set to 0) in effect size units?
stats	A matrix of means and se to use instead of finding them from the data
head	The number of largest values to report
tail	The number of smallest values to report
sort	Sort the groups/variables by value
decreasing	Should they be sorted in increasing or decreasing order (from top to bottom)
main	The caption for the figure
alpha	p value for confidence intervals
eyes	Draw catseyes for error limits
items	If showing results from best scales, show the items for a specified dv
min.n	If using describeBy or statsBy, what should be the minimum sample size to draw
max.labels	Length of labels (truncate after this value)
labels	Specify the labels versus find them from the row names
label.width	Truncate after labels.width
select	Scale the plot for all the variables, but just show the select variables
groups	ignored: to be added eventually
gdata	ignored
cex	The standard meaning of cex for graphics
pt.cex	ignored
pch	Plot character of the mean
gpch	ignored
bg	background color (of the dots showing the means)
fg	foreground color (of the line segments)
color	Color of the text labels
gcolor	ignored
lcolor	ignored until groups are implemented
xlab	Label the x axis, if NULL, the variable name is used
ylab	If NULL, then the group rownames are used
xlim	If NULL, then calculated to show nice values
add	If TRUE, will add the plot to a previous plot (e.g., from dotchart)
order	if sort=TRUE, if order is NULL, sort on values, otherwise, if order is returned from a previous figure, use that order.
...	And any other graphic parameters we have forgotten

Details

Adapted from the dot.chart function to include error bars and to use the output of describe, describeBy, statsBy, fa, bestScales or cohen.d. To speed up multiple plots, the function can work from the output of a previous run. Thus describeBy will be done and the results can be show for multiple variables.

If using the add=TRUE option to add an error.dots plot to a dotplot, note that the order of variables in dot plots goes from last to first (highest y value is actually the last value in a vector.) Also note that the xlim parameter should be set to make sure the plots line up correctly.

Value

Returns (invisibily) either a describeBy or describe object as well as the order if sorted

Author(s)

William Revelle

References

Used in particular for showing https://sapa-project.org output.

See Also

describe, describeBy, or statsBy as well as error.bars, error.bars.by, statsBy, bestScales or cohen.d

Examples

```
temp <- error.dots(bfi[1:25],sort=TRUE,
xlab="Mean score for the item, sorted by difficulty")
error.dots(bfi[1:25],sort=TRUE, order=temp$order,
 add=TRUE, eyes=TRUE) #over plot with eyes

#error.dots(psychTools::ability,eyes=TRUE, xlab="Mean score for the item")

cd <- cohen.d(bfi[1:26],"gender")
temp <- error.dots(cd, select=c(1:15,21:25),head=12,tail=13,
      main="Cohen d and confidence intervals of BFI by gender")
error.dots(cd,select=c(16:20),head=13,tail=12,col="blue",add=TRUE,fg="red" ,main="")
abline(v=0)
#now show cis for correlations
R <- corCi(attitude,plot=FALSE)
error.dots(R, sort=FALSE)
#the asymmetric case
R <- corr.test(attitude[,1:2],attitude[,3:7])
error.dots(R, sort=FALSE)
```

errorCircles *Two way plots of means, error bars, and sample sizes*

Description

Given a matrix or data frame, data, find statistics based upon a grouping variable and then plot x and y means with error bars for each value of the grouping variable. If the data are paired (e.g. by gender), then plot means and error bars for the two groups on all variables.

Usage

```
errorCircles(x, y, data, ydata = NULL, group=NULL, paired = FALSE, labels = NULL,
main = NULL, xlim = NULL, ylim = NULL, xlab = NULL, ylab = NULL,add=FALSE, pos = NULL,
 offset = 1, arrow.len = 0.2, alpha = 0.05, sd = FALSE, bars = TRUE, circles = TRUE,
 colors=NULL,col.arrows=NULL,col.text=NULL,circle.size=1, ...)
```

Arguments

x	The x variable (by name or number) to plot
y	The y variable (name or number) to plot
data	The matrix or data.frame to use for the x data
ydata	If plotting data from two data.frames, then the y variable of the ydata frame will be used.
group	If specified, then statsBy is called first to find the statistics by group
paired	If TRUE, plot all x and y variables for the two values of the grouping variable.
labels	Variable names
main	Main title for plot
xlim	xlim values if desired– defaults to min and max mean(x) +/- 2 se
ylim	ylim values if desired – defaults to min and max mean(y) +/- 2 se
xlab	label for x axis – grouping variable 1
ylab	label for y axis – grouping variable 2
add	If TRUE, add to the prior plot
pos	Labels are located where with respect to the mean?
offset	Labels are then offset from this location
arrow.len	Arrow length
alpha	alpha level of error bars
sd	if sd is TRUE, then draw means +/- 1 sd)
bars	Should error.bars be drawn for both x and y
circles	Should circles representing the relative sample sizes be drawn?
colors	Plot the points using colors – default is black

col.text	What color for the text labels (defaults to colors)
col.arrows	What color should the arrows and circles be? Defaults to colors
circle.size	A scaling parameter for the error.circles. Defaults to 1, but can be adjusted downwards to make them less instrusive.
...	Other parameters for plot

Details

When visualizing the effect of an experimental manipulation or the relationship of multiple groups, it is convenient to plot their means as well as their confidence regions in a two dimensional space.

The diameter of the enclosing circle (ellipse) scales as 1/sqrt(N) * the maximum standard error of all variables. That is to say, the area of the ellipse reflects sample size.

Value

If the group variable is specified, then the statistics from statsBy are (invisibly) returned.

Note

Basically this is a combination (and improvement) of statsBy with error.crosses. Can also serve some of the functionality of error.bars.by (see the last example).

Author(s)

William Revelle

See Also

statsBy, describeBy, error.crosses

Examples

```
#BFI scores for males and females
errorCircles(1:25,1:25,data=bfi,group="gender",paired=TRUE,ylab="female scores",
      xlab="male scores",main="BFI scores by gender")
 abline(a=0,b=1)
#drop the circles since all samples are the same sizes
errorCircles(1:25,1:25,data=bfi,group="gender",paired=TRUE,circles=FALSE,
     ylab="female scores",xlab="male scores",main="BFI scores by gender")
 abline(a=0,b=1)
 if(require(psychTools)) {

 data(psychTools::affect)
colors <- c("black","red","white","blue")
films <- c("Sad","Horror","Neutral","Happy")
affect.stats <- errorCircles("EA2","TA2",data=psychTools::affect[-c(1,20)],
group="Film",labels=films,
      xlab="Energetic Arousal",ylab="Tense Arousal",ylim=c(10,22),xlim=c(8,20),
      pch=16,cex=2,colors=colors, main ="EA and TA pre and post affective movies")
#now, use the stats from the prior run
```

```
errorCircles("EA1","TA1",data=affect.stats,labels=films,pch=16,cex=2,colors=colors,add=TRUE)

}
#show sample size with the size of the circles
errorCircles("SATV","SATQ",sat.act,group="education")

#Can also provide error.bars.by functionality
 errorCircles(2,5,group=2,data=sat.act,circles=FALSE,pch=16,colors="blue",
      ylim= c(200,800),main="SATV by education",labels="")
 #just do the breakdown and then show the points
# errorCircles(3,5,group=3,data=sat.act,circles=FALSE,pch=16,colors="blue",
#         ylim= c(200,800),main="SATV by age",labels="",bars=FALSE)
```

---

esem *Perform and Exploratory Structural Equation Model (ESEM) by using factor extension techniques*

Description

Structural Equation Modeling (SEM) is a powerful tool for confirming multivariate structures and is well done by the lavaan, sem, or OpenMx packages. Because they are confirmatory, SEM models test specific models. Exploratory Structural Equation Modeling (ESEM), on the other hand, takes a more exploratory approach. By using factor extension, it is possible to extend the factors of one set of variables (X) into the variable space of another set (Y). Using this technique, it is then possible to estimate the correlations between the two sets of latent variables, much the way normal SEM would do. Based upon exploratory factor analysis (EFA) this approach provides a quick and easy approach to do exploratory structural equation modeling.

Usage

```
esem(r, varsX, varsY, nfX = 1, nfY = 1, n.obs = NULL, fm = "minres",
      rotate = "oblimin", rotateY="oblimin", plot = TRUE, cor = "cor",
       use = "pairwise",weight=NULL,smooth=TRUE, ...)
esemDiagram(esem=NULL,labels=NULL,cut=.3,errors=FALSE,simple=TRUE,
 regression=FALSE,lr=TRUE, digits=1,e.size=.1,adj=2,
     main="Exploratory Structural Model", ...)
esem.diagram(esem=NULL,labels=NULL,cut=.3,errors=FALSE,simple=TRUE,
 regression=FALSE,lr=TRUE, digits=1,e.size=.1,adj=2,
     main="Exploratory Structural Model", ...) #deprecated
cancorDiagram(fit, cut=.1,digits=2, nfX = NULL, nfY = NULL,simple=FALSE,e.size=.1,
     main="Canonical Correlation",...)
interbattery(r, varsX, varsY, nfX = 1, nfY = 1, n.obs = NULL,cor = "cor",
       use = "pairwise",weight=NULL)
```

Arguments

r	A correlation matrix or a raw data matrix suitable for factor analysis
varsX	The variables defining set X
varsY	The variables defining set Y
nfX	The number of factors to extract for the X variables
nfY	The number of factors to extract for the Y variables
n.obs	Number of observations (needed for eBIC and chi square), can be ignored.
fm	The factor method to use, e.g., "minres", "mle" etc. (see fa for details)
rotate	Which rotation to use. (see fa for details)
rotateY	Which rotatation to use for the Y variables.
plot	If TRUE, draw the esemDiagram
cor	What options for to use for correlations (see fa for details)
use	"pairwise" for pairwise complete data, for other options see cor
weight	Weights to apply to cases when finding wt.cov
smooth	Should correlation matrices be smoothed?
...	other parameters to pass to fa or to esemDiagram functions.
esem	The object returned from esem and passed to esemDiagram
fit	The object returned by lmCor with canonical variates
labels	Variable labels
cut	Loadings with abs(loading) > cut will be shown
simple	Only the biggest loading per item is shown
errors	include error estimates (as arrows)
e.size	size of ellipses (adjusted by the number of variables)
digits	Round coefficient to digits
adj	loadings are adjusted by factor number mod adj to decrease likelihood of overlap
main	Graphic title, defaults to "Exploratory Structural Model"
lr	draw the graphic left to right (TRUE) or top to bottom (FALSE)
regression	Not yet implemented

Details

Factor analysis as implemented in fa attempts to summarize the covariance (correlational) structure of a set of variables with a small set of latent variables or “factors". This solution may be ‘extended’ into a larger space with more variables without changing the original solution (see fa.extension. Similarly, the factors of a second set of variables (the Y set) may be extended into the original (X) set. Doing so allows two independent measurement models, a measurement model for X and a measurement model for Y. These two sets of latent variables may then be correlated for an Exploratory Structural Equation Model. (This is exploratory because it is based upon exploratory factor analysis (EFA) rather than a confirmatory factor model (CFA) using more traditional Structural Equation Modeling packages such as sem, lavaan, or Mx.)

Although the output seems very similar to that of a normal EFA using fa, it is actually two independent factor analyses (of the X and the Y sets) that are then mutually extended into each other. That is, the loadings and structure matrices from sets X and Y are merely combined, and the correlations between the two sets of factors are found.

Interbattery factor analysis was developed by Tucker (1958) as a way of comparing the factors in common to two batteries of tests. (Currently under development and not yet complete). Using some straight forward linear algebra It is easy to find the factors of the intercorrelations between the two sets of variables. This does not require estimating communalities and is highly related to the procedures of canonical correlation.

The difference between the esem and the interbattery approach is that the first factors the X set and then relates those factors to factors of the Y set. Interbattery factor analysis, on the other hand, tries to find one set of factors that links both sets but is still distinct from factoring both sets together.

Value

communality	The amount of variance in each of the X and Y variables accounted for by the total model.
sumsq	The amount of variance accounted for by each factor – independent of the other factors.
dof	Degrees of freedom of the model.
esem.dof	Alternatively consider the df1 for the X model and df2 for the Y model. esem.dof = df1 + df2
null.dof	Degrees of freedom of the null model (the correlation matrix)
ENull	chi square of the null model
chi	chi square of the model. This is found by examining the size of the residuals compared to their standard error.
rms	The root mean square of the residuals.
nh	Harmonic sample size if using min.chi for factor extraction.
EPVAL	Probability of the Emprical Chi Square given the hypothesis of an identity matrix.
crms	Adjusted root mean square residual
EBIC	When normal theory fails (e.g., in the case of non-positive definite matrices), it useful to examine the empirically derived EBIC based upon the empirical χ^2 - 2 df.
ESABIC	Sample size adjusted empirical BIC
fit	sum of squared residuals versus sum of squared original values
fit.off	fit applied to the off diagonal elements
sd	standard deviation of the residuals
factors	Number of factors extracted
complexity	Item complexity
n.obs	Number of total observations
loadings	The factor pattern matrix for the combined X and Y factors

Structure	The factor structure matrix for the combined X and Y factors
loadsX	Just the X set of loadings (pattern) without the extension variables.
loadsY	Just the Y set of loadings (pattern) without the extension variables.
PhiX	The correlations of the X factors
PhiY	the correlations of the Y factors
Phi	the correlations of the X and Y factors within the selves and across sets.
fm	The factor method used
fx	The complete factor analysis output for the X set
fy	The complete factor analysis output for the Y set
residual	The residual correlation matrix (R - model). May be examined by a call to residual().
Call	Echo back the original call to the function.
model	model code for SEM and for lavaan to do subsequent confirmatory modeling

Note

Developed September, 2016, revised December, 2018 to produce code for lavaan and sem from the esem and esemDiagram functions. Suggestions or comments are most welcome.

This is clearly an exploratory approach and the degrees of freedom for the model are unclear.

In December 2023, I added the canCorDiagram function to help understand canonical correlations.

Author(s)

William Revelle

References

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at https://personality-project.org/r/book/

Tucker, Ledyard (1958) An inter-battery method of factor analysis, Psychometrika, 23, 111-136.

See Also

principal for principal components analysis (PCA). PCA will give very similar solutions to factor analysis when there are many variables. The differences become more salient as the number variables decrease. The PCA and FA models are actually very different and should not be confused. One is a model of the observed variables, the other is a model of latent variables.

lmCor which does a canonical correlation between the X and Y sets of variables.

irt.fa for Item Response Theory analyses using factor analysis, using the two parameter IRT equivalent of loadings and difficulties.

VSS will produce the Very Simple Structure (VSS) and MAP criteria for the number of factors, nfactors to compare many different factor criteria.

ICLUST will do a hierarchical cluster analysis alternative to factor analysis or principal components analysis.

predict.psych to find predicted scores based upon new data, fa.extension to extend the factor solution to new variables, omega for hierarchical factor analysis with one general factor. fa.multi for hierarchical factor analysis with an arbitrary number of higher order factors.

faRegression to do multiple regression from factor analysis solutions.

fa.sort will sort the factor loadings into echelon form. fa.organize will reorganize the factor pattern matrix into any arbitrary order of factors and items.

KMO and cortest.bartlett for various tests that some people like.

factor2cluster will prepare unit weighted scoring keys of the factors that can be used with scoreItems.

fa.lookup will print the factor analysis loadings matrix along with the item "content" taken from a dictionary of items. This is useful when examining the meaning of the factors.

anova.psych allows for testing the difference between two (presumably nested) factor models .

Examples

```
#make up a sem like problem using sim.structure
fx <-matrix(c( .9,.8,.6,rep(0,4),.6,.8,-.7),ncol=2)
fy <- matrix(c(.6,.5,.4),ncol=1)
rownames(fx) <- c("V","Q","A","nach","Anx")
rownames(fy)<- c("gpa","Pre","MA")
Phi <-matrix( c(1,0,.7,.0,1,.7,.7,.7,1),ncol=3)
gre.gpa <- sim.structural(fx,Phi,fy)
print(gre.gpa)

#now esem it:
example <- esem(gre.gpa$model,varsX=1:5,varsY=6:8,nfX=2,nfY=1,n.obs=1000,plot=FALSE)
example
esemDiagram(example,simple=FALSE)
#compare this to the canonical solution
mod <- lmCor(y=6:8, x =1:5, data=gre.gpa$model, plot=FALSE)
cancorDiagram(mod)  #does not work because of imaginary roots
#compare two alternative solutions to the first 2 factors of the neo.
#solution 1 is the normal 2 factor solution.
#solution 2 is an esem with 1 factor for the first 6 variables, and 1 for the second 6.

if(require(psychTools)){
f2 <- fa(psychTools::neo[1:12,1:12],2)
es2 <- esem(psychTools::neo,1:6,7:12,1,1)
summary(f2)
summary(es2)
fa.congruence(f2,es2)
}
interbattery(Thurstone.9,1:4,5:9,2,2) #compare to the solution of Tucker. We are not there yet.
```

fa *Exploratory Factor analysis using MinRes (minimum residual) as well as EFA by Principal Axis, Weighted Least Squares or Maximum Likelihood*

Description

Among the many ways to do latent variable exploratory factor analysis (EFA), one of the better is to use Ordinary Least Squares (OLS) to find the minimum residual (minres) solution. This produces solutions very similar to maximum likelihood even for badly behaved matrices. A variation on minres is to do weighted least squares (WLS). Perhaps the most conventional technique is principal axes (PAF). An eigen value decomposition of a correlation matrix is done and then the communalities for each variable are estimated by the first n factors. These communalities are entered onto the diagonal and the procedure is repeated until the sum(diag(r)) does not vary. Yet another estimate procedure is maximum likelihood. For well behaved matrices, maximum likelihood factor analysis (either in the fa or in the factanal function) is probably preferred. Bootstrapped confidence intervals of the loadings and interfactor correlations are found by fa with n.iter > 1.

Usage

```
fa(r, nfactors=1, n.obs = NA, n.iter=1, rotate="oblimin", scores="regression",
 residuals=FALSE, SMC=TRUE, covar=FALSE, missing=FALSE, impute="none",
 min.err = 0.001,  max.iter = 50, symmetric=TRUE, warnings=TRUE, fm="minres",
  alpha=.1, p=.05, oblique.scores=FALSE, np.obs=NULL, use="pairwise", cor="cor",
  correct=.5, weight=NULL, n.rotations=20, hyper=.15, smooth=TRUE,Grice=FALSE,...)

fac(r,nfactors=1,n.obs = NA, rotate="oblimin", scores="tenBerge", residuals=FALSE,
  SMC=TRUE, covar=FALSE, missing=FALSE, impute="median", min.err = 0.001,
 max.iter=50, symmetric=TRUE, warnings=TRUE, fm="minres", alpha=.1,
 oblique.scores=FALSE, np.obs=NULL, use="pairwise", cor="cor", correct=.5,
 weight=NULL, n.rotations=1, hyper=.15, smooth=TRUE,Grice=FALSE,...)

fa.pooled(datasets,nfactors=1,rotate="oblimin", scores="regression",
 residuals=FALSE, SMC=TRUE, covar=FALSE, missing=FALSE,impute="median", min.err =
 .001, max.iter=50,symmetric=TRUE, warnings=TRUE, fm="minres", alpha=.1,
 p =.05, oblique.scores=FALSE,np.obs=NULL, use="pairwise", cor="cor",
 correct=.5, weight=NULL,Grice=FALSE,...)

fa.sapa(r,nfactors=1, n.obs = NA,n.iter=1,rotate="oblimin",scores="regression",
 residuals=FALSE, SMC=TRUE, covar=FALSE, missing=FALSE, impute="median",
 min.err = .001, max.iter=50,symmetric=TRUE,warnings=TRUE, fm="minres", alpha=.1,
 p =.05, oblique.scores=FALSE, np.obs=NULL, use="pairwise", cor="cor", correct=.5,
 weight=NULL, frac=.1,...)
```

Arguments

r	A correlation or covariance matrix or a raw data matrix. If raw data, the correlation matrix will be found using pairwise deletion. If covariances are supplied, they will be converted to correlations unless the covar option is TRUE.
nfactors	Number of factors to extract, default is 1
n.obs	Number of observations used to find the correlation matrix if using a correlation matrix. Used for finding the goodness of fit statistics. Must be specified if using a correlaton matrix and finding confidence intervals.
np.obs	The pairwise number of observations. Used if using a correlation matrix and asking for a minchi solution.
rotate	"none", "varimax", "quartimax", "bentlerT", "equamax", "varimin", "geominT" and "bifactor" are orthogonal rotations. "Promax", "promax", "oblimin", "simplimax", "bentlerQ, "geominQ" and "biquartimin" and "cluster" are possible oblique transformations of the solution. The default is to do a oblimin transformation, although versions prior to 2009 defaulted to varimax. SPSS seems to do a Kaiser normalization before doing Promax, this is done here by the call to "promax" which does the normalization before calling Promax in GPArotation.
n.rotations	Number of random starts for the rotations. Defaults to 20. (see Note)
n.iter	Number of bootstrap interations to do in fa or fa.poly
residuals	Should the residual matrix be shown
scores	the default="regression" finds factor scores using regression. Alternatives for estimating factor scores include simple regression ("Thurstone"), correlation preserving ("tenBerge") as well as "Anderson" and "Bartlett" using the appropriate algorithms (factor.scores). Although scores="tenBerge" is probably preferred for most solutions, it will lead to problems with some improper correlation matrices.
SMC	Use squared multiple correlations (SMC=TRUE) or use 1 as initial communality estimate. Try using 1 if imaginary eigen values are reported. If SMC is a vector of length the number of variables, then these values are used as starting values in the case of fm='pa'.
covar	if covar is TRUE, factor the covariance matrix, otherwise factor the correlation matrix
missing	if scores are TRUE, and missing=TRUE, then impute missing values using either the median or the mean. Specifying impute="none" will not impute data points and thus will have some missing data in the factor scores.
impute	"median" or "mean" values are used to replace missing values
min.err	Iterate until the change in communalities is less than min.err
max.iter	Maximum number of iterations for convergence
symmetric	symmetric=TRUE forces symmetry by just looking at the lower off diagonal values
hyper	Count number of (absolute) loadings less than hyper.
warnings	warnings=TRUE => warn if number of factors is too many

fm	Factoring method fm="minres" will do a minimum residual as will fm="uls". Both of these use a first derivative. fm="ols" differs very slightly from "minres" in that it minimizes the entire residual matrix using an OLS procedure but uses the empirical first derivative. This will be slower. fm="wls" will do a weighted least squares (WLS) solution, fm="gls" does a generalized weighted least squares (GLS), fm="pa" will do the principal factor solution, fm="ml" will do a maximum likelihood factor analysis. fm="minchi" will minimize the sample size weighted chi square when treating pairwise correlations with different number of subjects per pair. fm ="minrank" will do a minimum rank factor analysis. fm="alpha" will do alpha factor analysis as described in Kaiser and Coffey (1965). "old.min" will do minimal residual the way it was done prior to April, 2017 (see discussion below).
alpha	alpha level for the confidence intervals for RMSEA
p	if doing iterations to find confidence intervals, what probability values should be found for the confidence intervals
oblique.scores	When factor scores are found, should they be based on the structure matrix (default) or the pattern matrix (oblique.scores=TRUE). Now it is always false. If you want oblique factor scores, use tenBerge.
weight	If not NULL, a vector of length n.obs that contains weights for each observation. The NULL case is equivalent to all cases being weighted 1.
use	How to treat missing data, use="pairwise" is the default". See cor for other options.
cor	How to find the correlations: "cor" is Pearson", "cov" is covariance, "tet" is tetrachoric, "poly" is polychoric, "mixed" uses mixedCor for a mixture of tetrachorics, polychorics, Pearsons, biserials, and polyserials, Yuleb is Yulebonett, Yuleq and YuleY are the obvious Yule coefficients as appropriate
correct	When doing tetrachoric, polycoric, or mixed cor, how should we treat empty cells. (See the discussion in the help for tetrachoric.)
frac	The fraction of data to sample n.iter times if showing stability across sample sizes
datasets	A list of replication data sets to analyse in fa.pooled. All need to have the same number of variables.
smooth	By default, improper correlations matrices are smoothed. This is not necessary for factor extraction using fm="pa" or fm="minres", but is needed to give some of the goodness of fit tests. (see notes)
Grice	Grice=TRUE to find factor score indeterminancy using the Grice approach. FALSE (the default) to use procedures discussed by Nicewander (2020). See notes as well as the discussion in fsi.
...	additional parameters, specifically, keys may be passed if using the target rotation, or delta if using geominQ, or whether to normalize if using Varimax

Details

Factor analysis is an attempt to approximate a correlation or covariance matrix with one of lesser rank. The basic model is that ${}_nR_n \approx_n F_{kk}F'_n + U^2$ where k is much less than n. There are many

ways to do factor analysis, and maximum likelihood procedures are probably the most commonly preferred (see factanal). The existence of uniquenesses is what distinguishes factor analysis from principal components analysis (e.g., principal). If variables are thought to represent a "true" or latent part then factor analysis provides an estimate of the correlations with the latent factor(s) representing the data. If variables are thought to be measured without error, then principal components provides the most parsimonious description of the data.

Factor loadings will be smaller than component loadings for the later reflect unique error in each variable. The off diagonal residuals for a factor solution will be superior (smaller) than that of a component model. Factor loadings can be thought of as the asymptotic component loadings as the number of variables loading on each factor increases. See Widaman, 1993, 2018, for very thorough discussions of why the factor model and the component model should not be confused.

The fa function will do factor analyses using one of six different algorithms: minimum residual (minres, aka ols, uls), principal axes, alpha factoring, weighted least squares, minimum rank, or maximum likelihood.

Principal axes factor analysis has a long history in exploratory analysis and is a straightforward procedure. Successive eigen value decompositions are done on a correlation matrix with the diagonal replaced with diag (FF') until $\sum(diag(FF'))$ does not change (very much). The current limit of max.iter =50 seems to work for most problems, but the Holzinger-Harmon 24 variable problem needs about 203 iterations to converge for a 5 factor solution.

Not all factor programs that do principal axes do iterative solutions. The example from the SAS manual (Chapter 33) is such a case. To achieve that solution, it is necessary to specify that the max.iter = 1. Comparing that solution to an iterated one (the default) shows that iterations improve the solution.

In addition, fm="mle" produces even better solutions for this example. Both the RMSEA and the root mean square of the residuals are smaller than the fm="pa" solution.

However, simulations of multiple problem sets suggest that fm="pa" tends to produce slightly smaller residuals while having slightly larger RMSEAs than does fm="minres" or fm="mle". That is, the sum of squared residuals for fm="pa" seem to be slightly smaller than those found using fm="minres" but the RMSEAs are slightly worse when using fm="pa". That is to say, the "true" minimal residual is probably found by fm="pa".

Following extensive correspondence with Hao Wu and Mikko Ronkko, in April, 2017 the derivative of the minres (and uls) fitting was modified. This leads to slightly smaller residuals (appropriately enough for a method claiming to minimize them) than the prior procedure. For consistency with prior analyses, "old.min" was added to give these slightly larger residuals. The differences between old.min and the newer "minres" and "ols" solutions are at the third to fourth decimal, but none the less, are worth noting. For comparison purposes, the fm="ols" uses empirical first derivatives, while uls and minres use equation based first derivatives. The results seem to be identical, but the minres and uls solutions require fewer iterations for larger problems and are faster. Thanks to Hao Wu for some very thoughtful help.

Although usually these various algorithms produce equivalent results, there are several data sets included that show large differences between the methods. Schutz produces Heywood and super Heywood cases, blant leads to very different solutions. In particular, the minres solution produces smaller residuals than does the mle solution, and the factor.congruence coefficients show very different solutions.

A very strong argument against using MLE is found in the chapter by MacCallum, Brown and Cai (2007) who show that OLS approaches produce equivalent solutions most of the time, and better

solutions some of the time. This particularly in the case of models with some unmodeled small factors. (See sim.minor to generate such data.)

Principal axes may be used in cases when maximum likelihood solutions fail to converge, although fm="minres" will also do that and tends to produce better (smaller RMSEA) solutions.

The fm="minchi" option is a variation on the "minres" (ols) solution and minimizes the sample size weighted residuals rather than just the residuals. This was developed to handle the problem of data with large variations in the number of observations in pairwise correlations produced by using the Massively Missing Completely at Random (MMCAR) data collection method. (This is the procedure used in the SAPA project.)

A traditional problem in factor analysis was to find the best estimate of the original communalities in order to speed up convergence. Using the Squared Multiple Correlation (SMC) for each variable will underestimate the original communalities, using 1s will over estimate. By default, the SMC estimate is used. Note that in the case of non-invertible matrices, the pseudo-inverse is found so smcs are still estimated. In either case, iterative techniques will tend to converge on a stable solution. If, however, a solution fails to be achieved, it is useful to try again using ones (SMC =FALSE). Alternatively, a vector of starting values for the communalities may be specified by the SMC option.

The iterated principal axes algorithm does not attempt to find the best (as defined by a maximum likelihood criterion) solution, but rather one that converges rapidly using successive eigen value decompositions. The maximum likelihood criterion of fit and the associated chi square value are reported, and will be (slightly) worse than that found using maximum likelihood procedures.

The minimum residual (minres) solution is an unweighted least squares solution that takes a slightly different approach. It uses the optim function and adjusts the diagonal elements of the correlation matrix to mimimize the squared residual when the factor model is the eigen value decomposition of the reduced matrix. MINRES and PA will both work when ML will not, for they can be used when the matrix is singular. Although before the change in the derivative, the MINRES solution was slightly more similar to the ML solution than is the PA solution. With the change in the derivative of the minres fit, the minres, pa and uls solutions are practically identical. To a great extent, the minres and wls solutions follow ideas in the factanal function with the change in the derivative.

The weighted least squares (wls) solution weights the residual matrix by 1/ diagonal of the inverse of the correlation matrix. This has the effect of weighting items with low communalities more than those with high communalities.

The generalized least squares (gls) solution weights the residual matrix by the inverse of the correlation matrix. This has the effect of weighting those variables with low communalities even more than those with high communalities.

The maximum likelihood solution takes yet another approach and finds those communality values that minimize the chi square goodness of fit test. The fm="ml" option provides a maximum likelihood solution following the procedures used in factanal but does not provide all the extra features of that function. It does, however, produce more expansive output.

The minimum rank factor model (MRFA) roughly follows ideas by Shapiro and Ten Berge (2002) and Ten Berge and Kiers (1991). It makes use of the glb.algebraic procedure contributed by Andreas Moltner. MRFA attempts to extract factors such that the residual matrix is still positive semi-definite.

Alpha factor analysis finds solutions based upon a correlation matrix corrected for communalities and then rescales these to the original correlation matrix. This procedure is described by Kaiser and Coffey, 1965.

Test cases comparing the output to SPSS suggest that the PA algorithm matches what SPSS calls uls, and that the wls solutions are equivalent in their fits. The wls and gls solutions have slightly larger eigen values, but slightly worse fits of the off diagonal residuals than do the minres or maximum likelihood solutions. Comparing the results to the examples in Harman (1976), the PA solution with no iterations matches what Harman calls Principal Axes (as does SAS), while the iterated PA solution matches his minres solution. The minres solution found in psych tends to have slightly smaller off diagonal residuals (as it should) than does the iterated PA solution.

Although for items, it is typical to find factor scores by scoring the salient items (using, e.g., `scoreItems`) factor scores can be estimated by regression as well as several other means. There are multiple approaches that are possible (see Grice, 2001) and one taken here was developed by ten-Berge et al. (see `factor.scores`). The alternative, which will match factanal is to find the scores using regression – Thurstone's least squares regression where the weights are found by $W = R^{-1}S$ and R is the correlation matrix of the variables and S is the structure matrix. Then, factor scores are just $Fs = XW$.

In the oblique case, the factor loadings are referred to as Pattern coefficients and are related to the Structure coefficients by $S = P\Phi$ and thus $P = S\Phi^{-1}$. When estimating factor scores, `fa` and `factanal` differ in that `fa` finds the factors from the Structure matrix while `factanal` seems to do it from the Pattern matrix. Thus, although in the orthogonal case, fa and factanal agree perfectly in their factor score estimates, they do not agree in the case of oblique factors. Setting oblique.scores = TRUE will produce factor score estimate that match those of `factanal`.

It is sometimes useful to extend the factor solution to variables that were not factored. This may be done using `fa.extension`. Factor extension is typically done in the case where some variables were not appropriate to factor, but factor loadings on the original factors are still desired. Factor extension is a very powerful procedure in that it allows one to find the factor-criterion correlations without using factor scores.

For dichotomous items or polytomous items, it is recommended to analyze the `tetrachoric` or `polychoric` correlations rather than the Pearson correlations. This may be done by specifying cor="poly" or cor="tet" or cor="mixed" if the data have a mixture of dichotomous, polytomous, and continous variables.

Analysis of dichotomous or polytomous data may also be done by using `irt.fa` or simply setting the cor="poly" option. In the first case, the factor analysis results are reported in Item Response Theory (IRT) terms, although the original factor solution is returned in the results. In the later case, a typical factor loadings matrix is returned, but the tetrachoric/polychoric correlation matrix and item statistics are saved for reanalysis by `irt.fa`. (See also the `mixed.cor` function to find correlations from a mixture of continuous, dichotomous, and polytomous items.)

Of the various rotation/transformation options, varimax, Varimax, quartimax, bentlerT, geominT, and bifactor do orthogonal rotations. Promax transforms obliquely with a target matix equal to the varimax solution. oblimin, quartimin, simplimax, bentlerQ, geominQ and biquartimin are oblique transformations. Most of these are just calls to the GPArotation package. The "cluster" option does a targeted rotation to a structure defined by the cluster representation of a varimax solution. With the optional "keys" parameter, the "target" option will rotate to a target supplied as a keys matrix. (See `target.rot`.)

oblimin is implemented in GPArotation by a call to quartimin with delta=0. This leads to confusion when people refer to quartimin solutions.

It is important to note a dirty little secret about factor rotation. That is the problem of local minima. Multiple restarts of the rotation algorithms are strongly encouraged (see Nguyen and Waller, N.

G. 2021). Until 2.6.4, I ignored this advice and just set n.rotations=1 as the default. However, an example from Keith Widaman convinced me otherwise. The default is now 20 at a small cost of increased run time.

Two additional target rotation options are available through calls to GPArotation. These are the targetQ (oblique) and targetT (orthogonal) target rotations of Michael Browne. See `target.rot` for more documentation.

The "bifactor" rotation implements the Jennrich and Bentler (2011) bifactor rotation by calling the GPForth function in the GPArotation package and using two functions adapted from the MatLab code of Jennrich and Bentler. This seems to have a problem with local minima and multiple starting values should be used.

There are two varimax rotation functions. One, Varimax, in the GPArotation package does not by default apply Kaiser normalization. The other, varimax, in the stats package, does. It appears that the two rotation functions produce slightly different results even when normalization is set. For consistency with the other rotation functions, Varimax is probably preferred.

The rotation matrix (rot.mat) is returned from all of these options. This is the inverse of the Th (theta?) object returned by the GPArotation package. The correlations of the factors may be found by $\Phi = \theta' \theta$

There are two ways to handle dichotomous or polytomous responses: `fa` with the cor="poly" option which will return the tetrachoric or polychoric correlation matrix, as well as the normal factor analysis output, and `irt.fa` which returns a two parameter irt analysis as well as the normal fa output.

When factor analyzing items with dichotomous or polytomous responses, the `irt.fa` function provides an Item Response Theory representation of the factor output. The factor analysis results are available, however, as an object in the irt.fa output.

`fa.poly` is deprecated, for its functioning is matched by setting cor="poly". It will produce normal factor analysis output but also will save the polychoric matrix (rho) and items difficulties (tau) for subsequent irt analyses. `fa.poly` will, by default, find factor scores if the data are available. The correlations are found using either `tetrachoric` or `polychoric` and then this matrix is factored. Weights from the factors are then applied to the original data to estimate factor scores.

The function `fa` will repeat the analysis n.iter times on a bootstrapped sample of the data (if they exist) or of a simulated data set based upon the observed correlation matrix. The mean estimate and standard deviation of the estimate are returned and will print the original factor analysis as well as the alpha level confidence intervals for the estimated coefficients. The bootstrapped solutions are rotated towards the original solution using target.rot. The factor loadings are z-transformed, averaged and then back transformed. This leads to an error in the case of Heywood cases. The probably better alternative is to just find the mean bootstrapped value and find the confidence intervals based upon the observed range of values. The default is to have n.iter =1 and thus not do bootstrapping.

If using polytomous or dichotomous items, it is perhaps more useful to find the Item Response Theory parameters equivalent to the factor loadings reported in fa.poly by using the `irt.fa` function.

For those who like SPSS type output, the measure of factoring adequacy known as the Kaiser-Meyer-Olkin `KMO` test may be found from the correlation matrix or data matrix using the `KMO` function. Similarly, the Bartlett's test of Sphericity may be found using the `cortest.bartlett` function.

For those who want to have an object of the variances accounted for, this is returned invisibly by the print function. (e.g., p <- print(fa(ability))$Vaccounted). This is now returned by the fa function as well (e.g. p <- fa(ability)$Vaccounted). Just as communalities may be found by the diagonal

of Pattern %*% t(Structure) so can the variance accounted for be found by diagonal (t(Structure) %*% Pattern. Note that these are referred to as SS loadings.

The output from the print.psych.fa function displays the factor loadings (from the pattern matrix, the h2 (communalities) the u2 (the uniquenesses), com (the complexity of the factor loadings for that variable (see below). In the case of an orthogonal solution, h2 is merely the row sum of the squared factor loadings. But for an oblique solution, it is the row sum of the orthogonal factor loadings (remember, that rotations or transformations do not change the communality).

In response to a request from Asghar Minaei who wanted to combine several imputation data sets from mice, fa.pooled was added. The separate data sets are combined into a list (datasets) which are then factored separately. Each solution is rotated towards the first set in the list. The results are then reported in terms of pooled loadings and confidence intervals based upon the replications.

`fa.sapa` simulates the process of doing SAPA (Synthetic Aperture Personality Assessment). It will do iterative solutions for successive random samples of fractions (frac) of the data set. This allows us to find the stability of solutions for various sample sizes and various sample rates. Need to specify the number of iterations (n.iter) as well as the percent of data sampled (frac).

Value

`values`	Eigen values of the common factor solution
`e.values`	Eigen values of the original matrix
`communality`	Communality estimates for each item. These are merely the sum of squared factor loadings for that item.
`communalities`	If using minrank factor analysis, these are the communalities reflecting the total amount of common variance. They will exceed the communality (above) which is the model estimated common variance.
`rotation`	which rotation was requested?
`n.obs`	number of observations specified or found
`loadings`	An item by factor (pattern) loading matrix of class “loadings" Suitable for use in other programs (e.g., GPA rotation or factor2cluster. To show these by sorted order, use `print.psych` with sort=TRUE
`complexity`	Hoffman’s index of complexity for each item. This is just $\frac{(\Sigma a_i^2)^2}{\Sigma a_i^4}$ where a_i is the factor loading on the ith factor. From Hofmann (1978), MBR. See also Pettersson and Turkheimer (2010).
`Structure`	An item by factor structure matrix of class “loadings". This is just the loadings (pattern) matrix times the factor intercorrelation matrix.
`fit`	How well does the factor model reproduce the correlation matrix. This is just $\frac{\Sigma r_{ij}^2 - \Sigma r_{ij}^{*2}}{\Sigma r_{ij}^2}$ (See `VSS`, `ICLUST`, and `principal` for this fit statistic.
`fit.off`	how well are the off diagonal elements reproduced?
`dof`	Degrees of Freedom for this model. This is the number of observed correlations minus the number of independent parameters. Let n=Number of items, nf = number of factors then $dof = n*(n-1)/2 - n*nf + nf*(nf-1)/2$

`objective` Value of the function that is minimized by a maximum likelihood procedures. This is reported for comparison purposes and as a way to estimate chi square goodness of fit. The objective function is $f = log(trace((FF' + U2)^{-1}R) - log(|(FF' + U2)^{-1}R|) - n.items$. When using MLE, this function is minimized. When using OLS (minres), although we are not minimizing this function directly, we can still calculate it in order to compare the solution to a MLE fit.

`STATISTIC` If the number of observations is specified or found, this is a chi square based upon the objective function, f (see above). Using the formula from `factanal`(which seems to be Bartlett's test) :
$\chi^2 = (n.obs - 1 - (2 * p + 5)/6 - (2 * factors)/3)) * f$

`PVAL` If n.obs > 0, then what is the probability of observing a chisquare this large or larger?

`Phi` If oblique rotations (e.g,m using oblimin from the GPArotation package or promax) are requested, what is the interfactor correlation?

`communality.iterations`
The history of the communality estimates (For principal axis only.) Probably only useful for teaching what happens in the process of iterative fitting.

`residual` The matrix of residual correlations after the factor model is applied. To display it conveniently, use the `residuals` command.

`chi` When normal theory fails (e.g., in the case of non-positive definite matrices), it useful to examine the empirically derived χ^2 based upon the sum of the squared residuals * N. This will differ slightly from the MLE estimate which is based upon the fitting function rather than the actual residuals. Until January, 2026 these values were calculated from the sum of all of the residuals, but now they are found from the sum of the lower off diagonal elements. This is more appropriate.

`rms` This is the sum of the squared (lower off diagonal residuals) divided by the degrees of freedom. Comparable to an RMSEA which, because it is based upon χ^2, requires the number of observations to be specified. The rms is an empirical value while the RMSEA is based upon normal theory and the non-central χ^2 distribution. That is to say, if the residuals are particularly non-normal, the rms value and the associated χ^2 and RMSEA can differ substantially.

`crms` rms adjusted for degrees of freedom

`RMSEA` The Root Mean Square Error of Approximation is based upon the non-central χ^2 distribution and the χ^2 estimate found from the MLE fitting function. With normal theory data, this is fine. But when the residuals are not distributed according to a noncentral χ^2, this can give very strange values. (And thus the confidence intervals can not be calculated.) The RMSEA is a conventional index of goodness (badness) of fit but it is also useful to examine the actual rms values.

`TLI` The Tucker Lewis Index of factoring reliability which is also known as the non-normed fit index.

`BIC` Based upon χ^2 with the assumption of normal theory and using the χ^2 found using the objective function defined above. This is just $\chi^2 - df * log(N)$

eBIC — When normal theory fails (e.g., in the case of non-positive definite matrices), it useful to examine the empirically derived eBIC based upon the empirical χ^2 - df *log(N).

R2 — The multiple R square between the factors and factor score estimates, if they were to be found. (From Nicewander, 2020 (default) or Grice, 2001). Derived from R2 is is the minimum correlation between any two factor estimates = 2R2-1. See `fsi` for more information about factor indeterminancy.

r.scores — The correlations of the factor score estimates using the specified model, if they were to be found. Comparing these correlations with that of the scores themselves will show, if an alternative estimate of factor scores is used (e.g., the tenBerge method), the problem of factor indeterminacy. For these correlations will not necessarily be the same.

weights — The beta weights to find the factor score estimates. These are also used by the `predict.psych` function to find predicted factor scores for new cases. These weights will depend upon the scoring method requested.

scores — The factor scores as requested. Note that these scores reflect the choice of the way scores should be estimated (see scores in the input). That is, simple regression ("Thurstone"), correlaton preserving ("tenBerge") as well as "Anderson" and "Bartlett" using the appropriate algorithms (see `factor.scores`). The correlation between factor score estimates (r.scores) is based upon using the regression/Thurstone approach. The actual correlation between scores will reflect the rotation algorithm chosen and may be found by correlating those scores. Although the scores are found by multiplying the standarized data by the weights matrix, this will not result in standard scores if using regression.

valid — The validity coffiecient of course coded (unit weighted) factor score estimates (From Grice, 2001)

score.cor — The correlation matrix of coarse coded (unit weighted) factor score estimates, if they were to be found, based upon the structure matrix rather than the weights or loadings matrix.

rot.mat — The rotation matrix as returned from GPArotation.

Note

Thanks to Erich Studerus for some very helpful suggestions about various rotation and factor scoring algorithms, and to Gumundur Arnkelsson for suggestions about factor scores for singular matrices.

The fac function is the original fa function which is now called by fa repeatedly to get confidence intervals.

SPSS will sometimes use a Kaiser normalization before rotating. This will lead to different solutions than reported here. To get the Kaiser normalized loadings, use `kaiser`.

The communality for a variable is the amount of variance accounted for by all of the factors. That is to say, for orthogonal factors, it is the sum of the squared factor loadings (rowwise). The communality is insensitive to rotation. However, if an oblique solution is found, then the communality is not the sum of squared pattern coefficients. In both cases (oblique or orthogonal) the communality is the diagonal of the reproduced correlation matrix where ${}_nR_n =_n P_{kk}\Phi_{kk}P'_n$ where P is the pattern matrix and Φ is the factor intercorrelation matrix. This is the same, of course to multiplying

the pattern by the structure: $R = PS'$ where the Structure matrix is $S = \Phi P$. Similarly, the sum of squared loading values are the diagonal of the product ${}_k\Phi_{kk} P'_{nn} P_k$.

Note that these SS loadings will not match the SS loadings returned by the stats::loading function which assumes orthogonal factors. Thus, just printing the loadings object returned by fa will not match the results of a normal print of the fa object nor those returned in the Vaccounted object.

A frequently asked question is why are the factor names of the rotated solution not in ascending order? That is, for example, if factoring the 25 items of the bfi, the factor names are MR2 MR3 MR5 MR1 MR4, rather than the seemingly more logical "MR1" "MR2" "MR3" "MR4" "MR5". This is for pedagogical reasons, in that factors as extracted are orthogonal and are in order of amount of variance accounted for. But when rotated (orthogonally) or transformed (obliquely) the simple structure solution does not preserve that order. The factors are still ordered according to variance accounted for, but because rotation changes how much variance each factor explains, the order may not the be the same as the original order. The factor names are, of course, arbitrary, and are kept with the original names to show the effect of rotation/transformation. To give them names associated with their ordinal position, simply paste("F", 1:nf, sep="") where nf is the number of factors. See the last example.

The print function for the fa output will return (invisibly) an object (Vaccounted) that matches the printed output for the variance accounted for by each factor, as well as the cumulative variance, and the percentage of variance accounted for by each factor.

Correction to documentation: as of September, 2014, the oblique.scores option is correctly explained. (It had been backwards.) The default (oblique.scores=FALSE) finds scores based upon the Structure matrix, while oblique.scores=TRUE finds them based upon the pattern matrix. The latter case matches factanal. This error was detected by Mark Seeto.

If the raw data are factored, factors scores are estimated. By default this will be done using 'regression' but alternatives are available. Although the scores are found by multiplying the standardized data by the weights, if using regression, the resulting factor scores will not necessarily have unit variance.

In the case of missing data for some items, the factor scores will return NA for any case where any item is missing. Specifying missing=TRUE will return factor scores for all cases using the imputation option chosen.

The minimum residual solution is done by finding those communalities that will minimize the off diagonal residual. The uls solution finds those communalities that minimize the total residuals.

The minres solution has been modified (April, 2107) following suggestions by Hao Wu. Although the fitting function was the minimal residual, the first derivative of the fitting function was incorrect. This has now been modified so that the results match those of SPSS and CEFA. The prior solutions are still available using fm="old.min".

Alpha factoring was added in August, 2017 to add to the numerous alternative models of factoring.

A few more lines of output were added in August 2017 to show the measures of factor adequacy for different rotations. This had been available in the results from `factor.scores` but now is added to the fa output. This was based upon the techniques of Grice (2001). By default, these measures are now found using methods discussed by Nicewander (2020). The Grice approach can be used by specifying Grice=TRUE. See the discussion of factor indeterminacy in `fsi`.

For a comparison of the `fa` to factor analysis using SPSS, see Grieder and Steiner (2021).

Some correlation matrices that arise from using pairwise deletion or from tetrachoric or polychoric matrices will not be proper. That is, they will not be positive semi-definite (all eigen values >= 0).

The cor.smooth function will adjust correlation matrices (smooth them) by making all negative eigen values slightly greater than 0, rescaling the other eigen values to sum to the number of variables, and then recreating the correlation matrix. See cor.smooth for an example of this problem using the burt data set. This set probably represents a typo when entering the published correlation matrix.

One reason for this problem when using tetrachorics or polychorics seems to be the adjustment for continuity. Setting correct=0 turns this off and seems to produce more proper matrices.

Although both fm ="pa" or fm="minres" will work with these “improper" matrices, the goodness of fit tests do not. Thus, by default, calls to fa.stats will pass the smooth parameter as TRUE. This may be prevented by forcing smooth=FALSE. Npt smoothing does not affect the pa or minres solution, but will affect the goodness of fit statistics slightly.

Problems in factor polychoric matrices: If getting error messages when factoring polychoric matrices, try turning off the correction for continuity (correct=0).

The use of n.rotations=20 was introduced in version 2.6.6. By forcing multiple restarts, this improves the rotation solution for some particular problems. This change has had the effect that previous solutions using the prior default of 1 will be different. See the test case the Widaman data set in faRotations.

Author(s)

William Revelle

References

Gorsuch, Richard, (1983) Factor Analysis. Lawrence Erlebaum Associates.

Grice, James W. (2001), Computing and evaluating factor scores. Psychological Methods, 6, 430-450

Grieder, Silvia and Steiner, Markus. D. (2021) Algorithmic jingle jungle: A comparison of implementations of principal axis factoring and promax rotation in R and SPSS. Behavior Research Methods. Doi: 10.3758/s13428-021-01581

Harman, Harry and Jones, Wayne (1966) Factor analysis by minimizing residuals (minres), Psychometrika, 31, 3, 351-368.

Hofmann, R. J. (1978) . Complexity and simplicity as objective indices descriptive of factor solutions. Multivariate Behavioral Research, 13, 247-250.

Kaiser, Henry F. and Caffrey, John. Alpha factor analysis, Psychometrika, (30) 1-14.

Nicewander, W. Alan (2020) A pespective on the mathematical and psychometric aspects of factor indeterminacy. Multivariate Behavioral Research. 55 DOI 10.1080/00273171.2019.1684872)

Nguyen, H. V. & Waller, N. G. (in press). Local minima and factor rotations in exploratory factor analysis. Psychological Methods.

Pettersson E, Turkheimer E. (2010) Item selection, evaluation, and simple structure in personality data. Journal of research in personality, 44(4), 407-420.

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at https://personality-project.org/r/book/

Shapiro, A. and ten Berge, Jos M. F, (2002) Statistical inference of minimum rank factor analysis. Psychometika, (67) 79-84.

ten Berge, Jos M. F. and Kiers, Henk A. L. (1991). A numerical approach to the approximate and the exact minimum rank of a covariance matrix. Psychometrika, (56) 309-315.

Widaman, K.F. (1993) Common Factor Analysis Versus Principal Component Analysis: Differential Bias in Representing Model Parameters? Multivariate Behavioral Research, 28 (3), 263-311.

Widaman, K.F. (2018) On common factor and principal component representations of data: Implications for theory and for confirmatory replications. Structural Equation Modeling: A Multidisciplinary Journal (25) 829-847.

See Also

`principal` for principal components analysis (PCA). PCA will give very similar solutions to factor analysis when there are many variables. The differences become more salient as the number variables decrease. The PCA and FA models are actually very different and should not be confused. One is a model of the observed variables, the other is a model of latent variables. Although some commercial packages (e.g., SPSS and SAS) refer to both as factor models, they are not. It is incorrect to report doing a factor analysis using principal components.

`CFA` Added in January, 2026. `CFA` will do a confirmatory factor analysis as specified by a model (using either lavaan syntax or an item keys list). Also available as a confirmatory model is `CFA.bifactor` which will extract a general factor and then a set of group factors. These algorithms are non-iterative and are limited to no-cross loadings (other than for a general factor).

`omega` estimates $\omega_{hierarchical}$ as well as ω_{total}.

`omegaStats` will take output from functions that report a general factor and group factors and calculate various omega statistics.

`irt.fa` for Item Response Theory analyses using factor analysis, using the two parameter IRT equivalent of loadings and difficulties.

`fa.random` applies a random intercepts model by removing the mean score for each subject prior to factoring.

`VSS` will produce the Very Simple Structure (VSS) and MAP criteria for the number of factors, `nfactors` to compare many different factor criteria.

`ICLUST` will do a hierarchical cluster analysis alternative to factor analysis or principal components analysis.

`factor.scores` to find factor scores with alternative options. `predict.psych` to find predicted scores based upon new data, `fa.extension` to extend the factor solution to new variables, `omega` for hierarchical factor analysis with one general factor. `fa.multi` for hierarchical factor analysis with an arbitrary number of 2nd order factors.

`fa.sort` will sort the factor loadings into echelon form. `fa.organize` will reorganize the factor pattern matrix into any arbitrary order of factors and items.

`KMO` and `cortest.bartlett` for various tests that some people like.

`factor2cluster` will prepare unit weighted scoring keys of the factors that can be used with `scoreItems`.

`fa.lookup` will print the factor analysis loadings matrix along with the item “content" taken from a dictionary of items. This is useful when examining the meaning of the factors.

`anova.psych` allows for testing the difference between two (presumably nested) factor models .

bassAckward will perform repeated factorings and organize them in a top-down structure suggested by Goldberg (2006) and Waller (2007).

fa.random to do a "random effects" factora analysis. This is just removing the person effect for each subject by mean centering for each subject. Functionally, this removes a general factor.

factor.rotate to do factor rotations "by hand".

Examples

```
#using the Harman 24 mental tests, compare a principal factor with a principal components solution
pc <- principal(Harman74.cor$cov,4,rotate="varimax")   #principal components
pa <- fa(Harman74.cor$cov,4,fm="pa" ,rotate="varimax")  #principal axis
uls <- fa(Harman74.cor$cov,4,rotate="varimax")        #unweighted least squares is minres
wls <- fa(Harman74.cor$cov,4,fm="wls")       #weighted least squares

#to show the loadings sorted by absolute value
print(uls,sort=TRUE)

#then compare with a maximum likelihood solution using factanal
mle <- factanal(covmat=Harman74.cor$cov,factors=4)
factor.congruence(list(mle,pa,pc,uls,wls))

#note that the order of factors and the sign of some of factors may differ

#finally, compare the unrotated factor, ml, uls, and  wls solutions
wls <- fa(Harman74.cor$cov,4,rotate="none",fm="wls")
pa <- fa(Harman74.cor$cov,4,rotate="none",fm="pa")
minres <-  factanal(factors=4,covmat=Harman74.cor$cov,rotation="none")
mle <- fa(Harman74.cor$cov,4,rotate="none",fm="mle")
uls <- fa(Harman74.cor$cov,4,rotate="none",fm="uls")
factor.congruence(list(minres,mle,pa,wls,uls))
#in particular, note the similarity of the mle and min res solutions
#note that the order of factors and the sign of some of factors may differ



#an example of where the ML and PA and MR models differ is found in Thurstone.33.
#compare the first two factors with the 3 factor solution
Thurstone.33 <- as.matrix(Thurstone.33)
mle2 <- fa(Thurstone.33,2,rotate="none",fm="mle")
mle3 <- fa(Thurstone.33,3 ,rotate="none",fm="mle")
pa2 <- fa(Thurstone.33,2,rotate="none",fm="pa")
pa3 <- fa(Thurstone.33,3,rotate="none",fm="pa")
mr2 <- fa(Thurstone.33,2,rotate="none")
mr3 <- fa(Thurstone.33,3,rotate="none")
factor.congruence(list(mle2,mr2,pa2,mle3,pa3,mr3))

#f5 <- fa(bfi[1:25],5)
#f5  #names are not in ascending numerical order (see note)
#colnames(f5$loadings) <- paste("F",1:5,sep="")
#f5

#Get the variance accounted for object from the print function
```

```
p <- print(mr3)
round(p$Vaccounted,2)

#pool data and fa the pooled result (not run)
#datasets.list <- list(bfi[1:500,1:25],bfi[501:1000,1:25],
#  bfi[1001:1500,1:25],bfi[1501:2000,1:25],bfi[2001:2500,1:25]) #five different data sets
#temp <- fa.pooled(datasets.list,5)    #do 5 factor analyses, pool the results
```

---

fa.diagram *Graph factor loading matrices*

Description

Factor analysis or principal components analysis results are typically interpreted in terms of the major loadings on each factor. These structures may be represented as a table of loadings or graphically, where all loadings with an absolute value > some cut point are represented as an edge (path). fa.diagram uses the various diagram functions to draw the diagram. fa.graph generates dot code for external plotting. fa.rgraph uses the Rgraphviz package (if available) to draw the graph. het.diagram will draw "heterarchy" diagrams of factor/scale solutions at different levels. All of these as well as some additional functions may be called by diagram which is a wrapper to the specific functions.

Usage

```
fa.diagram(fa.results,Phi=NULL,fe.results=NULL,sort=TRUE,labels=NULL,cut=.3,
     simple=TRUE, errors=FALSE,g=FALSE,digits=1,e.size=.05,rsize=.15,side=2,
   main,cex=NULL,l.cex=NULL, marg=c(.5,.5,1,.5),adj=1,ic=FALSE,dictionary=NULL, ...)
het.diagram(r,levels,cut=.3,digits=2,both=TRUE,
        main="Heterarchy diagram",l.cex,gap.size,...)
extension.diagram(fa.results,Phi=NULL,fe.results=NULL,sort=TRUE,labels=NULL,cut=.3,
    f.cut, e.cut=.1,simple=TRUE,e.simple=FALSE,errors=FALSE,g=FALSE,
    digits=1,e.size=.05,rsize=.15,side=2,main,cex=NULL, e.cex=NULL,
    marg=c(.5,.5,1,.5),adj=1,ic=FALSE, ...)

fa.graph(fa.results,out.file=NULL,labels=NULL,cut=.3,simple=TRUE,
   size=c(8,6), node.font=c("Helvetica", 14),
    edge.font=c("Helvetica", 10), rank.direction=c("RL","TB","LR","BT"),
     digits=1,main="Factor Analysis", ...)
fa.rgraph(fa.results,out.file=NULL,labels=NULL,cut=.3,simple=TRUE,
   size=c(8,6), node.font=c("Helvetica", 14),
    edge.font=c("Helvetica", 10), rank.direction=c("RL","TB","LR","BT"),
     digits=1,main="Factor Analysis",graphviz=TRUE, ...)
```

Arguments

fa.results — The output of factor analysis, principal components analysis, or ICLUST analysis. May also be a factor loading matrix from anywhere.

Phi	Normally not specified (it is is found in the FA, pc, or ICLUST, solution), this may be given if the input is a loadings matrix.
fe.results	the results of a factor extension analysis (if any)
out.file	If it exists, a dot representation of the graph will be stored here (fa.graph)
labels	Variable labels
cut	Loadings with abs(loading) > cut will be shown
f.cut	factor correlations > f.cut will be shown
e.cut	extension loadings with abs(loading) > e.cut will be shown
simple	Only the biggest loading per item is shown
e.simple	Only the biggest loading per extension item is shown
g	Does the factor matrix reflect a g (first) factor. If so, then draw this to the left of the variables, with the remaining factors to the right of the variables. It is useful to turn off the simple parameter in this case.
ic	If drawing a cluster analysis result, should we treat it as latent variable model (ic=FALSE) or as an observed variable model (ic=TRUE)
dictionary	Replace variable names with those found in the "Content" or "Item" or "item" column of a dictionary
r	A correlation matrix for the het.diagram function
levels	A list of the elements in each level
both	Should arrows have double heads (in het.diagram)
size	graph size
sort	sort the factor loadings before showing the diagram
errors	include error estimates (as arrows)
e.size	size of ellipses
rsize	size of rectangles
side	on which side should error arrows go?
cex	modify font size
e.cex	Modify the font size for the Dependent variables, defaults to cex
l.cex	modify the font size in arrows, defaults to cex
gap.size	The gap in the arrow for the label. Can be adjusted to compensate for variations in cex or l.cex
marg	sets the margins to be wider than normal, returns them to the normal size upon exit
adj	how many different positions (1-3) should be used for the numeric labels. Useful if they overlap each other.
node.font	what font should be used for nodes in fa.graph
edge.font	what font should be used for edges in fa.graph
rank.direction	parameter passed to Rgraphviz– which way to draw the graph
digits	Number of digits to show as an edgelable
main	Graphic title, defaults to "factor analyis" or "factor analysis and extension"
graphviz	Should we try to use Rgraphviz for output?
...	other parameters

Details

Path diagram representations have become standard in confirmatory factor analysis, but are not yet common in exploratory factor analysis. Representing factor structures graphically helps some people understand the structure.

By default the arrows come from the latent variables to the observed variables. This is, of course, the factor model. However, if the class of the object to be drawn is 'principal', then reverse the direction of the arrows, and the 'latent' variables are no longer latent, and are shown as boxes. For cluster models, the default is to treat them as factors, but if ic =TRUE, then we treat it as a components model.

fa.diagram does not use Rgraphviz and is the preferred function. fa.graph generates dot code to be used by an external graphics program. It does not have all the bells and whistles of fa.diagram, but these may be done in the external editor.

Hierarchical (bifactor) models may be drawn by specifying the g parameter as TRUE. This allows for an graphical displays of various factor transformations with a bifactor structure (e.g., `bifactor` and `biquartimin`. See `omega` for an alternative way to find these structures.

The `het.diagram` function will show the case of a hetarchical structure at multiple levels. It can also be used to show the patterns of correlations between sets of scales (e.g., EPI, NEO, BFI). The example is for showing the relationship between 3 sets of 4 variables from the Thurstone data set. The parameters l.cex and gap.size are used to adjust the font size of the labels and the gap in the lines.

`extension.diagram` will draw a `fa.extend` result with slightly more control than using `fa.diagram` or the more generic `diagram` function.

In fa.rgraph although a nice graph is drawn for the orthogonal factor case, the oblique factor drawing is acceptable, but is better if cleaned up outside of R or done using fa.diagram.

The normal input is taken from the output of either `fa` or `ICLUST`. This latter case displays the ICLUST results in terms of the cluster loadings, not in terms of the cluster structure. Actually an interesting option.

It is also possible to just give a factor loading matrix as input. In this case, supplying a Phi matrix of factor correlations is also possible.

It is possible, using fa.graph, to export dot code for an omega solution. fa.graph should be applied to the schmid$sl object with labels specified as the rownames of schmid$sl. The results will need editing to make fully compatible with dot language plotting.

To specify the model for a structural equation confirmatory analysis of the results, use `structure.diagram` instead.

Value

fa.diagram: A path diagram is drawn without using Rgraphviz. This is probably the more useful function.

fa.rgraph: A graph is drawn using rgraphviz. If an output file is specified, the graph instructions are also saved in the dot language.

fa.graph: the graph instructions are saved in the dot language.

Note

fa.rgraph requires Rgraphviz. Because there are occasional difficulties installing Rgraphviz from Bioconductor in that some libraries are misplaced and need to be relinked, it is probably better to use fa.diagram or fa.graph.

Author(s)

William Revelle

See Also

diagram, omega.graph, ICLUST.graph, bassAckward.diagram. structure.diagram will convert the factor diagram to sem modeling code.

For information on constructing a dictionary for the items, see fa.lookup.

Examples

```
test.simple <- fa(item.sim(16),2,rotate="oblimin")
#if(require(Rgraphviz)) {fa.graph(test.simple) }
fa.diagram(test.simple)
f3 <- fa(Thurstone,3,rotate="cluster")
fa.diagram(f3,cut=.4,digits=2)
f3l <- f3$loadings
fa.diagram(f3l,main="input from a matrix")
Phi <- f3$Phi
fa.diagram(f3l,Phi=Phi,main="Input from a matrix")
fa.diagram(ICLUST(Thurstone,2,title="Two cluster solution of Thurstone"),main="Input from ICLUST")
het.diagram(Thurstone,levels=list(1:4,5:8,3:7))

#add item content to the diagram
f5 <- fa(bfi[1:25],5)
fa.diagram(f5) #just show the variable names
#add item content to the diagram using a dictionary
fa.diagram(f5,dictionary=bfi.dictionary, rsize=.5,
   main="Factor analysis of the BFI: Item content") #set the rectangle size to be slightly bigger

#this example shows how to use the ic parameter
#show how caffiene increases arousal and tension
select <- c("active" , "alert" ,  "aroused", "sleepy",  "tired" ,
          "drowsy" , "anxious", "jittery" ,"nervous",
 "calm" ,   "relaxed" ,"at.ease" ,"gender",  "drug")
 #msq.small <- msqR[select]  requires psychTools

 msq.small<- small.msq   #a small data set in psych
fe <- fa.extend(msq.small, 2,1:12,14) #caffeine effects on mood
extension.diagram(fe, ic=TRUE) #drug causes arousal
```

`fa.extension` *Apply Dwyer's factor extension to find factor loadings for extended variables*

Description

Dwyer (1937) introduced a method for finding factor loadings for variables not included in the original analysis. This is basically finding the unattenuated correlation of the extension variables with the factor scores. An alternative, which does not correct for factor reliability was proposed by Gorsuch (1997). Both options are an application of exploratory factor analysis with extensions to new variables. Also useful for finding the validities of variables in the factor space.

Usage

```
fa.extension(Roe,fo,correct=TRUE)
fa.extend(r,nfactors=1,ov=NULL,ev=NULL,n.obs = NA, np.obs=NULL,
  correct=TRUE,rotate="oblimin",SMC=TRUE,   warnings=TRUE, fm="minres",
  alpha=.1,omega=FALSE,cor="cor",use="pairwise",cor.correct=.5,weight=NULL,
    missing=FALSE, smooth=TRUE, ...)
faRegression(r,nfactors=1,ov=NULL,dv=NULL, n.obs = NA
 , np.obs=NULL,correct=TRUE,rotate="oblimin",SMC=TRUE,warnings=TRUE, fm="minres",alpha=.1,
 omega=FALSE,cor="cor",use="pairwise",cor.correct=.5,weight=NULL,smooth=TRUE, ...)
#this is just an alias for
faReg(r,nfactors=1,ov=NULL,dv=NULL, n.obs = NA
 , np.obs=NULL,correct=TRUE,rotate="oblimin",SMC=TRUE,warnings=TRUE, fm="minres",alpha=.1,
 omega=FALSE,cor="cor",use="pairwise",cor.correct=.5,weight=NULL,smooth=TRUE, ...)
```

Arguments

`Roe`	The correlations of the original variables with the extended variables
`fo`	The output from the `fa` or `omega` functions applied to the original variables.
`correct`	correct=TRUE produces Dwyer's solution, correct=FALSE does not correct for factor reliability. This is not quite the Gorsuch technique.
`r`	A correlation or data matrix with all of the variables to be analyzed by fa.extend
`ov`	The original variables to factor
`ev`	The extension variables
`dv`	The dependent variables if doing faRegression
`nfactors`	Number of factors to extract, default is 1
`n.obs`	Number of observations used to find the correlation matrix if using a correlation matrix. Used for finding the goodness of fit statistics. Must be specified if using a correlaton matrix and finding confidence intervals.
`np.obs`	Pairwise number of observations. Required if using fm="minchi", suggested in other cases to estimate the empirical goodness of fit.

rotate	"none", "varimax", "quartimax", "bentlerT", "geominT" and "bifactor" are orthogonal rotations. "promax", "oblimin", "simplimax", "bentlerQ, "geominQ" and "biquartimin" and "cluster" are possible rotations or transformations of the solution. The default is to do a oblimin transformation, although versions prior to 2009 defaulted to varimax.
SMC	Use squared multiple correlations (SMC=TRUE) or use 1 as initial communality estimate. Try using 1 if imaginary eigen values are reported. If SMC is a vector of length the number of variables, then these values are used as starting values in the case of fm='pa'.
warnings	warnings=TRUE => warn if number of factors is too many
fm	factoring method fm="minres" will do a minimum residual (OLS), fm="wls" will do a weighted least squares (WLS) solution, fm="gls" does a generalized weighted least squares (GLS), fm="pa" will do the principal factor solution, fm="ml" will do a maximum likelihood factor analysis. fm="minchi" will minimize the sample size weighted chi square when treating pairwise correlations with different number of subjects per pair.
alpha	alpha level for the confidence intervals for RMSEA
omega	Do the extension analysis for an omega type analysis
cor	Pass the kind of correlation to fa (defaults to Pearson, can use mixed)
use	Option for the cor function on how to handle missing data.
cor.correct	The correction to be passed to mixed, tet, or polycor (defaults to .5)
weight	Should we weight the variables? (see fa)
missing	When finding factor scores, are missing data allowed?
smooth	Should we smooth the correlation matrix – smoothing produces bad output if there are NAs in the R matrix
...	Additional parameters, specifically, keys may be passed if using the target rotation, or delta if using geominQ, or whether to normalize if using Varimax

Details

It is sometimes the case that factors are derived from a set of variables (the Fo factor loadings) and we want to see what the loadings of an extended set of variables (Fe) would be. Given the original correlation matrix Ro and the correlation of these original variables with the extension variables of Roe, it is a straight forward calculation to find the loadings Fe of the extended variables on the original factors. This technique was developed by Dwyer (1937) for the case of adding new variables to a factor analysis without doing all the work over again. But, as discussed by Horn (1973) factor extension is also appropriate when one does not want to include the extension variables in the original factor analysis, but does want to see what the loadings would be anyway.

This could be done by estimating the factor scores and then finding the covariances of the extension variables with the factor scores. If raw data are available, this is done by faReg. But if the original data are not available, but just the covariance or correlation matrix is, then the use of fa.extension is most appropriate to estimate those correlations before doing the regression.

The factor analysis results from either fa or omega functions applied to the original correlation matrix is extended to the extended variables given the correlations (Roe) of the extended variables with the original variables.

fa.extension assumes that the original factor solution was found by the fa function.

For a very nice discussion of the relationship between factor scores, correlation matrices, and the factor loadings in a factor extension, see Horn (1973).

The fa.extend function may be thought of as a "seeded" factor analysis. That is, the variables in the original set are factored, this solution is then extended to the extension set, and the resulting output is presented as if both the original and extended variables were factored together. This may also be done for an omega analysis.

The example of fa.extend compares the extended solution to a direct solution of all of the variables using factor.congruence.

Another important use of factor extension is when parts of a correlation matrix are missing. Thus suppose you have a correlation matrix formed of variables 1 .. i, j..n where the first i variables correlate with each other (R11) and with the j... n (R12) variables, but the elements of of the R22 (j..n) are missing. That is, X has a missing data structure but we can find the correlation matrix R[1..n,1..n] =cor(X,na.rm=TRUE)

R[1..n,1..n] =

R[1..i,1..i]	R[1..i,j..n]
R[j..n,1..i]	NA

Factors can be found for the R11 matrix and then extended to the variables in the entire matrix. This allows for irt approaches to be applied even with significantly missing data.

f <- fa.extend(R,nf=1, ov=1:i,ev=j:n)

Then the loadings of n loadings of f may be found.

Combining this procedure with fa2irt allows us to find the irt based parameters of the n variables, even though we have substantially incomplete data.

Using the idea behind factor extension it is straightforward to apply these techniques to multiple regression, because the extended loadings are functionally beta weights. . faRegression just organizes the extension analysis in terms of a regression analysis. If the raw data are available, it first finds the factor scores and then correlates these with the dependent variable in the standard regression approach. But, if just the correlation matrix is available, it estimates the factor by dependent variable correlations by the use of link{fa.extend} before doing the regression.

Value

Factor Loadings of the exended variables on the original factors

Author(s)

William Revelle

References

Paul S. Dwyer (1937) The determination of the factor loadings of a given test from the known factor loadings of other tests. Psychometrika, 3, 173-178

Gorsuch, Richard L. (1997) New procedure for extension analysis in exploratory factor analysis, Educational and Psychological Measurement, 57, 725-740

Horn, John L. (1973) On extension analysis and its relation to correlations between variables and factor scores. Multivariate Behavioral Research, 8, (4), 477-489.

See Also

See Also as fa, principal, Dwyer

Examples

```
 #The Dwyer Example
Ro <- Dwyer[1:7,1:7]
Roe <- Dwyer[1:7,8]
fo <- fa(Ro,2,rotate="none")
fe <- fa.extension(Roe,fo)

#an example from simulated data
set.seed(42)
 d <- sim.item(12)$item    #two orthogonal factors
 R <- cor(d)
 Ro <- R[c(1,2,4,5,7,8,10,11),c(1,2,4,5,7,8,10,11)]
 Roe <- R[c(1,2,4,5,7,8,10,11),c(3,6,9,12)]
 fo <- fa(Ro,2)
 fe <- fa.extension(Roe,fo)
 fa.diagram(fo,fe=fe)

 #alternatively just specify the original variables and the extension variables
 fe = fa.extend(R, 2, ov =c(1,2,4,5,7,8,10,11), ev=c(3,6,9,12))
 fa.diagram(fe$fo, fe = fe$fe)

 #create two correlated factors
 fx <- matrix(c(.9,.8,.7,.85,.75,.65,rep(0,12),.9,.8,.7,.85,.75,.65),ncol=2)
 Phi <- matrix(c(1,.6,.6,1),2)
 sim.data <- sim.structure(fx,Phi,n=1000,raw=TRUE)
 R <- cor(sim.data$observed)
 Ro <- R[c(1,2,4,5,7,8,10,11),c(1,2,4,5,7,8,10,11)]
 Roe <- R[c(1,2,4,5,7,8,10,11),c(3,6,9,12)]
 fo <- fa(Ro,2)
 fe <- fa.extension(Roe,fo)
 fa.diagram(fo,fe=fe)

 #now show how fa.extend works with the same data set
#note that we have to make sure that the variables are in the order to do the factor congruence
 fe2 <- fa.extend(sim.data$observed,2,ov=c(1,2,4,5,7,8,10,11),ev=c(3,6,9,12))
 fa.diagram(fe2,main="factor analysis with extension variables")
 fa2 <- fa(sim.data$observed[,c(1,2,4,5,7,8,10,11,3,6,9,12)],2)
 factor.congruence(fe2,fa2)
 summary(fe2)

 #an example of extending an omega analysis
```

```
fload <- matrix(c(c(c(.9,.8,.7,.6),rep(0,20)),c(c(.9,.8,.7,.6),rep(0,20)),c(c(.9,.8,.7,.6),
        rep(0,20)),c(c(c(.9,.8,.7,.6),rep(0,20)),c(.9,.8,.7,.6))),ncol=5)
 gload <- matrix(rep(.7,5))
 five.factor <- sim.hierarchical(gload,fload,500,TRUE) #create sample data set
 ss <- c(1,2,3,5,6,7,9,10,11,13,14,15,17,18,19)
 Ro <- cor(five.factor$observed[,ss])
 Re <- cor(five.factor$observed[,ss],five.factor$observed[,-ss])
 om5 <-omega(Ro,5)   #the omega analysis
 om.extend <- fa.extension(Re,om5) #the extension analysis
 om.extend #show it
 #now, include it in an omega diagram
 combined.om <- rbind(om5$schmid$sl[,1:ncol(om.extend$loadings)],om.extend$loadings)
 class(combined.om) <-c("psych","extend")
 omega.diagram(combined.om,main="Extended Omega")

 #show how to use fa.extend to do regression analyses with the raw data
 b5 <- faReg (bfi, nfactors = 5, ov =1:25, dv =26:28)
 extension.diagram(b5)
 R <-cor(bfi,use="pairwise")
 b5.r <- faReg(R, nfactors = 5, ov =1:25, dv =26:28) # not identical to b5
 round(b5$regression$coefficients - b5.r$regression$coefficients,2)
```

---

fa.lookup *A set of functions for factorial and empirical scale construction*

Description

When constructing scales through rational, factorial, or empirical means, it is useful to examine the content of the items that relate most highly to each other (e.g., the factor loadings of fa.lookup of a set of items), or to some specific set of criteria (e.g., bestScales). Given a dictionary of item content, these routines will sort by factor loading, item means, or criteria correlations and display the item content.

Usage

```
lookup(x,y,criteria=NULL)
lookupItems(content=NULL,dictionary=NULL,search=c("Item","Content","item"))
fa.lookup(f,dictionary=NULL,digits=2,cut=.0,n=NULL,sort=TRUE,maxlength=NULL)
item.lookup(f,m, dictionary,cut=.3, digits = 2)
itemSort(m, dictionary, ascending=TRUE, digits = 2)
keys.lookup(keys.list,dictionary)
lookupFromKeys(keys.list,dictionary,n=20,cors=NULL,sort=TRUE,suppress.names=FALSE,
      digits=2)
lmCorLookup(x,dictionary=NULL,cut=0,digits=2,p=.05)
```

Arguments

x	A data matrix or data frame depending upon the function.
y	A data matrix or data frame or a vector
criteria	Which variables (by name or location) should be the empirical target for bestScales and bestItems. May be a separate object.
f	The object returned from either a factor analysis (fa) or a principal components analysis (principal)
content	The word(s) to search for from a dictionary
keys.list	A list of scoring keys suitable to use for make.keys
cut	Return all values in abs(x[,c1]) > cut.
n	Return the n best items per factor (as long as they have their highest loading on that factor)
cors	If provided (e.g. from scoreItems) will be added to the lookupFromKeys output
dictionary	a data.frame with rownames corresponding to rownames in the f$loadings matrix or colnames of the data matrix or correlation matrix, and entries (may be multiple columns) of item content. See Notes for how to construct a dictionary.
search	Column names of dictionary to search, defaults to "Item" or "Content" (dictionaries have different labels for this column), can search any column specified by search.
m	A data frame of item means
digits	round to digits
sort	Should the factors be sorted first?
suppress.names	In lookupFromKeys, should we suppress the column labels
p	Show lmCor regressions with probability < p
ascending	order to sort the means in itemSort – see dfOrder
maxlength	If specified, just print the the first maxlength characters

Details

fa.lookup and lookup are simple helper functions to summarize correlation matrices or factor loading matrices. bestItems will sort the specified column (criteria) of x on the basis of the (absolute) value of the column. The return as a default is just the rowname of the variable with those absolute values > cut. If there is a dictionary of item content and item names, then include the contents as a two column (or more) matrix with rownames corresponding to the item name and then as many fields as desired for item content. (See the example dictionary bfi.dictionary).

lookup is used by bestItems and will find values in c1 of y that match those in x. It returns those rows of y of that match x. Suppose that you have a "dictionary" of the many variables in a study but you want to consider a small subset of them in a data set x. Then, you can find the entries in the dictionary corresponding to x by lookup(rownames(x),y) If the column is not specified, then it will match by rownames(y).

fa.lookup is used when examining the output of a factor analysis and one wants the corresponding variable names and contents. The returned object may then be printed in LaTex by using the df2latex function with the char option set to TRUE.

fa.lookup will work with output from fa, pca or omega. For omega output, the items are sorted by the non-general factor loadings.

Similarly, given a correlation matrix, r, of the x variables, if you want to find the items that most correlate with another item or scale, and then show the contents of that item from the dictionary, bestItems(r,c1=column number or name of x, contents = y)

item.lookup combines the output from a factor analysis fa with simple descriptive statistics (a data frame of means) with a dictionary. Items are grouped by factor loadings > cut, and then sorted by item mean. This allows a better understanding of how a scale works, in terms of the meaning of the item endorsements. Note the means must be a one column matrix (with row names), not a vector (without rownames.)

itemSort Combine item means and item content and then sort them by the item means.

lookupItems searches a dictionary for all items that have a certain content. The rownames of the returned object are the item numbers which can then be used in other functions to find statistics (e.g. omega) of a scale with those items. If an scales by items correlation matrix is given, then the item correlation with that scale are also shown.

Value

bestItems returns a sorted list of factor loadings or correlations with the labels as provided in the dictionary.

lookup is a very simple implementation of the match function.

fa.lookup takes a factor/cluster analysis object (or just a keys like matrix), sorts it using fa.sort and then matches by row.name to the corresponding dictionary entries.

Note

Although empirical scale construction is appealing, it has the basic problem of capitalizing on chance. Thus, be careful of over interpreting the results unless working with large samples. Iteration and bootstrapping aggregation (bagging) gives information on the stability of the solutions. See bestScales

To create a dictionary, create an object with row names as the item numbers, and the columns as the item content. See the bfi.dictionary in the psychTools package as an example. The bfi.dictionary was constructed from a spreadsheet with multiple columns, the first of which was the column names of the bfi. See the first (not run) example.

Author(s)

William Revelle

References

Revelle, W. (in preparation) An introduction to psychometric theory with applications in R. Springer. (Available online at https://personality-project.org/r/book/).

See Also

fa, iclust,principal, bestScales and bestItems

Examples

```
#Tne following shows how to create a dictionary
#first, copy the spreadsheet to the clipboard
#the spreadsheet should have multiple columns
#col 1         col 2     col 3
#item        content   label
#A1           Am indifferent to the feelings of others.     (q_146)
#A2        Inquire about others' well-being. (q_1162
#...

# bfi.dictionary <- read.clipboard.tab()  #read from the clipboard
# rownames(bfi.dictionary) <- bfi.dictionary[1] #the first column had the names
# bfi.dictionary <- bfi.dictionary[-1]  #these are redundant, drop them

f5 <- fa(bfi,5)
m <- colMeans(bfi,na.rm=TRUE)
item.lookup(f5,m,dictionary=bfi.dictionary[2,drop=FALSE])
 #just show the item content, not the source of the items
fa.lookup(f5,dictionary=bfi.dictionary[2])

#just show the means and the items
#use the m vector we found above
itemSort(as.matrix(m),dictionary=bfi.dictionary[,2:3,drop=FALSE])


#show how to use lookupFromKeys
bfi.keys <-
list(agree=c("-A1","A2","A3","A4","A5"),conscientiousness=c("C1","C2","C3","-C4","-C5"),
extraversion=c("-E1","-E2","E3","E4","E5"),neuroticism=c("N1","N2","N3","N4","N5"),
openness = c("O1","-O2","O3","O4","-O5"))
bfi.over <- scoreOverlap(bfi.keys,bfi) #returns the corrected for overlap values
lookupFromKeys(bfi.keys,bfi.dictionary,n=5, cors=bfi.over$item.cor)
 #show the keying information
 if(require(psychTools)){
lookupItems("life",psychTools::spi.dictionary) #find those items with "life" in the item
}
```

`fa.multi` *Multi level (hierarchical) factor analysis*

Description

Some factor analytic solutions produce correlated factors which may in turn be factored. If the solution has one higher order, the omega function is most appropriate. But, in the case of multi higher order factors, then the faMulti function will do a lower level factoring and then factor the resulting correlation matrix. Multi level factor diagrams are also shown.

Usage

```
fa.multi(r, nfactors = 3, nfact2 = 1, n.obs = NA, n.iter = 1, rotate = "oblimin",
 scores = "regression", residuals = FALSE, SMC = TRUE, covar = FALSE, missing =
 FALSE,impute = "median", min.err = 0.001, max.iter = 50, symmetric = TRUE, warnings
 =TRUE, fm = "minres", alpha = 0.1, p = 0.05, oblique.scores = FALSE, np.obs = NULL,
 use ="pairwise", cor = "cor", ...)

fa.multi.diagram(multi.results,sort=TRUE,labels=NULL,flabels=NULL,f2labels=NULL,cut=.2,
 gcut=.2, simple=TRUE,errors=FALSE,
    digits=1,e.size=.1,rsize=.15,side=3,main=NULL,cex=NULL,color.lines=TRUE
    ,marg=c(.5,.5,1.5,.5),adj=2, ...)
```

Arguments

The arguments match those of the fa function.

r	A correlation matrix or raw data matrix
nfactors	The desired number of factors for the lower level
nfact2	The desired number of factors for the higher level
n.obs	Number of observations used to find the correlation matrix if using a correlation matrix. Used for finding the goodness of fit statistics. Must be specified if using a correlaton matrix and finding confidence intervals.
np.obs	The pairwise number of observations. Used if using a correlation matrix and asking for a minchi solution.
rotate	"none", "varimax", "quartimax", "bentlerT", "equamax", "varimin", "geominT" and "bifactor" are orthogonal rotations. "promax", "oblimin", "simplimax", "bentlerQ, "geominQ" and "biquartimin" and "cluster" are possible oblique transformations of the solution. The default is to do a oblimin transformation, although versions prior to 2009 defaulted to varimax.
n.iter	Number of bootstrap interations to do in fa or fa.poly
residuals	Should the residual matrix be shown
scores	the default="regression" finds factor scores using regression. Alternatives for estimating factor scores include simple regression ("Thurstone"), correlaton preserving ("tenBerge") as well as "Anderson" and "Bartlett" using the appropriate algorithms (see factor.scores). Although scores="tenBerge" is probably preferred for most solutions, it will lead to problems with some improper correlation matrices.
SMC	Use squared multiple correlations (SMC=TRUE) or use 1 as initial communality estimate. Try using 1 if imaginary eigen values are reported. If SMC is a vector of length the number of variables, then these values are used as starting values in the case of fm='pa'.
covar	if covar is TRUE, factor the covariance matrix, otherwise factor the correlation matrix
missing	if scores are TRUE, and missing=TRUE, then impute missing values using either the median or the mean

impute	"median" or "mean" values are used to replace missing values
min.err	Iterate until the change in communalities is less than min.err
max.iter	Maximum number of iterations for convergence
symmetric	symmetric=TRUE forces symmetry by just looking at the lower off diagonal values
warnings	warnings=TRUE => warn if number of factors is too many
fm	factoring method fm="minres" will do a minimum residual (OLS), fm="wls" will do a weighted least squares (WLS) solution, fm="gls" does a generalized weighted least squares (GLS), fm="pa" will do the principal factor solution, fm="ml" will do a maximum likelihood factor analysis. fm="minchi" will minimize the sample size weighted chi square when treating pairwise correlations with different number of subjects per pair.
alpha	alpha level for the confidence intervals for RMSEA
p	if doing iterations to find confidence intervals, what probability values should be found for the confidence intervals
oblique.scores	When factor scores are found, should they be based on the structure matrix (default) or the pattern matrix (oblique.scores=TRUE).
use	How to treat missing data, use="pairwise" is the default". See cor for other options.
cor	How to find the correlations: "cor" is Pearson", "cov" is covariance, "tet" is tetrachoric, "poly" is polychoric, "mixed" uses mixed cor for a mixture of tetrachorics, polychorics, Pearsons, biserials, and polyserials, Yuleb is Yulebonett, Yuleq and YuleY are the obvious Yule coefficients as appropriate
multi.results	The results from fa.multi
labels	variable labels
flabels	Labels for the factors (not counting g)
f2labels	The labels for the second order factors
size	size of graphics window
digits	Precision of labels
cex	control font size
color.lines	Use black for positive, red for negative
marg	The margins for the figure are set to be wider than normal by default
adj	Adjust the location of the factor loadings to vary as factor mod 4 + 1
main	main figure caption
...	additional parameters, specifically, keys may be passed if using the target rotation, or delta if using geominQ, or whether to normalize if using Varimax. In addition, for fa.multi.diagram, other options to pass into the graphics packages
e.size	the size to draw the ellipses for the factors. This is scaled by the number of variables.
cut	Minimum path coefficient to draw
gcut	Minimum general factor path to draw

simple	draw just one path per item
sort	sort the solution before making the diagram
side	on which side should errors be drawn?
errors	show the error estimates
rsize	size of the rectangles

Details

See fa and omega for a discussion of factor analysis and of the case of one higher order factor.

Value

f1	The standard output from a factor analysis from fa for the raw variables
f2	The standard output from a factor analysis from fa for the correlation matrix of the level 1 solution.

Note

This is clearly an early implementation (Feb 14 2016) which might be improved.

Author(s)

William Revelle

References

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at https://personality-project.org/r/book/

See Also

fa, omega

Examples

```
f31 <- fa.multi(Thurstone,3,1) #compare with \code{\link{omega}}
f31
fa.multi.diagram(f31)
```

`fa.parallel` *Scree plots of data or correlation matrix compared to random "parallel" matrices*

Description

One way to determine the number of factors or components in a data matrix or a correlation matrix is to examine the "scree" plot of the successive eigenvalues. Sharp breaks in the plot suggest the appropriate number of components or factors to extract. "Parallel" analysis is an alternative technique that compares the scree of factors of the observed data with that of a random data matrix of the same size as the original. This may be done for continuous , dichotomous, or polytomous data using Pearson, tetrachoric or polychoric correlations.

Usage

```
fa.parallel(x,n.obs=NULL,fm="minres",fa="both",nfactors=1,
 main="Parallel Analysis Scree Plots",
 n.iter=20,error.bars=FALSE,se.bars=FALSE,SMC=FALSE,ylabel=NULL,show.legend=TRUE,
 sim=TRUE,quant=.95,cor="cor",use="pairwise",plot=TRUE,correct=.5,sqrt=FALSE)
paSelect(keys,x,cor="cor", fm="minres",plot=FALSE)

fa.parallel.poly(x ,n.iter=10,SMC=TRUE,  fm = "minres",correct=TRUE,sim=FALSE,
       fa="both",global=TRUE)   #deprecated
## S3 method for class 'poly.parallel'
plot(x,show.legend=TRUE,fa="both",...)
```

Arguments

`x`	A data.frame or data matrix of scores. If the matrix is square, it is assumed to be a correlation matrix. Otherwise, correlations (with pairwise deletion) will be found.
`n.obs`	n.obs=0 implies a data matrix/data.frame. Otherwise, how many cases were used to find the correlations.
`fm`	What factor method to use. (minres, ml, uls, wls, gls, pa) See `fa` for details.
`fa`	show the eigen values for a principal components (fa="pc") or a principal axis factor analysis (fa="fa") or both principal components and principal factors (fa="both")
`nfactors`	The number of factors to extract when estimating the eigen values. Defaults to 1, which was the prior value used.
`main`	a title for the analysis
`n.iter`	Number of simulated analyses to perform
`use`	How to treat missing data, use="pairwise" is the default". See cor for other options.
`cor`	How to find the correlations: "cor" is Pearson", "cov" is covariance, "tet" is tetrachoric, "poly" is polychoric, "mixed" uses mixed cor for a mixture of tetrachorics, polychorics, Pearsons, biserials, and polyserials, Yuleb is Yulebonett,

Yuleq and YuleY are the obvious Yule coefficients as appropriate. This matches the call to fa.

keys — A list of scoring keys to allow fa.parallel on multiple subsets of the data

correct — For tetrachoric correlations, should a correction for continuity be applied. (See tetrachoric.) If set to 0, then no correction is applied, otherwise, the default is to add .5 observations to the cell.

sim — For continuous data, the default is to resample as well as to generate random normal data. If sim=FALSE, then just show the resampled results. These two results are very similar. This does not make sense in the case of correlation matrix, in which case resampling is impossible. In the case of polychoric or tetrachoric data, in addition to randomizing the real data, should we compare the solution to random simulated data. This will double the processing time, but will basically show the same result.

error.bars — Should error.bars be plotted (default = FALSE)

se.bars — Should the error bars be standard errors (se.bars=TRUE) or 1 standard deviation (se.bars=FALSE, the default). With many iterations, the standard errors are very small and some prefer to see the broader range. The default has been changed in version 1.7.8 to be se.bars=FALSE to more properly show the range.

SMC — SMC=TRUE finds eigen values after estimating communalities by using SMCs. smc = FALSE finds eigen values after estimating communalities with the first factor.

ylabel — Label for the y axis – defaults to “eigen values of factors and components", can be made empty to show many graphs

show.legend — the default is to have a legend. For multiple panel graphs, it is better to not show the legend

quant — if nothing is specified, the empirical eigen values are compared to the mean of the resampled or simulated eigen values. If a value (e.g., quant=.95) is specified, then the eigen values are compared against the matching quantile of the simulated data. Clearly the larger the value of quant, the few factors/components that will be identified. The default is to use quant=.95.

global — If doing polychoric analyses (fa.parallel.poly) and the number of alternatives differ across items, it is necessary to turn off the global option. fa.parallel.poly is deprecated but this choice is still relevant.

... — additional plotting parameters, for plot.poly.parallel

plot — By default, fa.parallel draws the eigen value plots. If FALSE, suppresses the graphic output

sqrt — If TRUE, take the squareroot of the eigen values to more understandably show the scale. Note, although providing more useful graphics the results will be the same. See DelGuidice, 2022

Details

Cattell’s “scree" test is one of most simple tests for the number of factors problem. Horn’s (1965) “parallel" analysis is an equally compelling procedure. Other procedures for determining the most optimal number of factors include finding the Very Simple Structure (VSS) criterion (VSS) and

Velicer's MAP procedure (included in VSS). Both the VSS and the MAP criteria are included in the nfactors function which also reports the mean item complexity and the BIC for each of multiple solutions. fa.parallel plots the eigen values for a principal components and the factor solution (minres by default) and does the same for random matrices of the same size as the original data matrix. For raw data, the random matrices are 1) a matrix of univariate normal data and 2) random samples (randomized across rows) of the original data.

fa.parallel with the cor=poly option will do what fa.parallel.poly explicitly does: parallel analysis for polychoric and tetrachoric factors. If the data are dichotomous, fa.parallel.poly will find tetrachoric correlations for the real and simulated data, otherwise, if the number of categories is less than 10, it will find polychoric correlations. Note that fa.parallel.poly is slower than fa.parallel because of the complexity of calculating the tetrachoric/polychoric correlations. The functionality of fa.parallel.poly is included in fa.parallel with cor=poly option (etc.) option but the older fa.parallel.poly is kept for those who call it directly.

That is, fa.parallel now will do tetrachorics or polychorics directly if the cor option is set to "tet" or "poly". As with fa.parallel.poly this will take longer.

The means of (ntrials) random solutions are shown. Error bars are usually very small and are suppressed by default but can be shown if requested. If the sim option is set to TRUE (default), then parallel analyses are done on resampled data as well as random normal data. In the interests of speed, the parallel analyses are done just on resampled data if sim=FALSE. Both procedures tend to agree.

As of version 1.5.4, I added the ability to specify the quantile of the simulated/resampled data, and to plot standard deviations or standard errors. By default, this is set to the 95th percentile.

Alternative ways to estimate the number of factors problem are discussed in the Very Simple Structure (Revelle and Rocklin, 1979) documentation (VSS) and include Wayne Velicer's MAP algorithm (Veicer, 1976).

Parallel analysis for factors is actually harder than it seems, for the question is what are the appropriate communalities to use. If communalities are estimated by the Squared Multiple Correlation (SMC) smc, then the eigen values of the original data will reflect major as well as minor factors (see sim.minor to simulate such data). Random data will not, of course, have any structure and thus the number of factors will tend to be biased upwards by the presence of the minor factors.

By default, fa.parallel estimates the communalities based upon a one factor minres solution. Although this will underestimate the communalities, it does seem to lead to better solutions on simulated or real (e.g., the bfi or Harman74) data sets.

For comparability with other algorithms (e.g, the paran function in the paran package), setting smc=TRUE will use smcs as estimates of communalities. This will tend towards identifying more factors than the default option.

Yet another option (suggested by Florian Scharf) is to estimate the eigen values based upon a particular factor model (e.g., specify nfactors > 1).

Printing the results will show the eigen values of the original data that are greater than simulated values.

A sad observation about parallel analysis is that it is sensitive to sample size. That is, for large data sets, the eigen values of random data are very close to 1. This will lead to different estimates of the number of factors as a function of sample size. Consider factor structure of the bfi data set (the first 25 items are meant to represent a five factor model). For samples of 200 or less, parallel analysis suggests 5 factors, but for 1000 or more, six factors and components are indicated. This is not due

to an instability of the eigen values of the real data, but rather the closer approximation to 1 of the random data as n increases.

Although with nfactors=1, 6 factors are suggested, when specifying nfactors =5, parallel analysis of the bfi suggests 12 factors should be extracted!

When simulating dichotomous data in fa.parallel.poly, the simulated data have the same difficulties as the original data. This functionally means that the simulated and the resampled results will be very similar. Note that fa.parallel.poly has functionally been replaced with fa.parallel with the cor="poly" option.

As with many psych functions, fa.parallel has been changed to allow for multicore processing. For running a large number of iterations, it is obviously faster to increase the number of cores to the maximum possible (using the options("mc.cores"=n) command where n is determined from detectCores().

Value

A plot of the eigen values for the original data, ntrials of resampling of the original data, and of a equivalent size matrix of random normal deviates. If the data are a correlation matrix, specify the number of observations.

Also returned (invisibly) are:

`fa.values`	The eigen values of the factor model for the real data.
`fa.sim`	The descriptive statistics of the simulated factor models.
`pc.values`	The eigen values of a principal components of the real data.
`pc.sim`	The descriptive statistics of the simulated principal components analysis.
`nfact`	Number of factors with eigen values > eigen values of random data
`ncomp`	Number of components with eigen values > eigen values of random data
`values`	The simulated values for all simulated trials

Note

Although by default the test is applied to the 95th percentile eigen values, this can be modified by setting the quant parameter to any particular quantile. The actual simulated data are also returned (invisibly) in the value object. Thus, it is possible to do descriptive statistics on those to choose a preferred comparison. See the last example (not run) The simulated and resampled data tend to be very similar, so for a slightly cleaner figure, set sim=FALSE.

For relatively small samples with dichotomous data and cor="tet" or cor="poly" if some cells are empty, or if the resampled matrices are not positive semi-definite, warnings are issued. this leads to serious problems if using multi.cores (the default if using a Mac). This is due to the correction for continuity. Setting correct=0 seems to solve the problem.

Final Note: It is important to realize that there is no one right to determine the number of factors. Although parallell analysis is useful, the results should be taken for what they are: model fits. No model is correct but some models are useful.

Note

Gagan Atreya reports a problem with the multi-core implementation of fa.parallel when running Microsoft Open R. This can be resolved by setMKLthreads(1) to set the number of threads to 1.

Author(s)

William Revelle

References

Del Giudice, M. (2022, October 21). The Square-Root Scree Plot: A Simple Improvement to a Classic Display. doi: 10.31234/osf.io/axubd

Floyd, Frank J. and Widaman, Keith. F (1995) Factor analysis in the development and refinement of clinical assessment instruments. Psychological Assessment, 7(3):286-299, 1995.

Horn, John (1965) A rationale and test for the number of factors in factor analysis. Psychometrika, 30, 179-185.

Humphreys, Lloyd G. and Montanelli, Richard G. (1975), An investigation of the parallel analysis criterion for determining the number of common factors. Multivariate Behavioral Research, 10, 193-205.

Revelle, William and Rocklin, Tom (1979) Very simple structure - alternative procedure for estimating the optimal number of interpretable factors. Multivariate Behavioral Research, 14(4):403-414.

Velicer, Wayne. (1976) Determining the number of components from the matrix of partial correlations. Psychometrika, 41(3):321-327, 1976.

See Also

fa, nfactors, VSS, VSS.plot, VSS.parallel, sim.minor

Examples

```
#test.data <- Harman74.cor$cov   #The 24 variable Holzinger - Harman problem
#fa.parallel(test.data,n.obs=145)
fa.parallel(Thurstone,n.obs=213)   #the 9 variable Thurstone problem

if(require(psychTools)) {
#set.seed(123)
#minor <- sim.minor(24,4,400) #4 large and 12 minor factors
#ffa.parallel(minor$observed) #shows 5 factors and 4 components -- compare with
#fa.parallel(minor$observed,SMC=FALSE) #which shows 6  and 4 components factors
#a demonstration of parallel analysis of a dichotomous variable
#fp <- fa.parallel(psychTools::ability)    #use the default Pearson correlation
#fpt <- fa.parallel(psychTools::ability,cor="tet")  #do a tetrachoric correlation
#fpt <- fa.parallel(psychTools::ability,cor="tet",quant=.95) #do a tetrachoric correlation and
#use the 95th percentile of the simulated results
#apply(fp$values,2,function(x) quantile(x,.95))  #look at the 95th percentile of values
#apply(fpt$values,2,function(x) quantile(x,.95))  #look at the 95th percentile of values
#describe(fpt$values)  #look at all the statistics of the simulated values
#paSelect(bfi.keys,bfi)#do 5 different runs, once for each subscale
#paSelect(ability.keys,ability,cor="poly",fm="minrank")
}
```

fa.poly *Deprecated Exploratory Factor analysis functions. Please use fa*

Description

After 6 years, it is time to stop using these deprecated functions! Please see fa which includes all of the functionality of these older functions.

Usage

```
fa.poly(x,nfactors=1,n.obs = NA, n.iter=1, rotate="oblimin", SMC=TRUE, missing=FALSE,
 impute="median", min.err = .001, max.iter=50, symmetric=TRUE, warnings=TRUE,
 fm="minres",alpha=.1, p =.05,scores="regression", oblique.scores=TRUE,
        weight=NULL,global=TRUE,...)  #deprecated

factor.minres(r, nfactors=1, residuals = FALSE, rotate = "varimax",n.obs = NA,
scores = FALSE,SMC=TRUE, missing=FALSE,impute="median",min.err = 0.001, digits = 2,
 max.iter = 50,symmetric=TRUE,warnings=TRUE,fm="minres")    #deprecated

factor.wls(r,nfactors=1,residuals=FALSE,rotate="varimax",n.obs = NA,
scores=FALSE,SMC=TRUE,missing=FALSE,impute="median", min.err = .001,
 digits=2,max.iter=50,symmetric=TRUE,warnings=TRUE,fm="wls")  #deprecated
```

Arguments

r	deprecated.
x	deprecated
nfactors	deprecated
n.obs	deprecated
rotate	deprecated
n.iter	deprecated
residuals	deprecated
scores	deprecated
SMC	deprecated
missing	deprecated
impute	deprecated
max.iter	deprecated
symmetric	deprecated
warnings	deprecated
fm	deprecated
alpha	deprecated
p	deprecated

oblique.scores	deprecated
weight	deprecated
global	deprecated
digits	deprecated
min.err	deprecated
...	deprecated

Details

Please see the writeup for fa for all of the functionality in these older functions.

Value

Pleases see the writeup for fa

Note

These functions have been deprecated for 8 years. Don't use them.

Author(s)

William Revelle

References

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at https://personality-project.org/r/book/

Examples

```
#none, you should see fa
#using the Harman 24 mental tests, compare a principal factor with a principal components solution
```

---

fa.random *A first approximation to Random Effects Exploratory Factor Analysis*

Description

Inspired, in part, by the wprifm function in the profileR package, fa.random removes between subject differences in mean level and then does a normal exploratory factor analysis of the ipsatized data. Functionally, this removes a general factor of the data before factoring. To prevent non-positive definiteness of the residual data matrix, a very small amount of random noise is added to each variable. This is just a call to fa after removing the between subjects effect. Read the help file for fa for a detailed explanation of all of the input parameters and the output objects.

Usage

```
fa.random(data, nfactors = 1, fix = TRUE, n.obs = NA, n.iter = 1, rotate = "oblimin",
 scores = "regression", residuals = FALSE, SMC = TRUE, covar = FALSE, missing = FALSE,
 impute = "median", min.err = 0.001, max.iter = 50, symmetric = TRUE, warnings = TRUE,
  fm = "minres", alpha = 0.1, p = 0.05, oblique.scores = FALSE, np.obs = NULL,
  use = "pairwise", cor = "cor", weight = NULL, ...)
```

Arguments

data	A raw data matrix (or data.frame)
nfactors	Number of factors to extract, default is 1
fix	If TRUE, then a small amount of random error is added to each observed variable to keep the matrix positive semi-definite. If FALSE, then this is not done but because the matrix is non-positive semi-definite it will need to be smoothed when finding the scores and the various statistics.
n.obs	Number of observations used to find the correlation matrix if using a correlation matrix. Used for finding the goodness of fit statistics. Must be specified if using a correlaton matrix and finding confidence intervals. Ignored.
np.obs	The pairwise number of observations. Used if using a correlation matrix and asking for a minchi solution.
rotate	"none", "varimax", "quartimax", "bentlerT", "equamax", "varimin", "geominT" and "bifactor" are orthogonal rotations. "Promax", "promax", "oblimin", "simplimax", "bentlerQ, "geominQ" and "biquartimin" and "cluster" are possible oblique transformations of the solution. The default is to do a oblimin transformation, although versions prior to 2009 defaulted to varimax. SPSS seems to do a Kaiser normalization before doing Promax, this is done here by the call to "promax" which does the normalization before calling Promax in GPArotation.
n.iter	Number of bootstrap interations to do in fa or fa.poly
residuals	Should the residual matrix be shown
scores	the default="regression" finds factor scores using regression. Alternatives for estimating factor scores include simple regression ("Thurstone"), correlaton preserving ("tenBerge") as well as "Anderson" and "Bartlett" using the appropriate algorithms (factor.scores). Although scores="tenBerge" is probably preferred for most solutions, it will lead to problems with some improper correlation matrices.
SMC	Use squared multiple correlations (SMC=TRUE) or use 1 as initial communality estimate. Try using 1 if imaginary eigen values are reported. If SMC is a vector of length the number of variables, then these values are used as starting values in the case of fm='pa'.
covar	if covar is TRUE, factor the covariance matrix, otherwise factor the correlation matrix
missing	if scores are TRUE, and missing=TRUE, then impute missing values using either the median or the mean
impute	"median" or "mean" values are used to replace missing values

min.err	Iterate until the change in communalities is less than min.err
max.iter	Maximum number of iterations for convergence
symmetric	symmetric=TRUE forces symmetry by just looking at the lower off diagonal values
warnings	warnings=TRUE => warn if number of factors is too many
fm	Factoring method fm="minres" will do a minimum residual as will fm="uls". Both of these use a first derivative. fm="ols" differs very slightly from "minres" in that it minimizes the entire residual matrix using an OLS procedure but uses the empirical first derivative. This will be slower. fm="wls" will do a weighted least squares (WLS) solution, fm="gls" does a generalized weighted least squares (GLS), fm="pa" will do the principal factor solution, fm="ml" will do a maximum likelihood factor analysis. fm="minchi" will minimize the sample size weighted chi square when treating pairwise correlations with different number of subjects per pair. fm ="minrank" will do a minimum rank factor analysis. "old.min" will do minimal residual the way it was done prior to April, 2017 (see discussion below).
alpha	alpha level for the confidence intervals for RMSEA
p	if doing iterations to find confidence intervals, what probability values should be found for the confidence intervals
oblique.scores	When factor scores are found, should they be based on the structure matrix (default) or the pattern matrix (oblique.scores=TRUE).
weight	If not NULL, a vector of length n.obs that contains weights for each observation. The NULL case is equivalent to all cases being weighted 1.
use	How to treat missing data, use="pairwise" is the default". See cor for other options.
cor	How to find the correlations: "cor" is Pearson", "cov" is covariance, "tet" is tetrachoric, "poly" is polychoric, "mixed" uses mixed cor for a mixture of tetrachorics, polychorics, Pearsons, biserials, and polyserials, Yuleb is Yulebonett, Yuleq and YuleY are the obvious Yule coefficients as appropriate
...	additional parameters, specifically, keys may be passed if using the target rotation, or delta if using geominQ, or whether to normalize if using Varimax

Details

This function is inspired by the wprifm function in the profileR package and the citation there to a paper by Davison, Kim and Close (2009). The basic logic is to extract a means vector from each subject and then to analyze the resulting ipsatized data matrix. This can be seen as removing acquiecence in the case of personality items, or the general factor, in the case of ability items. Factors composed of items that are all keyed the same way (e.g., Neuroticism in the `bfi` data set) will be most affected by this technique.

The output is identical to the normal `fa` output with the addition of two objects: subject and within.r. The subject object is just the vector of the mean score for each subject on all the items. within.r is just the correlation of each item with those scores.

Value

subject	A vector of the mean score on all items for each subject
within.r	The correlation of each item with the subject vector
values	Eigen values of the common factor solution
e.values	Eigen values of the original matrix
communality	Communality estimates for each item. These are merely the sum of squared factor loadings for that item.
communalities	If using minrank factor analysis, these are the communalities reflecting the total amount of common variance. They will exceed the communality (above) which is the model estimated common variance.
rotation	which rotation was requested?
n.obs	number of observations specified or found
loadings	An item by factor (pattern) loading matrix of class "loadings" Suitable for use in other programs (e.g., GPA rotation or factor2cluster. To show these by sorted order, use `print.psych` with sort=TRUE
complexity	Hoffman's index of complexity for each item. This is just $\frac{(\Sigma a_i^2)^2}{\Sigma a_i^4}$ where a_i is the factor loading on the ith factor. From Hofmann (1978), MBR. See also Pettersson and Turkheimer (2010).
Structure	An item by factor structure matrix of class "loadings". This is just the loadings (pattern) matrix times the factor intercorrelation matrix.
fit	How well does the factor model reproduce the correlation matrix. This is just $\frac{\Sigma r_{ij}^2 - \Sigma r_{ij}^{*2}}{\Sigma r_{ij}^2}$ (See `VSS`, `ICLUST`, and `principal` for this fit statistic.
fit.off	how well are the off diagonal elements reproduced?
dof	Degrees of Freedom for this model. This is the number of observed correlations minus the number of independent parameters. Let n=Number of items, nf = number of factors then $dof = n*(n-1)/2 - n*nf + nf*(nf-1)/2$
objective	Value of the function that is minimized by a maximum likelihood procedures. This is reported for comparison purposes and as a way to estimate chi square goodness of fit. The objective function is $f = log(trace((FF' + U2)^{-1}R) - log(\|(FF' + U2)^{-1}R\|) - n.items$. When using MLE, this function is minimized. When using OLS (minres), although we are not minimizing this function directly, we can still calculate it in order to compare the solution to a MLE fit.
STATISTIC	If the number of observations is specified or found, this is a chi square based upon the objective function, f (see above). Using the formula from `factanal`(which seems to be Bartlett's test) : $\chi^2 = (n.obs - 1 - (2*p+5)/6 - (2*factors)/3))*f$
PVAL	If n.obs > 0, then what is the probability of observing a chisquare this large or larger?
Phi	If oblique rotations (e.g,m using oblimin from the GPArotation package or promax) are requested, what is the interfactor correlation?

`communality.iterations`
: The history of the communality estimates (For principal axis only.) Probably only useful for teaching what happens in the process of iterative fitting.

`residual`
: The matrix of residual correlations after the factor model is applied. To display it conveniently, use the `residuals` command.

`chi`
: When normal theory fails (e.g., in the case of non-positive definite matrices), it useful to examine the empirically derived χ^2 based upon the sum of the squared residuals * N. This will differ slightly from the MLE estimate which is based upon the fitting function rather than the actual residuals.

`rms`
: This is the sum of the squared (off diagonal residuals) divided by the degrees of freedom. Comparable to an RMSEA which, because it is based upon χ^2, requires the number of observations to be specified. The rms is an empirical value while the RMSEA is based upon normal theory and the non-central χ^2 distribution. That is to say, if the residuals are particularly non-normal, the rms value and the associated χ^2 and RMSEA can differ substantially.

`crms`
: rms adjusted for degrees of freedom

`RMSEA`
: The Root Mean Square Error of Approximation is based upon the non-central χ^2 distribution and the χ^2 estimate found from the MLE fitting function. With normal theory data, this is fine. But when the residuals are not distributed according to a noncentral χ^2, this can give very strange values. (And thus the confidence intervals can not be calculated.) The RMSEA is a conventional index of goodness (badness) of fit but it is also useful to examine the actual rms values.

`TLI`
: The Tucker Lewis Index of factoring reliability which is also known as the non-normed fit index.

`BIC`
: Based upon χ^2 with the assumption of normal theory and using the χ^2 found using the objective function defined above. This is just $\chi^2 - 2df$

`eBIC`
: When normal theory fails (e.g., in the case of non-positive definite matrices), it useful to examine the empirically derived eBIC based upon the empirical χ^2 - 2 df.

`R2`
: The multiple R square between the factors and factor score estimates, if they were to be found. (From Grice, 2001). Derived from R2 is is the minimum correlation between any two factor estimates = 2R2-1.

`r.scores`
: The correlations of the factor score estimates using the specified model, if they were to be found. Comparing these correlations with that of the scores themselves will show, if an alternative estimate of factor scores is used (e.g., the tenBerge method), the problem of factor indeterminacy. For these correlations will not necessarily be the same.

`weights`
: The beta weights to find the factor score estimates. These are also used by the `predict.psych` function to find predicted factor scores for new cases. These weights will depend upon the scoring method requested.

`scores`
: The factor scores as requested. Note that these scores reflect the choice of the way scores should be estimated (see scores in the input). That is, simple regression ("Thurstone"), correlaton preserving ("tenBerge") as well as "Anderson" and "Bartlett" using the appropriate algorithms (see `factor.scores`). The

correlation between factor score estimates (r.scores) is based upon using the regression/Thurstone approach. The actual correlation between scores will reflect the rotation algorithm chosen and may be found by correlating those scores. Although the scores are found by multiplying the standarized data by the weights matrix, this will not result in standard scores if using regression.

`valid` The validity coffiecient of course coded (unit weighted) factor score estimates (From Grice, 2001)

`score.cor` The correlation matrix of course coded (unit weighted) factor score estimates, if they were to be found, based upon the loadings matrix rather than the weights matrix.

`rot.mat` The rotation matrix as returned from GPArotation.

Note

An interesting, but not necessarily good, idea. To see what this does if there is a general factor, consider the unrotated solutions to the ability data set. In particular, compare the first factor loading with its congruence to the ipsatized solution means vector correlated with the items (the within.r object).

Author(s)

William Revelle

References

Davison, Mark L. and Kim, Se-Kang and Close, Catherine (2009) Factor Analytic Modeling of Within Person Variation in Score Profiles. Multivariate Behavioral Research (44(5) 668-687.

See Also

fa

Examples

```
if(require(psychTools)) {
fa.ab <- fa(psychTools::ability,4,rotate="none")  #normal factor analysis
fa.ab.ip <- fa.random(psychTools::ability,3,rotate="none")
fa.congruence(list(fa.ab,fa.ab.ip,fa.ab.ip$within.r))

}
```

`fa.sort` *Sort factor analysis or principal components analysis loadings*

Description

Although the print.psych function will sort factor analysis loadings, sometimes it is useful to do this outside of the print function. fa.sort takes the output from the fa or principal functions and sorts the loadings for each factor. Items are located in terms of their greatest loading. The new order is returned as an element in the fa list. fa.organize allows for the columns or rows to be reorganized.

Usage

```
fa.sort(fa.results,polar=FALSE)
fa.organize(fa.results,o=NULL,i=NULL,cn=NULL,echelon=TRUE,flip=TRUE)
```

Arguments

`fa.results`	The output from a factor analysis or principal components analysis using `fa` or `principal`. Can also just be a matrix of loadings. Can also organize the output of cluster loadings from `iclust`.
`polar`	Sort by polar coordinates of first two factors (FALSE)
`o`	The order in which to order the factors
`i`	The order in which to order the items
`cn`	new factor names
`echelon`	Organize the factors so that they are in echelon form (variable 1 .. n on factor 1, n+1 ...n=k on factor 2, etc.)
`flip`	Flip factor loadings such that the colMean is positive.

Details

The fa.results$loadings are replaced with sorted loadings.

fa.organize takes a factor analysis or components output and reorganizes the factors in the o order. Items are organized in the i order. This is useful when comparing alternative factor solutions.

The flip option works only for the case of matrix input, not for full `fa` objects. Use the `reflect` function.

Value

A sorted factor analysis, principal components analysis, or omega loadings matrix.

These sorted values are used internally by the various diagram functions.

The values returned are the same as `fa`, except in sorted order. In addition, the order is returned as an additional element in the fa list.

Author(s)

William Revelle

See Also

See Also as fa, pca, iclust, print.psych, fa.diagram,

Examples

```
test.simple <- fa(sim.item(16)$item,2)
fa.sort(test.simple)
fa.organize(test.simple,c(2,1))  #the factors but not the items have been rearranged
```

---

faCor — *Correlations between two factor analysis solutions*

Description

Given two factor analysis or pca solutions to a data matrix or correlation, what are the similarities between the two solutions. This may be found by factor correlations as well as factor congruences. Factor correlations are found by the matrix product of the factor weights and the correlation matrix and are estimates of what the factor score correlations would be. Factor congruence (aka Tucker or Burt coefficient) is the cosine of the vectors of factor loadings.

Usage

```
faCor(r, nfactors = c(1, 1), fm = c("minres", "minres"), rotate =
 c("oblimin", "oblimin"), scores = c("tenBerge", "tenBerge"), adjust=c(TRUE,TRUE),
   use = "pairwise", cor = "cor", weight = NULL, correct = 0.5,Target=list(NULL,NULL))
```

Arguments

r	A correlation matrix or a data matrix suitable for factoring
nfactors	Number of factors in each solution to extract
fm	Factor method. The default is 'minres' factoring. To compare with pca solutions, can also be (fm ="pca")
rotate	What type of rotations to apply. The default for factors is oblimin, for pca is varimax.
scores	What factor scoring algorithm should be used. Defaults to tenBerge for both cases.
adjust	Should the factor intercorrelations be corrected by the lack of factor deterimi-nancy (i.e. divide by the square root of the factor R2)
use	How to treat missing data. Use='pairwise" finds pairwise complete correlations.
cor	What kind of correlation to find. The default is Pearson.
weight	Should cases be weighted? Default, no.

`correct`	If finding tetrachoric or polychoric correlations, what correction should be applied to empty cells (defaults to .5)
`Target`	If doing target rotations (e.g., TargetQ or TargetT), then the Target must be specified. If TargetT, this may be a matrix, if TargetQ, this must be a list. (Strange property of GPARotation.)

Details

The factor correlations are found using the approach discussed by Gorsuch (1983) and uses the weights matrices found by $w = SR^{-1}$ and $r = w'Rw$ where S is the structure matrix and is $s = F\Phi$. The resulting correlations may be adjusted for the factor score variances (the diagonal of r) (the default).

For factor loading vectors of F1 and F2 the measure of factor congruence, phi, is

$$\phi = \frac{\sum F_1 F_2}{\sqrt{\sum(F_1^2)\sum(F_2^2)}}.$$

and is also found in `factor.congruence`.

For comparisons of factor solutions from 1 to n, use `bassAckward`. This function just compares two solutions from the same correlation/data matrix. `factor.congruence` can be used to compare any two sets of factor loadings.

Note that alternative ways of finding weights (e.g., regression, Bartlett, tenBerge) will produce somewhat different results. tenBerge produces weights that maintain the factor correlations in the factor scores.

Value

`Call`	The function call
`r`	The factor intercorrelations
`congruence`	The Burt/Tucker coefficient of congruence
`f1`	The first factor analysis
`f2`	The second factor analysis

Note

Useful for comparing factor solutions from the same data. Will not work for different data sets

Author(s)

William Revelle

References

Gorsuch, Richard, (1983) Factor Analysis. Lawrence Erlebaum Associates.

Burt, Cyril (1948) The factorial study of temperamental traits. British Journal of Statistical Psychology, 1(3) 178-203.

Horn, John L. (1973) On extension analysis and its relation to correlations between variables and factor scores. Multivariate Behavioral Research, 8, (4), 477-489.

Lorenzo-Seva, U. and ten Berge, J. M. F. (2006). Tucker's congruence coefficient as a meaningful index of factor similarity. Methodology: European Journal of Research Methods for the Behavioral and Social Sciences, 2(2):57-64.

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at https://personality-project.org/r/book/

See Also

fa, pca, omega and iclust, and {link{bassAckward} for alternative hierarchical solutions. fa.extend and fa.extension for other uses of factor - item correlations.

Examples

```
faCor(Thurstone,nfactors=c(2,3)) #compare two solutions to the Thurstone problem
faCor(bfi[1:25],nfactors=c(5,5),fm=c("minres","pca")) #compare pca and fa solutions
#compare two levels of factor extraction, and then find the correlations of the scores
faCor(bfi[1:25],nfactors=c(3,5)) #based upon linear algebra
f3 <- fa(bfi[1:25],3,scores="tenBerge")
f5 <- fa(bfi[1:25],5 ,scores="tenBerge")
cor2(f3$scores,f5$scores) #the correlation between the factor score estimates
```

---

factor.congruence *Coefficient of factor congruence*

Description

Given two sets of factor loadings, report their degree of congruence (vector cosine). Although first reported by Burt (1937,1 1948), this is frequently known as the Tucker index of factor congruence. Cohen's Profile similarity may be found as well.

Usage

```
factor.congruence(x, y=NULL,digits=2,use=NULL,structure=FALSE)
fa.congruence(x, y=NULL,digits=2,use=NULL,structure=FALSE)
```

Arguments

x	A matrix of factor loadings or a list of matrices of factor loadings
y	A second matrix of factor loadings (if x is a list, then y may be empty)
digits	Round off to digits
use	If NULL, then no loading matrices may contain missing values. If use="complete" then variables with any missing loadings are dropped (with a warning)
structure	If TRUE, find the factor congruences based upon the Structure matrix (if available), otherwise based upon the pattern matrix.

Details

Find the coefficient of factor congruence between two sets of factor loadings.

Factor congruences are the cosines of pairs of vectors defined by the loadings matrix and based at the origin. Thus, for loadings that differ only by a scaler (e.g. the size of the eigen value), the factor congruences will be 1.

For factor loading vectors of F1 and F2 the measure of factor congruence, phi, is

$$\phi = \frac{\sum F_1 F_2}{\sqrt{\sum(F_1^2)\sum(F_2^2)}}.$$

It is an interesting exercise to compare factor congruences with the correlations of factor loadings. Factor congruences are based upon the raw cross products, while correlations are based upon centered cross products. That is, correlations of factor loadings are cosines of the vectors based at the mean loading for each factor.

$$\phi = \frac{\sum(F_1 - a)(F_2 - b)}{\sqrt{\sum((F_1 - a)^2)\sum((F_2 - b)^2)}}.$$

.

For congruence coefficients, a = b= 0. For correlations a=mean F1, b= mean F2.

Input may either be matrices or factor analysis or principal components analyis output (which includes a loadings object), or a mixture of the two.

To compare more than two solutions, x may be a list of matrices, all of which will be compared.

Normally, all factor loading matrices should be complete (have no missing loadings). In the case where some loadings are missing, if the use option is specified, then variables with missing loadings are dropped.

If the data are zero centered, this is the correlation, if the data are centered around the scale midpoint (M), this is Cohen's Similarity coefficient. See examples. If M is not specified, it is found as the midpoint of the items in x and y.

`cohen.profile` applies the `congruence` function to data centered around M. M may be specified, or found from the data. The last example is taken from Cohen (1969).

`distance` finds the generalized distance as a function of r. City block (r=1), Euclidean (r=2) or weighted towards maximimum (r >2).

Value

A matrix of factor congruences or distances.

Author(s)

`<revelle@northwestern.edu>`
https://personality-project.org/revelle.html

References

Burt, Cyril (1948) Methods of factor-analysis with and without successive approximation. British Journal of Educational Psychology, 7 (2) 172-195.

Burt, Cyril (1948) The factorial study of temperamental traits. British Journal of Statistical Psychology, 1(3) 178-203.

Cohen, Jacob (1969), rc: A profile similarity coefficient invariant over variable reflection. Psychological Bulletin, 71 (4) 281-284.

Lorenzo-Seva, U. and ten Berge, J. M. F. (2006). Tucker's congruence coefficient as a meaningful index of factor similarity. Methodology: European Journal of Research Methods for the Behavioral and Social Sciences, 2(2):57-64.

Gorsuch, Richard, (1983) Factor Analysis. Lawrence Erlebaum Associates.

Revelle, W. (In preparation) An Introduction to Psychometric Theory with applications in R (https://personality-project.org/r/book/)

See Also

principal, fa. faCor will find factor correlations as well as congruences.

Examples

```
#factor congruence of factors and components, both rotated
#fa <- fa(Harman74.cor$cov,4)
#pc <- principal(Harman74.cor$cov,4)
#factor.congruence(fa,pc)
 #    RC1  RC3  RC2  RC4
#MR1 0.98 0.41 0.28 0.32
#MR3 0.35 0.96 0.41 0.31
#MR2 0.23 0.16 0.95 0.28
#MR4 0.28 0.38 0.36 0.98



#factor congruence without rotation
#fa <- fa(Harman74.cor$cov,4,rotate="none")
#pc <- principal(Harman74.cor$cov,4,rotate="none")
#factor.congruence(fa,pc)   #just show the beween method congruences
#     PC1   PC2   PC3   PC4
#MR1 1.00 -0.04 -0.06 -0.01
#MR2 0.15  0.97 -0.01 -0.15
#MR3 0.31  0.05  0.94  0.11
#MR4 0.07  0.21 -0.12  0.96

#factor.congruence(list(fa,pc))  #this shows the within method congruence as well

 #     MR1   MR2  MR3   MR4  PC1   PC2   PC3   PC4
#MR1  1.00  0.11 0.25  0.06 1.00 -0.04 -0.06 -0.01
#MR2  0.11  1.00 0.06  0.07 0.15  0.97 -0.01 -0.15
#MR3  0.25  0.06 1.00  0.01 0.31  0.05  0.94  0.11
#MR4  0.06  0.07 0.01  1.00 0.07  0.21 -0.12  0.96
```

```
#PC1  1.00  0.15 0.31  0.07 1.00  0.00  0.00  0.00
#PC2 -0.04  0.97 0.05  0.21 0.00  1.00  0.00  0.00
#PC3 -0.06 -0.01 0.94 -0.12 0.00  0.00  1.00  0.00
#PC4 -0.01 -0.15 0.11  0.96 0.00  0.00  0.00  1.00

#pa <- fa(Harman74.cor$cov,4,fm="pa")
# factor.congruence(fa,pa)
#         PA1  PA3  PA2  PA4
#Factor1 1.00 0.61 0.46 0.55
#Factor2 0.61 1.00 0.50 0.60
#Factor3 0.46 0.50 1.00 0.57
#Factor4 0.56 0.62 0.58 1.00


#compare with
#round(cor(fa$loading,pc$loading),2)
#      RC1   RC3   RC2   RC4
#MR1  0.99 -0.18 -0.33 -0.34
#MR3 -0.33  0.96 -0.16 -0.43
#MR2 -0.29 -0.46  0.98 -0.21
#MR4 -0.44 -0.30 -0.22  0.98
```

`factor.fit` *How well does the factor model fit a correlation matrix. Part of the VSS package*

Description

The basic factor or principal components model is that a correlation or covariance matrix may be reproduced by the product of a factor loading matrix times its transpose: F'F or P'P. One simple index of fit is the 1 - sum squared residuals/sum squared original correlations. This fit index is used by `VSS`, `ICLUST`, etc.

Usage

```
factor.fit(r, f)
```

Arguments

`r`	a correlation matrix
`f`	A factor matrix of loadings.

Details

There are probably as many fit indices as there are psychometricians. This fit is a plausible estimate of the amount of reduction in a correlation matrix given a factor model. Note that it is sensitive to the size of the original correlations. That is, if the residuals are small but the original correlations are small, that is a bad fit.

Let

$$R* = R - FF'$$

$$fit = 1 - \frac{\sum(R*^2)}{\sum(R^2)}$$

.

The sums are taken for the off diagonal elements.

Value

fit

Author(s)

William Revelle

See Also

VSS, ICLUST

Examples

```
## Not run:
#compare the fit of 4 to 3 factors for the Harman 24 variables
fa4 <- factanal(x,4,covmat=Harman74.cor$cov)
round(factor.fit(Harman74.cor$cov,fa4$loading),2)
#[1] 0.9
fa3 <- factanal(x,3,covmat=Harman74.cor$cov)
round(factor.fit(Harman74.cor$cov,fa3$loading),2)
#[1] 0.88


## End(Not run)
```

---

factor.model *Find R = F F' + U2 is the basic factor model*

Description

The basic factor or principal components model is that a correlation or covariance matrix may be reproduced by the product of a factor loading matrix times its transpose. Find this reproduced matrix. Used by factor.fit, VSS, ICLUST, etc.

Usage

```
factor.model(f,Phi=NULL,U2=TRUE)
```

Arguments

f	A matrix of loadings.
Phi	A matrix of factor correlations
U2	Should the diagonal be model by ff' (U2 = TRUE) or replaced with 1's (U2 = FALSE)

Value

A correlation or covariance matrix.

Author(s)

<revelle@northwestern.edu >
https://personality-project.org/revelle.html

References

Gorsuch, Richard, (1983) Factor Analysis. Lawrence Erlebaum Associates.
Revelle, W. In preparation) An Introduction to Psychometric Theory with applications in R (https://personality-project.org/r/book/)

See Also

ICLUST.graph,ICLUST.cluster, cluster.fit , VSS, omega

Examples

```
f2 <- matrix(c(.9,.8,.7,rep(0,6),.6,.7,.8),ncol=2)
mod <- factor.model(f2)
round(mod,2)
```

---

factor.residuals $R^* = R - F F'$

Description

The basic factor or principal components model is that a correlation or covariance matrix may be reproduced by the product of a factor loading matrix times its transpose. Find the residuals of the original minus the reproduced matrix. Used by factor.fit, VSS, ICLUST, etc.

Usage

```
factor.residuals(r, f)
```

Arguments

r	A correlation matrix
f	A factor model matrix or a list of class loadings

Details

The basic factor equation is ${}_nR_n \approx_n F_{kk}F'_n + U^2$. Residuals are just R* = R - F'F. The residuals should be (but in practice probably rarely are) examined to understand the adequacy of the factor analysis. When doing Factor analysis or Principal Components analysis, one usually continues to extract factors/components until the residuals do not differ from those expected from a random matrix.

Value

rstar is the residual correlation matrix.

Author(s)

Maintainer: William Revelle <revelle@northwestern.edu>

See Also

fa, principal, VSS, ICLUST

Examples

```
fa2 <- fa(Harman74.cor$cov,2,rotate="oblimin")
fa2resid <- factor.residuals(Harman74.cor$cov,fa2)
 fa2resid[1:4,1:4] #residuals with two factors extracted
 fa4 <- fa(Harman74.cor$cov,4,rotate="oblimin")
 fa4resid <- factor.residuals(Harman74.cor$cov,fa4)
 fa4resid[1:4,1:4] #residuals with 4 factors extracted
```

---

factor.rotate *"Hand" rotate a factor loading matrix*

Description

Given a factor or components matrix, it is sometimes useful to do arbitrary rotations of particular pairs of variables. This supplements the much more powerful rotation package GPArotation and is meant for specific requirements to do unusual rotations.

Usage

```
factor.rotate(f, angle, col1=1, col2=2,plot=FALSE,...)
```

Arguments

f	original loading matrix or a data frame (can be output from a factor analysis function).
angle	angle (in degrees!) to rotate
col1	column in factor matrix defining the first variable
col2	column in factor matrix defining the second variable
plot	plot the original (unrotated) and rotated factors
...	parameters to pass to fa.plot

Details

Partly meant as a demonstration of how rotation works, factor.rotate is useful for those cases that require specific rotations that are not available in more advanced packages such as GPArotation. If the plot option is set to TRUE, then the original axes are shown as dashed lines.

The rotation is in degrees counter clockwise.

Value

The resulting rotated matrix of loadings.

Note

For a complete rotation package, see GPArotation.

Author(s)

Maintainer: William Revelle <revelle@northwestern.edu >

References

https://personality-project.org/r/book/

Examples

```
#using the Harman 24 mental tests, rotate the 2nd and 3rd factors 45 degrees
f4<- fa(Harman74.cor$cov,4,rotate="none")
f4r45 <- factor.rotate(f4,45,2,3)
f4r90 <- factor.rotate(f4r45,45,2,3)
print(factor.congruence(f4,f4r45),digits=3) #poor congruence with original
print(factor.congruence(f4,f4r90),digits=3) #factor 2 and 3 have been exchanged and 3 flipped

#a graphic example
data(Harman23.cor)
f2 <- fa(Harman23.cor$cov,2,rotate="none")
op <- par(mfrow=c(1,2))
cluster.plot(f2,xlim=c(-1,1),ylim=c(-1,1),title="Unrotated ")
f2r <- factor.rotate(f2,-33,plot=TRUE,xlim=c(-1,1),ylim=c(-1,1),title="rotated -33 degrees")
```

```
op <- par(mfrow=c(1,1))
```

---

factor.scores *Various ways to estimate factor scores for the factor analysis model*

Description

A fundamental problem with factor analysis is that although the model is defined at the structural level, it is indeterminate at the data level. This problem of factor indeterminancy leads to alternative ways of estimating factor scores, none of which is ideal. Following Grice (2001) four different methods are available here.

Usage

```
factor.scores(x, f, Phi = NULL, method = c("Thurstone", "tenBerge", "Anderson",
       "Bartlett", "Harman","components"),rho=NULL,missing=FALSE,
       impute="none", Grice=FALSE)
```

Arguments

x	Either a matrix of data if scores are to be found, or a correlation matrix if just the factor weights are to be found.
f	The output from the fa, irt.fa, factor.rotate functions, or a factor loading matrix.
Phi	If a pattern matrix is provided, then what were the factor intercorrelations. Does not need to be specified if f is the output from the fa or irt.fa functions.
method	Which of four factor score estimation procedures should be used. Defaults to "tenBerge" which is not the method used by fa which uses Thurstone. See details below for the other four methods.
rho	If x is a set of data and rho is specified, then find scores based upon the fa results and the correlations reported in rho. Used when scoring fa.poly results.
missing	If missing is TRUE, missing items are imputed using either the median or mean. If missing is FALSE, the default, scores are found based upon the mean of the available items for each subject.
impute	By default, only complete cases are scored. But, missing data can be imputed using "median" or "mean". The number of missing by subject is reported. If impute = "none", missing data are not scored.
Grice	If true, use the Grice approach to factor indeterminancy – see fsi

Details

Although the factor analysis model is defined at the structural level, it is undefined at the data level. This is a well known but little discussed problem with factor analysis.

Factor scores represent estimates of common part of the variables and should not be thought of as identical to the factors themselves. If a factor is thought of as a chop stick stuck into the center of an ice cream cone and factor score estimates are represented by straws anywhere along the edge of the cone the problem of factor indeterminacy becomes clear, for depending on the shape of the cone, two straws can be negatively correlated with each other. (The imagery is taken from Niels Waller, adapted from Stanley Mulaik). In a very clear discussion of the problem of factor score indeterminacy, Grice (2001) reviews several alternative ways of estimating factor scores and considers weighting schemes that will produce uncorrelated factor score estimates as well as the effect of using coarse coded (unit weighted) factor weights.

`factor.scores` uses four different ways of estimate factor scores. In all cases, the factor score estimates are based upon the data matrix, X, times a weighting matrix, W, which weights the observed variables.

For polytomous or dichotmous data, factor scores can be estimated using Item Response Theory techniques (e.g., using `link{irt.fa}` and then `link{scoreIrt}`. Such scores are still just factor score estimates, for the IRT model is a latent variable model equivalent to factor analysis.

- method="Thurstone" finds the regression based weights: $W = R^{-1}F$ where R is the correlation matrix and F is the factor loading matrix. This the default for factor scores in `fa` function.
- method="tenBerge" finds weights such that the correlation between factors for an oblique solution is preserved. This the default for `factor.scores`. Note that formula 8 in Grice has a typo in the formula for C and should be: $L = F\Phi^(1/2)$ $C = R^(-1/2)L(L'R^(-1)L)^(-1/2)$ $W = R^(-1/2)C\Phi^(1/2)$
- method="Anderson" finds weights such that the factor scores will be uncorrelated: $W = U^{-2}F(F'U^{-2}RU^{-2}F)^{-1/2}$ where U is the diagonal matrix of uniquenesses. The Anderson method works for orthogonal factors only, while the tenBerge method works for orthogonal or oblique solutions.
- method = "Bartlett" finds weights given $W = U^{-2}F(F'U^{-2}F)^{-1}$
- method="Harman" finds weights based upon socalled "idealized" variables: $W = F(t(F)F)^{-1}$.
- method="components" uses weights that are just component loadings.

Value

- scores (the factor scores if the raw data is given)
- weights (the factor weights)
- r.scores (The correlations of the factor score estimates.)
- missing A vector of the number of missing observations per subject

Author(s)

William Revelle

References

Grice, James W.,2001, Computing and evaluating factor scores, Psychological Methods, 6,4, 430-450. (note the typo in equation 8)

ten Berge, Jos M.F., Wim P. Krijnen, Tom Wansbeek and Alexander Shapiro (1999) Some new results on correlation-preserving factor scores prediction methods. Linear Algebra and its Applications, 289, 311-318.

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at https://personality-project.org/r/book/

See Also

fa, factor.stats

Examples

```
f3 <- fa(Thurstone,3 )
f3$weights  #just the scoring weights
f5 <- fa(bfi,5) #this does the factor analyis
ten.scores <- factor.scores(bfi,f5, method="tenBerge")$scores
#compare the tenBerge factor score correlation to the factor correlations
sum(cor(ten.scores,use="pairwise") - f5$Phi)^2  #compare to the f5$Phi values
#compare the default (regression) score correlations to the factor correlations
sum(cor(f5$scores,use="pairwise")  - f5$Phi)^2
#compare to the f5 solution
cor2(f5$scores,ten.scores)
```

---

factor.stats — *Find various goodness of fit statistics for factor analysis and principal components*

Description

Chi square and other goodness of fit statistics are found based upon the fit of a factor or components model to a correlation matrix. Although these statistics are normally associated with a maximum likelihood solution, they can be found for minimal residual (OLS), principal axis, or principal component solutions as well. Primarily called from within these functions, factor.stats can be used by itself. Measures of factorial adequacy and validity follow the paper by Grice, 2001.

Usage

```
fa.stats(r=NULL,f,phi=NULL,n.obs=NA,np.obs=NULL,alpha=.05,fm=NULL,
     smooth=TRUE, coarse=TRUE, dof=NULL,Grice=FALSE)
factor.stats(r=NULL,f,phi=NULL,n.obs=NA,np.obs=NULL,alpha=.1,fm=NULL,
      smooth=TRUE, coarse=TRUE, dof = NULL,Grice=FALSE)
```

Arguments

r	A correlation matrix or a data frame of raw data
f	A factor analysis loadings matrix or the output from a factor or principal components analysis. In which case the r matrix need not be specified.
phi	A factor intercorrelation matrix if the factor solution was oblique.
n.obs	The number of observations for the correlation matrix. If not specified, and a correlation matrix is used, chi square will not be reported. Not needed if the input is a data matrix.
np.obs	The pairwise number of subjects for each pair in the correlation matrix. This is used for finding observed chi square.
alpha	alpha level of confidence intervals for RMSEA (twice the confidence at each tail)
fm	flag if components are being given statistics
smooth	Should the corelation matrix be smoothed before finding the stats
coarse	By default, find the coarse coded statistics.
dof	for EFA, found from data, for CFA passed from efa
Grice	If FALSE, use the Nicewander approach to factor indeterminancy. If TRUE, use the Grice approach.

Details

Combines the goodness of fit tests used in fa and principal into one function. If the matrix is singular, will smooth the correlation matrix before finding the fit functions. Now will find the RMSEA (root mean square error of approximation) and the alpha confidence intervals similar to a SEM function. Also reports the root mean square residual.

Chi square is found two ways. The first (STATISTIC) applies the goodness of fit test from Maximum Likelihood objective function (see below). This assumes multivariate normality. The second is the empirical chi square based upon the observed residual correlation matrix and the observed sample size for each correlation. This is found by summing the squared residual correlations time the sample size.

Value

fit	How well does the factor model reproduce the correlation matrix. (See VSS, ICLUST, and principal for this fit statistic.
fit.off	how well are the off diagonal elements reproduced? This is just 1 - the relative magnitude of the squared off diagonal residuals to the squared off diagonal original values.
dof	Degrees of Freedom for this model. This is the number of observed correlations minus the number of independent parameters. Let n=Number of items, nf = number of factors then $dof = n * (n-1)/2 - n * nf + nf * (nf-1)/2$

`objective` value of the function that is minimized by maximum likelihood procedures. This is reported for comparison purposes and as a way to estimate chi square goodness of fit. The objective function is
$f = log(trace((FF' + U2)^{-1}R) - log(|(FF' + U2)^{-1}R|) - n.items.$

`STATISTIC` If the number of observations is specified or found, this is a chi square based upon the objective function, f. Using the formula from `factanal`(which seems to be Bartlett's test) :
$\chi^2 = (n.obs - 1 - (2 * p + 5)/6 - (2 * factors)/3)) * f$
Note that this is different from the chi square reported by the sem package which seems to use $\chi^2 = (n.obs - 1 - (2 * p + 5)/6 - (2 * factors)/3)) * f$

`PVAL` If n.obs > 0, then what is the probability of observing a chisquare this large or larger?

`Phi` If oblique rotations (using oblimin from the GPArotation package or promax) are requested, what is the interfactor correlation.

`R2` The multiple R square between the factors and factor score estimates, if they were to be found. (From Grice, 2001)

`r.scores` The correlations of the factor score estimates, if they were to be found.

`weights` The beta weights to find the factor score estimates

`valid` The validity coffiecient of coarse coded (unit weighted) factor score estimates (From Grice, 2001)

`score.cor` The correlation matrix of coarse coded (unit weighted) factor score estimates, if they were to be found, based upon the loadings matrix. Note that these are not the same as the correlation of the factor score estimates r.scores

`RMSEA` The Root Mean Square Error of Approximation and the alpha confidence intervals. Based upon the chi square non-centrality parameter. This is found as $\sqrt{f/dof - 1(/-1)}$

`rms` The empirically found square root of the squared residuals. This does not require sample size to be specified nor does it make assumptions about normality.

`crms` While the rms uses the number of correlations to find the average, the crms uses the number of degrees of freedom. Thus, there is a penalty for having too complex a model.

Note

The problem of factor and factor score estimates leads to multiple different estimates of the correlations between the factors. Phi is the factor intercorrelation matrix from the rotations, r.scores is the correlation of the factor score estimates (if they were to be found from the data), score.cor is the correlation of the coarse coded factor score estimates, (if they were to be found). and of course the correlation of the factor score estimates themselves. By default, the first three of these are found.

Author(s)

William Revelle

References

Grice, James W.,2001, Computing and evaluating factor scores, Psychological Methods, 6,4, 430-450.

See Also

fa with fm="pa" for principal axis factor analysis, fa with fm="minres" for minimum residual factor analysis (default). factor.pa also does principal axis factor analysis, but is deprecated, as is factor.minres for minimum residual factor analysis. See principal for principal components.

Examples

```
v9 <- sim.hierarchical()
f3 <- fa(v9,3)
factor.stats(v9,f3,n.obs=500)
f3o <- fa(v9,3,fm="pa",rotate="Promax")
factor.stats(v9,f3o,n.obs=500)
```

---

factor2cluster *Extract cluster definitions from factor loadings*

Description

Given a factor or principal components loading matrix, assign each item to a cluster corresponding to the largest (signed) factor loading for that item. Essentially, this is a Very Simple Structure approach to cluster definition that corresponds to what most people actually do: highlight the largest loading for each item and ignore the rest.

Usage

```
factor2cluster(loads, cut = 0,aslist=FALSE)
```

Arguments

loads	either a matrix of loadings, or the result of a factor analysis/principal components analyis with a loading component
cut	Extract items with absolute loadings > cut
aslist	if TRUE, Return a keys list, else return a keys matrix (old style)

Details

A factor/principal components analysis loading matrix is converted to a cluster (-1,0,1) definition matrix where each item is assigned to one and only one cluster. This is a fast way to extract items that will be unit weighted to form cluster composites. Use this function in combination with cluster.cor to find the corrleations of these composite scores.

A typical use in the SAPA project is to form item composites by clustering or factoring (see ICLUST, principal), extract the clusters from these results (factor2cluster), and then form the composite correlation matrix using cluster.cor. The variables in this reduced matrix may then be used in multiple R procedures using mat.regress.

The input may be a matrix of item loadings, or the output from a factor analysis which includes a loadings matrix.

Value

a keys list (new style or a matrix of -1,0,1 cluster definitions for each item.

Author(s)

https://personality-project.org/revelle.html

Maintainer: William Revelle < revelle@northwestern.edu >

References

https://personality-project.org/r/r.vss.html

See Also

cluster.cor, factor2cluster, fa, principal, ICLUST make.keys, keys2list

Examples

```
#matches the factanal example
f4  <- fa(Harman74.cor$cov,4,rotate="varimax")
factor2cluster(f4)
```

---

faRotations *Multiple rotations of factor loadings to find local minima*

Description

A dirty little secret of factor rotation algorithms is the problem of local minima (Nguyen and Waller, 2022). Following ideas in that article, we allow for multiple random restarts and then return the global optimal solution. Used as part of the fa function or available as a stand alone function.

Usage

```
faRotations(loadings, r = NULL, rotate = "oblimin", hyper = 0.15, n.rotations = 10,...)
```

Arguments

loadings	Factor loadings matrix from fa or pca or any N x k loadings matrix
r	The correlation matrix used to find the factors. (Used to find the factor indeterminancy of the solution)
rotate	"none", "varimax", "Varimax","quartimax", "bentlerT", "equamax", "varimin", "geominT" and "bifactor" are orthogonal rotations. "Promax", "promax", "oblimin", "simplimax", "bentlerQ, "geominQ" and "biquartimin" and "cluster" are possible oblique transformations of the solution. Defaults to oblimin.
hyper	The value defining when a loading is in the "hyperplane".
n.rotations	The number of random restarts to use.
...	additional parameters, specifically, keys may be passed if using the target rotation, or delta if using geominQ, or whether to normalize if using Varimax

Details

Nguyen and Waller review the problem of local minima in factor analysis. This is a problem for all rotation algorithms, but is more so for some. faRotations generates n.rotations different starting values and then applies the specified rotation to the original loadings using multiple start values. Hyperplane counts and complexity indices are reported for each starting matrix, and the one with the highest hyoerplane count and the lowest complexity is returned.

Different rotation algorithms are sensitive to local minima. The examples show this for a particularly problematic correlation matrix offered by Keith Widaman.

Value

loadings	The best rotated solution
Phi	Factor correlations
rotation.stats	Hyperplane count, complexity.
rot.mat	The rotation matrix used.

Note

Adapted from the fungible package by Waller

Author(s)

William Revelle

References

Nguyen, H. V., & Waller, N. G. (2022, January 6). Local Minima and Factor Rotations in Exploratory Factor Analysis. Psychological Methods. Advance online publication. doi 10.1037/met0000467

See Also

fa, pca

Examples

```
f5 <- fa(bfi[,1:25],5,rotate="none")
f5r <- faRotations(f5,n.rotations=10,rotate="oblimin")   #
faRotations(f5$loadings)  #matrix input
geo <- faRotations(f5,rotate="geominQ",n.rotation=10)
 # a popular alternative, but more sensitive to local minima
describe(f5r$rotation.stats[,1:3])
describe(geo$rotation.stats[,1:3])

# An interesting problem from Keith Widaman
model <- 'f1 =~.8*v1+ .6*v2 + .4 *v3
          f2 =~ .8*v4 + .6*v5 + .4*v6
          f1 ~ .5*f2'

lavParse(model, phi=TRUE)     #show the model
widaman <- sim(model)$model
widaman     #show the correlations

test.1 <- fa(widaman, fm = "minres", nfactors = 2, rotate = "simplimax", n.rotations=1)
test.10 <- fa(widaman, fm = "minres", nfactors = 2, rotate = "simplimax")  #note the improvement
test.ob1 <- fa(widaman, fm = "minres", nfactors = 2,n.rotations=1)   #default is oblimin

test.ob <- fa(widaman, fm = "minres", nfactors = 2)   #default is oblimin
#stats::promax version
test.pr <- fa(widaman, fm = "minres", nfactors = 2, rotate = "promax")
#GPArotation version
test.pro <-  fa(widaman, fm = "minres", nfactors = 2, rotate = "Promax")
test.pro1 <- fa(widaman, fm = "minres", nfactors = 2,rotate = "Promax",n.rotations=1)

#pca is not factor analysis, don't confuse them

test.pc1 <- pca(widaman,nfactors=2,rotate="Promax")   #defaults to one rotation
test.pc10 <- pca(widaman,nfactors=2,rotate="Promax",n.rotations=10)
test.pcob <-  pca(widaman,nfactors=2,rotate="oblimin",n.rotations=10)

#now compare the alternative solutions
#first create a data frame
phi.mat <- data.frame(rotation=cs(Truth,Simplimax.1,Simplimax.10,oblimin.1,oblimin.10,
Promax.stats,Promax.1,Promax.10, pc.Promax.1, pc.Promax.10, pc.oblimin.10),
Phi=round(c(.5,test.1$Phi[1,2],test.10$Phi[1,2],test.ob1$Phi[1,2],test.ob$Phi[1,2],
test.pr$Phi[1,2],test.pro1$Phi[1,2],test.pro$Phi[1,2],
test.pc1$Phi[1,2], test.pc10$Phi[1,2],test.pcob$Phi[1,2]),2))

#then show it
phi.mat

#Problems with Promax
pa_promax.10 <- fa(Harman74.cor$cov, 4,fm="pa" ,rotate="Promax")
```

```
pa_promax.1 <- fa(Harman74.cor$cov, 4,fm="pa" ,rotate="Promax",n.rotations=1)
round(pa_promax.10$Phi,2)
round(pa_promax.1$Phi,2)
pa_oblimin <- fa(Harman74.cor$cov, 4,fm="pa" ,rotate="oblimin")
round(pa_oblimin$Phi,2)
```

`fisherz` *Transformations of r, d, and t including Fisher r to z and z to r and confidence intervals*

Description

Convert a correlation to a z or t, or d, or chi or covariance matrix or z to r using the Fisher transformation or find the confidence intervals for a specified correlation. r2d converts a correlation to an effect size (Cohen's d) and d2r converts a d into an r. g2r converts Hedge's g to a correlation. t2r converts a t test to r, r2t converts a correlation to a t-test value. chi2r converts a chi square to r, r2chi converts it back. r2c and cor2cov convert a correlation matrix to a covariance matrix. d2t and t2d convert cohen's d into a t and a t into a cohen d. See `cohen.d` for other conversions.

Usage

```
fisherz(rho)
fisherz2r(z)
r.con(rho,n,p=.95,twotailed=TRUE)
r2t(rho,n)
t2r(t,df)
g2r(g,df,n)
chi2r(chi2,n)
r2chi(rho,n)
r2c(rho,sigma)
cor2cov(rho,sigma)
r2p(rho,n)
```

Arguments

`rho`	a Pearson r
`z`	A Fisher z
`n`	Sample size for confidence intervals
`df`	degrees of freedom for t, or g
`p`	Confidence interval

twotailed	Treat p as twotailed p
g	An effect size (Hedge's g)
t	A student's t value
chi2	A chi square
sigma	a vector of standard deviations to be used to convert a correlation matrix to a covariance matrix

Value

z	value corresponding to r (fisherz)
r	r corresponding to z (fisherz2r)
r.con	lower and upper p confidence intervals (r.con)
t	t with n-2 df (r2t)
r	r corresponding to effect size d or d corresponding to r.
r2c	r2c is the reverse of the cor2con function of base R. It just converts a correlation matrix to the corresponding covariance matrix given a vector of standard deviations.

Author(s)

Maintainer: William Revelle <revelle@northwestern.edu >

Examples

```
n <- 30
r <- seq(0,.9,.1)
d <- r2d(r)
rc <- matrix(r.con(r,n),ncol=2)
t <- r*sqrt(n-2)/sqrt(1-r^2)
p <- (1-pt(t,n-2))*2
r1 <- t2r(t,(n-2))
r2 <- d2r(d)
chi <- r2chi(r,n)
r3 <- chi2r(chi,n)
r.rc <- data.frame(r=r,z=fisherz(r),lower=rc[,1],upper=rc[,2],t=t,p=p,d=d,
    chi2=chi,d2r=r2,t2r=r1,chi2r=r3)
round(r.rc,2)
```

`fparse` *Parse and extend formula input from a model and return the DV(s), IV(s), and associated terms.*

Description

Formula input from e.g., lm, may be extended to include mediators, quadratic and partial terms using a standard syntax. This is used by (e.g.,) `lmCor`, `mediate`, and `error.bars.by`. `lavParse` will take a limited number of lavaan syntax commands and return as a model matrix for `CFA`, `CFA.bifactor`, `sim`, and `sim.structure`

Usage

```
fparse(expr)
lavParse(model, phi=FALSE)
```

Arguments

`expr`	A legitimate expression in the form y ~ x1 , etc. (see details)
`model`	A lavaan style factor specification. (see examples)
`phi`	If TRUE, the model can include the factor correlations.

Details

The basic formula input given as DV1 + DV2 ~ IV1 + IV2 + (IV3) + I(IV4^2) - IV5 will be parsed to return 2 DVs (1 and 2), two normal IVs (1 and 2), a mediator (IV3) a quadratic (IV4) and a variable to be partialed (IV5). See the various examples in `lmCor` and `mediate`.

To allow rapid comparisons with sem and cfa output from lavaan, it is convenient to use lavaan style syntax.

`CFA` and`CFA` can specify the model using lavaan syntax as well as the item keys format.

Value

`v.mat`	A matrix of variable weights for each factor
`y`	A list of elements from the left side of the formula
`x`	A list of elements from the right side of the formula
`m`	A list of those elements of the formula included in ()
`prod`	A list of elements separated by a * sign
`ex`	A list of elements marked by I()
`model`	A matrix with cols for the factors, rows for the variable, values specified

Author(s)

William Revelle

Examples

```
fparse(DV  ~ IV1 + IV2 * IV2*IV3 + (IV4) + I(IV5^2) )
#somewhat more complicated
fparse(DV1 + DV2 ~ IV1 + IV2 + IV3*IV4 + I(IV5^2) + I(Iv6^2) + (IV7) + (IV8) - IV9)


model <- ' F1 =~ .9*V1 + .8*V2 + .7*V3
          F2 =~ .8 * V4 + .7*V5 + .6*V6
         F3 =~ .9*V7 + .7*V8 + .5*V9
         F1 ~ .3 * F2 + .6* F3
         F2 ~ .5*F3'
lavParse(model,phi=TRUE)   #will return a 9 x 3 matrix loadings and 3x3 phi matrix for sim

#by specifying the coefficients, we can run sim more readily
sim.data <-sim(model,n=500)
cx <- CFA(model,sim.data$observed)

#The CFA function will not use the coefficients nor the phi matrix, just the path model
#So we can use the same model for simulations and CFA
#But, lavaan needs to drop the coefficients
modelg <- 'g =~ .4*V1 +.4*V2+.4*V3 +.4*V4+.4*V5+.4*V6 +.4*V7+.4*V8+.4*V9
F1 =~ .9*V1 + .8*V2 + .7*V3
          F2 =~ .8 * V4 + .7*V5 + .6*V6
         F3 =~ .9*V7 + .7*V8 + .5*V9   '

modelg <- 'g=~V1 + V2 + V3 + V4 + V5 + V6 + V7 + V8 +V9
     F1 =~  V1 + V2 + V3
      F2 =~  V4 + V5 + V6
        F3 =~ V7 + V8 + V9'
lavParse(modelg)
#model without coefficients to run in lavaan
```

`fsi` *Factor Score Indeterminacy estimates)*

Description

Factor analysis either using Exploratory Factor Analysis `fa` or Confirmatory Factor analysis `CFA` estimates factor loadings to best fit a latent variable model that is underdetermined. An infinite number of factor scores from such a model all will fit equally well (Grice, 2001, Nicewander, 2020, Waller, 2023). `fsi` calculates an estimated R2 of observed factor scores with latent factors using either the Nicewander approach (default) or Grice approach (to match previous versions.)

Usage

```
fsi(f, phi = NULL, r = NULL, Grice = FALSE,short=TRUE)
```

Arguments

f	A factor loading matrix
phi	Factor correlations
r	The correlations of the raw data – needed for the Grice option
Grice	If TRUE, find the Grice (2001) solution.
short	By default, return a vector of R2 values for each factor. Alternatively report the R2 values and H matrix (see Nicewander, 2020).

Details

A dirty little secret of factor analysis is that the factor scores are not perfectly correlated with their corresponding factors. In fact, there are an infinite number of factor score solutions that all fit equally well. The factor scores are, in fact, indeterminant. Although known for almost 100 years, the lack of determinancy is rarely reported. "for any set of p, imperfectly measured variables that fit Spearman's model, a research can construct an infinite set of factor scores, each of which fits the model perfectly." Waller,2023, p 245.

Reviews of the factor score inderminancy (FSI) problem and possible solutions include Grice (2001), Nicewander (2020) and Waller (2023).

The Grice (2001) approach is highly cited, but recent work by Nicewander (2020) and Waller (2023) suggests an alternative approach, which is more highly grounded in statistical theory. `fsi` will report both of these solutions. Nicewander approach is used in the `CFA` function and has replaced (as of version 2.6.1) the Grice approach used in `fa`.

The `fsi` will report the R2 between putative factor scores and the latent factors using either the Nicewander (2020) or the Grice (2001) approach. Prior versions of `fa` in psych reported just the Grice solution. As of psych_2.6.1 the default for `fa` is to report the Nicewander estimate, but with an option to use the Grice estimate.

Value

R2	Muliple R2 of the factor model with the factors
H	The first nfactor elments of the H matrix of factor correlations in the nvar space. See Nicewander, 2020.

Author(s)

William Revelle

References

Grice, James W. (2001) Computing and evaluating factors scores. Psychological Methods, 6, p 430-450. 10.1037/1082-989X.6.4.430

Nicewander, W. Alan (2020) A pespective on the mathematical and psychometric aspects of factor indeterminacy. Multivariate Behavioral Research. 55 DOI 10.1080/00273171.2019.1684872)

Waller, N. G. (2023) Breaking our silence on factor score indeterminacy. Journal of Educational and Behavioral Statistics. 48, p 244-261. doi; 10.3102/1076998622112.

See Also

fa and CFA.

Examples

```
#matrix from Waller example for fsIndeterminacy from fungible
Lambda <- matrix(c(.8,  0,
                   .7,  0,
                   .6,  0,
                    0, .5,
                    0, .4,
                    0, .3), 6, 2, byrow=TRUE)

 fsi(Lambda)  #should be .7675 and .3837

 A <- matrix( c(.8, 0,
                .8, 0,
                 0,  .6,
                 0,  .6), 4,2, byrow=TRUE)

 fsi(A)   #should be .781 .529  (from Nicewander Appendix 1)
```

Garcia *Data from the sexism (protest) study of Garcia, Schmitt, Branscome, and Ellemers (2010)*

Description

Garcia, Schmitt, Branscombe, and Ellemers (2010) report data for 129 subjects on the effects of perceived sexism on anger and liking of women's reactions to ingroup members who protest discrimination. This data set is also used as the 'protest' data set by Hayes (2013 and 2018). It is a useful example of mediation and moderation in regression. It may also be used as an example of plotting interactions.

Usage

```
data("GSBE")
```

Format

A data frame with 129 observations on the following 6 variables.

`protest` 0 = no protest, 1 = Individual Protest, 2 = Collective Protest

`sexism` Means of an 8 item Modern Sexism Scale.

`anger` Anger towards the target of discrimination. "I feel angry towards Catherine".

`liking` Mean rating of 6 liking ratings of the target.

`respappr` Mean of four items of appropriateness of the target's response.

`prot2` A recoding of protest into two levels (to match Hayes, 2013).

Details

The reaction of women to women who protest discriminatory treatment was examined in an experiment reported by Garcia et al. (2010). 129 women were given a description of sex discrimination in the workplace (a male lawyer was promoted over a clearly more qualified female lawyer). Subjects then read that the target lawyer felt that the decision was unfair. Subjects were then randomly assigned to three conditions: Control (no protest), Individual Protest ("They are treating me unfairly") , or Collective Protest ("The firm is is treating women unfairly").

Participants were then asked how much they liked the target (liking), how angry they were to the target (anger) and to evaluate the appropriateness of the target's response (respappr).

Garcia et al (2010) report a number of interactions (moderation effects) as well as moderated-mediation effects.

This data set is used as an example in Hayes (2013) for moderated mediation. It is used here to show how to do moderation (interaction terms) in regression (see `setCor`) , how to do moderated mediation (see `mediate`) and how draw interaction graphs (see help).

Source

The data were downloaded from the webpages of Andrew Hayes (https://www.afhayes.com/public/hayes2018data.zip) supporting the first and second edition of his book. The second edition includes 6 variables, the first, just four. The protest variable in 2018 has three levels, but just two in the 2013 source.

The data are used by kind permission of Donna M. Garcia, Michael T. Schmitt, Nyla R. Branscombe, and Naomi Ellemers.

References

Garcia, Donna M. and Schmitt, Michael T. and Branscombe, Nyla R. and Ellemers, Naomi (2010). Women's reactions to ingroup members who protest discriminatory treatment: The importance of beliefs about inequality and response appropriateness. European Journal of Social Psychology, (40) 733-745.

Hayes, Andrew F. (2013) Introduction to mediation, moderation, and conditional process analysis: A regression-based approach. Guilford Press.

Examples

```
data(GSBE)  #alias to Garcia data set

## Just do regressions with interactions
lmCor(respappr ~ prot2 * sexism,std=FALSE,data=Garcia,main="Moderated (mean centered )")
lmCor(respappr ~ prot2 * sexism,std=FALSE,data=Garcia,main="Moderated (don't center)", zero=FALSE)
#demonstrate interaction plots
plot(respappr ~ sexism, pch = 23- protest, bg = c("black","red", "blue")[protest],
data=Garcia, main = "Response to sexism varies as type of protest")
by(Garcia,Garcia$protest, function(x) abline(lm(respappr ~ sexism,
   data =x),lty=c("solid","dashed","dotted")[x$protest+1]))
text(6.5,3.5,"No protest")
text(3,3.9,"Individual")
text(3,5.2,"Collective")
```

```
#compare two models  (bootstrapping n.iter set to 50 for speed
# 1) mean center the variables prior to taking product terms
mod1 <- mediate(respappr ~ prot2 * sexism +(sexism),data=Garcia,n.iter=50
 ,main="Moderated mediation (mean centered)")
# 2) do not mean center
mod2 <- mediate(respappr ~ prot2 * sexism +(sexism),data=Garcia,zero=FALSE, n.iter=50,
    main="Moderated  mediation (not centered")

summary(mod1)
summary(mod2)
```

`geometric.mean` *Find the geometric mean of a vector or columns of a data.frame.*

Description

The geometric mean is the nth root of n products or e to the mean log of x. Useful for describing non-normal, i.e., geometric distributions.

Usage

```
geometric.mean(x,na.rm=TRUE)
```

Arguments

`x`	a vector or data.frame
`na.rm`	remove NA values before processing

Details

Useful for teaching how to write functions, also useful for showing the different ways of estimating central tendency.

Value

geometric mean(s) of x or x.df.

Note

Not particularly useful if there are elements that are <= 0.

Author(s)

William Revelle

See Also

harmonic.mean, mean

Examples

```
x <- seq(1,5)
x2 <- x^2
x2[2] <- NA
X <- data.frame(x,x2)
geometric.mean(x)
geometric.mean(x2)
geometric.mean(X)
geometric.mean(X,na.rm=FALSE)
```

---

glb.algebraic *Find the greatest lower bound to reliability.*

Description

The greatest lower bound solves the "educational testing problem". That is, what is the reliability of a test? (See guttman for a discussion of the problem). Although there are many estimates of a test reliability (Guttman, 1945) most underestimate the true reliability of a test.

For a given covariance matrix of items, C, the function finds the greatest lower bound to reliability of the total score using the csdp function from the Rcsdp package.

Usage

```
glb.algebraic(Cov, LoBounds = NULL, UpBounds = NULL)
```

Arguments

Cov	A p * p covariance matrix. Positive definiteness is not checked.
LoBounds	A vector $l = (l_1, \ldots, l_p)$ of length p with lower bounds to the diagonal elements x_i. The default l=(0, . . . , 0) does not imply any constraint, because positive semidefiniteness of the matrix $\tilde{C} + Diag(x)$ implies $0 \leq x_i$
UpBounds	A vector u =(u1, . . . , up) of length p with upper bounds to the diagonal elements xi. The default is u = v.

Details

If C is a p * p-covariance matrix, v = diag(C) its diagonal (i. e. the vector of variances $v_i = c_{ii}$), $\tilde{C} = C - Diag(v)$ is the covariance matrix with 0s substituted in the diagonal and x = the vector $x_1, \ldots, x_n$ the educational testing problem is (see e. g., Al-Homidan 2008)

$$\sum_{i=1}^{p} x_i \to \min$$

s.t.

$$\tilde{C} + Diag(x) \geq 0$$

(i.e. positive semidefinite) and $x_i \leq v_i, i = 1, \ldots, p$. This is the same as minimizing the trace of the symmetric matrix

$$\tilde{C} + diag(x) = \begin{pmatrix} x_1 & c_{12} & \ldots & c_{1p} \\ c_{12} & x_2 & \ldots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{1p} & c_{2p} & \ldots & x_p \end{pmatrix}$$

s. t. $\tilde{C} + Diag(x)$ is positive semidefinite and $x_i \leq v_i$.

The greatest lower bound to reliability is

$$\frac{\sum_{ij} \bar{c_{ij}} + \sum_i x_i}{\sum_{ij} c_{ij}}$$

Additionally, function glb.algebraic allows the user to change the upper bounds $x_i \leq v_i$ to $x_i \leq u_i$ and add lower bounds $l_i \leq x_i$.

The greatest lower bound to reliability is applicable for tests with non-homogeneous items. It gives a sharp lower bound to the reliability of the total test score.

Caution: Though glb.algebraic gives exact lower bounds for exact covariance matrices, the estimates from empirical matrices may be strongly biased upwards for small and medium sample sizes.

glb.algebraic is wrapper for a call to function csdp of package Rcsdp (see its documentation).

If Cov is the covariance matrix of subtests/items with known lower bounds, rel, to their reliabilities (e. g. Cronbachs α), LoBounds can be used to improve the lower bound to reliability by setting LoBounds <- rel*diag(Cov).

Changing UpBounds can be used to relax constraints $x_i \leq v_i$ or to fix x_i-values by setting LoBounds[i] < -z; UpBounds[i] <- z.

Value

`glb`	The algebraic greatest lower bound
`solution`	The vector x of the solution of the semidefinite program. These are the elements on the diagonal of C.
`status`	Status of the solution. See documentation of csdp in package Rcsdp. If status is 2 or greater or equal than 4, no glb and solution is returned. If status is not 0, a warning message is generated.
`Call`	The calling string

Author(s)

Andreas Moltner
Center of Excellence for Assessment in Medicine/Baden-Wurttemberg
University of Heidelberg

William Revelle
Department of Psychology
Northwestern University Evanston, Illiniois
https://personality-project.org/revelle.html

References

Al-Homidan S (2008). Semidefinite programming for the educational testing problem. Central European Journal of Operations Research, 16:239-249.

Bentler PM (1972) A lower-bound method for the dimension-free measurement of internal consistency. Soc Sci Res 1:343-357.

Fletcher R (1981) A nonlinear programming problem in statistics (educational testing). SIAM J Sci Stat Comput 2:257-267.

Shapiro A, ten Berge JMF (2000). The asymptotic bias of minimum trace factor analysis, with applications to the greatest lower bound to reliability. Psychometrika, 65:413-425.

ten Berge, Socan G (2004). The greatest bound to reliability of a test and the hypothesis of unidimensionality. Psychometrika, 69:613-625.

See Also

For an alternative estimate of the greatest lower bound, see `glb.fa`. For multiple estimates of reliablity, see `guttman`

Examples

```
Cv<-matrix(c(215, 64, 33, 22,
             64, 97, 57, 25,
             33, 57,103, 36,
             22, 25, 36, 77),ncol=4)

Cv                    # covariance matrix of a test with 4 subtests
Cr<-cov2cor(Cv)       # Correlation matrix of tests
if(!require(Rcsdp)) {print("Rcsdp must be installed to find the glb.algebraic")} else {
 glb.algebraic(Cv)     # glb of total score
glb.algebraic(Cr)      # glb of sum of standardized scores

 w<-c(1,2,2,1)         # glb of weighted total score
 glb.algebraic(diag(w) %*% Cv %*% diag(w))
alphas <- c(0.8,0,0,0) # Internal consistency of first test is known

glb.algebraic(Cv,LoBounds=alphas*diag(Cv))

                      # Fix all diagonal elements to 1 but the first:

lb <- glb.algebraic(Cr,LoBounds=c(0,1,1,1),UpBounds=c(1,1,1,1))
lb$solution[1]        # should be the same as the squared mult. corr.
smc(Cr)[1]
}
```

`Gleser` *Example data from Gleser, Cronbach and Rajaratnam (1965) to show basic principles of generalizability theory.*

Description

Gleser, Cronbach and Rajaratnam (1965) discuss the estimation of variance components and their ratios as part of their introduction to generalizability theory. This is a adaptation of their "illustrative data for a completely matched G study" (Table 3). 12 patients are rated on 6 symptoms by two judges. Components of variance are derived from the ANOVA.

Usage

```
data(Gleser)
```

Format

A data frame with 12 observations on the following 12 variables. J item by judge:

`J11` a numeric vector

`J12` a numeric vector

`J21` a numeric vector

`J22` a numeric vector

`J31` a numeric vector

`J32` a numeric vector

`J41` a numeric vector

`J42` a numeric vector

`J51` a numeric vector

`J52` a numeric vector

`J61` a numeric vector

`J62` a numeric vector

Details

Generalizability theory is the application of a components of variance approach to the analysis of reliability. Given a G study (generalizability) the components are estimated and then may be used in a D study (Decision). Different ratios are formed as appropriate for the particular D study.

Source

Gleser, G., Cronbach, L., and Rajaratnam, N. (1965). Generalizability of scores influenced by multiple sources of variance. Psychometrika, 30(4):395-418. (Table 3, rearranged to show increasing patient severity and increasing item severity.

References

Gleser, G., Cronbach, L., and Rajaratnam, N. (1965). Generalizability of scores influenced by multiple sources of variance. Psychometrika, 30(4):395-418.

Examples

```
#Find the MS for each component:
#First, stack the data
data(Gleser)
stack.g <- stack(Gleser)
st.gc.df <- data.frame(stack.g,Persons=rep(letters[1:12],12),
Items=rep(letters[1:6],each=24),Judges=rep(letters[1:2],each=12))
#now do the ANOVA
anov <- aov(values ~ (Persons*Judges*Items),data=st.gc.df)
summary(anov)
```

`Gorsuch` *Example data set from Gorsuch (1997) for an example factor extension.*

Description

Gorsuch (1997) suggests an alternative to the classic Dwyer (1937) factor extension technique. This data set is taken from that article. Useful for comparing `link{fa.extension}` with and without the correct=TRUE option.

Usage

```
data(Gorsuch)
```

Details

Gorsuc (1997) suggested an alternative model for factor extension. His method is appropriate for the case of repeated variables. This is handled in `link{fa.extension}` with correct=FALSE

Source

Richard L. Gorsuch (1997) New Procedure for Extension Analysis in Exploratory Factor Analysis. Educational and Psychological Measurement, 57, 725-740.

References

Dwyer, Paul S. (1937), The determination of the factor loadings of a given test from the known factor loadings of other tests. Psychometrika, 3, 173-178

Examples

```
data(Gorsuch)

Ro <- Gorsuch[1:6,1:6]
Roe <- Gorsuch[1:6,7:10]
fo <- fa(Ro,2,rotate="none")
fa.extension(Roe,fo,correct=FALSE)
```

Harman *Six data sets from Harman (1967). 9 cognitive variables from Holzinger and 8 emotional variables from Burt*

Description

Classic data sets reported by Harman (1967) are 9 psychological (cognitive) variables taken from Holzinger and 8 emotional variables taken from Burt. Two others are socioeconomic and political data sets. Additionally, 8 physical variables. All five of these are used for tests and demonstrations of various factoring algortithms. One set of five variables (Haman_5) is included as a demonstration of a Heywood case.

Usage

```
data(Harman)
data(Harman.5)
data(Harman_5)
data(Harman.political)
data(Harman.8)
```

Details

- Harman.Holzinger: 9 x 9 correlation matrix of ability tests, N = 696.
- Harman.Burt: a 8 x 8 correlation matrix of “emotional" items. N = 172
- Harman.5: 12 census tracts for 5 socioeconomic data (Harman p 14)
- Harman_5: 5 variables with a Heywood case (see Cooperman and Waller, 2022)
- Harman.political: p 166.
- Harman.8 8 physical measures

Harman.Holzinger. The nine psychological variables from Harman (1967, p 244) are taken from unpublished class notes of K.J. Holzinger with 696 participants. This is a subset of 12 tests with 4 factors. It is yet another nice example of a bifactor solution. Bentler (2007) uses this data set to discuss reliablity analysis. The data show a clear bifactor structure and are a nice example of the various estimates of reliability included in the `omega` function. Should not be confused with the `Holzinger` or `Holzinger.9` data sets in `bifactor`.

See also the Holzinger-Swineford data set of 301 subjects with 26 variables in `holzinger.swineford`. These data were provided by Keith Widaman. "The Holzinger and Swineford (1939) data have been

used as a model data set by many investigators. For example, Harman (1976) used the "24 Psychological Variables" example prominently in his authoritative text on multiple factor analysis, and the data presented under this rubric consisted of 24 of the variables from the Grant-White school (N = 145). Meredith (1964a, 1964b) used several variables from the Holzinger and Swineford study in his work on factorial invariance under selection. Joreskog (1971) based his work on multiple-group confirmatory factor analysis using the Holzinger and Swineford data, subsetting the data into four groups.

Rosseel, who developed the "lavaan" package for R , included 9 of the manifest variables from Holzinger and Swineford (1939) as a "resident" data set when one downloads the lavaan package. Several background variables are included in this "resident" data set in addition to 9 of the psychological tests (which are named x1 - x9 in the data set). When analyzing these data, I found the distributions of the variables (means, SDs) did not match the sample statistics from the original article. For example, in the "resident" data set in lavaan, scores on all manifest variables ranged between 0 and 10, sample means varied between 3 and 6, and sample SDs varied between 1.0 and 1.5. In the original data set, scores ranges were rather different across tests, with some variables having scores that ranged between 0 and 20, but other manifest variables having scores ranging from 50 to over 300 - with obvious attendant differences in sample means and SDs."

Harman.Burt. Eight “emotional" variables are taken from Harman (1967, p 164) who in turn adapted them from Burt (1939). They are said be from 172 normal children aged nine to twelve. As pointed out by Harman, this correlation matrix is singular and has squared multiple correlations > 1. Because of this problem, it is a nice test case for various factoring algorithms. (For instance, omega will issue warning messages for fm="minres" or fm="pa" but will fail for fm="ml".)

The Eight Physical Variables problem is taken from Harman (1976) and represents the correlations between eight physical variables for 305 girls. The two correlated clusters represent four measures of "lankiness" and then four measures of "stockiness". The original data were selected from 17 variables reported in an unpublished dissertation by Mullen (1939).

Variable 6 ("Bitrochanteric diamter") is the distance between the outer points of the hips.

The row names match the original Harman paper, the column names have been abbreviated.

See also the `usaf` data set for other physical measurements.

The `fa` solution for principal axes (fm="pa") matches the reported minres solution, as does the fm="minres".

For those interested in teaching examples using various body measurements, see the body data set in the gclus package.

The Burt data set probably has a typo in the original correlation matrix. Changing the Sorrow-Tenderness correlation from .87 to .81 makes the correlation positive definite.

As pointed out by Jan DeLeeuw, the Burt data set is a subset of 8 variables from the original 11 reported by Burt in 1915. That matrix has the same problem. See `burt`.

Other example data sets that are useful demonstrations of factor analysis are the seven bifactor examples in `Bechtoldt` and the 24 ability measures in `Harman74.cor`

There are several other Harman examples in the psych package (i.e., Harman.8) as well as in the dataseta and GPArotation packages. The Harman 24 mental tests problem is in the basic datasets package at Harman74.cor.

Other Harman data sets are 5 socioeconomic variables for 12 census tracts Harman.5 used by John Loehlin as an example for EFA. Another one of the many Harman (1967) data sets is Harman.political. This contains 8 political variables taken over 147 election areas. The principal factor

method with SMCs as communalities match those of table 8.18. The data are used by Dziubian and Shirkey as an example of the Kaiser-Meyer-Olkin test of factor adequacy.

The Harman_5 correlation matrix is discussed by Cooperman and Waller and is nice example of a Heywood case.

Source

Harman (1967 p 164 and p 244.)

H. Harman and W.Jones. (1966) Factor analysis by minimizing residuals (minres). Psychometrika, 31(3):351-368.

References

Harman, Harry Horace (1967), Modern factor analysis. Chicago, University of Chicago Press. Bentler, P. Covariance structure models for maximal reliability of unit-weighted composites. In Handbook of latent variable and related models, pages 1–17. North Holland, 2007.

Burt, C. General and Specific Factors underlying the Primary Emotions. Reports of the British Association for the Advancement of Science, 85th meeting, held in Manchester, September 7-11, 1915. London, John Murray, 1916, p. 694-696 (retrieved from the web at https://www.biodiversitylibrary.org/item/95822#790

Cooperman, A.W. and N. G. Waller (2022) Heywood You Go Away! Examining Causes, Effects, and Treatments for Heywood Cases in Exploratory Factor Analysis, Psychological Methods, (27) 156-176.

See Also

See also the original `burt` data set, the Harman.5 and Harman.political data sets.

Examples

```
data(Harman)
cor.plot(Harman.Holzinger)
cor.plot(Harman.Burt)
smc(Harman.Burt)  #note how this produces impossible results
f2 <- fa(Harman.8,2, rotate="none")  #minres matches Harman and Jones
```

`harmonic.mean` *Find the harmonic mean of a vector, matrix, or columns of a data.frame*

Description

The harmonic mean is merely the reciprocal of the arithmetic mean of the reciprocals.

Usage

```
harmonic.mean(x,na.rm=TRUE,zero=TRUE)
```

Arguments

x	a vector, matrix, or data.frame
na.rm	na.rm=TRUE remove NA values before processing
zero	If TRUE, then if there are any zeros, return 0, else, return the harmonic mean of the non-zero elements

Details

Included as an example for teaching about functions. As well as for a discussion of how to estimate central tendencies. Also used in statsBy to weight by the harmonic mean.

Values of 0 can be included (in which case the harmonic.mean = 0) or converted to NA according to the zero option.

Added the zero option, March, 2017.

Value

The harmonic mean(s)

Note

Included as a simple demonstration of how to write a function

Examples

```
x <- seq(1,5)
x2 <- x^2
x2[2] <- NA
y <- x - 1
X <- data.frame(x,x2,y)
harmonic.mean(x)
harmonic.mean(x2)
harmonic.mean(X)
harmonic.mean(X,na.rm=FALSE)
harmonic.mean(X,zero=FALSE)
```

---

headTail *Combine calls to head and tail*

Description

A quick way to show the first and last n lines of a data.frame, matrix, or a text object. Just a pretty call to head and tail or View

Usage

```
headTail(x, top=4,bottom=4,from=1,to=NULL, digits=2, hlength = 4, tlength =4,
ellipsis=TRUE)
headtail(x,hlength=4,tlength=4,digits=2,ellipsis=TRUE,from=1,to=NULL)
topBottom(x, top=4,bottom=4,from=1,to=NULL, digits=2, hlength = 4, tlength = 4)
quickView(x,top=8,bottom=8,from=1,to=NULL)
```

Arguments

x	A matrix or data frame or free text
top	The number of lines at the beginning to show
bottom	The number of lines at the end to show
digits	Round off the data to digits
ellipsis	Separate the head and tail with dots (ellipsis)
from	The first column to show (defaults to 1)
to	The last column to show (defaults to the number of columns)
hlength	The number of lines at the beginning to show (an alias for top)
tlength	The number of lines at the end to show (an alias for bottom)

Value

The first top and last bottom lines of a matrix or data frame with an ellipsis in between. If the input is neither a matrix nor data frame, the output will be the first top and last bottom lines.

For each line, just columns starting at from and going to to will be displayed. Bt default, from = 1 and to = the last column.

topBottom is just a call to headTail with ellipsis = FALSE and returning a matrix output.

quickView is a call to `View` which opens a viewing window which is scrollable (if needed because the number of lines listed is more than a screen's worth). View (and therefore quickView) is slower than `headTail` or `topBottom`.

See Also

`head` and `tail`

Examples

```
if(require(psychTools)) {
headTail(psychTools::iqitems,4,8,to=6) #the first 4 and last 6 items from 1 to 6
topBottom(psychTools::ability,from =2, to = 6) #the first and last 4 items from 2 to 6
}
headTail(bfi,top=4, bottom=4,from =6,to=10) #the first and last 4 from 6 to 10
#not shown
#quickView(ability,hlength=10,tlength=10)  #this loads a spreadsheet like table
```

ICC *Intraclass Correlations (ICC1, ICC2, ICC3 from Shrout and Fleiss)*

Description

The Intraclass correlation is used as a measure of association when studying the reliability of raters. Shrout and Fleiss (1979) outline 6 different estimates, that depend upon the particular experimental design. All are implemented and given confidence limits. Uses either aov or lmer depending upon options. lmer allows for missing values.

Usage

```
ICC(x,missing=TRUE,alpha=.05,lmer=TRUE,check.keys=FALSE)
```

Arguments

x	a matrix or dataframe of ratings
missing	if TRUE, remove missing data – work on complete cases only (aov only)
alpha	The alpha level for significance for finding the confidence intervals
lmer	Should we use the lmer function from lme4? This handles missing data and gives variance components as well. TRUE by default.
check.keys	If TRUE reverse those items that do not correlate with total score. This is not done by default.

Details

Shrout and Fleiss (1979) consider six cases of reliability of ratings done by k raters on n targets. McGraw and Wong (1996) consider 10, 6 of which are identical to Shrout and Fleiss and 4 are conceptually different but use the same equations as the 6 in Shrout and Fleiss.

The intraclass correlation is used if raters are all of the same “class". That is, there is no logical way of distinguishing them. Examples include correlations between pairs of twins, correlations between raters. If the variables are logically distinguishable (e.g., different items on a test), then the more typical coefficient is based upon the inter-class correlation (e.g., a Pearson r) and a statistic such as alpha or omega might be used. alpha and ICC3k are identical.

Where the data are laid out in terms of Rows (subjects) and Columns (rater or tests), the various ICCs are found by the ratio of various estimates of variance components. In all cases, subjects are taken as varying at random, and the residual variance is also random. The distinction between models 2 and 3 is whether the judges (items/tests) are seen as random or fixed. A further distinction is whether the emphasis is upon absolute agreement of the judges, or merely consistency.

As discussed by Liljequist et al. (2019), McGraw and Wong lay out 5 models which use just three forms of the ICC equations.

Model 1 is a one way model with

ICC1: Each target is rated by a different judge and the judges are selected at random.

$$ICC(1,1) = \rho_{1,1} = \frac{\sigma_r^2}{\sigma_r^2 + \sigma_w^2}$$

(This is a one-way ANOVA fixed effects model and is found by (MSB- MSW)/(MSB+ (nr-1)*MSW))

ICC2: A random sample of k judges rate each target. The measure is one of absolute agreement in the ratings.

$$ICC(2,1) = \rho_{2,1} = \frac{\sigma_r^2}{\sigma_r^2 + \sigma_c^2 + \sigma_{rc}^2 + \sigma_e^2}$$

Found as (MSB- MSE)/(MSB + (nr-1)*MSE + nr*(MSJ-MSE)/nc)

ICC3: A fixed set of k judges rate each target. There is no generalization to a larger population of judges.

$$ICC(3,1) = \rho_{3,1} = \frac{\sigma_r^2}{\sigma_r^2 + \sigma_c^2 + \sigma_e^2}$$

(MSB - MSE)/(MSB+ (nr-1)*MSE)

Then, for each of these cases, is reliability to be estimated for a single rating or for the average of k ratings? (The 1 rating case is equivalent to the average intercorrelation, the k rating case to the Spearman Brown adjusted reliability.)

ICC1 is sensitive to differences in means between raters and is a measure of absolute agreement.

ICC2 and ICC3 remove mean differences between judges, but are sensitive to interactions of raters by judges. The difference between ICC2 and ICC3 is whether raters are seen as fixed or random effects.

ICC1k, ICC2k, ICC3K reflect the means of k raters.

If using the lmer option, then missing data are allowed. In addition the lme object returns the variance decomposition. (This is simliar to `testRetest` which works on the items from two occasions.

The check.keys option by default reverses items that are negatively correlated with total score. A message is issued.

Value

`results`	A matrix of 6 rows and 8 columns, including the ICCs, F test, p values, and confidence limits
`summary`	The anova summary table or the lmer summary table
`stats`	The anova statistics (converted from lmer if using lmer)
`MSW`	Mean Square Within based upon the anova
`lme`	The variance decomposition if using the lmer option

Note

The results for the Lower and Upper Bounds for ICC(2,k) do not match those of SPSS 9 or 10, but do match the definitions of Shrout and Fleiss. SPSS seems to have been using the formula in McGraw and Wong, but not the errata on p 390. They seem to have fixed it in more recent releases (15).

Starting with psych 1.4.2, the confidence intervals are based upon (1-alpha)% at both tails of the confidence interval. This is in agreement with Shrout and Fleiss. Prior to 1.4.2 the confidence intervals were (1-alpha/2)%. However, at some point, this error slipped back again. It has been fixed in version 1.9.5 (5/21/19).

However, following some very useful comments from Mijke Rhumtulla, I should divide by 2 after all. This has been fixed (again) 3/05/22 for version 2.2.2.

An occasionally asked question: How do these 6 measures compare to the 10 ICCsreported by SPSS? The answer is that the 6 Shrout and Fleiss are the same as the 10 by McGraw and Wong.

A very helpful discussion of the formulae used in the calculations is by Liljequist et al, 2019.

Although M & W distinguish between two way random effects and two way mixed effects for consistency of a single rater, examining their formula show that both of these are S& F ICC(3,1).

Similarly, the S&F ICC(2,1) is called both Two way random effects, absolute agreement and Two way mixed effects, absolute agreement by M&W.

The same confusion occurs with multiple raters (3,k) are both two way random effects and mixed effects for consistency and (2,k) are both two way random and two way mixed absolute agreement.

As M&W say "The importance of the random-fixed effects distinction is in its effect on the interpretation, but not calculation, of an ICC. Namely, when levels of the column factor are randomly sampled, one can generalize beyond one's data, but not when they are fixed. In either case, however, the value of the ICC is the same," (p 37)

To quote from IBM support:

"Problem

I'm using the SPSS RELIABILITY procedure to compute intraclass correlation coefficients (ICCs), and am trying to make sure I understand the correspondence between the measures available in SPSS and those discussed in Shrout and Fleiss. How do the two sets of models relate to each other?

Resolving The Problem Shrout and Fleiss (1979) discussed six ICC measures, which consist of pairs of measures for the reliability of a single rating and that of the average of k ratings (where k is the number of raters) for three different models: the one-way random model (called Case 1), the two-way random model (Case 2), and the two-way mixed model (Case 3). The measures implemented in SPSS were taken from McGraw and Wong (1996), who discussed these six measures, plus four others. The additional ones in McGraw and Wong are actually numerically identical to the four measures for the two-way cases discussed by Shrout and Fleiss, differing only in their interpretation.

The correspondence between the measures available in SPSS and those discussed in Shrout and Fleiss is as follows:

S&F(1,1) = SPSS One-Way Random Model, Absolute Agreement, Single Measure

S&F(1,k) = SPSS One-Way Random Model, Absolute Agreement, Average Measure

S&F(2,1) = SPSS Two-Way Random Model, Absolute Agreement, Single Measure

S&F(2,k) = SPSS Two-Way Random Model, Absolute Agreement, Average Measure

S&F(3,1) = SPSS Two-Way Mixed Model, Consistency, Single Measure

S&F(3,k) = SPSS Two-Way Mixed Model, Consistency, Average Measure

SPSS also offers consistency measures for the two-way random case (numerically equivalent to the consistency measures for the two-way mixed case, but differing in interpretation), and absolute agreement measures for the two-way mixed case (numerically equivalent to the absolute agreement measures for the two-way random case, again differing in interpretation)

Author(s)

William Revelle

References

Shrout, Patrick E. and Fleiss, Joseph L. Intraclass correlations: uses in assessing rater reliability. Psychological Bulletin, 1979, 86, 420-3428.

McGraw, Kenneth O. and Wong, S. P. (1996), Forming inferences about some intraclass correlation coefficients. Psychological Methods, 1, 30-46. + errata on page 390.

Liljequist David, Elfving Britt and Skavberg Kirsti (2019) Intraclass correlation-A discussion and demonstration of basic features. PLoS ONE 14(7): e0219854. https://doi.org/10.1371/journal.

Revelle, W. (in prep) An introduction to psychometric theory with applications in R. Springer. (working draft available at https://personality-project.org/r/book/

Examples

```
sf <- matrix(c(
9,    2,   5,    8,
6,    1,   3,    2,
8,    4,   6,    8,
7,    1,   2,    6,
10,   5,   6,    9,
6,   2,   4,    7),ncol=4,byrow=TRUE)
colnames(sf) <- paste("J",1:4,sep="")
rownames(sf) <- paste("S",1:6,sep="")
sf  #example from Shrout and Fleiss (1979)
ICC(sf,lmer=FALSE)  #just use the aov procedure

#data(sai)
if(require(psychTools)) {sai <- psychTools::sai
sai.xray <- subset(sai,(sai$study=="XRAY") & (sai$time==1))
xray.icc <- ICC(sai.xray[-c(1:3)],lmer=TRUE,check.keys=TRUE)
xray.icc
xray.icc$lme  #show the variance components as well
}
```

`iclust` *iclust: Item Cluster Analysis – Hierarchical cluster analysis using psychometric principles*

Description

A common data reduction technique is to cluster cases (subjects). Less common, but particularly useful in psychological research, is to cluster items (variables). This may be thought of as an alternative to factor analysis, based upon a much simpler and more intuitive model. The cluster model is that the correlations between variables reflect that each item loads on at most one cluster, and that items that load on those clusters correlate as a function of their respective loadings on that cluster and items that define different clusters correlate as a function of their respective cluster loadings and the intercluster correlations. Essentially, the cluster model is a Very Simple Structure factor model of complexity one (see VSS).

iclust function applies the iclust algorithm (Revelle, 1979) to hierarchically cluster items to form composite scales. Clusters are combined if coefficients alpha and beta will increase in the new cluster.

α, the mean split half correlation, and β, an estimate of the general factor saturation based upon the correlation between the most distinct partions of the correlation matrix, are estimates of the reliability and general factor saturation of the test. (See also the omega function to estimate McDonald's coeffients ω_h and ω_t). Other reliability estimates are found in the reliability and splitHalf.

Usage

```
iclust(r.mat, nclusters=0, alpha=3, beta=1,
 beta.size=4, alpha.size=3,
 correct=TRUE,correct.cluster=TRUE,
 reverse=TRUE, beta.min=.5, output=1,
 digits=2,labels= NULL,cut=0,
 n.iterations =0,
 title="ICLUST",
 cor="cor", use="pairwise",
 plot=TRUE,
 weighted=TRUE,
 cor.gen=TRUE,SMC=TRUE,purify=TRUE,diagonal=FALSE,
 n.obs = NA, reliability=FALSE)

ICLUST(r.mat, nclusters=0, alpha=3, beta=1,
 beta.size=4, alpha.size=3,
 correct=TRUE,correct.cluster=TRUE,
 reverse=TRUE, beta.min=.5, output=1,
 digits=2,labels=NULL,cut=0,
 n.iterations = 0,
 title="ICLUST",
 cor="cor", use="pairwise", plot=TRUE,
 weighted=TRUE,
 cor.gen=TRUE,SMC=TRUE,purify=TRUE,diagonal=FALSE,
 n.obs=NA,reliability=FALSE)


#iclust(r.mat)    #use all defaults
#iclust(r.mat,nclusters =3)   #use all defaults and if possible stop at 3 clusters
#ICLUST(r.mat, output =3)     #long output shows clustering history
#ICLUST(r.mat, n.iterations =3)  #clean up solution by item reassignment
```

Arguments

r.mat	A correlation matrix or data matrix/data.frame. (If r.mat is not square i.e, is not a correlation matrix, the data are correlated using the options specified in cor. The default is to use pairwise deletion for Pearson correlations. Alternatives include tetrachoric or polychoric correlations.
nclusters	Extract clusters until nclusters remain (default will extract until the other criteria

are met or 1 cluster, whichever happens first). See the discussion below for alternative techniques for specifying the number of clusters.

`alpha` Apply the increase in alpha criterion (0) never or for (1) the smaller, 2) the average, or 3) the greater of the separate alphas. (default = 3).

`beta` Apply the increase in beta criterion (0) never or for (1) the smaller, 2) the average, or 3) the greater of the separate betas. (default =1). By setting a larger nvalue, the application of the beta criterion becomes more stringent. See the examples.

`beta.size` Apply the beta criterion after clusters are of beta.size (default = 4). Setting this to a smaller values is more strigent.

`alpha.size` Apply the alpha criterion after clusters are of size alpha.size (default =3).

`correct` Correct correlations for reliability (default = TRUE).

`correct.cluster` Correct cluster -sub cluster correlations for reliability of the sub cluster , default is TRUE))

`reverse` Reverse negative keyed items (default = TRUE). This is the principal that a reversed score item makes sense. Probably not appropriate if clustering people.

`beta.min` Stop clustering if the beta is not greater than beta.min (default = .5)

`output` 1) short, 2) medium, 3) long, 4) very long output. (default =1). To see the cluster statistics at each level, use 3. To see the within cluster correlations at each level, use 4. (This produces a great deal of output, but is useful to understand the algorithm.)

`labels` Vector of item content or labels. If NULL, then the colnames are used. If FALSE, then labels are V1 .. Vn. See the examples for how to use a dictionary.

`cut` Sort cluster loadings > absolute(cut) (default = 0)

`n.iterations` Iterate the solution n.iterations times to "purify" the clusters (default = 0)

`digits` Precision of digits of output (default = 2)

`title` Title for this run.

`cor` What kind of correlation should be applied to raw data, defaults to "pearson" with pairwise complete, options include "tet", "poly", "mixed","spearman".

`use` See options for use in the cor function. Defaults to "pairwise".

`plot` Should ICLUST.diagram be called automatically for plotting (does not require Rgraphviz default=TRUE)

`weighted` Weight the intercluster correlation by the size of the two clusters (TRUE) or do not weight them (FALSE). A third option ("min"), added 6/11/24 is to estimate the general saturation by the minimum of the within and between interitem cluster correlations.

`cor.gen` When correlating clusters with subclusters, base the correlations on the general factor (default) or general + group (cor.gen=FALSE)

`SMC` When estimating cluster-item correlations, use the smcs as the estimate of an item communality (SMC=TRUE) or use the maximum correlation (SMC=FALSE).

`purify`	Should clusters be defined as the original groupings (purify = FALSE) or by the items with the highest loadings on those original clusters? (purify = TRUE)
`diagonal`	Should the diagonal be included in the fit statistics. The default is not to include it. Prior to 1.2.8, the diagonal was included.
`n.obs`	Number of observations (if using a correlation matrix). Specify if using a correlation matrix in order to some fit statistics that depend upon sample size (e.g., chi square, RMSEA, etc.)
`reliability`	Report various alternative estimates of reliability and return the reliability object. This will lead to problems if some clusters are just items. For well formed clusters, set reliability =TRUE or just use the reliability function on the keys object.

Details

Extensive documentation and justification of the algorithm is available in the original MBR 1979 https://personality-project.org/revelle/publications/iclust.pdf paper. Further discussion of the algorithm and sample output is available on the personality-project.org web page: https://personality-project.org/r/r.ICLUST.html

A common problem in the social sciences is to construct scales or composites of items to measure constructs of theoretical interest and practical importance. This process frequently involves administering a battery of items from which those that meet certain criteria are selected. These criteria might be rational, empirical,or factorial. A similar problem is to analyze the adequacy of scales that already have been formed and to decide whether the putative constructs are measured properly. Both of these problems have been discussed in numerous texts, as well as in myriad articles. Proponents of various methods have argued for the importance of face validity, discriminant validity, construct validity, factorial homogeneity, and theoretical importance.

Revelle (1979) proposed that hierachical cluster analysis could be used to estimate a new coefficient (beta) that was an estimate of the general factor saturation of a test. More recently, Zinbarg, Revelle, Yovel and Li (2005) compared McDonald's Omega to Cronbach's alpha and Revelle's beta. They conclude that ω_h hierarchical is the best estimate. An algorithm for estimating `omega` is available as part of the psych package.

Revelle and Zinbarg (2009) discuss alpha, beta, and omega, as well as other estimates of reliability.

Reise and Haviland (2024) give a nice discussion of alpha and beta for applied scale construction.

The original ICLUST program was written in FORTRAN in the early 1970s to run on CDC and IBM mainframes and was then modified to run in PC-DOS. The R version of iclust is a completely new version written for the psych package.

A requested feature (not yet available) is to specify certain items as forming a cluster. That is, to do confirmatory cluster analysis.

The program currently has three primary functions: cluster, loadings, and graphics. β. Conceptually β is similar to the worst split half reliability (see e.g. `splitHalf`) but it is better conceived of as an estimate of the general factor saturation of a test.

Consider the matrix M composed of four submatrices of size n x n and m x m

$$M = \begin{matrix} Rx & Rxy \\ Rxy & Ry \end{matrix}$$

with average within and between item correlations of

$$\text{av.r} = \begin{matrix} \text{rxx} & \text{rxy} \\ \text{rxy} & \text{ryy} \end{matrix}$$

Then $Rx = \Sigma rxx$ with n^2 elements and $Rxy = \Sigma rxy$ with n * m elements. The total variance of the M (Vm) matrix is just the sum of the four submatrices. If M has been partitioned such that rxy is minimized (which is the goal of clustering), then the correlations in Rxy reflect just the general factor in this test. Thus

$\beta = \frac{n*m*rxy}{Vm}$.

That is, the general variance of the test is assumed to be what the two most unrelated parts share.

An important element of `iclust` is the ability to estimate coefficient β. In June, 2009, the option of weighted versus unweighted beta was introduced. Unweighted beta calculates beta based upon the correlation between two clusters, corrected for test length using the Spearman-Brown prophecy formala, while weighted beta finds the average interitem correlation between the items within two clusters and then finds beta from this. That is, for two clusters X and Y of size N and M with between average correlation rxy, weighted beta is (N+M)^2 rxy/(Vx +y + 2Cxy). Raw (unweighted) beta is 2Rxy/(1+Rxy) where Rxy = Cxy/sqrt(VxVy). Weighted beta seems a more appropriate estimate and is now the default. Unweighted beta is still available for consistency with prior versions.

Added in June, 2024 is yet a third option , the minimum of rxx, ryy, and rxy where rxy is the average between cluster correlation. This is logically the best option, but is still being tested. weighted=TRUE is still the default.

Also modified in June, 2009 was the way of correcting for item overlap when calculating the cluster-subcluster correlations for the graphic output. This does not affect the final cluster solution, but does produce slightly different path values. In addition, there are two ways to solve for the cluster - subcluster correlation.

Given the covariance between two clusters, Cab with average rab = Cab/(N*M), and cluster variances Va and Vb with Va = N + N*(N-1)*ra then the correlation of cluster A with the combined cluster AB is either

a) (((N+M)^2)rab + Cab)/sqrt(Va*Va) (option cor.gen=TRUE) or
b) (Va - N + Nra + Cab)/sqrt(Vab*Va) (option cor.gen=FALSE)

The default is to use cor.gen=TRUE.

Although iclust will give what it thinks is the best solution in terms of the number of clusters to extract, the user will sometimes disagree. To get more clusters than the default solution, just set the nclusters parameter to the number desired. However, to get fewer than meet the alpha and beta criteria, it is sometimes necessary to set alpha=0 and beta=0 and then set the nclusters to the desired number.

Hierarchical clustering is not guaranteed to give the "best" solution. Thus, following the original clustering, items are correlated with all clusters and reassigned based upon their highest correlation. This is shown in item by cluster Structure matrix. The first column (O) is the original cluster definition, the second column (P) is the purified cluster solution. This is seen in the example clustering

of bfi data set. The figure shows the original clustering and matches the O column of the Cluster Structure.

Thus, there are two sets of scoring keys reported: the purified keys, keys, and the original, unpurified keys, keys.org. The reliability object reported is based upon the purified keys. The alphas reported in the iclust.diagram are based upon the original keys. To find the other statistics reported in the reliability object, run the reliability function on the keys.org object. (see the bfi cluster example below.)

Clustering 24 tests of mental ability

A sample output using the 24 variable problem by Harman can be represented both graphically and in terms of the cluster order. The default is to produce graphics using the diagram functions. An alternative is to use the Rgraphviz package (from BioConductor). Because this package is sometimes hard to install, there is an alternative option (ICLUST.graph to write dot language instructions for subsequent processing. This will create a graphic instructions suitable for any viewing program that uses the dot language. ICLUST.rgraph produces the dot code for Graphviz. Somewhat lower resolution graphs with fewer options are available in the ICLUST.diagram function which does not require Rgraphviz. Dot code can be viewed directly in Graphviz or can be tweaked using commercial software packages (e.g., OmniGraffle)

Note that for the Harman 24 variable problem, with the default parameters, the data form one large cluster. (This is consistent with the Very Simple Structure (VSS) output as well, which shows a clear one factor solution for complexity 1 data.)

An alternative solution is to ask for a somewhat more stringent set of criteria and require an increase in the size of beta for all clusters greater than 3 variables. This produces a 4 cluster solution.

It is also possible to use the original parameter settings, but ask for a 4 cluster solution.

At least for the Harman 24 mental ability measures, it is interesting to compare the cluster pattern matrix with the oblique rotation solution from a factor analysis. The factor congruence of a four factor oblique pattern solution with the four cluster solution is > .99 for three of the four clusters and > .97 for the fourth cluster. The cluster pattern matrix is returned as an invisible object in the output.

In September, 2012, the fit statistics (pattern fit and cluster fit) were slightly modified to (by default) not consider the diagonal (diagonal=FALSE). Until then, the diagonal was included in the cluster fit statistics. The pattern fit is analogous to factor analysis and is based upon the model = P x Structure where Structure is Pattern * Phi. Then R* = R - model and fit is the ratio of sum(r*^2)/sum(r^2) for the off diagonal elements.

The results are best visualized using ICLUST.graph, the results of which can be saved as a dot file for the Graphviz program. https://www.graphviz.org/. The iclust.diagram is called automatically to produce cluster diagrams. The resulting diagram is not quite as pretty as what can be achieved in dot code but is quite adequate if you don't want to use an external graphics program. With the installation of Rgraphviz, ICLUST can also provide cluster graphs.

As of May, 2024, it is much easier to take the cluster results and if using raw data to find cluster scores. ICLUST now returns two keys lists which can be passed to various scoring functions. See the discussion above about raw keys and purified keys.

As of June, 2024, following a discussion with Steven Reise, I have added a call to the reliability function which is passed the keys.list for each cluster. This allows for alternative estimates of reliability for the clusters.

Value

`title` Name of this analysis

`results` A list containing the step by step cluster history, including which pair was grouped, what were the alpha and betas of the two groups and of the combined group.

Note that the alpha values are "standardized alphas" based upon the correlation matrix, rather than the raw alphas that will come from `scoreItems`

The print.psych and summary.psych functions will print out just the must important results.

`corrected` The raw and corrected for alpha reliability cluster intercorrelations.

`clusters` a matrix of -1, 0, and 1 values to define cluster membership.

`purified` A list of the cluster definitions and cluster loadings of the purified solution. These are sorted by importance (the eigenvalues of the clusters). The cluster membership from the original (O) and purified (P) clusters are indicated along with the cluster structure matrix. These item loadings are the same as those found by the `scoreItems` function and are found by correcting the item-cluster correlation for item overlap by summing the item-cluster covariances with all except that item and then adding in the smc for that item. These resulting correlations are then corrected for scale reliability.

To show just the most salient items, use the cutoff option in `print.psych`

`cluster.fit`, `structure.fit`, `pattern.fit`

There are a number of ways to evaluate how well any factor or cluster matrix reproduces the original matrix. Cluster fit considers how well the clusters fit if only correlations with clusters are considered. Structure fit evaluates R = CC' while pattern fit evaluate R = C inverse (phi) C' where C is the cluster loading matrix, and phi is the intercluster correlation matrix.

`pattern` The pattern matrix loadings. Pattern is just C inverse (Phi). The pattern matrix is conceptually equivalent to that of a factor analysis, in that the pattern coefficients are b weights of the cluster to the variables, while the normal cluster loadings are correlations of the items with the cluster. The four cluster and four factor pattern matrices for the Harman problem are very similar.

`order` This is a vector of the variable names in the order of the cluster diagram. This is useful as a tool for sorting correlations matrices. See the last example.

`keys` A list of keys for scoring. These are the keys representing the "purified" clusters

`keys.org` A list of scoring keys for the orginal (unpurified) solution.

`reliability` The output of the `reliability` function applied to each cluster using the keys list. This is a useful to examine splithalf reliablities – which may or may not be the same as beta.

`stats` A list of various fit statistics. Similar to those from `fa`.

Note

iclust draws graphical displays with or without using Rgraphviz. Because of difficulties installing Rgraphviz on many systems, the default it not even try using it. With the introduction of the `diagram`

functions, iclust now draws using diagram which is not as pretty as using Rgraphviz, but more stable. However, Rgraphviz can be used by using ICLUST.rgraph to produces slightly better graphics. It is also possible to export dot code in the dot language for further massaging of the graphic. This may be done using ICLUST.graph. This last option is probably preferred for nice graphics which can be massaged in any dot code program (e.g., graphviz (https://graphviz.org) or a commercial program such as OmniGraffle.

To view the cluster structure more closely, it is possible to save the graphic output as a pdf and then magnify this using a pdf viewer. This is useful when clustering a large number of variables.

In order to sort the clusters by cluster loadings, use iclust.sort.

By default, the correlations used for the similarity matrix are Pearson correlations. It is of course possible to use tetrachoric or polychoric to form the correlation matrix for later analysis. This option is now included directly in the function and can be specified with the cor option(e.g., cor="tet").

iclust can also be used to organize complex correlation matrices. Thus, by clustering the items in a correlation matrix, sorting that matrix by the cluster loadings using mat.sort, and then plotting with corPlot. See the penultimate example. An alterative way of ordering the variables is to use the order object which is just a vector of the variable names sorted by the way they appear in the tree diagram. (See the final example.)

Author(s)

William Revelle

References

Reise, S.P. and Haviland, M. G. (2024) Understanding Alpha and Beta and Sources of Common Variance: Theoretical Underpinnings and a Practical Example. Journal of Personality Assessment <doi.org:10.1080/00223891.2024.2420175>

Revelle, W. Hierarchical Cluster Analysis and the Internal Structure of Tests. Multivariate Behavioral Research, 1979, 14, 57-74.

Revelle, W. and Zinbarg, R. E. (2009) Coefficients alpha, beta, omega and the glb: comments on Sijtsma. Psychometrika, 2009.

https://personality-project.org/revelle/publications/iclust.pdf
See also more extensive documentation at https://personality-project.org/r/r.ICLUST.html and
Revelle, W. (in prep) An introduction to psychometric theory with applications in R. To be published by Springer. (working draft available at https://personality-project.org/r/book/

See Also

iclust.sort, ICLUST.diagram, ICLUST.graph, ICLUST.cluster, cluster.fit , VSS, omega

Examples

```
test.data <- Harman74.cor$cov
ic.out <- iclust(test.data,title="ICLUST of the Harman data")
summary(ic.out)
```

```
#use all defaults and stop at 4 clusters
ic.out4 <- iclust(test.data,nclusters =4,title="Force 4 clusters", reliability=FALSE)
summary(ic.out4)
ic.out1 <- iclust(test.data,beta=3,beta.size=3)  #use more stringent criteria
ic.out  #more complete output
plot(ic.out4)    #this shows the spatial representation
#use a dot graphics viewer on the out.file
#dot.graph <- ICLUST.graph(ic.out,out.file="test.ICLUST.graph.dot")
#show the equivalent of a factor solution
fa.diagram(ic.out4$pattern,Phi=ic.out4$Phi,main="Pattern taken from iclust")


#Try running the above example with the Harman74.cor data set from datasets.


#demonstrating the use of labels using the bfi data set

ic<- iclust(bfi[,1:25], labels=bfi.dictionary[1:25,2] ,
    title="ICLUST of bfi data set")
ic  #show the output
cluster.scores <- scoreItems(ic$keys, bfi[,1:25])  #find cluster scores
original.scores <- scoreItems(ic$keys.org,bfi[,1:25]) #find cluster scores

#The effect of a more stringent beta criterion
ic.beta3 <- iclust(bfi[1:25], beta=3, title="ICLUST with more stringent beta")
ic.beta3

#beta is not the worst split half, but rather a general factor estimate
#consider 9 items representing 3 factors
F <- matrix(c(rep(.6,3),rep(0,9),rep(.6,3),rep(0,9),rep(.6,3)), ncol=3)
R <- F %*% t(F)
diag(R) <- 1
ic <- iclust(R)
ic$beta     #0 because there is nothing in common between these three clusters
#but
splitHalf(R) # split half = .19  because the split is formed from V1.. V4 and V5 .. V9.

#organize a correlation matrix based upon cluster solution.
R <- cor(bfi, use="pairwise")
ic <- iclust(R,plot=FALSE, reliability = FALSE) #suppress the plot
R.s <- mat.sort(R, ic)
corPlot(R.s, main ="bfi sorted by iclust loadings")
#compare with
corPlot(R[ic$order,ic$order] ,main="bfi sorted by iclust order")
```

`ICLUST.cluster` *Function to form hierarchical cluster analysis of items*

Description

The guts of the ICLUST algorithm. Called by ICLUST See ICLUST for description.

Usage

```
ICLUST.cluster(r.mat, ICLUST.options,smc.items)
```

Arguments

r.mat	A correlation matrix
ICLUST.options	A list of options (see ICLUST)
smc.items	passed from the main program to speed up processing

Details

See ICLUST

Value

A list of cluster statistics, described more fully in ICLUST

comp1	Description of 'comp1'
comp2	Description of 'comp2'

...

Note

Although the main code for ICLUST is here in ICLUST.cluster, the more extensive documentation is for ICLUST.

Author(s)

William Revelle

References

Revelle, W. 1979, Hierarchical Cluster Analysis and the Internal Structure of Tests. Multivariate Behavioral Research, 14, 57-74. https://personality-project.org/revelle/publications/iclust.pdf
See also more extensive documentation at https://personality-project.org/r/r.ICLUST.html

See Also

ICLUST.graph,ICLUST, cluster.fit , VSS, omega

iclust.diagram *Draw an ICLUST hierarchical cluster structure diagram*

Description

Given a cluster structure determined by ICLUST, create a graphic structural diagram using graphic functions in the psych package To create dot code to describe the ICLUST output with more precision, use ICLUST.graph. If Rgraphviz has been successfully installed, the alternative is to use ICLUST.rgraph.

Usage

```
iclust.diagram(ic, labels = NULL, short = FALSE, digits = 2, cex = NULL,
    min.size = NULL,
     e.size =1,
     colors=c("black","blue"),
     main = "ICLUST diagram",
     cluster.names=NULL,marg=c(.5,.5,1.5,.5),plot=TRUE, bottomup=TRUE,
                both=TRUE,pos=NULL,...)
```

Arguments

ic	Output from ICLUST
labels	labels for variables (if not specified as rownames in the ICLUST output. See examples.
short	if short=TRUE, variable names are replaced with Vn
digits	Round the path coefficients to digits accuracy
cex	The standard graphic control parameter for font size modifications. This can be used to make the labels bigger or smaller than the default values.
min.size	Don't provide statistics for clusters less than min.size
e.size	size of the ellipses with the cluster statistics.
colors	postive and negative
main	The main graphic title
cluster.names	Normally, clusters are named sequentially C1 ... Cn. If cluster.names are specified, then these values will be used instead.
marg	Sets the margins to be narrower than the default values. Resets them upon return
plot	If plot is TRUE, then draw the diagram, if FALSE, then just return the veriable order from the plot
bottomup	Which way to draw the arrows. TRUE means from the items to the clusters. See note.
both	if TRUE report alpha and beta. if "alpha", just report the alpha, or if "beta" just report beta.
pos	Not used
...	To allow parameters to be passed to lower level graphics

Details

iclust.diagram provides most of the power of ICLUST.rgraph without the difficulties involved in installing Rgraphviz. It is called automatically from ICLUST.

Following a request by Michael Kubovy, cluster.names may be specified to replace the normal C1 ... Cn names.

If access to a dot language graphics program is available, it is probably better to use the iclust.graph function to get dot output for offline editing.

Until 3/11/23 arrows went from clusters to items. The default value for bottomup has been changed to draw from items to clusters. To draw the old way, set bottomup=TRUE.

Value

Graphical output summarizing the hierarchical cluster structure. The graph is drawn using the diagram functions (e.g., dia.curve, dia.arrow, dia.rect, dia.ellipse) created as a work around to Rgraphviz.

Also returned (invisibly) is a vector of variable names ordered by their location in the tree diagram. The plot option suppresses the plot for speed.

Note

Suggestions for improving the graphic output are welcome. Thus, the request from Steven Reise to just display alpha and not beta was added in June, 2024.

Author(s)

William Revelle

References

Revelle, W. Hierarchical Cluster Analysis and the Internal Structure of Tests. Multivariate Behavioral Research, 1979, 14, 57-74.

See Also

ICLUST

Examples

```
v9 <- sim.hierarchical()
v9c <- ICLUST(v9)
test.data <- Harman74.cor$cov
ic.out <- ICLUST(test.data)
#now show how to relabel clusters
ic.bfi <- iclust(bfi[1:25],beta=3) #find the clusters
cluster.names <- rownames(ic.bfi$results) #get the old names
#change the names to the desired ones
cluster.names[c(16,19,18,15,20)] <- c("Neuroticism","Extra-Open","Agreeableness",
      "Conscientiousness","Open")
#now show the new names
```

```
iclust.diagram(ic.bfi,cluster.names=cluster.names,min.size=4,e.size=1.75)
#just use the diagram function (which calls iclust.diagram).
#Add item content using the labels parameter
diagram(ic.bfi,labels=bfi.dictionary[,2],cluster.names=cluster.names,
       min.size=4,e.size=1.5, main="ICLUST of the BFI")
```

ICLUST.graph *create control code for ICLUST graphical output*

Description

Given a cluster structure determined by ICLUST, create dot code to describe the ICLUST output. To use the dot code, use either https://www.graphviz.org/ Graphviz or a commercial viewer (e.g., OmniGraffle). This function parallels ICLUST.rgraph which uses Rgraphviz.

Usage

```
ICLUST.graph(ic.results, out.file,min.size=1, short = FALSE,labels=NULL,
size = c(8, 6), node.font = c("Helvetica", 14), edge.font = c("Helvetica", 12),
 rank.direction=c("RL","TB","LR","BT"), digits = 2, title = "ICLUST", ...)
```

Arguments

ic.results	output list from ICLUST
out.file	name of output file (defaults to console)
min.size	draw a smaller node (without all the information) for clusters < min.size – useful for large problems
short	if short==TRUE, don't use variable names
labels	vector of text labels (contents) for the variables
size	size of output
node.font	Font to use for nodes in the graph
edge.font	Font to use for the labels of the arrows (edges)
rank.direction	LR or RL
digits	number of digits to show
title	any title
...	other options to pass

Details

Will create (or overwrite) an output file and print out the dot code to show a cluster structure. This dot file may be imported directly into a dot viewer (e.g., https://www.graphviz.org/). The "dot" language is a powerful graphic description language that is particulary appropriate for viewing cluster output. Commercial graphics programs (e.g., OmniGraffle) can also read (and clean up) dot files.

ICLUST.graph takes the output from ICLUST results and processes it to provide a pretty picture of the results. Original variables shown as rectangles and ordered on the left hand side (if rank direction is RL) of the graph. Clusters are drawn as ellipses and include the alpha, beta, and size of the cluster. Edges show the cluster intercorrelations.

It is possible to trim the output to not show all cluster information. Clusters < min.size are shown as small ovals without alpha, beta, and size information.

Although it would be nice to process the dot code directly in R, the Rgraphviz package is difficult to use on all platforms and thus the dot code is written directly.

Value

Output is a set of dot commands written either to console or to the output file. These commands may then be used as input to any "dot" viewer, e.g., Graphviz.

Author(s)

<revelle@northwestern.edu >
https://personality-project.org/revelle.html

References

ICLUST: https://personality-project.org/r/r.ICLUST.html

See Also

VSS.plot, ICLUST

Examples

```
## Not run:
test.data <- Harman74.cor$cov
ic.out <- ICLUST(test.data)
#out.file <- file.choose(new=TRUE)   #create a new file to write the plot commands to
#ICLUST.graph(ic.out,out.file)
#now go to graphviz (outside of R) and open the out.file you created
print(ic.out,digits=2)

## End(Not run)


#test.data <- Harman74.cor$cov
#my.iclust <- ICLUST(test.data)
#ICLUST.graph(my.iclust)
#
#
#digraph ICLUST {
#  rankdir=RL;
#  size="8,8";
#  node [fontname="Helvetica" fontsize=14 shape=box, width=2];
#  edge [fontname="Helvetica" fontsize=12];
# label = "ICLUST";
```

```
# fontsize=20;
#V1  [label = VisualPerception];
#V2  [label = Cubes];
#V3  [label = PaperFormBoard];
#V4  [label = Flags];
#V5  [label = GeneralInformation];
#V6  [label = PargraphComprehension];
#V7  [label = SentenceCompletion];
#V8  [label = WordClassification];
#V9  [label = WordMeaning];
#V10  [label = Addition];
#V11  [label = Code];
#V12  [label = CountingDots];
#V13  [label = StraightCurvedCapitals];
#V14  [label = WordRecognition];
#V15  [label = NumberRecognition];
#V16  [label = FigureRecognition];
#V17  [label = ObjectNumber];
#V18  [label = NumberFigure];
#V19  [label = FigureWord];
#V20  [label = Deduction];
#V21  [label = NumericalPuzzles];
#V22  [label = ProblemReasoning];
#V23  [label = SeriesCompletion];
#V24  [label = ArithmeticProblems];
#node [shape=ellipse, width ="1"];
#C1-> V9 [ label = 0.78 ];
#C1-> V5 [ label = 0.78 ];
#C2-> V12 [ label = 0.66 ];
#C2-> V10 [ label = 0.66 ];
#C3-> V18 [ label = 0.53 ];
#C3-> V17 [ label = 0.53 ];
#C4-> V23 [ label = 0.59 ];
#C4-> V20 [ label = 0.59 ];
#C5-> V13 [ label = 0.61 ];
#C5-> V11 [ label = 0.61 ];
#C6-> V7 [ label = 0.78 ];
#C6-> V6 [ label = 0.78 ];
#C7-> V4 [ label = 0.55 ];
#C7-> V1 [ label = 0.55 ];
#C8-> V16 [ label = 0.5 ];
#C8-> V14 [ label = 0.49 ];
#C9-> C1 [ label = 0.86 ];
#C9-> C6 [ label = 0.86 ];
#C10-> C4 [ label = 0.71 ];
#C10-> V22 [ label = 0.62 ];
#C11-> V21 [ label = 0.56 ];
#C11-> V24 [ label = 0.58 ];
#C12-> C10 [ label = 0.76 ];
#C12-> C11 [ label = 0.67 ];
#C13-> C8 [ label = 0.61 ];
#C13-> V15 [ label = 0.49 ];
#C14-> C2 [ label = 0.74 ];
```

```
#C14-> C5 [ label = 0.72 ];
#C15-> V3 [ label = 0.48 ];
#C15-> C7 [ label = 0.65 ];
#C16-> V19 [ label = 0.48 ];
#C16-> C3 [ label = 0.64 ];
#C17-> V8 [ label = 0.62 ];
#C17-> C12 [ label = 0.8 ];
#C18-> C17 [ label = 0.82 ];
#C18-> C15 [ label = 0.68 ];
#C19-> C16 [ label = 0.66 ];
#C19-> C13 [ label = 0.65 ];
#C20-> C19 [ label = 0.72 ];
#C20-> C18 [ label = 0.83 ];
#C21-> C20 [ label = 0.87 ];
#C21-> C9 [ label = 0.76 ];
#C22-> 0 [ label = 0 ];
#C22-> 0 [ label = 0 ];
#C23-> 0 [ label = 0 ];
#C23-> 0 [ label = 0 ];
#C1  [label =   "C1\n  alpha= 0.84\n beta=  0.84\nN= 2"] ;
#C2  [label =   "C2\n  alpha= 0.74\n beta=  0.74\nN= 2"] ;
#C3  [label =   "C3\n  alpha= 0.62\n beta=  0.62\nN= 2"] ;
#C4  [label =   "C4\n  alpha= 0.67\n beta=  0.67\nN= 2"] ;
#C5  [label =   "C5\n  alpha= 0.7\n beta=  0.7\nN= 2"] ;
#C6  [label =   "C6\n  alpha= 0.84\n beta=  0.84\nN= 2"] ;
#C7  [label =   "C7\n  alpha= 0.64\n beta=  0.64\nN= 2"] ;
#C8  [label =   "C8\n  alpha= 0.58\n beta=  0.58\nN= 2"] ;
#C9  [label =   "C9\n  alpha= 0.9\n beta=  0.87\nN= 4"] ;
#C10  [label =   "C10\n  alpha= 0.74\n beta=  0.71\nN= 3"] ;
#C11  [label =   "C11\n  alpha= 0.62\n beta=  0.62\nN= 2"] ;
#C12  [label =   "C12\n  alpha= 0.79\n beta=  0.74\nN= 5"] ;
#C13  [label =   "C13\n  alpha= 0.64\n beta=  0.59\nN= 3"] ;
#C14  [label =   "C14\n  alpha= 0.79\n beta=  0.74\nN= 4"] ;
#C15  [label =   "C15\n  alpha= 0.66\n beta=  0.58\nN= 3"] ;
#C16  [label =   "C16\n  alpha= 0.65\n beta=  0.57\nN= 3"] ;
#C17  [label =   "C17\n  alpha= 0.81\n beta=  0.71\nN= 6"] ;
#C18  [label =   "C18\n  alpha= 0.84\n beta=  0.75\nN= 9"] ;
#C19  [label =   "C19\n  alpha= 0.74\n beta=  0.65\nN= 6"] ;
#C20  [label =   "C20\n  alpha= 0.87\n beta=  0.74\nN= 15"] ;
#C21  [label =   "C21\n  alpha= 0.9\n beta=  0.77\nN= 19"] ;
#C22  [label =   "C22\n  alpha= 0\n beta=  0\nN= 0"] ;
#C23  [label =   "C23\n  alpha= 0\n beta=  0\nN= 0"] ;
#{ rank=same;
#V1;V2;V3;V4;V5;V6;V7;V8;V9;V10;V11;V12;V13;V14;V15;V16;V17;V18;V19;V20;V21;V22;V23;V24;}}
#
#copy the above output to Graphviz and draw it
#see \url{https://personality-project.org/r/r.ICLUST.html} for an example.
```

`ICLUST.rgraph` *Draw an ICLUST graph using the Rgraphviz package*

Description

Given a cluster structure determined by ICLUST, create a rgraphic directly using Rgraphviz. To create dot code to describe the ICLUST output with more precision, use ICLUST.graph. As an option, dot code is also generated and saved in a file. To use the dot code, use either https://www.graphviz.org/ Graphviz or a commercial viewer (e.g., OmniGraffle).

Usage

```
ICLUST.rgraph(ic.results, out.file = NULL, min.size = 1, short = FALSE,
    labels = NULL, size = c(8, 6), node.font = c("Helvetica", 14),
   edge.font = c("Helvetica", 10), rank.direction=c("RL","TB","LR","BT"),
    digits = 2, title = "ICLUST",label.font=2, ...)
```

Arguments

ic.results	output list from ICLUST
out.file	File name to save optional dot code.
min.size	draw a smaller node (without all the information) for clusters < min.size – useful for large problems
short	if short==TRUE, don't use variable names
labels	vector of text labels (contents) for the variables
size	size of output
node.font	Font to use for nodes in the graph
edge.font	Font to use for the labels of the arrows (edges)
rank.direction	LR or TB or RL
digits	number of digits to show
title	any title
label.font	The variable labels can be a different size than the other nodes. This is particularly helpful if the number of variables is large or the labels are long.
...	other options to pass

Details

Will create (or overwrite) an output file and print out the dot code to show a cluster structure. This dot file may be imported directly into a dot viewer (e.g., https://www.graphviz.org/). The "dot" language is a powerful graphic description language that is particulary appropriate for viewing cluster output. Commercial graphics programs (e.g., OmniGraffle) can also read (and clean up) dot files.

ICLUST.rgraph takes the output from ICLUST results and processes it to provide a pretty picture of the results. Original variables shown as rectangles and ordered on the left hand side (if rank direction is RL) of the graph. Clusters are drawn as ellipses and include the alpha, beta, and size of the cluster. Edges show the cluster intercorrelations.

It is possible to trim the output to not show all cluster information. Clusters < min.size are shown as small ovals without alpha, beta, and size information.

Value

Output is a set of dot commands written either to console or to the output file. These commands may then be used as input to any "dot" viewer, e.g., Graphviz.

ICLUST.rgraph is a version of ICLUST.graph that uses Rgraphviz to draw on the screen as well.

Additional output is drawn to main graphics screen.

Note

Requires Rgraphviz

Author(s)

<revelle@northwestern.edu >
https://personality-project.org/revelle.html

References

ICLUST: https://personality-project.org/r/r.ICLUST.html

See Also

VSS.plot, ICLUST

Examples

```
test.data <- Harman74.cor$cov
ic.out <- ICLUST(test.data)   #uses iclust.diagram instead
```

ICLUST.sort *Sort items by absolute size of cluster loadings*

Description

Given a cluster analysis or factor analysis loadings matrix, sort the items by the (absolute) size of each column of loadings. Used as part of ICLUST and SAPA analyses. The columns are rearranged by the

Usage

```
ICLUST.sort(ic.load, cut = 0, labels = NULL,keys=FALSE,clustsort=TRUE)
```

Arguments

`ic.load`	The output from a factor or principal components analysis, or from ICLUST, or a matrix of loadings.
`cut`	Do not include items in clusters with absolute loadings less than cut
`labels`	labels for each item.
`keys`	should cluster keys be returned? Useful if clusters scales are to be scored.
`clustsort`	TRUE will will sort the clusters by their eigenvalues

Details

When interpreting cluster or factor analysis outputs, is is useful to group the items in terms of which items have their biggest loading on each factor/cluster and then to sort the items by size of the absolute factor loading.

A stable cluster solution will be one in which the output of these cluster definitions does not vary when clusters are formed from the clusters so defined.

With the keys=TRUE option, the resulting cluster keys may be used to score the original data or the correlation matrix to form clusters from the factors.

Value

`sorted`	A data.frame of item numbers, item contents, and item x factor loadings.
`cluster`	A matrix of -1, 0, 1s defining each item by the factor/cluster with the row wise largest absolute loading.

...

Note

Although part of the ICLUST set of programs, this is also more useful for factor or principal components analysis. See also `fa.sort` for similar functionality.

Author(s)

William Revelle

References

https://personality-project.org/r/r.ICLUST.html

See Also

`ICLUST.graph`,`ICLUST.cluster`, `cluster.fit` , `VSS`, `factor2cluster`

`interp.median` *Find the interpolated sample median, quartiles, or specific quantiles for a vector, matrix, or data frame*

Description

For data with a limited number of response categories (e.g., attitude items), it is useful treat each response category as range with width, w and linearly interpolate the median, quartiles, or any quantile value within the median response.

Usage

```
interp.median(x, w = 1,na.rm=TRUE)
interp.quantiles(x, q = .5, w = 1,na.rm=TRUE)
interp.quartiles(x,w=1,na.rm=TRUE)
interp.boxplot(x,w=1,na.rm=TRUE)
interp.values(x,w=1,na.rm=TRUE)
interp.qplot.by(y,x,w=1,na.rm=TRUE,xlab="group",ylab="dependent",
               ylim=NULL,arrow.len=.05,typ="b",add=FALSE,...)
```

Arguments

`x`	input vector
`q`	quantile to estimate (0 < q < 1
`w`	category width
`y`	input vector for interp.qplot.by
`na.rm`	should missing values be removed
`xlab`	x label
`ylab`	Y label
`ylim`	limits for the y axis
`arrow.len`	length of arrow in interp.qplot.by
`typ`	plot type in interp.qplot.by
`add`	add the plot or not
`...`	additional parameters to plotting function

Details

If the total number of responses is N, with median, M, and the number of responses at the median value, Nm >1, and Nb= the number of responses less than the median, then with the assumption that the responses are distributed uniformly within the category, the interpolated median is M - .5w + w*(N/2 - Nb)/Nm.

The generalization to 1st, 2nd and 3rd quartiles as well as the general quantiles is straightforward.

A somewhat different generalization allows for graphic presentation of the difference between interpolated and non-interpolated points. This uses the interp.values function.

If the input is a matrix or data frame, quantiles are reported for each variable.

Value

im	interpolated median(quantile)
v	interpolated values for all data points

See Also

median

Examples

```
interp.median(c(1,2,3,3,3))  # compare with median = 3
interp.median(c(1,2,2,5))
interp.quantiles(c(1,2,2,5),.25)
x <- sample(10,100,TRUE)
interp.quartiles(x)
#
x <-  c(1,1,2,2,2,3,3,3,3,4,5,1,1,1,2,2,3,3,3,3,4,5,1,1,1,2,2,3,3,3,3,4,2)
y <-  c(1,2,3,3,3,3,4,4,4,4,4,1,2,3,3,3,3,4,4,4,4,5,1,5,3,3,3,3,4,4,4,4,4)
x <-  x[order(x)]   #sort the data by ascending order to make it clearer
y <- y[order(y)]
xv <- interp.values(x)
yv <- interp.values(y)
barplot(x,space=0,xlab="ordinal position",ylab="value")
lines(1:length(x)-.5,xv)
points(c(length(x)/4,length(x)/2,3*length(x)/4),interp.quartiles(x))
barplot(y,space=0,xlab="ordinal position",ylab="value")
lines(1:length(y)-.5,yv)
points(c(length(y)/4,length(y)/2,3*length(y)/4),interp.quartiles(y))
if(require(psychTools)) {
data(psychTools::galton)
galton <- psychTools::galton
interp.median(galton)
interp.qplot.by(galton$child,galton$parent,ylab="child height"
,xlab="Mid parent height")
}
```

---

irt.1p — *Item Response Theory estimate of theta (ability) using a Rasch (like) model*

Description

Item Response Theory models individual responses to items by estimating individual ability (theta) and item difficulty (diff) parameters. This is an early and crude attempt to capture this modeling procedure. A better procedure is to use irt.fa.

Usage

```
irt.person.rasch(diff, items)
irt.0p(items)
irt.1p(delta,items)
irt.2p(delta,beta,items)
```

Arguments

diff	A vector of item difficulties –probably taken from irt.item.diff.rasch
items	A matrix of 0,1 items nrows = number of subjects, ncols = number of items
delta	delta is the same as diff and is the item difficulty parameter
beta	beta is the item discrimination parameter found in irt.discrim

Details

A very preliminary IRT estimation procedure. Given scores xij for ith individual on jth item Classical Test Theory ignores item difficulty and defines ability as expected score : abilityi = theta(i) = x(i.) A zero parameter model rescales these mean scores from 0 to 1 to a quasi logistic scale ranging from - 4 to 4 This is merely a non-linear transform of the raw data to reflect a logistic mapping.

Basic 1 parameter (Rasch) model considers item difficulties (delta j): p(correct on item j for the ith subject |theta i, deltaj) = 1/(1+exp(deltaj - thetai)) If we have estimates of item difficulty (delta), then we can find theta i by optimization

Two parameter model adds item sensitivity (beta j): p(correct on item j for subject i |thetai, deltaj, betaj) = 1/(1+exp(betaj *(deltaj- theta i))) Estimate delta, beta, and theta to maximize fit of model to data.

The procedure used here is to first find the item difficulties assuming theta = 0 Then find theta given those deltas Then find beta given delta and theta.

This is not an "official" way to do IRT, but is useful for basic item development. See irt.fa and score.irt for far better options.

Value

a data.frame with estimated ability (theta) and quality of fit. (for irt.person.rasch)
a data.frame with the raw means, theta0, and the number of items completed

Note

Not recommended for serious use. This code is under development. Much better functions are in the ltm and eRm packages. Similar analyses can be done using irt.fa and score.irt.

Author(s)

William Revelle

See Also

sim.irt, sim.rasch, logistic, irt.fa, tetrachoric, irt.item.diff.rasch

---

irt.fa *Item Response Analysis by Exploratory Factor Analysis of tetrachoric/polychoric correlations*

Description

Although exploratory factor analysis and Item Response Theory seem to be very different models of binary data, they can provide equivalent parameter estimates of item difficulty and item discrimination. Tetrachoric or polychoric correlations of a data set of dichotomous or polytomous items may be factor analysed using a minimum residual or maximum likelihood factor analysis and the result loadings transformed to item discrimination parameters. The tau parameter from the tetrachoric/polychoric correlations combined with the item factor loading may be used to estimate item difficulties.

Usage

```
irt.fa(x,nfactors=1,correct=TRUE,plot=TRUE,n.obs=NULL,rotate="oblimin",fm="minres",
        sort=FALSE,...)
irt.select(x,y)
fa2irt(f,rho,plot=TRUE,n.obs=NULL)
```

Arguments

x	A data matrix of dichotomous or discrete items, or the result of tetrachoric or polychoric
nfactors	Defaults to 1 factor
correct	If true, then correct the tetrachoric correlations for continuity. (See tetrachoric).
plot	If TRUE, automatically call the plot.irt or plot.poly functions.
y	the subset of variables to pick from the rho and tau output of a previous irt.fa analysis to allow for further analysis.
n.obs	The number of subjects used in the initial analysis if doing a second analysis of a correlation matrix. In particular, if using the fm="minchi" option, this should be the matrix returned by count.pairwise.
rotate	The default rotation is oblimin. See fa for the other options.
fm	The default factor extraction is minres. See fa for the other options.
f	The object returned from fa
rho	The object returned from polychoric or tetrachoric. This will include both a correlation matrix and the item difficulty levels.
sort	Should the factor loadings be sorted before preparing the item information tables. Defaults to FALSE as this is more useful for scoring items. For tabular output it is better to have sort=TRUE.
...	Additional parameters to pass to the factor analysis function

Details

irt.fa combines several functions into one to make the process of item response analysis easier. Correlations are found using either tetrachoric or polychoric. Exploratory factor analyeses with all the normal options are then done using fa. The results are then organized to be reported in terms of IRT parameters (difficulties and discriminations) as well as the more conventional factor analysis output. In addition, because the correlation step is somewhat slow, reanalyses may be done using the correlation matrix found in the first step. In this case, if it is desired to use the fm="minchi" factoring method, the number of observations needs to be specified as the matrix resulting from pairwiseCount.

The tetrachoric correlation matrix of dichotomous items may be factored using a (e.g.) minimum residual factor analysis function fa and the resulting loadings, λ_i are transformed to discriminations by $\alpha = \frac{\lambda_i}{\sqrt{1-\lambda_i^2}}$.

The difficulty parameter, δ is found from the τ parameter of the tetrachoric or polychoric function.

$\delta_i = \frac{\tau_i}{\sqrt{1-\lambda_i^2}}$

Similar analyses may be done with discrete item responses using polychoric correlations and distinct estimates of item difficulty (location) for each item response.

The results may be shown graphically using plot.irt for dichotomous items or plot.poly for polytomous items. These are called by plotting the irt.fa output, see the examples). For plotting there are three options: type = "ICC" will plot the item characteristic response function. type = "IIC" will plot the item information function, and type= "test" will plot the test information function. Invisible output from the plot function will return tables of item information as a function of several levels of the trait, as well as the standard error of measurement and the reliability at each of those levels.

The normal input is just the raw data. If, however, the correlation matrix has already been found using tetrachoric, polychoric, or a previous analysis using irt.fa then that result can be processed directly. Because irt.fa saves the rho and tau matrices from the analysis, subsequent analyses of the same data set are much faster if the input is the object returned on the first run. A similar feature is available in omega.

The output is best seen in terms of graphic displays. Plot the output from irt.fa to see item and test information functions.

The print function will print the item location and discriminations. The additional factor analysis output is available as an object in the output and may be printed directly by specifying the $fa object.

The irt.select function is a helper function to allow for selecting a subset of a prior analysis for further analysis. First run irt.fa, then select a subset of variables to be analyzed in a subsequent irt.fa analysis. Perhaps a better approach is to just plot and find the information for selected items.

The plot function for an irt.fa object will plot ICC (item characteristic curves), IIC (item information curves), or test information curves. In addition, by using the "keys" option, these three kinds of plots can be done for selected items. This is particularly useful when trying to see the information characteristics of short forms of tests based upon the longer form factor analysis.

The plot function will also return (invisibly) the informaton at multiple levels of the trait, the average information (area under the curve) as well as the location of the peak information for each item. These may be then printed or printed in sorted order using the sort option in print.

Value

irt	A list of Item location (difficulty) and discrimination
fa	A list of statistics for the factor analyis
rho	The tetrachoric/polychoric correlation matrix
tau	The tetrachoric/polychoric cut points

Note

irt.fa makes use of the tetrachoric or tetrachoric functions. Both of these will use multiple cores if this is an option. To set these use options("mc.cores"=x) where x is the number of cores to use. (Macs default to 2, PCs seem to default to 1).

In comparing irt.fa to the ltm function in the ltm package or to the analysis reported in Kamata and Bauer (2008) the discrimination parameters are not identical, because the irt.fa reports them in units of the normal curve while ltm and Kamata and Bauer report them in logistic units. In addition, Kamata and Bauer do their factor analysis using a logistic error model. Their results match the irt.fa results (to the 2nd or 3rd decimal) when examining their analyses using a normal model. (With thanks to Akihito Kamata for sharing that analysis.)

irt.fa reports parameters in normal units. To convert them to conventional IRT parameters, multiply by 1.702. In addition, the location parameter is expressed in terms of difficulty (high positive scores imply lower frequency of response.)

The results of irt.fa can be used by score.irt for irt based scoring. First run irt.fa and then score the results using a two parameter model using score.irt.

There is also confusion in the literature of how to interpret the tau parameter. Here I treat it the item threshold for a response, which is to say that tau reflects item difficulty.

Further note that the rho parameter in the fa2irt function is the result from a tetrachoric or polychoric analysis and reflects both the correlations and the item thresholds.

Author(s)

William Revelle

References

Kamata, Akihito and Bauer, Daniel J. (2008) A Note on the Relation Between Factor Analytic and Item Response Theory Models Structural Equation Modeling, 15 (1) 136-153.

McDonald, Roderick P. (1999) Test theory: A unified treatment. L. Erlbaum Associates.

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at https://personality-project.org/r/book/

See Also

fa, sim.irt, tetrachoric, polychoric as well as plot.psych for plotting the IRT item curves. The use of fa.extend is helpful for the case of an incomplete block design to find the factor solution.

See also score.irt for scoring items based upon these parameter estimates. irt.responses will plot the empirical response curves for the alternative response choices for multiple choice items.

Examples

```
## Not run:
set.seed(17)
d9 <- sim.irt(9,1000,-2.5,2.5,mod="normal") #dichotomous items
test <- irt.fa(d9$items)
test
op <- par(mfrow=c(3,1))
plot(test,type="ICC")
plot(test,type="IIC")
plot(test,type="test")
par(op)

#compare this result to first finding the correlations, then factoring them, and
# then applying fa2irt.
R  <- tetrachoric(d9$items,correct=TRUE)
f <- fa(R$rho)
test2 <- fa2irt(f,R)
test2
test2$plot$sumInfo[[1]] - test$plot$sumInfo[[1]]  #identical results

set.seed(17)
items <- sim.congeneric(N=500,short=FALSE,categorical=TRUE) #500 responses to 4 discrete items
d4 <- irt.fa(items$observed)  #item response analysis of congeneric measures
d4    #show just the irt output
d4$fa  #show just the factor analysis output


op <- par(mfrow=c(2,2))
plot(d4,type="ICC")
par(op)


if(require(psychTools)) {
#using the iq data set for an example of real items
#first need to convert the responses to tf
data(iqitems)
iq.keys <- c(4,4,4, 6, 6,3,4,4,  5,2,2,4,  3,2,6,7)

iq.tf <- score.multiple.choice(iq.keys,psychTools::iqitems,score=FALSE)  #just the responses
iq.irt <- irt.fa(iq.tf)
print(iq.irt,short=FALSE) #show the IRT as well as factor analysis output
p.iq <- plot(iq.irt)  #save the invisible summary table
p.iq  #show the summary table of information by ability level
#select a subset of these variables
small.iq.irt <- irt.select(iq.irt,c(1,5,9,10,11,13))
small.irt <- irt.fa(small.iq.irt)
plot(small.irt)
#find the information for three subset of iq items
keys <- make.keys(16,list(all=1:16,some=c(1,5,9,10,11,13),others=c(1:5)))
plot(iq.irt,keys=keys)
}
#compare output to the ltm package or Kamata and Bauer   -- these are in logistic units
```

```
ls <- irt.fa(lsat6)
#library(ltm)
# lsat.ltm <- ltm(lsat6~z1)
#  round(coefficients(lsat.ltm)/1.702,3)  #convert to normal (approximation)
#
#   Dffclt Dscrmn
#Q1 -1.974  0.485
#Q2 -0.805  0.425
#Q3 -0.164  0.523
#Q4 -1.096  0.405
#Q5 -1.835  0.386


#Normal results  ("Standardized and Marginal")(from Akihito Kamata )
#Item       discrim             tau
#  1       0.4169             -1.5520
#  2       0.4333             -0.5999
#  3       0.5373             -0.1512
#  4       0.4044             -0.7723
#  5       0.3587             -1.1966
#compare to ls

  #Normal results  ("Standardized and conditional") (from Akihito Kamata )
#item            discrim   tau
#  1           0.3848    -1.4325
#  2           0.3976    -0.5505
#  3           0.4733    -0.1332
#  4           0.3749    -0.7159
#  5           0.3377    -1.1264
#compare to ls$fa and ls$tau

#Kamata and Bauer (2008) logistic estimates
#1   0.826    2.773
#2   0.723    0.990
#3   0.891    0.249
#4   0.688    1.285
#5   0.657    2.053



## End(Not run)
```

---

irt.item.diff.rasch *Simple function to estimate item difficulties using IRT concepts*

Description

Steps toward a very crude and preliminary IRT program. These two functions estimate item difficulty and discrimination parameters. A better procedure is to use irt.fa or the ltm package.

Usage

```
irt.item.diff.rasch(items)
irt.discrim(item.diff,theta,items)
```

Arguments

items	a matrix of items
item.diff	a vector of item difficulties (found by irt.item.diff)
theta	ability estimate from irt.person.theta

Details

Item Response Theory (aka "The new psychometrics") models individual responses to items with a logistic function and an individual (theta) and item difficulty (diff) parameter.

irt.item.diff.rasch finds item difficulties with the assumption of theta=0 for all subjects and that all items are equally discriminating.

irt.discrim takes those difficulties and theta estimates from irt.person.rasch to find item discrimination (beta) parameters.

A far better package with these features is the ltm package. The IRT functions in the psych-package are for pedagogical rather than production purposes. They are believed to be accurate, but are not guaranteed. They do seem to be slightly more robust to missing data structures associated with SAPA data sets than the ltm package.

The irt.fa function is also an alternative. This will find tetrachoric or polychoric correlations and then convert to IRT parameters using factor analysis (fa).

Value

a vector of item difficulties or item discriminations.

Note

Under development. Not recommended for public consumption. See irt.fa and score.irt for far better options.

Author(s)

William Revelle

See Also

irt.fa, irt.person.rasch

irt.responses *Plot probability of multiple choice responses as a function of a latent trait*

Description

When analyzing ability tests, it is important to consider how the distractor alternatives vary as a function of the latent trait. The simple graphical solution is to plot response endorsement frequencies against the values of the latent trait found from multiple items. A good item is one in which the probability of the distractors decrease and the keyed answer increases as the latent trait increases.

Usage

```
irt.responses(theta,items, breaks = 11,show.missing=FALSE, show.legend=TRUE,
legend.location="topleft", colors=NULL,...)
```

Arguments

theta	The estimated latent trait (found, for example by using score.irt).
items	A matrix or data frame of the multiple choice item responses.
breaks	The number of levels of the theta to use to form the probability estimates. May be increased if there are enough cases.
show.legend	Show the legend
show.missing	For some SAPA data sets, there are a very large number of missing responses. In general, we do not want to show their frequency.
legend.location	Choose among c("bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center","none"). The default is "topleft".
colors	if NULL, then use the default colors, otherwise, specify the color choices. The basic color palette is c("black", "blue", "red", "darkgreen", "gold2", "gray50", "cornflowerblue", "mediumorchid2").
...	Other parameters for plots and points

Details

This function is a convenient way to analyze the quality of item alternatives in a multiple choice ability test. The typical use is to first score the test (using, e.g., score.multiple.choice according to some scoring key and to then find the score.irt based scores. Response frequencies for each alternative are then plotted against total score. An ideal item is one in which just one alternative (the correct one) has a monotonically increasing response probability.

Because of the similar pattern of results for IRT based or simple sum based item scoring, the function can be run on scores calculated either by score.irt or by score.multiple.choice. In the latter case, the number of breaks should not exceed the number of possible score alternatives.

Value

Graphic output

Author(s)

William Revelle

References

Revelle, W. An introduction to psychometric theory with applications in R (in prep) Springer. Draft chapters available at https://personality-project.org/r/book/

See Also

score.multiple.choice, score.irt

Examples

```
if(require(psychTools) ) {
data(iqitems)
iq.keys <- c(4,4,4, 6,6,3,4,4,  5,2,2,4,  3,2,6,7)
scores <- score.multiple.choice(iq.keys,iqitems,score=TRUE,short=FALSE)
#note that for speed we can just do this on simple item counts rather
# than IRT based scores.
op <- par(mfrow=c(2,2))  #set this to see the output for multiple items
irt.responses(scores$scores,psychTools::iqitems[1:4],breaks=11)
op <-  par(op)
}
```

---

kaiser — *Apply the Kaiser normalization when rotating factors*

Description

Kaiser (1958) suggested normalizing factor loadings before rotating them, and then denormalizing them after rotation. The GPArotation package does not (by default) normalize, nor does the fa function. Then, to make it more confusing, varimax in stats does,Varimax in GPArotation does not. kaiser will take the output of a non-normalized solution and report the normalized solution.

Usage

```
kaiser(f, rotate = "oblimin",m=4,pro.m=4)
```

Arguments

f	A factor analysis output from fa or a factor loading matrix.
rotate	Any of the standard rotations avaialable in the GPArotation package.
m	a parameter to pass to Promax
pro.m	A redundant parameter, which is used to replace m in calls to Promax

Details

Best results if called from an unrotated solution. Repeated calls using a rotated solution will produce incorrect estimates of the correlations between the factors.

Value

See the values returned by GPArotation functions

Note

Prepared in response to a question about why `fa` oblimin results are different from SPSS.

Author(s)

William Revelle

References

Kaiser, H. F. (1958) The varimax criterion for analytic rotation in factor analysis. Psychometrika 23, 187-200.

See Also

`fa`, `Promax`

Examples

```
f3 <- fa(Thurstone,3)
f3n <- kaiser(fa(Thurstone,3,rotate="none"))
f3p <- kaiser(fa(Thurstone,3,rotate="none"),rotate="Promax",m=3)
factor.congruence(list(f3,f3n,f3p))
```

KMO — *Find the Kaiser, Meyer, Olkin Measure of Sampling Adequacy*

Description

Henry Kaiser (1970) introduced an Measure of Sampling Adequacy (MSA) of factor analytic data matrices. Kaiser and Rice (1974) then modified it. This is just a function of the squared elements of the 'anti-image' matrix compared to the squares of the original correlations. The overall MSA as well as estimates for each item are found. The index is known as the Kaiser-Meyer-Olkin (KMO) index.

Usage

```
KMO(r)
```

Arguments

r	A correlation matrix or a data matrix (correlations will be found)

Details

Let $S^2 = diag(R^{-1})^{-1}$ and $Q = SR^{-1}S$. Then Q is said to be the anti-image intercorrelation matrix. Let $sumr^2 = \sum R^2$ and $sumq^2 = \sum Q^2$ for all off diagonal elements of R and Q, then $SMA = sumr^2/(sumr^2 + sumq^2)$. Although originally MSA was 1 - sumq^2/sumr^2 (Kaiser, 1970), this was modified in Kaiser and Rice, (1974) to be $SMA = sumr^2/(sumr^2 + sumq^2)$. This is the formula used by Dziuban and Shirkey (1974) and by SPSS.

In his delightfully flamboyant style, Kaiser (1975) suggested that KMO > .9 were marvelous, in the .80s, mertitourious, in the .70s, middling, in the .60s, medicore, in the 50s, miserable, and less than .5, unacceptable.

An alternative measure of whether the matrix is factorable is the Bartlett test `cortest.bartlett` which tests the degree that the matrix deviates from an identity matrix.

Note that except for the reversal of signs, the anti-image correlation matrix is the same as that returned by `partial.r`.

Value

- MSA: The overall Measure of Sampling Adequacy
- MSAi: The measure of sampling adequacy for each item
- Image: The anti-image correlation matrix (Q)

Author(s)

William Revelle

References

H.~F. Kaiser. (1970) A second generation little jiffy. Psychometrika, 35(4):401–415.

H.~F. Kaiser and J.~Rice. (1974) Little jiffy, mark iv. Educational and Psychological Measurement, 34(1):111–117.

H.F. Kaiser. 1974) An index of factor simplicity. Psychometrika, 39 (1) 31-36.

Dziuban, Charles D. and Shirkey, Edwin C. (1974) When is a correlation matrix appropriate for factor analysis? Some decision rules. Psychological Bulletin, 81 (6) 358 - 361.

See Also

See Also as `fa`, `cortest.bartlett`, `Harman.political`, `partial.r`

Examples

```
KMO(Thurstone)
k.m <- KMO(Harman.political)   #compare to the results in Dziuban and Shirkey (1974)
k.m
lowerMat(k.m$Image)
```

```
lowerMat(partial.r(Harman.political))   #identical to image, except for sign
```

---

lmCor *Multiple Regression, Canonical and Set Correlation from matrix or raw input*

Description

A generalization of the lm function to correlation/covariance matrix input. Given a correlation matrix or a matrix or dataframe of raw data, find the multiple regressions and draw a path diagram relating a set of y variables as a function of a set of x variables. A set of covariates (z) can be partialled from the x and y sets. Regression diagrams are automatically included. The model can be specified in conventional formula form, or in terms of x variables and y variables. Multiplicative models (interactions) and quadratic terms may be specified in the formula mode if using raw data. By default, the data will be zero centered before finding the interactions. Will also find Cohen's Set Correlation between a predictor set of variables (x) and a criterion set (y). Also finds the canonical correlations between the x and y sets. Associated functions allow for cross validation of these regression models.

Usage

```
lmCor(y,x,data,z=NULL,n.obs=NULL,use="pairwise",std=TRUE,square=FALSE,
     main="Regression Models",plot=TRUE,show=FALSE,zero=TRUE, alpha = .05,part=FALSE)


lmDiagram(sc,main="Regression model",digits=2,show=FALSE,cex=1,l.cex=1,...)


#does not handle formula input  just does regressions from matrix input
matReg(x,y,C,m=NULL,z=NULL,n.obs=0,means=NULL,std=FALSE,raw=TRUE,part=FALSE)

#useful helper functions
crossValidation(model,data,options=NULL,select=NULL)
crossValidationBoot(y,x,data,n.iter=100)

matPlot(x, type = "b", minlength=6, xlas=0,legend=NULL,
      lab=NULL,pch=16,col=1:6,lty=NULL,...)


# Over the years, the names have changed

 #the prior names for this function which will be deprecated

 lmCor.diagram(sc,main="Regression model",digits=2,show=FALSE,cex=1,l.cex=1,...)

setCor(y,x,data,z=NULL,n.obs=NULL,use="pairwise",std=TRUE,square=FALSE,
```

```
     main="Regression Models",plot=TRUE,show=FALSE,zero=TRUE, alpha = .05,part=FALSE)
#a deprecated  alias to lmCor

set.cor(y,x,data,z=NULL,n.obs=NULL,use="pairwise",std=TRUE,square=FALSE,
 main="Regression Models",plot=TRUE,show=FALSE,zero=TRUE,part=FALSE)   #an old alias

mat.regress(y, x,data, z=NULL,n.obs=NULL,use="pairwise",square=FALSE) #the old form
```

Arguments

y	Three options: 'formula' form (similar to lm) or either the column numbers of the y set (e.g., c(2,4,6) or the column names of the y set (e.g., c("Flags","Addition"). See notes and examples for each.
x	either the column numbers of the x set (e.g., c(1,3,5) or the column names of the x set (e.g. c("Cubes","PaperFormBoard"). x and y may also be set by use of the formula style of lm.
data	A matrix or data.frame of correlations or, if not square, of raw data. Missing values are allowed (see use)
C	A variance/covariance matrix, or a correlation matrix
m	The column name or numbers of the set of mediating variables (see mediate).
z	the column names or numbers of the set of covariates.
n.obs	If specified, then confidence intervals, etc. are calculated, not needed if raw data are given.
n.iter	Number of bootstrap resamples to take in crossValidationBoot
use	Find the correlations using "pairwise" (default) or just use "complete" cases (to match the lm function)
std	Report standardized betas (based upon the correlations) or raw bs (based upon covariances)
part	If z is specified should part (TRUE) or partial (default) correlations be found.
raw	Are data from a correlation matrix or data matrix? Used in matReg
means	A vector of means for the data in matReg if giving matrix input
main	The title for lmCor.diagram
square	if FALSE, then square matrices are treated as correlation matrices not as data matrices. In the rare case that one has as many cases as variables, then set square=TRUE.
sc	The output of lmCor or setCor may be used for drawing diagrams
digits	How many digits should be displayed in the lmCor.diagram?
show	Show the unweighted matrix correlation between the x and y sets?
zero	zero center the x variables before finding the interaction terms.
alpha	p value of the confidence intervals for the beta coefficients

plot	By default, lmCor makes a plot of the results, set to FALSE to suppress the plot
cex	Text size of boxes displaying the variables in the diagram
l.cex	Text size of numbers in arrows, defaults to cex
...	Additional graphical parameters for lmCor.diagram
model	The resulting object from lmCor or bestScales
options	If using a bestScales object, which set of keys to use ("best.key","weights","optimal.key","optimal.weights
select	Not yet implemented option to crossValidation
type	type="b" draws both lines and points
xlas	Orientation of the x labels (3 to make vertical)
minlength	When abbreviating x labels how many characters as a minimum
legend	If not NULL, the draw a legend at the appropriate location
pch	What plot character(s) to use in matPlot (defaults to 18. should be less than 26 -nvar)
col	colors to use in matPlot
lty	line types to use in matPlot
lab	What labels should be supplied to the lines in matPlot. (Defaults to colnames of the variables.)

Details

Although it is more common to calculate multiple regression and canonical correlations from the raw data, it is, of course, possible to do so from a matrix of correlations or covariances. In this case, the input to the function is a square covariance or correlation matrix, as well as the column numbers (or names) of the x (predictor), y (criterion) variables, and if desired z (covariates). The function will find the correlations if given raw data.

Input is either conventional formula mode, or the set of y variables and the set of x variables. Formula mode can be written in the standard formula style of lm (see last example). In this case, pairwise or higher interactions (product terms) may also be specified. By default, when finding product terms, the predictive variables are zero centered (Cohen, Cohen, West and Aiken, 2003), although this option can be turned off (zero=FALSE) to match the results of `lm` or the results discussed in Hayes (2013).

Covariates to be removed are specified by a negative sign in the formula input or by using the z variable. Note that when specifying covariates, the regressions are done as if the regressions were done on the partialled variables. This means that the degrees of freedom and the R2 reflect the regressions of the partialled variables. (See the examples.)

If using covariates, should they be removed from the dependent as well as the independent variables (part = FALSE, the default which means partial correlations or just the independent variables (part=TRUE or part aka semi-partial correlations). The difference between part correlations and partial correlations is whether the variance of the covariate is removed from both the DV and IVs (a partial correlation) or from just the IVs (a part correlation). The slopes of the regressions remain the same, but the amount of variance in the DV (and thus the standard errors) is larger when using part correlations. Using partial correlations for the covariates is equivalent to using the covariate in the regression when interpreting the effects of the other variables.

The output is a set of multiple correlations, one for each dependent variable in the y set, as well as the set of canonical correlations. Since 2.3.12, the canonical loadings for the X and Y variables are returned (Xmat,YMat).

An additional output is the R2 found using Cohen's set correlation (Cohen, 1982). This is a measure of how much variance and the x and y set share.

Cohen (1982) introduced the set correlation, a multivariate generalization of the multiple correlation to measure the overall relationship between two sets of variables. It is an application of canonicial correlation (Hotelling, 1936) and $1 - \prod(1 - \rho_i^2)$ where ρ_i^2 is the squared canonical correlation. Set correlation is the amount of shared variance (R2) between two sets of variables. With the addition of a third, covariate set, set correlation will find multivariate R2, as well as partial and semi partial R2. (The semi and bipartial options are not yet implemented.) Details on set correlation may be found in Cohen (1982), Cohen (1988) and Cohen, Cohen, Aiken and West (2003).

R2 between two sets is just

$$R^2 = 1 - \frac{|R_{yx}|}{|R_y|\,|R_x|} = 1 - \prod(1 - \rho_i^2)$$

where R is the complete correlation matrix of the x and y variables and Rx and Ry are the two sets involved.

Unfortunately, the R2 is sensitive to one of the canonical correlations being very high. An alternative, T2, is the proportion of additive variance and is the average of the squared canonicals. (Cohen et al., 2003), see also Cramer and Nicewander (1979). This average, because it includes some very small canonical correlations, will tend to be too small. Cohen et al. admonition is appropriate: "In the final analysis, however, analysts must be guided by their substantive and methodological conceptions of the problem at hand in their choice of a measure of association." (p613).

Yet another measure of the association between two sets is just the simple, unweighted correlation between the two sets. That is,

$$R_{uw} = \frac{1R_{xy}1'}{(1R_{yy}1')^{.5}(1R_{xx}1')^{.5}}$$

where Rxy is the matrix of correlations between the two sets. This is just the simple (unweighted) sums of the correlations in each matrix. This technique exemplifies the robust beauty of linear models and is particularly appropriate in the case of one dimension in both x and y, and will be a drastic underestimate in the case of items where the betas differ in sign.

When finding the unweighted correlations, as is done in `alpha`, items are flipped so that they all are positively signed.

A typical use in the SAPA project is to form item composites by clustering or factoring (see `fa`,`ICLUST`, `principal`), extract the clusters from these results (`factor2cluster`), and then form the composite correlation matrix using `cluster.cor`. The variables in this reduced matrix may then be used in multiple R procedures using `lmCor`.

Although the overall matrix can have missing correlations, the correlations in the subset of the matrix used for prediction must exist.

If the number of observations is entered, then the conventional confidence intervals, statistical significance, and shrinkage estimates are reported.

If the input is rectangular (not square), correlations or covariances are found from the data. By default, these are pairwise complete correlations. This may be specified in the 'use' option.

The print function reports t and p values for the beta weights, the summary function just reports the beta weights.

The Variance Inflation Factor is reported but should be taken with the normal cautions of interpretation discussed by Guide and Ketokivi. That is to say, VIF > 10 is not a magic cuttoff to define colinearity. It is merely 1/(1-smc(R(x)).

The Guide and Ketokivi article is well worth reading for all who want to use various regression models.

anova.psych allows for comparisons between alternative models.

crossValidation can be used to take the results from lmCor or bestScales and apply the weights to a different data set.

crossValidationBoot will bootstrap resample data and perform a lmCor and then return the crossValidation R values. Although it will handle covariates (z), it will not handle product terms.

matPlot can be used to plot the crossValidation values (just a call to matplot with the xaxis given labels). matPlot has been improved to draw legends and to allow for specifications of the line types.

lmCorLookup will sort the beta weights and report them with item contents if given a dictionary.

matReg is primarily a helper function for mediate but is a general multiple regression function given a covariance matrix and the specified x, y and z variables. Its output includes betas, se, t, p and R2. The call includes m for mediation variables, but these are only used to adjust the degrees of freedom. matReg does not work on data matrices, nor does it take formula input. It is really just a helper function for mediate

Value

beta	the beta weights for each variable in X for each variable in Y
R	The multiple R for each equation (the amount of change a unit in the predictor set leads to in the criterion set).
R2	The multiple R2 (% variance acounted for) for each equation
VIF	The Variance Inflation Factor which is just 1/(1-smc(x))
se	Standard errors of beta weights (if n.obs is specified)
t	t value of beta weights (if n.obs is specified)
Probability	Probability of beta = 0 (if n.obs is specified)
shrunkenR2	Estimated shrunken R2 (if n.obs is specified)
r0	The zero order correlation between the predictors and the criteria
setR2	The multiple R2 of the set correlation between the x and y sets
residual	The residual correlation matrix of Y with x and z removed
ruw	The unit weighted multiple correlation for each dependent variable
Ruw	The unit weighted set correlation
cancor	The canonical correlations between the X and Y set.
Xmat	The path coefficients from the X variables to the canonical variables.
Ymat	The path coefficients from the Y variables to the canonical variables.

Note

As of April 30, 2011, the order of x and y was swapped in the call to be consistent with the general y ~ x syntax of the lm and aov functions. In addition, the primary name of the function was switched to setCor from mat.regress to reflect the estimation of the set correlation. This was changed to lmCor in the March, 2023 release to make the equivalence to lm more obvious.

In October, 2017 I added the ability to specify the input in formula mode and allow for higher level and multiple interactions.

The denominator degrees of freedom for the set correlation does not match that reported by Cohen et al., 2003 in the example on page 621 but does match the formula on page 615, except for the typo in the estimation of F (see Cohen 1982). The difference seems to be that they are adding in a correction factor of df 2 = df2 + df1.

Following a suggestion by Keith Widaman, when zero centering for product terms, the y variables are no longer zero centered. This puts the regression in the units of the DV. (2/22/22).

In December, 2023 I added the Xmat and Ymat coefficients for those who want to examine canonical correlations. This follows the example in Tabachnick and Fidell. Also added `cancorDiagram` to show the canonical path coefficients.

Author(s)

William Revelle

Maintainer: William Revelle <revelle@northwestern.edu>

References

J. Cohen (1982) Set correlation as a general multivariate data-analytic method. Multivariate Behavioral Research, 17(3):301-341.

J. Cohen, P. Cohen, S.G. West, and L.S. Aiken. (2003) Applied multiple regression/correlation analysis for the behavioral sciences. L. Erlbaum Associates, Mahwah, N.J., 3rd ed edition.

H. Hotelling. (1936) Relations between two sets of variates. Biometrika 28(3/4):321-377.

E.Cramer and W. A. Nicewander (1979) Some symmetric, invariant measures of multivariate association. Psychometrika, 44:43-54.

V. Daniel R. Guide Jr. and M. Ketokivim (2015) Notes from the Editors: Redefining some methodological criteria for the journal. Journal of Operations Management. 37. v-viii.

B.G. Tabacnick and L.S. Fidell (2007) Using Multivariate Statistics. Allyn and Bacon.

See Also

`mediate` for an alternative regression model with 'mediation'. `predict` to find predicted values given regression weights. `partial.r` for partial and part correlations. `cluster.cor`, `factor2cluster`,`principal`,`ICLUST`, `link{cancor}` and cca in the yacca package. `GSBE` for further demonstrations of mediation and moderation.

Examples

```
#First compare to lm using data input
summary(lm(rating ~ complaints + privileges, data = attitude))
lmCor(rating ~ complaints + privileges, data = attitude, std=FALSE) #do not standardize
z.attitude <- data.frame(scale(attitude))  #standardize the data before doing lm
summary(lm(rating ~ complaints + privileges, data = z.attitude))  #regressions on z scores
lmCor(rating ~ complaints + privileges, data = attitude)  #by default we standardize and
# the results are the same as the standardized lm


R <- cor(attitude) #find the correlations
#Do the regression on the correlations
#Note that these match the regressions on the standard scores of the data
lmCor(rating ~ complaints + privileges, data =R, n.obs=30)

#now, partial out learning and critical
lmCor(rating ~ complaints + privileges - learning - critical, data =R, n.obs=30)
#compare with the full regression:
lmCor(rating ~ complaints + privileges + learning + critical, data =R, n.obs=30)



#Canonical correlations:

#The first Kelley data set from Hotelling
kelley1 <- structure(c(1, 0.6328, 0.2412, 0.0586, 0.6328, 1, -0.0553, 0.0655,
0.2412, -0.0553, 1, 0.4248, 0.0586, 0.0655, 0.4248, 1), dim = c(4L,
4L), dimnames = list(c("reading.speed", "reading.power", "math.speed",
"math.power"), c("reading.speed", "reading.power", "math.speed",
"math.power")))
lowerMat(kelley1)
mod1 <- lmCor(y = math.speed + math.power ~ reading.speed + reading.power,
    data = kelley1, n.obs=140)
mod1$cancor
#Hotelling reports .3945 and .0688  we get  0.39450592 0.06884787

#the second Kelley data from Hotelling
kelley <- structure(list(speed = c(1, 0.4248, 0.042, 0.0215, 0.0573), power = c(0.4248,
1, 0.1487, 0.2489, 0.2843), words = c(0.042, 0.1487, 1, 0.6693,
0.4662), symbols = c(0.0215, 0.2489, 0.6693, 1, 0.6915), meaningless = c(0.0573,
0.2843, 0.4662, 0.6915, 1)), class = "data.frame", row.names = c("speed",
"power", "words", "symbols", "meaningless"))


lowerMat(kelley)

mod2 <- lmCor(power + speed ~ words + symbols + meaningless,data=kelley)  #formula mode
#lmCor(y= 1:2,x = 3:5,data = kelley) #order of variables input
cancorDiagram(mod2, cut=.05)
#Hotelling reports canonical correlations of .3073 and .0583  or squared correlations of
# 0.09443329 and 0.00339889 vs. our values of cancor = 0.3076 0.0593  with squared values
#of  0.0946 0.0035,
```

```
lmCor(y=c(7:9),x=c(1:6),data=Thurstone,n.obs=213)  #easier to just list variable
                                                #locations if we have long names
#now try partialling out some variables
lmCor(y=c(7:9),x=c(1:3),z=c(4:6),data=Thurstone) #compare with the previous
#compare complete print out with summary printing
sc <- lmCor(SATV + SATQ ~ gender + education,data=sat.act) # regression from raw data
sc
summary(sc)

lmCor(Pedigrees ~ Sentences + Vocabulary - First.Letters - Four.Letter.Words ,
data=Thurstone)  #showing formula input with two covariates

#Do some regressions with real data (rather than correlation matrices)
lmCor(reaction ~ cond + pmi + import, data = Tal.Or)

#partial out importance
lmCor(reaction ~ cond + pmi - import, data = Tal.Or, main="Partial out importance")

#compare with using lm by partialling
mod1 <- lm(reaction ~ cond + pmi + import, data = Tal.Or)
reaction.import <- lm(reaction~import,data=Tal.Or)$resid
cond.import <-  lm(cond~import,data=Tal.Or)$resid
pmi.import <-   lm(pmi~import,data=Tal.Or)$resid
mod.partial <- lm(reaction.import ~ cond.import + pmi.import)
summary(mod.partial)
#lm uses raw scores, so set std = FALSE for lmCor
print(lmCor(y = reaction ~ cond + pmi - import, data = Tal.Or,std = FALSE,
 main = "Partial out importance"),digits=4)
#notice that the dfs of the partial approach using lm are 1 more than the lmCor dfs

 #Show how to find quadratic terms
sc <- lmCor(reaction ~ cond + pmi + I(import^2), data = Tal.Or)
sc
#pairs.panels(sc$data) #show the SPLOM of the data

#Consider an example of a derivation and cross validation sample
set.seed(42)
ss <- sample(1:2800,1400)
model <- lmCor(y=26:28,x=1:25,data=bfi[ss,],plot=FALSE)
original.fit <- crossValidation(model,bfi[ss,]) #the derivation set
cross.fit <- crossValidation(model,bfi[-ss,])  #the cross validation set
summary(original.fit)
summary(cross.fit)
predicted <- predict(model,bfi[-ss,])
cor2(predicted,bfi[-ss,26:28])  #these are the correlations of the predicted with observed

#now do it for the correlations matrices - these should be the same
R1 <- lowerCor(bfi[ss,], show=FALSE)
R2 <- lowerCor(bfi[-ss,], show=FALSE)
mod1 <- lmCor(y=26:28,x=1:25,data=R1,plot=FALSE)
original <-  crossValidation(model,R1) #the derivation set capitalizes on chance
cross <- crossValidation(model,R2)  #the cross validation set -- values are lower
```

```
summary(original)  #inflated by chance
summary(cross)    #cross validated values are better estimates
```

---

logistic *Logistic transform from x to p and logit transform from p to x*

Description

The logistic function (1/(1+exp(-x)) and logit function (log(p/(1-p)) are fundamental to Item Response Theory. Although just one line functions, they are included here for ease of demonstrations and in drawing IRT models. Also included is the logistic.grm for a graded response model.

Usage

```
logistic(x,d=0, a=1,c=0, z=1)
logit(p)
logistic.grm( x,d=0,a=1.5,c=0,z=1,r=2,s=c(-1.5,-.5,.5,1.5))
```

Arguments

x	Any integer or real value
d	Item difficulty or delta parameter
a	The slope of the curve at x=0 is equivalent to the discrimination parameter in 2PL models or alpha parameter. Is either 1 in 1PL or 1.702 in 1PN approximations.
c	Lower asymptote = guessing parameter in 3PL models or gamma
z	The upper asymptote — in 4PL models
p	Probability to be converted to logit value
r	The response category for the graded response model
s	The response thresholds

Details

These three functions are provided as simple helper functions for demonstrations of Item Response Theory. The one parameter logistic (1PL) model is also known as the Rasch model. It assumes items differ only in difficulty. 1PL, 2PL, 3PL and 4PL curves may be drawn by choosing the appropriate d (delta or item difficulty), a (discrimination or slope), c (gamma or guessing) and z (zeta or upper asymptote).

logit is just the inverse of logistic.

logistic.grm will create the responses for a graded response model for the rth category where cutpoints are in s.

Value

p	logistic returns the probability associated with x
x	logit returns the real number associated with p

Author(s)

William Revelle

Examples

```
curve(logistic(x,a=1.702),-3,3,ylab="Probability of x",
 main="Logistic transform of x",xlab="z score units")
 #logistic with a=1.702 is almost the same as pnorm

curve(pnorm(x),add=TRUE,lty="dashed")
curve(logistic(x),add=TRUE)
text(2,.8, expression(alpha ==1))
text(2,1.0,expression(alpha==1.7))
curve(logistic(x),-4,4,ylab="Probability of x",
            main = "Logistic transform of x in logit units",xlab="logits")
curve(logistic(x,d=-1),add=TRUE)
curve(logistic(x,d=1),add=TRUE)
curve(logistic(x,c=.2),add=TRUE,lty="dashed")
text(1.3,.5,"d=1")
text(.3,.5,"d=0")
text(-1.5,.5,"d=-1")
text(-3,.3,"c=.2")
#demo of graded response model
 curve(logistic.grm(x,r=1),-4,4,ylim=c(0,1),main="Five level response scale",
           ylab="Probability of endorsement",xlab="Latent attribute on logit scale")
 curve(logistic.grm(x,r=2),add=TRUE)
 curve(logistic.grm(x,r=3),add=TRUE)
 curve(logistic.grm(x,r=4),add=TRUE)
 curve(logistic.grm(x,r=5),add=TRUE)

 text(-2.,.5,1)
 text(-1.,.4,2)
 text(0,.4,3)
 text(1.,.4,4)
  text(2.,.4,5)
```

---

lowerUpper — *Combine two square matrices to have a lower off diagonal for one, upper off diagonal for the other*

Description

When reporting correlation matrices for two samples (e.g., males and females), it is convenient to show them as one matrix, with entries below the diagonal representing one matrix, and entries above the diagonal the other matrix. It is also useful to compare a correlation matrix with the residuals from a fitted (e.g., factor) model.

Usage

```
lowerUpper(lower, upper=NULL, diff=FALSE)
```

Arguments

lower	A square matrix
upper	A square matrix of the same size as the first (if omitted, then the matrix is converted to two symmetric matrices).
diff	Find the difference between the first and second matrix and put the results in the above the diagonal entries.

Details

If just one matrix is provided (i.e., upper is missing), it is decomposed into two square matrices, one equal to the lower off diagonal entries, the other to the upper off diagonal entries. In the normal case two symmetric matrices are provided and combined into one non-symmetric matrix with the lower off diagonals representing the lower matrix and the upper off diagonals representing the upper matrix.

If diff is true, the upper off diagonal matrix reflects the differences between the two matrices.

Value

Either one matrix or a list of two

Author(s)

William Revelle

See Also

read.clipboard.lower, corPlot

Examples

```
 b1 <- Bechtoldt.1
 b2 <- Bechtoldt.2
 b12 <- lowerUpper(b1,b2)
 cor.plot(b12)
 diff12 <- lowerUpper(b1,b2,diff=TRUE)

 corPlot(t(diff12),numbers=TRUE,main="Bechtoldt1 and the differences from Bechtoldt2")

 #Compare r and partial r
 lower <- lowerCor(sat.act)
 upper <- partial.r(sat.act)
 both = lowerUpper(lower,upper)
 corPlot(both,numbers=TRUE,main="r and partial r for the sat.act data set")
#now show the differences
 both = lowerUpper(lower,upper,diff=TRUE)
 corPlot(both,numbers=TRUE,main="Differences between r and partial r for the sat.act data set")
```

make.keys *Create a keys matrix for use by score.items or cluster.cor*

Description

When scoring items by forming composite scales either from the raw data using scoreItems or from the correlation matrix using cluster.cor, it used to be necessary to create a keys matrix. This is no longer necessary as most of the scoring functions will directly use a keys list. make.keys is just a short cut for creating a keys matrix. The keys matrix is a nvar x nscales matrix of -1,0, 1 and defines the membership for each scale. Items can be specified by location or by name.

Usage

```
make.keys(nvars, keys.list, item.labels = NULL, key.labels = NULL)
keys2list(keys,sign=TRUE)
selectFromKeys(keys.list)
makePositiveKeys(keys.list,sign=FALSE)
```

Arguments

nvars	Number of variables items to be scored, or the name of the data.frame/matrix to be scored
keys.list	A list of the scoring keys, one element for each scale
item.labels	Typically, just the colnames of the items data matrix.
key.labels	Labels for the scales can be specified here, or in the key.list
keys	A keys matrix returned from make.keys
sign	if TRUE, prefix negatively keyed items with - (e.g., "-E2")

Details

The easiest way to prepare keys for scoreItems, scoreOverlap, scoreIrt.1pl, or scoreIrt.2pl is to specify a keys.list. This is just a list specifying the name of the scales to be scores and the direction of the items to be used.

In earlier versions (prior to 1.6.9) keys were formed as a matrix of -1, 0, and 1s for all the items using make.keys. This is no longer necessary, but make.keys is kept for compatibility with earlier versions.

There are three ways to create keys for the scoreItems, scoreOverlap, scoreIrt.1pl, or scoreIrt.2pl functions. One is to laboriously do it in a spreadsheet and then copy them into R. The other is to just specify them by item number in a list. make.keys allows one to specify items by name or by location or a mixture of both.

keys2list reverses the make.keys process and returns a list of scoring keys with the item names for each item to be keyed. If sign=FALSE, this is just a list of the items to be scored. (Useful for scoreIrt.2pl

selectFromKeys will strip the signs from a keys.list and create a vector of item names (deleting duplicates) associated with those keys. This is useful if using a keys.list to define scales and then

just selecting those items that are in subset of the keys.list. This is now done in the scoring functions in the interest of speed.

Since these scoring functions scoreItems, scoreOverlap, scoreIrt.1pl, or scoreIrt.2pl can now (> version 1.6.9) just take a keys.list as input, make.keys is not as important, but is kept for documentation purposes.

To address items by name it is necessary to specify item names, either by using the item.labels value, or by putting the name of the data file or the colnames of the data file to be scored into the first (nvars) position.

If specifying by number (location), then nvars is the total number of items in the object to be scored, not just the number of items used.

See the examples for the various options.

Note that make.keys was revised in Sept, 2013 to allow for keying by name.

It is also possible to do several make.keys operations and then combine them using superMatrix. The alternative, if using the keys.list features is just to concatenate them.

makePositiveKeys is useful for taking subsets of keys (e.g. from bestScales)and create separate keys for the positively and negatively keyed items.

Value

keys — a nvars x nkeys matrix of -1, 0, or 1s describing how to score each scale. nkeys is the length of the keys.list

See Also

scoreItems, scoreOverlap, cluster.cor superMatrix

Examples

```
data(attitude)  #specify the items by location
 key.list <- list(all=c(1,2,3,4,-5,6,7),
                  first=c(1,2,3),
                  last=c(4,5,6,7))
 keys <- make.keys(7,key.list,item.labels = colnames(attitude))
 keys
 #now, undo this
new.keys.list <- keys2list(keys)  #note, these are now given the variable names

select <- selectFromKeys(key.list)


 #scores <- score.items(keys,attitude)
 #scores

# data(bfi)
 #first create the keys by location (the conventional way)
 keys.list <- list(agree=c(-1,2:5),conscientious=c(6:8,-9,-10),
 extraversion=c(-11,-12,13:15),neuroticism=c(16:20),openness = c(21,-22,23,24,-25))
 keys <- make.keys(25,keys.list,item.labels=colnames(bfi)[1:25])
 new.keys.list <- keys2list(keys)  #these will be in the form of variable names
```

```
 #alternatively, create by a mixture of names and locations
 keys.list <- list(agree=c("-A1","A2","A3","A4","A5"),
conscientious=c("C1","C2","C2","-C4","-C5"),extraversion=c("-E1","-E2","E3","E4","E5"),
neuroticism=c(16:20),openness = c(21,-22,23,24,-25))
keys <- make.keys(bfi, keys.list) #specify the data file to be scored (bfi)
#or
keys <- make.keys(colnames(bfi),keys.list) #specify the names of the variables
#to be used
#or
#specify the number of variables to be scored and their names in all cases
keys <- make.keys(28,keys.list,colnames(bfi))


 scores <- scoreItems(keys,bfi)
 summary(scores)
```

---

manhattan *"Manhattan" plots of correlations with a set of criteria.*

Description

A useful way of showing the strength of many correlations with a particular criterion is the Manhattan plot. This is just a plot of correlations ordered by some keying variable. Useful to understand the basis of items used in bestScales.

Usage

```
manhattan(x, criteria = NULL, keys = NULL,raw=TRUE,n.obs=NULL, abs = TRUE,
  ylab = NULL, labels = NULL, log.p = FALSE,ci=.05, pch = 21,
 main = "Manhattan Plot of", adjust="holm",ylim = NULL,digits=2,dictionary=NULL, ...)
```

Arguments

x	A matrix or data.frame of items or a correlation matrix.
criteria	What column names should be predicted. If a separate file, what are the variables to predict.
keys	a keys.list similar to that used in scoreItems
raw	The default is raw data, the alternative is a correlation matrix
n.obs	If given a correlation matrix, and showing log.p, we need the number of observations
abs	Should we show the absolute value of the correlations.
ylab	If NULL, will label as either correlations or log (10) of correlations
labels	if NULL, will use the names of the keys

log.p	Should we show the correlations (log.p = FALSE) or the log of the probabilities of the correlations (TRUE)
ci	The probability for the upper and lower confidence intervals – bonferroni adjusted
pch	The default plot chararcter is a filled circle
main	The title for each criterion
adjust	Which adjustment for multiple correlations should be applied ("holm", "bonferroni", "none")
ylim	If NULL will be the min and max of the data
digits	Round off the results to digits
dictionary	A dictionary of items
...	Other graphic parameters

Details

When exploring the correlations of many items with a few criteria, it is useful to form scales from the most correlated items (see bestScales. To get a feeling of the distribution of items across various measures, we can display their correlations (or the log of the probabilities) grouped by some set of scale keys. May also be used to display and order correlations (rows) with a criteria (columns) if given a correlation as input (raw=FALSE).

Value

The correlations or the log p values are returned (invisibily)

Author(s)

William Revelle

See Also

bestScales, error.dots

Examples

```
op <- par(mfrow=(c(2,3))) #we want to compare two different sets of plots
manhattan(bfi[1:25],bfi[26:28]
,labels=colnames(bfi)[1:25], dictionary=bfi.dictionary)
manhattan(bfi[1:25],bfi[26:28],log.p=TRUE,
    dictionary=bfi.dictionary)

#Do it again, but now show items by the keys.list
bfi.keys <-
  list(agree=c("-A1","A2","A3","A4","A5"),conscientious=c("C1","C2","C3","-C4","-C5"),
 extraversion=c("-E1","-E2","E3","E4","E5"),neuroticism=c("N1","N2","N3","N4","N5"),
 openness = c("O1","-O2","O3","O4","-O5"))
man <-  manhattan(bfi[1:25],bfi[26:28],keys=bfi.keys,
     dictionary=bfi.dictionary[1:2])
```

```
manhattan(bfi[1:25],bfi[26:28],keys=bfi.keys,log.p=TRUE,
dictionary=bfi.dictionary[1:2])

#Alternatively, use a matrix as input
R <-cor(bfi[1:25],bfi[26:28],use="pairwise")
manhattan(R,cs(gender,education,age),keys=bfi.keys,
      dictionary=bfi.dictionary[1:2], raw=FALSE,abs=FALSE)
par <- op



psych:::dfOrder(man,1,ascending=FALSE)  #print out the items sorted on gender
```

`mardia` *Calculate univariate or multivariate (Mardia's test) skew and kurtosis for a vector, matrix, or data.frame*

Description

Find the skew and kurtosis for each variable in a data.frame or matrix. Unlike skew and kurtosis in e1071, this calculates a different skew for each variable or column of a data.frame/matrix. mardia applies Mardia's tests for multivariate skew and kurtosis

Usage

```
skew(x, na.rm = TRUE,type=3)
kurtosi(x, na.rm = TRUE,type=3)
mardia(x,na.rm = TRUE,plot=TRUE)
```

Arguments

`x`	A data.frame or matrix
`na.rm`	how to treat missing data
`type`	See the discussion in describe
`.`	
`plot`	Plot the expected normal distribution values versus the Mahalanobis distance of the subjects.

Details

given a matrix or data.frame x, find the skew or kurtosis for each column (for skew and kurtosis) or the multivariate skew and kurtosis in the case of mardia.

As of version 1.2.3,when finding the skew and the kurtosis, there are three different options available. These match the choices available in skewness and kurtosis found in the e1071 package (see Joanes and Gill (1998) for the advantages of each one).

If we define $m_r = [\sum(X - mx)^r]/n$ then

Type 1 finds skewness and kurtosis by $g_1 = m_3/(m_2)^{3/2}$ and $g_2 = m_4/(m_2)^2 - 3$.

Type 2 is $G1 = g1 * \sqrt{n * (n-1)}/(n-2)$ and $G2 = (n-1) * [(n+1)g2 + 6]/((n-2)(n-3))$.

Type 3 is $b1 = [(n-1)/n]^{3/2} m_3/m_2^{3/2}$ and $b2 = [(n-1)/n]^{3/2} m_4/m_2^2)$.

For consistency with e1071 and with the Joanes and Gill, the types are now defined as above.

However, from revision 1.0.93 to 1.2.3, kurtosi by default gives an unbiased estimate of the kurtosis (DeCarlo, 1997). Prior versions used a different equation which produced a biased estimate. (See the kurtosis function in the e1071 package for the distinction between these two formulae. The default, type 1 gave what is called type 2 in e1071. The other is their type 3.) For comparison with previous releases, specifying type = 2 will give the old estimate. These type numbers are now changed.

Value

`skew`	if input is a matrix or data.frame, skew is a vector of skews
`kurtosi`	if input is a matrix or data.frame, kurtosi is a vector of kurtosi
`bp1`	Mardia's bp1 estimate of multivariate skew
`bp2`	Mardia's bp2 estimate of multivariate kurtosis
`skew`	Mardia's skew statistic
`small.skew`	Mardia's small sample skew statistic
`p.skew`	Probability of skew
`p.small`	Probability of small.skew
`kurtosis`	Mardia's multivariate kurtosis statistic
`p.kurtosis`	Probability of kurtosis statistic
`D`	Mahalanobis distance of cases from centroid

Note

The mean function supplies means for the columns of a data.frame, but the overall mean for a matrix. Mean will throw a warning for non-numeric data, but colMeans stops with non-numeric data. Thus, the function uses either mean (for data frames) or colMeans (for matrices). This is true for skew and kurtosi as well.

Note

Probability values less than 10^-300 are set to 0.

Author(s)

William Revelle

References

Joanes, D.N. and Gill, C.A (1998). Comparing measures of sample skewness and kurtosis. The Statistician, 47, 183-189.

L.DeCarlo. 1997) On the meaning and use of kurtosis, Psychological Methods, 2(3):292-307,

K.V. Mardia (1970). Measures of multivariate skewness and kurtosis with applications. Biometrika, 57(3):pp. 519-30, 1970.

See Also

describe, describe.by, mult.norm in QuantPsyc, Kurt in QuantPsyc

Examples

```
round(skew(attitude),2)   #type 3 (default)
round(kurtosi(attitude),2)  #type 3 (default)
#for the differences between the three types of skew and kurtosis:
round(skew(attitude,type=1),2)  #type 1
round(skew(attitude,type=2),2)  #type 2
mardia(attitude)
x <- matrix(rnorm(1000),ncol=10)
describe(x)
mardia(x)
```

---

mat.sort *Sort the elements of a correlation matrix to reflect factor loadings*

Description

To see the structure of a correlation matrix, it is helpful to organize the items so that the similar items are grouped together. One such grouping technique is factor analysis. mat.sort will sort the items by a factor model (if specified), or any other order, or by the loadings on the first factor (if unspecified)

Usage

```
mat.sort(m, f = NULL)
matSort(m, f = NULL)
```

Arguments

m	A correlation matrix
f	A factor analysis output (i.e., one with a loadings matrix) or a matrix of weights

Details

The factor analysis output is sorted by size of the largest factor loading for each variable and then the matrix items are organized by those loadings. The default is to sort by the loadings on the first factor. Alternatives allow for ordering based upon any vector or matrix.

Value

A sorted correlation matrix, suitable for showing with corPlot.

Author(s)

William Revelle

See Also

fa, corPlot

Examples

```
data(Bechtoldt.1)
sorted <- mat.sort(Bechtoldt.1,fa(Bechtoldt.1,5))
corPlot(sorted,xlas=2) #vertical xaxis names
```

---

matrix.addition *A function to add two vectors or matrices*

Description

It is sometimes convenient to add two vectors or matrices in an operation analogous to matrix multiplication. For matrices nXm and mYp, the matrix sum of the i,jth element of nSp = sum(over m) of iXm + mYj.

Usage

```
x %+% y
```

Arguments

x	a n by m matrix (or vector if m= 1)
y	a m by p matrix (or vector if m = 1)

Details

Used in such problems as Thurstonian scaling. Although not technically matrix addition, as pointed out by Krus, there are many applications where the sum or difference of two vectors or matrices is a useful operation. An alternative operation for vectors is outer(x ,y , FUN="+") but this does not work for matrices.

Value

a n by p matix of sums

Author(s)

William Revelle

References

Krus, D. J. (2001) Matrix addition. Journal of Visual Statistics, 1, (February, 2001).

Examples

```
x <- seq(1,4)
z <- x %+% -t(x)
x
z
#compare with outer(x,-x,FUN="+")
x <- matrix(seq(1,6),ncol=2)
y <- matrix(seq(1,10),nrow=2)
z <- x %+% y
x
y
z
#but compare this with outer(x ,y,FUN="+")
```

---

mediate — *Estimate and display direct and indirect effects of mediators and moderator in path models*

Description

Find the direct and indirect effects of a predictor in path models of mediation and moderation. Bootstrap confidence intervals for the indirect effects. Mediation models are just extended regression models making explicit the effect of particular covariates in the model. Moderation is done by multiplication of the predictor variables. This function supplies basic mediation/moderation analyses for some of the classic problem types.

Usage

```
mediate(y, x, m=NULL, data, mod = NULL, z = NULL, n.obs = NULL, use = "pairwise",
 n.iter = 5000,  alpha = 0.05, std = FALSE,plot=TRUE,zero=TRUE,part=FALSE,
  main="Mediation")
mediate.diagram(medi,digits=2,ylim=c(3,7),xlim=c(-1,10),show.c=TRUE,
     main="Mediation model",cex=1,l.cex=1,...)
moderate.diagram(medi,digits=2,ylim=c(2,8),main="Moderation model", cex=1,l.cex=1,...)
```

Arguments

y	The dependent variable (or a formula suitable for a linear model), If a formula, then this is of the form y ~ x +(m) -z (see details)
x	One or more predictor variables

m	One (or more) mediating variables
data	A data frame holding the data or a correlation or covariance matrix.
mod	A moderating variable, if desired
z	Variables to partial out, if desired
n.obs	If the data are from a correlation or covariance matrix, how many observations were used. This will lead to simulated data for the bootstrap.
use	use="pairwise" is the default when finding correlations or covariances
n.iter	Number of bootstrap resamplings to conduct
alpha	Set the width of the confidence interval to be 1 - alpha
std	standardize the covariances to find the standardized betas
part	if part=TRUE, find part correlations otherwise find partial correlations when partialling
plot	Plot the resulting paths
zero	By default, will zero center the data before doing moderation
digits	The number of digits to report in the mediate.diagram.
medi	The output from mediate may be imported into mediate.diagram
ylim	The limits for the y axis in the mediate and moderate diagram functions
xlim	The limits for the x axis. Make the minimum more negative if the x by x correlations do not fit.
show.c	If FALSE, do not draw the c lines, just the partialed (c') lines
main	The title for the mediate and moderate functions
cex	Adjust the text size (defaults to 1)
l.cex	Adjust the text size in arrows, defaults to cex which in turn defaults to 1
...	Additional graphical parameters to pass to mediate.diagram

Details

When doing linear modeling, it is frequently convenient to estimate the direct effect of a predictor controlling for the indirect effect of a mediator. See Preacher and Hayes (2004) for a very thorough discussion of mediation. The mediate function will do some basic mediation and moderation models, with bootstrapped confidence intervals for the mediation/moderation effects.

Functionally, this is just regular linear regression and partial correlation with some different output.

In the case of two predictor variables, X and M, and a criterion variable Y, then the direct effect of X on Y, labeled with the path c, is said to be mediated by the effect of x on M (path a) and the effect of M on Y (path b). This partial effect (a b) is said to mediate the direct effect of X –c–> Y: X –a -> M –b–> Y with X –c'–> Y where c' = c - ab.

Testing the significance of the ab mediation effect is done through bootstrapping many random resamples (with replacement) of the data.

For moderation, the moderation effect of Z on the relationship between X -> Y is found by taking the (centered) product of X and Z and then adding this XZ term into the regression. By default, the data are zero centered before doing moderation (product terms). This is following the advice

of Cohen, Cohen, West and Aiken (2003). However, to agree with the analyses reported in Hayes (2013) we can set the zero=FALSE option to not zero center the data.

To partial out variables, either define them in the z term, or express as negative entries in the formula mode:

y1 ~ x1 + x2 + (m1)+ (m2) -z will look for the effect of x1 and x2 on y, mediated through m1 and m2 after z is partialled out.

Moderated mediation is done by specifying a product term.

y1 ~ x1 + x2*x3 + (m1)+ (m2) -z will look for the effect of x1, x2, x3 and the product of x2 and x3 on y, mediated through m1 and m2 after z is partialled out.

In the case of being provided just a correlation matrix, the bootstrapped values are based upon bootstrapping from data matching the original covariance/correlation matrix with the addition of normal errors. This allows us to test the mediation/moderation effect even if not given raw data. Moderation can not be done with just correlation matrix.

The function has been tested against some of the basic cases and examples in Hayes (2013) and the associated data sets.

Unless there is a temporal component that allows one to directly distinguish causal paths (time does not reverse direction), interpreting mediation models is problematic. Some people find it useful to compare the differences between mediation models where the causal paths (arrows) are reversed. This is a mistake and should not be done (Thoemmes, 2015).

For fine tuning the size of the graphic output, xlim and ylim can be specified in the mediate.diagram function. Otherwise, the graphics produced by mediate and moderate use the default xlim and ylim values.

Interaction terms (moderation) or mediated moderation can be specified as product terms.

Value

`total`	The total direct effect of x on y (c)
`direct`	The beta effects of x (c') and m (b) on y
`indirect`	The indirect effect of x through m on y (c-ab)
`mean.boot`	mean bootstrapped value of indirect effect
`sd.boot`	Standard deviation of bootstrapped values
`ci.quant`	The upper and lower confidence intervals based upon the quantiles of the bootstrapped distribution.
`boot`	The bootstrapped values themselves.
`a`	The effect of x on m
`b`	The effect of m on y
`b.int`	The interaction of x and mod (if specified)
`data`	The original data plus the product term (if specified)

Note

There are a number of other packages that do mediation analysis (e.g., sem and lavaan) and they are probably preferred for more complicated models. This function is supplied for the more basic cases, with 1..k y variables, 1..n x variables, 1 ..j mediators and 1 ..z variables to partial. The number of moderated effects is not limited, but more than 3rd order interactions are not very meaningful. It will not do two step mediation.

The current version will not correctly handle more than one DV.

Author(s)

William Revelle

References

J. Cohen, P. Cohen, S.G. West, and L.S. Aiken. (2003) Applied multiple regression/correlation analysis for the behavioral sciences. L. Erlbaum Associates, Mahwah, N.J., 3rd ed edition.

Hayes, Andrew F. (2013) Introduction to mediation, moderation, and conditional process analysis: A regression-based approach. Guilford Press.

Preacher, Kristopher J and Hayes, Andrew F (2004) SPSS and SAS procedures for estimating indirect effects in simple mediation models. Behavior Research Methods, Instruments, & Computers 36, (4) 717-731.

Thoemmes, Felix (2015) Reversing arrows in mediation models does not distinguish plausible models. Basic and applied social psychology, 27: 226-234.

Data from Hayes (2013), Preacher and Hayes (2004), and from Kerchoff (1974).

The Tal_Or data set is from Nurit Tal-Or and Jonathan Cohen and Yariv Tsfati and Albert C. Gunther, (2010) “Testing Causal Direction in the Influence of Presumed Media Influence", Communication Research, 37, 801-824 and is used with their kind permission. It is adapted from the webpage of A.F. Hayes. (www.afhayes.com/public/hayes2013data.zip).

The Garcia data set is from Garcia, Donna M. and Schmitt, Michael T. and Branscombe, Nyla R. and Ellemers, Naomi (2010). Women’s reactions to ingroup members who protest discriminatory treatment: The importance of beliefs about inequality and response appropriateness. European Journal of Social Psychology, (40) 733-745 and is used with their kind permission. It was downloaded from the Hayes (2013) website.

For an example of how to display the sexism by protest interaction, see the examples in the `GSBE` (Garcia) data set.

See the “how to do mediation and moderation" at personality-project.org/r/psych/HowTo/mediation.pdf as well as the introductory vignette.

See Also

`lmCor` and `lmCor.diagram` for regression and moderation, `Garcia` for further demonstrations of mediation and moderation.

Examples

```
# A simple mediation example is the Tal_Or data set (pmi for Hayes)
#The pmi data set from Hayes is available as the Tal_Or data set.
mod4 <- mediate(reaction ~ cond + (pmi), data =Tal_Or,n.iter=50)
summary(mod4)
#Two mediators (from Hayes model 6 (chapter 5))
mod6 <- mediate(reaction ~ cond + (pmi) + (import), data =Tal_Or,n.iter=50)
summary(mod6)

#Moderated mediation is done for the Garcia (Garcia, 2010) data set.
# (see Hayes, 2013 for the protest data set
#n.iter set to 50 (instead of default of 5000) for speed of example
#no mediation, just an interaction
mod7 <- mediate(liking ~  sexism * prot2 , data=Garcia, n.iter = 50)
summary(mod7)
data(GSBE)   #The Garcia et al data set (aka GSBE)
mod11.4 <- mediate(liking ~  sexism * prot2 + (respappr), data=Garcia,
        n.iter = 50,zero=FALSE)   #to match Hayes
summary(mod11.4)
#to see this interaction graphically, run the examples in ?Garcia


#Simulated data from Preacher and Hayes (2004)
sobel <- structure(list(SATIS = c(-0.59, 1.3, 0.02, 0.01, 0.79, -0.35,
-0.03, 1.75, -0.8, -1.2, -1.27, 0.7, -1.59, 0.68, -0.39, 1.33,
-1.59, 1.34, 0.1, 0.05, 0.66, 0.56, 0.85, 0.88, 0.14, -0.72,
0.84, -1.13, -0.13, 0.2), THERAPY = structure(c(0, 1, 1, 0, 1,
1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1,
1, 1, 1, 0), value.labels = c(cognitive = 1, standard = 0)),
    ATTRIB = c(-1.17, 0.04, 0.58, -0.23, 0.62, -0.26, -0.28,
    0.52, 0.34, -0.09, -1.09, 1.05, -1.84, -0.95, 0.15, 0.07,
    -0.1, 2.35, 0.75, 0.49, 0.67, 1.21, 0.31, 1.97, -0.94, 0.11,
    -0.54, -0.23, 0.05, -1.07)), row.names = c(NA, -30L),
     class = "data.frame", variable.labels = c(SATIS = "Satisfaction",
THERAPY = "Therapy", ATTRIB = "Attributional Positivity"))
 #n.iter set to 50 (instead of default of 5000) for speed of example

#There are several forms of input.  The original specified y, x , and the mediator
#mediate(1,2,3,sobel,n.iter=50)  #The example in Preacher and Hayes
#As of October, 2017 we can specify this in a formula mode
mediate (SATIS ~ THERAPY + (ATTRIB),data=sobel, n.iter=50) #specify the mediator by
# adding parentheses

#A.C. Kerchoff, (1974) Ambition and Attainment: A Study of Four Samples of American Boys.
#Data from sem package taken from Kerckhoff (and in turn, from Lisrel manual)
R.kerch <- structure(list(Intelligence = c(1, -0.1, 0.277, 0.25, 0.572,
0.489, 0.335), Siblings = c(-0.1, 1, -0.152, -0.108, -0.105,
-0.213, -0.153), FatherEd = c(0.277, -0.152, 1, 0.611, 0.294,
0.446, 0.303), FatherOcc = c(0.25, -0.108, 0.611, 1, 0.248, 0.41,
0.331), Grades = c(0.572, -0.105, 0.294, 0.248, 1, 0.597, 0.478
), EducExp = c(0.489, -0.213, 0.446, 0.41, 0.597, 1, 0.651),
    OccupAsp = c(0.335, -0.153, 0.303, 0.331, 0.478, 0.651, 1
```

```
    )), class = "data.frame", row.names = c("Intelligence", "Siblings",
"FatherEd", "FatherOcc", "Grades", "EducExp", "OccupAsp"))


 #n.iter set to 50 (instead of default of 5000) for speed of demo
#mod.k <- mediate("OccupAsp","Intelligence",m= c(2:5),data=R.kerch,n.obs=767,n.iter=50)
#new style
mod.k <- mediate(OccupAsp ~ Intelligence + (Siblings) + (FatherEd) + (FatherOcc) +
(Grades), data = R.kerch, n.obs=767, n.iter=50)

mediate.diagram(mod.k)
#print the path values
mod.k

#Compare the following solution to the path coefficients found by the sem package

#mod.k2 <- mediate(y="OccupAsp",x=c("Intelligence","Siblings","FatherEd","FatherOcc"),
#     m= c(5:6),data=R.kerch,n.obs=767,n.iter=50)
#new format
mod.k2 <- mediate(OccupAsp ~ Intelligence + Siblings + FatherEd + FatherOcc + (Grades) +
(EducExp),data=R.kerch, n.obs=767, n.iter=50)
mediate.diagram(mod.k2,show.c=FALSE) #simpler output
#print the path values
mod.k2

#Several interesting test cases are taken from analyses of the Spengler data set
#This is temporarily added to psych from psychTools to help build for CRAN
#Although the sample sizes are actually very large in the first wave,  I use the
#sample sizes from the last wave
#We set the n.iter to be 50 instead of the default value of 5,000
if(require("psychTools")) {
 mod1 <- mediate(Income.50 ~ IQ + Parental+ (Ed.11) ,data=Spengler,
    n.obs = 1952, n.iter=50)
 mod2 <- mediate(Income.50 ~ IQ + Parental+ (Ed.11)  + (Income.11)
  ,data=Spengler,n.obs = 1952, n.iter=50)

  #Now, compare these models
anova(mod1,mod2)

 #Current version does not support two DVs
 #mod22 <-  mediate(Income.50 + Educ.50 ~ IQ + Parental+ (Ed.11)  + (Income.11)
 #    ,data=Spengler,n.obs = 1952, n.iter=50)

}
```

`mixedCor` *Find correlations for mixtures of continuous, polytomous, and dichotomous variables*

Description

For data sets with continuous, polytomous and dichotmous variables, the absolute Pearson correlation is downward biased from the underlying latent correlation. mixedCor finds Pearson correlations for the continous variables, polychorics for the polytomous items, tetrachorics for the dichotomous items, and the polyserial or biserial correlations for the various mixed variables. Results include the complete correlation matrix, as well as the separate correlation matrices and difficulties for the polychoric and tetrachoric correlations.

Usage

```
mixedCor(data=NULL,c=NULL,p=NULL,d=NULL,smooth=TRUE,correct=.5,global=TRUE,ncat=8,
             use="pairwise",method="pearson",weight=NULL)

mixed.cor(x = NULL, p = NULL, d=NULL,smooth=TRUE, correct=.5,global=TRUE,
        ncat=8,use="pairwise",method="pearson",weight=NULL)  #deprecated
```

Arguments

data	The data set to be analyzed (either a matrix or dataframe)
c	The names (or locations) of the continuous variables) (may be missing)
x	A set of continuous variables (may be missing) or, if p and d are missing, the variables to be analyzed.
p	A set of polytomous items (may be missing)
d	A set of dichotomous items (may be missing)
smooth	If TRUE, then smooth the correlation matix if it is non-positive definite
correct	When finding tetrachoric correlations, what value should be used to correct for continuity?
global	For polychorics, should the global values of the tau parameters be used, or should the pairwise values be used. Set to local if errors are occurring.
ncat	The number of categories beyond which a variable is considered "continuous".
use	The various options to the cor function include "everything", "all.obs", "complete.obs", "na.or.complete", or "pairwise.complete.obs". The default here is "pairwise"
method	The correlation method to use for the continuous variables. "pearson" (default), "kendall", or "spearman"
weight	If specified, this is a vector of weights (one per participant) to differentially weight participants. The NULL case is equivalent of weights of 1 for all cases.

Details

This function is particularly useful as part of the Synthetic Apeture Personality Assessment (SAPA) (https://www.sapa-project.org/) data sets where continuous variables (age, SAT V, SAT Q, etc) and mixed with polytomous personality items taken from the International Personality Item Pool (IPIP) and the dichotomous experimental IQ items that have been developed as part of SAPA (see, e.g., Revelle, Wilt and Rosenthal, 2010 or Revelle, Dworak and Condon, 2020.).

This is a very computationally intensive function which can be speeded up considerably by using multiple cores and using the parallel package. (See the note for timing comparisons.) This adjusts the number of cores to use when doing polychoric or tetrachoric. The greatest step in speed is going from 1 core to 2. This is about a 50% savings. Going to 4 cores seems to have about at 66% savings, and 8 a 75% savings. The number of parallel processes defaults to 2 but can be modified by using the `options` command: options("mc.cores"=4) will set the number of cores to 4.

Item response analyses using `irt.fa` may be done separately on the polytomous and dichotomous items in order to develop internally consistent scales. These scale may, in turn, be correlated with each other using the complete correlation matrix found by mixed.cor and using the `score.items` function.

This function is not quite as flexible as the hetcor function in John Fox's polychor package.

Note that the variables may be organized by type of data: continuous, polytomous, and dichotomous. This is done by simply specifying c, p, and d. This is advantageous in the case of some continuous variables having a limited number of categories because of subsetting.

`mixedCor` is essentially a wrapper for `cor`, `polychoric`, `tetrachoric`, `polydi` and `polyserial`. It first identifies the types of variables, organizes them by type (continuous, polytomous, dichotomous), calls the appropriate correlation function, and then binds the resulting matrices together.

Value

`rho`	The complete matrix
`rx`	The Pearson correlation matrix for the continuous items
`poly`	the polychoric correlation (poly$rho) and the item difficulties (poly$tau)
`tetra`	the tetrachoric correlation (tetra$rho) and the item difficulties (tetra$tau)

Note

mixedCor was designed for the SAPA project (`https://www.sapa-project.org/`) with large data sets with a mixture of continuous, dichotomous, and polytomous data. For smaller data sets, it is sometimes the case that the global estimate of the tau parameter will lead to unstable solutions. This may be corrected by setting the global parameter = FALSE.

`mixedCor` was added in April, 2017 to be slightly more user friendly. `mixed.cor` was deprecated in February, 2018.

When finding correlations between dummy coded SAPA data (e.g., of occupations), the real correlations are all slightly less than zero because of the ipsatized nature of the data. This leads to a non-positive definite correlation matrix because the matrix is no longer of full rank. Smoothing will correct this, even though this might not be desired. Turn off smoothing in this case.

Note that the variables no longer need to be organized by type of data: first continuous, then polytomous, then dichotomous. However, this automatic detection will lead to problems if the variables such as age are limited to less than 8 categories but those category values differ from the polytomous items. The fall back is to specify x, p, and d.

If the number of alternatives in the polychoric data differ and there are some dicthotomous data, it is advisable to set correct=0.

Timing: For large data sets, `mixedCor` takes a while. Progress messages `progressBar` report what is happening but do not actually report the rate of progress. (The steps are initializing, Pearson r,

polychorics, tetrachoric, polydi). It is recommended to use the multicore option although the benefit between 2, 4 and 8 cores seems fairly small: For large data sets (e.g., 255K subjects, 950 variables), with 4 cores running in parallel (options("mc.cores=4") on a MacBook Pro with 2.8 Ghz Intel Core I7, it took 2,873/2,887 seconds elapsed time, 8,152/7,718 secs of user time, and 1,762/1,489 of system time (with and without smoothing). This is noticeabably better than the 4,842 elapsed time (7,313 user, 1,459 system) for 2 cores but not much worse than running 8 virtual cores, with an elapsed time of 2,629, user time of 13,460, and system time of 2,679. On a Macbook Pro with 2 physical cores and a 3.3 GHz Intel Cor I7, running 4 multicores took 4,423 seconds elapsed time, 12,781 seconds of user time, and 2,605 system time. Running with 2 multicores, took slightly longer: 6,193 seconds elapsed time, 10,099 user time and 2,413 system time.

Note

Updated 9/21/26 so that the weight option actually works! Weights are applied when finding the Pearson correlations as well as the biserial, polyserial, and polychoric correlations. This has been a recurring problem since mixedCor was first released.

Author(s)

William Revelle

References

W.Revelle, J.Wilt, and A.Rosenthal. Personality and cognition: The personality-cognition link. In A.Gruszka, G. Matthews, and B. Szymura, editors, Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control, chapter 2, pages 27-49. Springer, 2010.

W Revelle, D. M. Condon, J. Wilt, J.A. French, A. Brown, and L G. Elleman(2016) Web and phone based data collection using planned missing designs in Nigel G. Fielding and Raymond M. Lee and Grant Blank (eds) SAGE Handbook of Online Research Methods, Sage Publications, Inc.

W. Revelle, E.M. Dworak and D.M. Condon (2020) Exploring the persome: The power of the item in understanding personality structure. Personality and Individual Differences, /10.1016/j.paid.2020.109905

See Also

polychoric, tetrachoric, scoreItems, scoreOverlap scoreIrt

Examples

```
data(bfi)
r <- mixedCor(data=bfi[,c(1:5,26,28)])
r
#this is the same as
r <- mixedCor(data=bfi,p=1:5,c=28,d=26)
r #note how the variable order reflects the original order in the data
#compare to raw Pearson
#note that the biserials and polychorics are not attenuated
rp <- cor(bfi[c(1:5,26,28)],use="pairwise")
lowerMat(rp)
```

mssd *Find von Neuman's Mean Square of Successive Differences*

Description

Von Neuman et al. (1941) discussed the Mean Square of Successive Differences as a measure of variability that takes into account gradual shifts in mean. This is appropriate when studying errors in ballistics or variability and stability in mood when studying affect. For random data, this will be twice the variance, but for data with a sequential order and a positive autocorrelation, this will be much smaller. Since the mssd is just twice the variance - the autocorrelation, it is thus possible to also find the autocorrelation for a particular lag.

Usage

```
mssd(x,group=NULL, lag = 1,na.rm=TRUE)
rmssd(x,group=NULL, lag=1, na.rm=TRUE)
autoR(x,group=NULL,lag=1,na.rm=TRUE,use="pairwise")
```

Arguments

x	a vector, data.frame or matrix
lag	the lag to use when finding diff
group	A column of the x data.frame to be used for grouping
na.rm	Should missing data be removed?
use	How to handle missing data in autoR
.	

Details

When examining multiple measures within subjects, it is sometimes useful to consider the variability of trial by trial observations in addition to the over all between trial variation. The Mean Square of Successive Differences (mssd) and root mean square of successive differences (rmssd) is just

$\sigma^2 = \Sigma(x_i - x_{i+lag})^2/(n - lag)$

Where n-lag is used because there are only n-lag cases.

In the case of multiple subjects (groups) with multiple observations per subject (group), specify the grouping variable will produce output for each group.

Similar functions are available in the matrixStats package. However, the varDiff function in that package is variance of the difference rather than the MeanSquare. This is just the variance and standard deviation applied to the result from the diff function.

Perhaps useful when studying mood, the autoR function finds the autocorrelation for each item for the specified lag. It also returns the rmssd (root means square successive difference). This is done by finding the correlation of the lag data.

Value

The average squared successive difference (mssd) and the square root of the average squared successive difference (rmssd). Note that this is not the same as the standard deviation of the lagged differences.

Author(s)

William Revelle

References

Jahng, Seungmin and Wood, Phillip K and Trull, Timothy J. Analysis of affective instability in ecological momentary assessment: Indices using successive difference and group comparison via multilevel modeling. Psychological methods (2008) 13, 354-375.

Von Neumann, J., Kent, R., Bellinson, H., and Hart, B. (1941). The mean square successive difference. The Annals of Mathematical Statistics, pages 153-162.

See Also

See Also `rmssd` for the standard deviation or `describe` for more conventional statistics. `describeBy` and `statsBy` give group level statistics. See also `link{mlr}`, `link{mlreliability}`, `link{mlPlot}` for other ways of examining within person variability over time.

Examples

```
t <- seq(-pi, pi, .1)
trial <- 1: length(t)
gr <- trial %% 8
c <- cos(t)
ts <- sample(t,length(t))
cs <- cos(ts)
x.df <- data.frame(trial,gr,t,c,ts,cs)
rmssd(x.df)
rmssd(x.df,gr)
autoR(x.df,gr)
describe(x.df)
#pairs.panels(x.df)
#mlPlot(x.df,grp="gr",Time="t",items=c(4:6))
```

---

multi.hist — *Multiple histograms with density and normal fits on one page*

Description

Given a matrix or data.frame, produce histograms for each variable in a "matrix" form. Include normal fits and density distributions for each plot.

The number of rows and columns may be specified, or calculated. May be used for single variables.

Usage

```
multi.hist(x,nrow=NULL,ncol=NULL,density=TRUE,freq=FALSE,bcol="white",
      dcol=c("black","black"),dlty=c("dashed","dotted"),
      main=NULL,mar=c(2,1,1,1),breaks=21,global=TRUE,...)
histBy(x,var,group,data=NULL,density=TRUE,alpha=.5,breaks=21,col,xlab,
            main="Histograms by group",freq=FALSE,...)
```

Arguments

x	matrix or data.frame
var	The variable in x to plot in histBy
group	The name of the variable in x to use as the grouping variable
data	Needs to be specified if using formula input to histBy
nrow	number of rows in the plot
ncol	number of columns in the plot
density	density=TRUE, show the normal fits and density distributions
freq	freq=FALSE shows probability densities and density distribution, freq=TRUE shows frequencies
bcol	Color for the bars
dcol	The color(s) for the normal and the density fits. Defaults to black.
dlty	The line type (lty) of the normal and density fits. (specify the optional graphic parameter lwd to change the line size)
main	title for each panel will be set to the column name unless specified
mar	Specify the lower, left, upper and right hand side margin in lines – set to be tighter than normal default of c(5,4,4,2) + .1
xlab	Label for the x variable
breaks	The number of breaks in histBy (see hist)
global	If TRUE, use the same x-axis for all plots
alpha	The degree of transparency of the overlapping bars in histBy
col	A vector of colors in histBy (defaults to the rainbow)
...	additional graphic parameters (e.g., col)

Details

This allows for quick summaries of multiple distributions. Particularly useful when examining the results of multiple-split halves that come from the `reliability` function.

By default, will try to make a square plot with equal number of rows and columns. However, the number of columns and rows may be specified for a particular plot.

Author(s)

William Revelle

See Also

bi.bars for drawing pairwise histograms and scatterHist for bivariate scatter and histograms. densityBy, violinBy and violin for density plots.

Examples

```
multi.hist(sat.act)
multi.hist(sat.act,bcol="red")
multi.hist(sat.act,dcol="blue")  #make both lines blue
multi.hist(sat.act,dcol= c("blue","red"),dlty=c("dotted", "solid"))
multi.hist(sat.act,freq=TRUE)   #show the frequency plot
multi.hist(sat.act,nrow=2)
histBy(sat.act,"SATQ","gender") #input by variable names
histBy(SATQ~ gender, data=sat.act) #formula input
```

---

multilevel.reliability

Find and plot various reliability/gneralizability coefficients for multilevel data

---

Description

Various indicators of reliability of multilevel data (e.g., items over time nested within subjects) may be found using generalizability theory. A basic three way anova is applied to the data from which variance components are extracted. Random effects for a nested design are found by lme. These are, in turn, converted to several reliability/generalizability coefficients. An optional call to lme4 to use lmer may be used for unbalanced designs with missing data. mlArrange is a helper function to convert wide to long format. Data can be rearranged from wide to long format, and multiple lattice plots of observations overtime for multiple variables and multiple subjects are created.

Usage

```
mlr(x, grp = "id", Time = "time", items = c(3:5),alpha=TRUE,icc=FALSE, aov=TRUE,
      lmer=FALSE,lme = TRUE,long=FALSE,values=NA,na.action="na.omit",plot=FALSE,
         main="Lattice Plot by subjects over time")
mlArrange(x, grp = "id", Time = "time", items = c(3:5),extra=NULL)
mlPlot(x, grp = "id", Time = "time", items = c(3:5),extra=NULL,
   col=c("blue","red","black","grey"),type="b",
    main="Lattice Plot by subjects over time",...)
multilevel.reliability(x, grp = "id", Time = "time", items = c(3:5),alpha=TRUE,icc=FALSE,
 aov=TRUE,lmer=FALSE,lme = TRUE,long=FALSE,values=NA,na.action="na.omit",
   plot=FALSE,main="Lattice Plot by subjects over time") #alias for mlr
```

Arguments

x	A data frame with persons, time, and items.
grp	Which variable specifies people (groups)

`Time`	Which variable specifies the temporal sequence?
`items`	Which items should be scored? Note that if there are multiple scales, just specify the items on one scale at a time. An item to be reversed scored can be specified by a minus sign. If long format, this is the column specifying item number.
`alpha`	If TRUE, report alphas for every subject (default)
`icc`	If TRUE, find ICCs for each person – can take a while
`aov`	if FALSE, and if icc is FALSE, then just draw the within subject plots
`lmer`	Should we use the lme4 package and lmer or just do the ANOVA? Requires the lme4 package to be installed. Necessary to do crossed designs with missing data but takes a very long time.
`lme`	If TRUE, will find the nested components of variance. Relatively fast.
`long`	Are the data in wide (default) or long format.
`values`	If the data are in long format, which column name (number) has the values to be analyzed?
`na.action`	How to handle missing data. Passed to the lme function.
`plot`	If TRUE, show a lattice plot of the data by subject
`extra`	Names or locations of extra columns to include in the long output. These will be carried over from the wide form and duplicated for all items. See example.
`col`	Color for the lines in mlPlot. Note that items are categorical and thus drawn in alphabetical order. Order the colors appropriately.
`type`	The standard type for lines (p,l, b)
`main`	The main title for the plot (if drawn)
`...`	Other parameters to pass to xyplot

Details

Classical reliabiity theory estimates the amount of variance in a set of observations due to a true score that varies over subjects. Generalizability theory extends this model to include other sources of variance, specifically, time. The classic studies using this approach are people measured over multiple time points with multiple items. Then the question is, how stable are various individual differences. Intraclass correlations (ICC) are found for each subject over items, and for each subject over time. Alpha reliabilities are found for each subject for the items across time.

More importantly, components of variance for people, items, time, and their interactions are found either by classical analysis of variance (aov) or by multilevel mixed effect modeling (lme). These are then used to form several different estimates of generalizability. Very thoughtful discussions of these procedure may be found in chapters by Shrout and Lane.

The variance components are the Between Person Variance σ_P^2, the variance between items σ_I^2, over time σ_T^2, and their interactions.

Then, RKF is the reliability of average of all ratings across all items and times (Fixed time effects). (Shrout and Lane, Equation 6):

$$R_{kF} = \frac{\sigma_P^2 + \sigma_{PI}^2/n.I}{\sigma_P^2 + \sigma_{PI}^2/n.I + \sigma_e^2/(n.In.P}$$

The generalizability of a single time point across all items (Random time effects) is just

$$R_{1R} = \frac{\sigma_P^2 + \sigma_{PI}^2/n.I}{\sigma_P^2 + \sigma_{PI}^2/n.I + \sigma_T^2 + \sigma_{PT}^2 + \sigma_e^2/(n.I)}$$

(Shrout and Lane equation 7 with a correction per Sean Lane.)

Generalizability of average time points across all items (Random effects). (Shrout and Lane, equation 8)

$$R_{kR} = \frac{\sigma_P^2 + \sigma_{PI}^2/n.I}{\sigma_P^2 + \sigma_{PI}^2/n.I + \sigma_T^2/n.T + \sigma_{PT}^2/n.T + \sigma_e^2/n.I}$$

Generalizability of change scores (Shrout and Lane, equation 9)

$$R_C = \frac{\sigma_{PT}^2}{\sigma_{PT}^2 + \sigma_e^2/n.I}$$

.

If the design may be thought of as fully crossed, then either aov or lmer can be used to estimate the components of variance. With no missing data and a balanced design, these will give identical answers. However aov breaks down with missing data and seems to be very slow and very memory intensive for large problems (5,919 seconds for 209 cases with with 88 time points and three items on a Mac Powerbook with a 2.8 GHZ Intel Core I7). The slowdown probably is memory related, as the memory demands increased to 22.62 GB of compressed memory. lmer will handle this design but is not nearly as slow (242 seconds for the 209 cases with 88 time points and three items) as the aov approach.

If the design is thought of as nested, rather than crossed, the components of variance are found using the lme function from nlme. This is very fast (114 cases with 88 time points and three items took 3.5 seconds).

The nested design leads to the generalizability of K random effects Nested (Shrout and Lane, equation 10):

$$R_{KRN} = \frac{\sigma_P^2}{\sigma_P^2 + \sigma_{T(P)}^2/n.I + \sigma_e^2/(n.In.P}$$

And, finally, to the reliability of between person differences, averaged over items. (Shrout and Lane, equation 11).

$$R_{CN} = \frac{\sigma_{T(P)}^2}{\sigma_{T(P)}^2 + \sigma_e^2/(n.I}$$

Unfortunately, when doing the nested analysis, lme will sometimes issue an obnoxious error about failing to converge. To fix this, turning off lme and just using lmer seems to solve the problem (i.e., set lme=FALSE and lmer=TRUE). (lme is part of core R and its namespace is automatically attached when loading psych). For many problems, lmer is not necessary and is thus not loaded. However sometimes it is useful. To use lmer it is necessary to have the lme4 package installed. It will be automatically loaded if it is installed and requested. In the interests of making a 'thin' package, lmer is suggested,not required.

The input can either be in 'wide' or 'long' form. If in wide form, then specify the grouping variable, the 'time' variable, and the the column numbers or names of the items. (See the first example). If in

long format, then what is the column (name or number) of the dependent variable. (See the second example.)

mlArrange takes a wide data.frame and organizes it into a 'long' data.frame suitable for a lattice xyplot. This is a convenient alternative to stack, particularly for unbalanced designs. The wide data frame is reorganized into a long data frame organized by grp (typically a subject id), by Time (typically a time varying variable, but can be anything, and then stacks the items within each person and time. Extra variables are carried over and matched to the appropriate grp and Time.

Thus, if we have N subjects over t time points for k items, in wide format for N * t rows where each row has k items and e extra pieces of information, we get a N x t * k row by 4 + e column dataframe. The first four columns in the long output are id, time, values, and item names, the remaining columns are the extra values. These could be something such as a trait measure for each subject, or the situation in which the items are given.

mlPlot plots k items over the t time dimensions for each subject.

Value

n.obs	Number of individuals
n.time	Maximum number of time intervals
n.items	Number of items
components	Components of variance associated with individuals, Time, Items, and their interactions.
RkF	Reliability of average of all ratings across all items and times (fixed effects).
R1R	Generalizability of a single time point across all items (Random effects)
RkR	Generalizability of average time points across all items (Random effects)
Rc	Generalizability of change scores over time.
RkRn	Generalizability of between person differences averaged over time and items
Rcn	Generalizability of within person variations averaged over items (nested structure)
ANOVA	The summary anova table from which the components are found (if done),
s.lmer	The summary of the lmer analysis (if done),
s.lme	The summary of the lme analysis (if done),
alpha	Within subject alpha over items and time.
summary.by.person	Summary table of ICCs organized by person,
summary.by.time	Summary table of ICCs organized by time.
ICC.by.person	A rather long list of ICCs by person.
ICC.by.time	Another long list of ICCs, this time for each time period,
long	The data (x) have been rearranged into long form for graphics or for further analyses using lme, lmer, or aov that require long form.

Author(s)

William Revelle

References

Bolger, Niall and Laurenceau, Jean-Phillippe, (2013) Intensive longitudinal models. New York. Guilford Press.

Cranford, J. A., Shrout, P. E., Iida, M., Rafaeli, E., Yip, T., & Bolger, N. (2006). A procedure for evaluating sensitivity to within-person change: Can mood measures in diary studies detect change reliably? Personality and Social Psychology Bulletin, 32(7), 917-929.

Revelle, W. and Condon, D.M. (2019) Reliability from alpha to omega: A tutorial. Psychological Assessment, 31, 12, 1395-1411. https://doi.org/10.1037/pas0000754. https://personality-project.org/revelle/publications/rc.pa.19.pdf Preprint available from personality-proejct.

Revelle, W. and Wilt, J. (2017) Analyzing dynamic data: a tutorial. Personality and Individual Differences. DOI: 10.1016/j.paid.2017.08.020

Shrout, Patrick and Lane, Sean P (2012), Psychometrics. In M.R. Mehl and T.S. Conner (eds) Handbook of research methods for studying daily life, (p 302-320) New York. Guilford Press

See Also

sim.multi and sim.multilevel to generate multilevel data, statsBy a for statistics for multi level analysis.

Examples

```
#data from Shrout and Lane, 2012.

shrout <- structure(list(Person = c(1L, 2L, 3L, 4L, 5L, 1L, 2L, 3L, 4L,
5L, 1L, 2L, 3L, 4L, 5L, 1L, 2L, 3L, 4L, 5L), Time = c(1L, 1L,
1L, 1L, 1L, 2L, 2L, 2L, 2L, 2L, 3L, 3L, 3L, 3L, 3L, 4L, 4L, 4L,
4L, 4L), Item1 = c(2L, 3L, 6L, 3L, 7L, 3L, 5L, 6L, 3L, 8L, 4L,
4L, 7L, 5L, 6L, 1L, 5L, 8L, 8L, 6L), Item2 = c(3L, 4L, 6L, 4L,
8L, 3L, 7L, 7L, 5L, 8L, 2L, 6L, 8L, 6L, 7L, 3L, 9L, 9L, 7L, 8L
), Item3 = c(6L, 4L, 5L, 3L, 7L, 4L, 7L, 8L, 9L, 9L, 5L, 7L,
9L, 7L, 8L, 4L, 7L, 9L, 9L, 6L)), class = "data.frame", row.names = c(NA,
-20L))

#make shrout super wide
#Xwide <- reshape(shrout,v.names=c("Item1","Item2","Item3"),timevar="Time",
#direction="wide",idvar="Person")
#add more helpful Names
#colnames(Xwide ) <- c("Person",c(paste0("Item",1:3,".T",1),paste0("Item",1:3,".T",2),
#paste0("Item",1:3,".T",3),paste0("Item",1:3,".T",4)))
#make superwide into normal form  (i.e., just return it to the original shrout data
#Xlong <-Xlong <- reshape(Xwide,idvar="Person",2:13)

#Now use these data for a multilevel repliability study, use the normal wide form output
mg <- mlr(shrout,grp="Person",Time="Time",items=3:5)
#which is the same as
#mg <- multilevel.reliability(shrout,grp="Person",Time="Time",items=
#         c("Item1","Item2","Item3"),plot=TRUE)
#to show the lattice plot by subjects, set plot = TRUE
```

```
#Alternatively for long input (returned in this case from the prior run)
mlr(mg$long,grp="id",Time ="time",items="items", values="values",long=TRUE)

#example of mlArrange
#First, add two new columns to shrout and
#then convert to long output using mlArrange
total <- rowSums(shrout[3:5])
caseid <- rep(paste0("ID",1:5),4)
new.shrout <- cbind(shrout,total=total,case=caseid)
#now convert to long
new.long <- mlArrange(new.shrout,grp="Person",Time="Time",items =3:5,extra=6:7)
headTail(new.long,6,6)
```

---

omega *Calculate McDonald's omega estimates of general and total factor saturation*

Description

McDonald has proposed coefficient omega as an estimate of the general factor saturation of a test. One way to find omega is to do a factor analysis of the original data set, rotate the factors obliquely, do a Schmid Leiman transformation, and then find omega. This function estimates omega as suggested by McDonald by using hierarchical factor analysis (following Jensen). A related option is to define the model using omega and then perform a confirmatory (bi-factor) analysis using the sem or lavaan packages. This is done by omegaSem and omegaFromSem. omegaFromSem will convert appropriate sem/lavaan objects to find omega. Yet another option is to do the direct Schmid-Leiman of Waller.

Usage

```
omega(m,nfactors=3,fm="minres",n.iter=1,p=.05,poly=FALSE,key=NULL,
    flip=TRUE,digits=2, title="Omega",sl=TRUE,labels=NULL,
   plot=TRUE,n.obs=NA,rotate="oblimin",Phi=NULL,option="equal",covar=FALSE, ...)
omegaSem(m,nfactors=3,fm="minres",key=NULL,flip=TRUE,digits=2,title="Omega",
  sl=TRUE,labels=NULL, plot=TRUE,n.obs=NA,rotate="oblimin",
  Phi = NULL, option="equal",lavaan=TRUE,...)

omegah(m,nfactors=3,fm="minres",key=NULL,flip=TRUE,
digits=2,title="Omega",sl=TRUE,labels=NULL, plot=TRUE,
  n.obs=NA,rotate="oblimin",Phi = NULL,option="equal",covar=FALSE,two.ok=FALSE,...)

omegaFromSem(fit,m=NULL,flip=TRUE,plot=TRUE)
omegaDirect(m,nfactors=3,fm="minres",rotate="oblimin",cut=.3,
   plot=TRUE,main="Direct Schmid Leiman")
directSl(m,nfactors=3,fm="minres",rotate="oblimin",cut=.3)
```

Arguments

m	A correlation matrix, or a data.frame/matrix of data, or (if Phi) is specified, an oblique factor pattern matrix
nfactors	Number of factors believed to be group factors
n.iter	How many replications to do in omega for bootstrapped estimates
fm	factor method (the default is minres) fm="pa" for principal axes, fm="minres" for a minimum residual (OLS) solution, or fm="ml" for maximum likelihood. Although fm="pc" for principal components is possible, it is not a latent variable model and should be avoided (see note). Using fm="pc" will inflate the correct values.
poly	should the correlation matrix be found using polychoric/tetrachoric or normal Pearson correlations
key	a vector of +/- 1s to specify the direction of scoring of items. The default is to assume all items are positively keyed, but if some items are reversed scored, then key should be specified.
flip	If flip is TRUE, then items are automatically flipped to have positive correlations on the general factor. Items that have been reversed are shown with a - sign.
p	probability of two tailed conference boundaries
digits	if specified, round the output to digits
title	Title for this analysis
main	main for this analysis (directSl)
cut	Loadings greater than cut are used in directSl
sl	If plotting the results, should the Schmid Leiman solution be shown or should the hierarchical solution be shown? (default sl=TRUE)
labels	If plotting, what labels should be applied to the variables? If not specified, will default to the column names.
plot	plot=TRUE (default) calls omega.diagram, plot =FALSE does not. If Rgraphviz is available, then omega.graph may be used separately.
n.obs	Number of observations - used for goodness of fit statistic
rotate	What rotation to apply? The default is oblimin, the alternatives include simplimax, Promax, cluster and target. target will rotate to an optional keys matrix (See target.rot)
Phi	If specified, then omega is found from the pattern matrix (m) and the factor intercorrelation matrix (Phi).
option	In the two factor case (not recommended), should the loadings be equal, emphasize the first factor, or emphasize the second factor. See in particular the option parameter in schmid for treating the case of two group factors.
covar	defaults to FALSE and the correlation matrix is found (standardized variables.) If TRUE, the do the calculations on the unstandardized variables and use covariances.
two.ok	If TRUE, do not give a warning about 3 factors being required.

`lavaan`	if FALSE, will use John Fox's sem package to do the omegaSem. If TRUE, will use Yves Rosseel's lavaan package.
`fit`	The fitted object from lavaan or sem. For lavaan, this includes the correlation matrix and the variable names and thus m needs not be specified.
`...`	Allows additional parameters to be passed through to the factor routines.

Details

"Many scales are assumed by their developers and users to be primarily a measure of one latent variable. When it is also assumed that the scale conforms to the effect indicator model of measurement (as is almost always the case in psychological assessment), it is important to support such an interpretation with evidence regarding the internal structure of that scale. In particular, it is important to examine two related properties pertaining to the internal structure of such a scale. The first property relates to whether all the indicators forming the scale measure a latent variable in common.

The second internal structural property pertains to the proportion of variance in the scale scores (derived from summing or averaging the indicators) accounted for by this latent variable that is common to all the indicators (Cronbach, 1951; McDonald, 1999; Revelle, 1979). That is, if an effect indicator scale is primarily a measure of one latent variable common to all the indicators forming the scale, then that latent variable should account for the majority of the variance in the scale scores. Put differently, this variance ratio provides important information about the sampling fluctuations when estimating individuals' standing on a latent variable common to all the indicators arising from the sampling of indicators (i.e., when dealing with either Type 2 or Type 12 sampling, to use the terminology of Lord, 1956). That is, this variance proportion can be interpreted as the square of the correlation between the scale score and the latent variable common to all the indicators in the infinite universe of indicators of which the scale indicators are a subset. Put yet another way, this variance ratio is important both as reliability and a validity coefficient. This is a reliability issue as the larger this variance ratio is, the more accurately one can predict an individual's relative standing on the latent variable common to all the scale's indicators based on his or her observed scale score. At the same time, this variance ratio also bears on the construct validity of the scale given that construct validity encompasses the internal structure of a scale." (Zinbarg, Yovel, Revelle, and McDonald, 2006).

McDonald has proposed coefficient omega_hierarchical (ω_h) as an estimate of the general factor saturation of a test. Zinbarg, Revelle, Yovel and Li (2005) https://personality-project.org/revelle/publications/zinbarg.revelle.pmet.05.pdf compare McDonald's ω_h to Cronbach's α and Revelle's β. They conclude that ω_h is the best estimate. (See also Zinbarg et al., 2006 and Revelle and Zinbarg (2009)).

One way to find ω_h is to do a factor analysis of the original data set, rotate the factors obliquely, factor that correlation matrix, do a Schmid-Leiman (schmid) transformation to find general factor loadings, and then find ω_h. Here we present code to do that.

ω_h differs as a function of how the factors are estimated. Four options are available, three use the `fa` function but with different factoring methods: the default does a minres factor solution, fm="pa" does a principle axes factor analysis fm="mle" does a maximum likelihood solution; fm="pc" does a principal components analysis using (`principal`). (But see Widaman, 1993, 2018 for important discussions of PCA should not be confused with factor analytic techniques.)

For ability items, it is typically the case that all items will have positive loadings on the general factor. However, for non-cognitive items it is frequently the case that some items are to be scored positively, and some negatively. Although probably better to specify which directions the items are

to be scored by specifying a key vector, if flip =TRUE (the default), items will be reversed so that they have positive loadings on the general factor. The keys are reported so that scores can be found using the `scoreItems` function. Arbitrarily reversing items this way can overestimate the general factor. (See the example with a simulated circumplex).

β, an alternative to ω_h, is defined as the worst split half reliability (Revelle, 1979). It can be estimated by using `ICLUST` (a hierarchical clustering algorithm originally developed for main frames and written in Fortran and that is now part of the psych package. (For a very complimentary review of why the ICLUST algorithm is useful in scale construction, see Cooksey and Soutar, 2005)).

The `omega` function uses exploratory factor analysis to estimate the ω_h coefficient. It is important to remember that "A recommendation that should be heeded, regardless of the method chosen to estimate ω_h, is to always examine the pattern of the estimated general factor loadings prior to estimating ω_h. Such an examination constitutes an informal test of the assumption that there is a latent variable common to all of the scale's indicators that can be conducted even in the context of EFA. If the loadings were salient for only a relatively small subset of the indicators, this would suggest that there is no true general factor underlying the covariance matrix. Just such an informal assumption test would have afforded a great deal of protection against the possibility of misinterpreting the misleading ω_h estimates occasionally produced in the simulations reported here." (Zinbarg et al., 2006, p 137).

A simple demonstration of the problem of an omega estimate reflecting just one of two group factors can be found in the last example.

Diagnostic statistics that reflect the quality of the omega solution include a comparison of the relative size of the g factor eigen value to the other eigen values, the percent of the common variance for each item that is general factor variance (p2), the mean of p2, and the standard deviation of p2. Further diagnostics can be done by describing (describe) the $schmid$sl results.

Although omega_h is uniquely defined only for cases where 3 or more subfactors are extracted, it is sometimes desired to have a two factor solution. By default this is done by forcing the schmid extraction to treat the two subfactors as having equal loadings.

There are three possible options for this condition: setting the general factor loadings between the two lower order factors to be "equal" which will be the sqrt(oblique correlations between the factors) or to "first" or "second" in which case the general factor is equated with either the first or second group factor. A message is issued suggesting that the model is not really well defined. This solution discussed in Zinbarg et al., 2007. To do this in omega, add the option="first" or option="second" to the call.

Although obviously not meaningful for a 1 factor solution, it is of course possible to find the sum of the loadings on the first (and only) factor, square them, and compare them to the overall matrix variance. This is done, with appropriate complaints.

In addition to ω_h, another of McDonald's coefficients is ω_t. This is an estimate of the total reliability of a test.

McDonald's ω_t, which is similar to Guttman's λ_6, `guttman` but uses the estimates of uniqueness (u^2) from factor analysis to find e_j^2. This is based on a decomposition of the variance of a test score, V_x into four parts: that due to a general factor, $\vec{g}$, that due to a set of group factors, $\vec{f}$, (factors common to some but not all of the items), specific factors, $\vec{s}$ unique to each item, and $\vec{e}$, random error. (Because specific variance can not be distinguished from random error unless the test is given at least twice, some combine these both into error).

Letting $\vec{x} = \vec{cg} + \vec{Af} + \vec{Ds} + \vec{e}$ then the communality of item$_j$, based upon general as well as group factors, $h_j^2 = c_j^2 + \sum f_{ij}^2$ and the unique variance for the item $u_j^2 = \sigma_j^2(1 - h_j^2)$ may be used to

estimate the test reliability. That is, if h_j^2 is the communality of item$_j$, based upon general as well as group factors, then for standardized items, $e_j^2 = 1 - h_j^2$ and

$$\omega_t = \frac{\vec{1}c\vec{c}'\vec{1} + \vec{1}A\vec{A}'\vec{1}'}{V_x} = 1 - \frac{\sum(1-h_j^2)}{V_x} = 1 - \frac{\sum u^2}{V_x}$$

Because $h_j^2 \geq r_{smc}^2$, $\omega_t \geq \lambda_6$.

It is important to distinguish here between the two ω coefficients of McDonald, 1978 and Equation 6.20a of McDonald, 1999, ω_t and ω_h. While the former is based upon the sum of squared loadings on all the factors, the latter is based upon the sum of the squared loadings on the general factor.

$$\omega_h = \frac{\vec{1}c\vec{c}'\vec{1}}{V_x}$$

Another estimate reported is the omega for an infinite length test with a structure similar to the observed test (omega H asymptotic). This is found by

$$\omega_{limit} = \frac{\vec{1}c\vec{c}'\vec{1}}{\vec{1}c\vec{c}'\vec{1} + \vec{1}A\vec{A}'\vec{1}'}$$

.

Following suggestions by Steve Reise, the Explained Common Variance (ECV) is also reported. This is the ratio of the general factor eigen value to the sum of all of the eigen values. As such, it is a better indicator of unidimensionality than of the amount of test variance accounted for by a general factor. (But, the *u* or unidim statistic is probably better. See `unidim` and `reliability` functions.)

The input to omega may be a correlation matrix or a raw data matrix, or a factor pattern matrix with the factor intercorrelations (Phi) matrix.

`omega` is an exploratory factor analysis function that uses a Schmid-Leiman transformation. `omegaSem` first calls `omega` and then takes the Schmid-Leiman solution, converts this to a confirmatory sem model and then calls the sem package to conduct a confirmatory model. ω_h is then calculated from the CFA output. Although for well behaved problems, the efa and cfa solutions will be practically identical, the CFA solution will not always agree with the EFA solution. In particular, the estimated R^2 will sometimes exceed 1. (An example of this is the Harman 24 cognitive abilities problem.)

In addition, not all EFA solutions will produce workable CFA solutions. Model misspecifications will lead to very strange CFA estimates.

It is also possible to give `omega` a factor pattern matrix and the associated factor intercorrelation. In this case, the analysis will be done on these matrices. This is particularly useful if one is not satisfied with the exploratory EFA solutions and rotation options and somehow comes up with an alternative. (For instance, one might want to do a EFA using fm='pa' with a Kaiser normalized Promax solution with a specified m value.)

`omegaFromSem` takes the output from a sem model and uses it to find ω_h. The estimate of factor indeterminacy, found by the multiple R^2 of the variables with the factors, will not match that found by the EFA model. In particular, the estimated R^2 will sometimes exceed 1. (An example of this is the Harman 24 cognitive abilities problem.)

The notion of omega may be applied to the individual factors as well as the overall test. A typical use of omega is to identify subscales of a total inventory. Some of that variability is due to the

general factor of the inventory, some to the specific variance of each subscale. Thus, we can find a number of different omega estimates: what percentage of the variance of the items identified with each subfactor is actually due to the general factor. What variance is common but unique to the subfactor, and what is the total reliable variance of each subfactor. These results are reported in omega.group object and in the last few lines of the normal output.

Finally, and still being tested, is omegaDirect adapted from Waller (2017). This is a direct rotation to a Schmid-Leiman like solution without doing the hierarchical factoring (directSl). This rotation is then interpreted in terms of omega. It is included here to allow for comparisons with the alternative procedures omega and omegaSem. Preliminary analyses suggests that it produces inappropriate solutions for the case where there is no general factor.

Moral: Finding omega_h is tricky and one should probably compare omega, omegaSem, omegaDirect and even iclust solutions to understand the differences.

The summary of the omega object is a reduced set of the most useful output.

The various objects returned from omega include:

Value

omega hierarchical
: The ω_h coefficient

omega.lim
: The limit of ω_h as the test becomes infinitly large

omega total
: The $omega_t$ coefficient

alpha
: Cronbach's α

schmid
: The Schmid Leiman transformed factor matrix and associated matrices

schmid$sl
: The g factor loadings as well as the residualized factors

schmid$orthog
: Varimax rotated solution of the original factors

schmid$oblique
: The oblimin or promax transformed factors

schmid$phi
: the correlation matrix of the oblique factors

schmid$gloading
: The loadings on the higher order, g, factor of the oblimin factors

key
: A vector of -1 or 1 showing which direction the items were scored.

model
: a list of two elements, one suitable to give to the sem function for structure equation models, the other, to give to the lavaan package.

sem
: The output from a sem analysis

omega.group
: The summary statistics for the omega total, omega hierarchical (general) and omega within each group.

scores
: Factor score estimates are found for the Schmid-Leiman solution. To get scores for the hierarchical model see the note.

various fit statistics
: various fit statistics, see output

OmegaSem
: is an object that contains the fits for the OmegaSem output.

loadings
: The direct SL rotated object (from omegaDirect)

orth.f
: The original, unrotated solution from omegaDirect

Target
: The cluster based target for rotation in directSl

Note

Requires the GPArotation package.

The default rotation uses oblimin from the GPArotation package. Alternatives include the simplimax function, as well as Promax or the promax rotations. promax will do a Kaiser normalization before applying Promax rotation.

If the factor solution leads to an exactly orthogonal solution (probably only for demonstration data sets), then use the rotate="Promax" option to get a solution.

omegaSem requires the sem or lavaan packages. omegaFromSem uses the output from the sem or lavaan package.

omega may be run on raw data (finding either Pearson or tetrachoric/polychoric corrlations, depending upon the poly option) a correlation matrix, a polychoric correlation matrix (found by e.g., polychoric), or the output of a previous omega run. This last case is particularly useful when working with categorical data using the poly=TRUE option. For in this case, most of the time is spent in finding the correlation matrix. The matrix is saved as part of the omega output and may be used as input for subsequent runs. A similar feature is found in irt.fa where the output of one analysis can be taken as the input to the subsequent analyses.

However, simulations based upon tetrachoric and polychoric correlations suggest that although the structure is better defined, that the estimates of omega are inflated over the true general factor saturation.

Omega returns factor scores based upon the Schmid-Leiman transformation. To get the hierarchical factor scores, it is necessary to do this outside of omega. See the example (not run).

Consider the case of the raw data in an object data. Then

f3 <- fa(data,3,scores="tenBerge", oblique.rotation=TRUE

f1 <- fa(f3$scores)

hier.scores <- data.frame(f1$scores,f3$scores)

When doing fm="pc" option, principal components are done for the original correlation matrix, but minres is used when examining the intercomponent correlations. A warning is issued that the method was changed to minres for the higher order solution. omega is a factor model, and finding loadings using principal components will overestimate the resulting solution. This is particularly problematic for the amount of group saturation, and thus the omega.group statistics are overestimates.

The last three lines of omega report "Total, General and Subset omega for each subset". These are available as the omega.group object in the output.

The last of these (omega group) is effectively what Steve Reise calls omegaS for the subset omega.

The omega general is the amount of variance in the group that is accounted for by the general factor, the omega total is the amount of variance in the group accounted for by general + group.

This is based upon a cluster solution (that is to say, every item is assigned to one group) and this is why for first column the omega general and group do not add up to omega total. Some of the variance is found in the cross loadings between groups.

Reise and others like to report the ratio of the second line to the first line (what portion of the reliable variance is general factor) and the third row to the first (what portion of the reliable variance is within group but not general. This may be found by using the omega.group object that is returned by omega. (See the last example.)

If using the lavaan=TRUE option in omegaSem please note that variable names can not start with a digit (e.g. 4.Letter.Words in the Thurstone data set. The leading digit needs to be removed.

omegaSem will do an exploratory efa and omega, create (and return) the commands for doing either a sem or lavaan analysis. The commands are returned as the model object. This can be used for further sem/lavaan analyses.

Omega can also be found from an analysis done using lavaan or sem directly by calling omegaFromSem with the original correlation matrix and the fit of the sem/lavaan model. See the last (not run) example)

Author(s)

https://personality-project.org/revelle.html
Maintainer: William Revelle < revelle@northwestern.edu >

References

https://personality-project.org/r/r.omega.html

Jensen, Arthur R. and Li-Jen Weng (1994) What is a good g? Intelligence, 18, 3, 231-258

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at https://personality-project.org/r/book/

Revelle, W. (1979). Hierarchical cluster analysis and the internal structure of tests. Multivariate Behavioral Research, 14, 57-74. (https://personality-project.org/revelle/publications/iclust.pdf)

Revelle, W. and Condon, D.M. (2019) Reliability from alpha to omega: A tutorial. Psychological Assessment, 31, 12, 1395-1411. https://doi.org/10.1037/pas0000754. https://osf.io/preprints/psyarxiv/2y3w9 Preprint available from PsyArxiv

Revelle, W. and Zinbarg, R. E. (2009) Coefficients alpha, beta, omega and the glb: comments on Sijtsma. Psychometrika, 74, 1, 145-154. (https://personality-project.org/revelle/publications/rz09.pdf

Waller, N. G. (2017) Direct Schmid-Leiman Transformations and Rank-Deficient Loadings Matrices. Psychometrika. DOI: 10.1007/s11336-017-9599-0

Widaman, K.F. (1993) Common Factor Analysis Versus Principal Component Analysis: Differential Bias in Representing Model Parameters? Multivariate Behavioral Research, 28 (3), 263-311.

Widaman, K.F. (2018) On common factor and principal component representations of data: Implications for theory and for confirmatory replications. Structural Equation Modeling: A Multidisciplinary Journal (25) 829-847

Zinbarg, R.E., Revelle, W., Yovel, I., & Li. W. (2005). Cronbach's Alpha, Revelle's Beta, McDonald's Omega: Their relations with each and two alternative conceptualizations of reliability. Psychometrika. 70, 123-133. https://personality-project.org/revelle/publications/zinbarg.revelle.pmet.05.pdf

Zinbarg, R., Yovel, I. & Revelle, W. (2007). Estimating omega for structures containing two group factors: Perils and prospects. Applied Psychological Measurement. 31 (2), 135-157.

Zinbarg, R., Yovel, I., Revelle, W. & McDonald, R. (2006). Estimating generalizability to a universe of indicators that all have one attribute in common: A comparison of estimators for omega. Applied Psychological Measurement, 30, 121-144. DOI: 10.1177/0146621605278814

See Also

omega.graph for a dot code graphic. ICLUST, ICLUST.diagram for hierarchical cluster analysis, VSS, nfactors, fa.parallel for alternative ways of estimating the appropriate number of factors. schmid for the decomposition used in omega. make.hierarchical to simulate a hierarchical model.

fa.multi for hierarchical factor analysis with an arbitrary number of 2nd order factors.

reliability to do multiple omega analysis at the same time on different subsets of items.

Examples

```
## Not run:
 test.data <- Harman74.cor$cov
# if(!require(GPArotation)) {message("Omega requires GPA rotation" )} else {
      my.omega <- omega(test.data)
      print(my.omega,digits=2)
#}

#create 9 variables with a hierarchical structure
v9 <- sim.hierarchical()
#with correlations of
round(v9,2)
#find omega
v9.omega <- omega(v9,digits=2)
v9.omega

#create 8 items with a two factor solution, showing the use of the flip option
sim2 <- item.sim(8)
omega(sim2)  #an example of misidentification-- remember to look at the loadings matrices.
omega(sim2,2)  #this shows that in fact there is no general factor
omega(sim2,2,option="first") #but, if we define one of the two group factors
     #as a general factor, we get a falsely high omega
#apply omega to analyze 6 mental ability tests
data(ability.cov)   #has a covariance matrix
omega(ability.cov$cov)

#om <- omega(Thurstone)
#round(om$omega.group,2)
#round(om$omega.group[2]/om$omega.group[1],2) #fraction of reliable that is general variance
# round(om$omega.group[3]/om$omega.group[1],2) #fraction of reliable that is group variance

#To find factor score estimates for the hierarchical model it is necessary to
#do two extra steps.

#Consider the case of the raw data in an object data.  (An example from simulation)
# set.seed(42)
# gload <- matrix(c(.9,.8,.7),nrow=3)
# fload <- matrix(c(.8,.7,.6,rep(0,9),.7,.6,.5,rep(0,9),.7,.6,.4),   ncol=3)
# data <- sim.hierarchical(gload=gload,fload=fload, n=100000, raw=TRUE)
#
# f3 <- fa(data$observed,3,scores="tenBerge", oblique.scores=TRUE)
# f1 <- fa(f3$scores)
```

```
# om <- omega(data$observed,sl=FALSE) #draw the hierarchical figure
# The scores from om are based upon the Schmid-Leiman factors and although the g factor
# is identical, the group factors are not.
# This is seen in the following correlation matrix
# hier.scores <- cbind(om$scores,f1$scores,f3$scores)
# lowerCor(hier.scores)
#
#this next set of examples require lavaan
#jensen <- sim.hierarchical()   #create a hierarchical structure (same as v9 above)
#om.jen <- omegaSem(jensen,lavaan=TRUE)  #do the exploratory omega with confirmatory as well
#lav.mod <- om.jen$omegaSem$model$lavaan #get the lavaan code or create it yourself
# lav.mod <- 'g =~ +V1+V2+V3+V4+V5+V6+V7+V8+V9
#              F1=~  + V1 + V2 + V3
#              F2=~  + V4 + V5 + V6
#              F3=~  + V7 + V8 + V9 '
#lav.jen <- cfa(lav.mod,sample.cov=jensen,sample.nobs=500,orthogonal=TRUE,std.lv=TRUE)
# omegaFromSem(lav.jen,jensen)
#the directSl solution
#direct.jen <- directSl(jen)
#direct.jen

#try a one factor solution -- this is not recommended, but sometimes done
#it will just give omega_total
# lav.mod.1 <- 'g =~ +V1+V2+V3+V4+V5+V6+V7+V8+V9 '
#lav.jen.1<- cfa(lav.mod.1,sample.cov=jensen,sample.nobs=500,orthogonal=TRUE,std.lv=TRUE)
# omegaFromSem(lav.jen.1,jensen)




## End(Not run)
```

---

omega.graph *Graph hierarchical factor structures*

Description

Hierarchical factor structures represent the correlations between variables in terms of a smaller set of correlated factors which themselves can be represented by a higher order factor.

Two alternative solutions to such structures are found by the omega function. The correlated factors solutions represents the effect of the higher level, general factor, through its effect on the correlated factors. The other representation makes use of the Schmid Leiman transformation to find the direct effect of the general factor upon the original variables as well as the effect of orthogonal residual group factors upon the items.

Graphic presentations of these two alternatives are helpful in understanding the structure. omega.graph and omega.diagram draw both such structures. Graphs are drawn directly onto the graphics window or expressed in “dot" commands for conversion to graphics using implementations of Graphviz (if using omega.graph).

Using Graphviz allows the user to clean up the Rgraphviz output. However, if Graphviz and Rgraphviz are not available, use omega.diagram.

See the other structural diagramming functions, fa.diagram and structure.diagram.

In addition

Usage

```
omega.diagram(om.results,sl=TRUE,sort=TRUE,labels=NULL,flabels=NULL,cut=.2,
gcut=.2,simple=TRUE,  errors=FALSE, digits=1,e.size=.1,rsize=.15,side=3,
    main=NULL,cex=NULL,color.lines=TRUE,marg=c(.5,.5,1.5,.5),adj=2, ...)
omega.graph(om.results, out.file = NULL,  sl = TRUE, labels = NULL, size = c(8, 6),
    node.font = c("Helvetica", 14), edge.font = c("Helvetica", 10),
    rank.direction=c("RL","TB","LR","BT"), digits = 1, title = "Omega", ...)
```

Arguments

om.results	The output from the omega function
out.file	Optional output file for off line analysis using Graphviz
sl	Orthogonal clusters using the Schmid-Leiman transform (sl=TRUE) or oblique clusters
labels	variable labels
flabels	Labels for the factors (not counting g)
size	size of graphics window
node.font	What font to use for the items
edge.font	What font to use for the edge labels
rank.direction	Defaults to left to right
digits	Precision of labels
cex	control font size
color.lines	Use black for positive, red for negative
marg	The margins for the figure are set to be wider than normal by default
adj	Adjust the location of the factor loadings to vary as factor mod 4 + 1
title	Figure title
main	main figure caption
...	Other options to pass into the graphics packages
e.size	the size to draw the ellipses for the factors. This is scaled by the number of variables.
cut	Minimum path coefficient to draw
gcut	Minimum general factor path to draw
simple	draw just one path per item
sort	sort the solution before making the diagram
side	on which side should errors be drawn?
errors	show the error estimates
rsize	size of the rectangles

Details

While omega.graph requires the Rgraphviz package, omega.diagram does not. omega requires the GPArotation package.

Value

clust.graph	A graph object
sem	A matrix suitable to be run throughe the sem function in the sem package.

Note

omega.graph requires rgraphviz. – omega requires GPArotation

Author(s)

https://personality-project.org/revelle.html
Maintainer: William Revelle < revelle@northwestern.edu >

References

https://personality-project.org/r/r.omega.html

Revelle, W. (in preparation) An Introduction to Psychometric Theory with applications in R. https://personality-project.org/r/book/

Revelle, W. (1979). Hierarchical cluster analysis and the internal structure of tests. Multivariate Behavioral Research, 14, 57-74. (https://personality-project.org/revelle/publications/iclust.pdf)

Zinbarg, R.E., Revelle, W., Yovel, I., & Li. W. (2005). Cronbach's Alpha, Revelle's Beta, McDonald's Omega: Their relations with each and two alternative conceptualizations of reliability. Psychometrika. 70, 123-133. https://personality-project.org/revelle/publications/zinbarg.revelle.pmet.05.pdf

Zinbarg, R., Yovel, I., Revelle, W. & McDonald, R. (2006). Estimating generalizability to a universe of indicators that all have one attribute in common: A comparison of estimators for omega. Applied Psychological Measurement, 30, 121-144. DOI: 10.1177/0146621605278814

See Also

omega, make.hierarchical, ICLUST.rgraph

Examples

```
#24 mental tests from Holzinger-Swineford-Harman
if(require(GPArotation) ) {om24 <- omega(Harman74.cor$cov,4) } #run omega

#
#example hierarchical structure from Jensen and Weng
if(require(GPArotation) ) {jen.omega <- omega(make.hierarchical())}
```

omegaStats *Omega Statistics from a Factor Loading Matrix*

Description

Among the many ways of estimating internal consistency, the family of model based statistics known as omega (omega_hierarchical, omega_total, omega_subtest, omega_bifactor) provide a number of solutions. omegaStats provides common output for several different estimation procedures. The omega function will find three versions of omega, as will the CFA function. Because of their different assumptions, the omega_hierarchical and omega_bifactor will provide different estimates of the general factor saturation of a test. Users of other packages (e.g., lavaan, may find the basic statistics by providing a correlation matrix and factor loadings matrix. See the last example.

Usage

```
omegaStats(r, loads = NULL, Phi = NULL, gload = NULL, group = NULL, h2 = NULL,
    covar = FALSE, lavaan=NULL)
```

Arguments

r	A correlation matrix
loads	A factor loadings matrix with the first column representing a general factor. Or alternatively users may specify the general factor and group factor loadings separately.
Phi	Factor intercorrelations
gload	A matrix of loadings on a general factor (if loads is not specified)
group	A matrix of group factor loadings (if loads is not specified)
h2	Communalities of the variables (r and loads are not specified)
covar	If TRUE, treat the r matrix as a covariance matrix to find u2
lavaan	To find omega from lavaan output, specify the lavaan object (see example)

Details

Model based estimates of internal consistency include the family of estimates known as omega. Omega_hierarchical (found by the omega function) estimates the general factor saturation of a test, while CFA finds the amount of variance accounted by the first factor in a bifactor solution. Although both procedures agree in terms of their estimates for the total amount of reliable variance, they disagree with respect to the amount that is common to all items.

lavaan based estimates differ yet again (see the final example which is not run).

Value

omega hierarchical	The ω_h coefficient
omega.lim	The limit of ω_h as the test becomes infinitly large
omega total	The ω_t coefficient
stats	as found by fa.stats
...	Many other statistics as reported by fa
Call	echoes the call to the function

Note

What is interesting is finding omega from the bifactor solution of non-correlated group factors. Direct CFA.bifactor will find a general factor (even though not there), although conventional fa procedures will fail to find a general factor.

Author(s)

William Revelle

References

Jensen, Arthur R. and Li-Jen Weng (1994) What is a good g? Intelligence, 18, 3, 231-258

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at https://personality-project.org/r/book/

Revelle, W. (1979). Hierarchical cluster analysis and the internal structure of tests. Multivariate Behavioral Research, 14, 57-74. (https://personality-project.org/revelle/publications/iclust.pdf)

Revelle, W. and Condon, D.M. (2019) Reliability from alpha to omega: A tutorial. Psychological Assessment, 31, 12, 1395-1411. https://doi.org/10.1037/pas0000754. https://osf.io/preprints/psyarxiv/2y3w9 Preprint available from PsyArxiv

Revelle, W. and Zinbarg, R. E. (2009) Coefficients alpha, beta, omega and the glb: comments on Sijtsma. Psychometrika, 74, 1, 145-154. (https://personality-project.org/revelle/publications/rz09.pdf

Waller, N. G. (2017) Direct Schmid-Leiman Transformations and Rank-Deficient Loadings Matrices. Psychometrika. DOI: 10.1007/s11336-017-9599-0

Zinbarg, R.E., Revelle, W., Yovel, I., & Li. W. (2005). Cronbach's Alpha, Revelle's Beta, McDonald's Omega: Their relations with each and two alternative conceptualizations of reliability. Psychometrika. 70, 123-133. https://personality-project.org/revelle/publications/zinbarg.revelle.pmet.05.pdf

Zinbarg, R., Yovel, I. & Revelle, W. (2007). Estimating omega for structures containing two group factors: Perils and prospects. Applied Psychological Measurement. 31 (2), 135-157.

Zinbarg, R., Yovel, I., Revelle, W. & McDonald, R. (2006). Estimating generalizability to a universe of indicators that all have one attribute in common: A comparison of estimators for omega. Applied Psychological Measurement, 30, 121-144. DOI: 10.1177/0146621605278814

See Also

fa, CFA, omega

Examples

```
#Create a 3 correlated factor model using default values and 10000 observations
v9 <- sim.hierarchical(n=10000)

model <- 'F1=~ V1 + V2 + V3
          F2=~ V4 + V5 + V6
          F3 =~ V7 +V8 + V9'
test <-  CFA.bifactor(model,v9$observed)
test   #show the results
omegaStats(test$r,test$loadings)


om <- omega(v9$observed)
omegaStats(r=om$R,loads=om$schmid$sl[,1:4])

##Now creat a three factor model with orthogonal factors
model <- 'F1=~ V1 + V2 + V3
          F2=~ V4 + V5 + V6
          F3 =~ V7 +V8 + V9'



model9 <- 'F1 =~ .9*V1 + .8*V2 + .7*V3
           F2 =~ .8*V4 + .7*V5 +.6*V6
           F3 =~ .7*V7 + .6*V8 +.5*V9'

set.seed(42)
v9s <- sim(model9,n=500)
 test <- CFA(model9,v9s$observed )  #do a cfa using Lavaan syntax
 test.bi <- CFA.bifactor(model9,v9s$observed)
 test.hi <- CFA.bifactor(model9,v9s$observed,g=TRUE)
 om <- omega(v9s$observed)
 omegaStats(test.bi$r,test.bi$loadings)

###do not run the following test of lavaan
if(FALSE) {
lav.model <- 'g=~ V1 + V2 + V3+ V4 + V5 + V6+V7 +V8 + V9
          F1=~ V1 + V2 + V3
          F2=~ V4 + V5 + V6
          F3 =~ V7 +V8 + V9'
library(lavaan)
fit <- cfa(lav.model,data=v9$observed,orthogonal=TRUE,std.lv=TRUE, estimator="ULS")
omegaStats(r=v9$r,lavaan=fit)   #from lavaan
fa.congruence(fit,test)
}  #end of lavaan test
```

`outlier` *Find and graph Mahalanobis squared distances to detect outliers*

Description

The Mahalanobis distance is $D^2 = (x - \mu)'\Sigma^-1(x - \mu)$ where Σ is the covariance of the x matrix. D2 may be used as a way of detecting outliers in distribution. Large D2 values, compared to the expected Chi Square values indicate an unusual response pattern. The mahalanobis function in stats does not handle missing data.

Usage

```
outlier(x, plot = TRUE, bad = 5,na.rm = TRUE, xlab, ylab, ...)
```

Arguments

`x`	A data matrix or data.frame
`plot`	Plot the resulting QQ graph
`bad`	Label the bad worst values
`na.rm`	Should missing data be deleted
`xlab`	Label for x axis
`ylab`	Label for y axis
`...`	More graphic parameters, e.g., cex=.8

Details

Adapted from the mahalanobis function and help page from stats.

Value

The D2 values for each case

Author(s)

William Revelle

References

Yuan, Ke-Hai and Zhong, Xiaoling, (2008) Outliers, Leverage Observations, and Influential Cases in Factor Analysis: Using Robust Procedures to Minimize Their Effect, Sociological Methodology, 38, 329-368.

See Also

`mahalanobis`

Examples

```
#first, just find and graph the outliers
d2 <- outlier(sat.act)
#combine with the data frame and plot it with the outliers highlighted in blue
sat.d2 <- data.frame(sat.act,d2)
pairs.panels(sat.d2,bg=c("yellow","blue")[(d2 > 25)+1],pch=21)
```

`p.rep` *Find the probability of replication for an F, t, or r and estimate effect size*

Description

The probability of replication of an experimental or correlational finding as discussed by Peter Killeen (2005) is the probability of finding an effect in the same direction upon an exact replication. For articles submitted to Psychological Science, p.rep needs to be reported.

F, t, p and r are all estimates of the size of an effect. But F, t, and p also are also a function of the sample size. Effect size, d prime, may be expressed as differences between means compared to within cell standard deviations, or as a correlation coefficient. These functions convert p, F, and t to d prime and the r equivalent.

Usage

```
p.rep(p = 0.05, n=NULL,twotailed = FALSE)
p.rep.f(F,df2,twotailed=FALSE)
p.rep.r(r,n,twotailed=TRUE)
p.rep.t(t,df,df2=NULL,twotailed=TRUE)
```

Arguments

`p`	conventional probability of statistic (e.g., of F, t, or r)
`F`	The F statistic
`df`	Degrees of freedom of the t-test, or of the first group if unequal sizes
`df2`	Degrees of freedom of the denominator of F or the second group in an unequal sizes t test
`r`	Correlation coefficient
`n`	Total sample size if using r
`t`	t-statistic if doing a t-test or testing significance of a regression slope
`twotailed`	Should a one or two tailed test be used?

Details

The conventional Null Hypothesis Significance Test (NHST) is the likelihood of observing the data given the null hypothesis of no effect. But this tells us nothing about the probability of the null hypothesis. Peter Killeen (2005) introduced the probability of replication as a more useful measure. The probability of replication is the probability that an exact replication study will find a result in the *same direction* as the original result.

p.rep is based upon a 1 tailed probability value of the observed statistic.

Other frequently called for statistics are estimates of the effect size, expressed either as Cohen's d, Hedges g, or the equivalent value of the correlation, r.

For p.rep.t, if the cell sizes are unequal, the effect size estimates are adjusted by the ratio of the mean cell size to the harmonic mean cell size (see Rownow et al., 2000).

Value

`p.rep`	Probability of replication
`dprime`	Effect size (Cohen's d) if more than just p is specified
`prob`	Probability of F, t, or r. Note that this can be either the one-tailed or two tailed probability value.
`r.equivalent`	For t-tests, the r equivalent to the t (see Rosenthal and Rubin(2003), Rosnow, Rosenthal, and Rubin, 2000))

.

Note

The p.rep value is the one tailed probability value of obtaining a result in the same direction.

References

Cummings, Geoff (2005) Understanding the average probability of replication: comment on Killeen 2005). Psychological Science, 16, 12, 1002-1004).

Killeen, Peter H. (2005) An alternative to Null-Hypothesis Significance Tests. Psychological Science, 16, 345-353

Rosenthal, R. and Rubin, Donald B.(2003), r-sub(equivalent): A Simple Effect Size Indicator. Psychological Methods, 8, 492-496.

Rosnow, Ralph L., Rosenthal, Robert and Rubin, Donald B. (2000) Contrasts and correlations in effect-size estimation, Psychological Science, 11. 446-453.

Examples

```
p.rep(.05)  #probability of replicating a result if the original study had a  p = .05
p.rep.f(9.0,98)  #probability of replicating  a result with F = 9.0 with 98 df
p.rep.r(.4,50)    #probability of replicating a result if r =.4 with n = 50
p.rep.t(3,98)   #probability of replicating a result if t = 3 with df =98
p.rep.t(2.14,84,14) #effect of equal sample sizes (see Rosnow et al.)
```

paired.r *Test the difference between (un)paired correlations*

Description

Test the difference between two (paired or unpaired) correlations. Given 3 variables, x, y, z, is the correlation between xy different than that between xz? If y and z are independent, this is a simple t-test of the z transformed rs. But, if they are dependent, it is a bit more complicated.

Usage

```
paired.r(xy, xz, yz=NULL, n, n2=NULL,twotailed=TRUE)
```

Arguments

xy	r(xy)
xz	r(xz)
yz	r(yz)
n	Number of subjects for first group
n2	Number of subjects in second group (if not equal to n)
twotailed	Calculate two or one tailed probability values

Details

To find the z of the difference between two independent correlations, first convert them to z scores using the Fisher r-z transform and then find the z of the difference between the two correlations. The default assumption is that the group sizes are the same, but the test can be done for different size groups by specifying n2.

If the correlations are not independent (i.e., they are from the same sample) then the correlation with the third variable r(yz) must be specified. Find a t statistic for the difference of thee two dependent correlations.

Value

a list containing the calculated t or z values and the associated two (or one) tailed probability.

t	t test of the difference between two dependent correlations
p	probability of the t or of the z
z	z test of the difference between two independent correlations

Author(s)

William Revelle

See Also

r.test for more tests of independent as well as dependent (paired) tests. p.rep.r for the probability of replicating a particular correlation. cor.test from stats for testing a single correlation and corr.test for finding the values and probabilities of multiple correlations. See also set.cor to do multiple correlations from matrix input.

Examples

```
paired.r(.5,.3, .4, 100) #dependent correlations
paired.r(.5,.3,NULL,100) #independent correlations same sample size
paired.r(.5,.3,NULL, 100, 64) # independent correlations, different sample sizes
```

---

pairs.panels *SPLOM, histograms and correlations for a data matrix*

Description

Adapted from the help page for pairs, pairs.panels shows a scatter plot of matrices (SPLOM), with bivariate scatter plots below the diagonal, histograms on the diagonal, and the Pearson correlation above the diagonal. Useful for descriptive statistics of small data sets. If lm=TRUE, linear regression fits are shown for both y by x and x by y. Correlation ellipses are also shown. Points may be given different colors depending upon some grouping variable. Robust fitting is done using lowess or loess regression. Confidence intervals of either the lm or loess are drawn if requested.

Usage

```
## S3 method for class 'panels'
pairs(x, smooth = TRUE, scale = FALSE, density=TRUE,ellipses=TRUE,
    digits = 2,method="pearson", pch = 20, lm=FALSE,cor=TRUE,jiggle=FALSE,factor=2,
    hist.col="cyan",show.points=TRUE,rug=TRUE, breaks = "Sturges",cex.cor=1,wt=NULL,
      smoother=FALSE,stars=FALSE,ci=FALSE,alpha=.05,hist.border="black" ,
      line.col="blue",ci.col="light blue",...)
```

Arguments

x	a data.frame or matrix
smooth	TRUE draws loess smooths
scale	TRUE scales the correlation font by the size of the absolute correlation.
density	TRUE shows the density plots as well as histograms
ellipses	TRUE draws correlation ellipses
lm	Plot the linear fit rather than the LOESS smoothed fits.
digits	the number of digits to show
method	method parameter for the correlation ("pearson","spearman","kendall")
pch	The plot character (defaults to 20 which is a '.').

cor	If plotting regressions, should correlations be reported?
jiggle	Should the points be jittered before plotting?
factor	factor for jittering (1-5)
hist.col	What color should the histogram on the diagonal be?
show.points	If FALSE, do not show the data points, just the data ellipses and smoothed functions
rug	if TRUE (default) draw a rug under the histogram, if FALSE, don't draw the rug
breaks	If specified, allows control for the number of breaks in the histogram (see the hist function)
cex.cor	If this is specified, this will change the size of the text in the correlations. this allows one to also change the size of the points in the plot by specifying the normal cex values. If just specifying cex, it will change the character size, if cex.cor is specified, then cex will function to change the point size.
wt	If specified, then weight the correlations by a weights matrix (see note for some comments)
smoother	If TRUE, then smooth.scatter the data points – slow but pretty with lots of subjects
stars	For those people who like to show the significance of correlations by using magic astricks, set stars=TRUE
ci	Draw confidence intervals for the linear model or for the loess fit, defaults to ci=FALSE. If confidence intervals are not drawn, the fitting function is lowess.
alpha	The alpha level for the confidence regions, defaults to .05
hist.border	Set to "NA" to not have lines between the histogram values
line.col	Color of the lowess or lm fit
ci.col	Fill color of the confidence intervals for lowess and lm fits
...	other options for pairs

Details

Shamelessly adapted from the pairs help page. Uses panel.cor, panel.cor.scale, and panel.hist, all taken from the help pages for pairs. Also adapts the ellipse function from John Fox's car package.

pairs.panels is most useful when the number of variables to plot is less than about 6-10. It is particularly useful for an initial overview of the data.

To show different groups with different colors, use a plot character (pch) between 21 and 25 and then set the background color to vary by group. (See the second example).

When plotting more than about 10 variables, it is useful to set the gap parameter to something less than 1 (e.g., 0). Alternatively, consider using cor.plot

In addition, when plotting more than about 100-200 cases, it is useful to set the plotting character to be a point. (pch=".")

Sometimes it useful to draw the correlation ellipses and best fitting loess without the points. (points.false=TRUE).

Value

A scatter plot matrix (SPLOM) is drawn in the graphic window. The lower off diagonal draws scatter plots, the diagonal histograms, the upper off diagonal reports the Pearson correlation (with pairwise deletion).

If lm=TRUE, then the scatter plots are drawn above and below the diagonal, each with a linear regression fit. Useful to show the difference between regression lines.

Note

If the data are either categorical or character, this is flagged with an astrix for the variable name. If character, they are changed to factors before plotting.

The wt parameter allows for scatter plots of the raw data while showing the weighted correlation matrix (found by using `cor.wt`). The current implementation uses the first two columns of the weights matrix for all analyses. This is useful, but not perfect. The use of this option would be to plot the means from a `statsBy` analysis and then display the weighted correlations by specifying the means and ns from the statsBy run. See the final (not run) example.

See Also

`pairs` which is the base from which pairs.panels is derived, `cor.plot` to do a heat map of correlations, and `scatter.hist` to draw a single correlation plot with histograms and best fitted lines.

Also see `outlier` for an example of coloring a subset of the points.

To find the probability "significance" of the correlations using normal theory, use `corr.test`. To find confidence intervals using boot strapping procedures, use `cor.ci`. To graphically show confidence intervals, see `cor.plot.upperLowerCi`.

Examples

```
pairs.panels(attitude)   #see the graphics window
data(iris)
pairs.panels(iris[1:4],bg=c("red","yellow","blue")[iris$Species],
        pch=21,main="Fisher Iris data by Species") #to show color grouping

pairs.panels(iris[1:4],bg=c("red","yellow","blue")[iris$Species],
  pch=21+as.numeric(iris$Species),main="Fisher Iris data by Species",hist.col="red")
   #to show changing the diagonal

#now show confidence intervals
pairs.panels(iris[1:4],bg=c("red","yellow","blue")[iris$Species], ci=TRUE,
  pch=21+as.numeric(iris$Species),main="Fisher Iris data by Species",hist.col="red")
   #to show changing the diagonal

#to show 'significance'
   pairs.panels(iris[1:4],bg=c("red","yellow","blue")[iris$Species],
 pch=21+as.numeric(iris$Species),main="Fisher Iris data by Species",hist.col="red",stars=TRUE)



#demonstrate not showing the data points
```

```
data(sat.act)
pairs.panels(sat.act,show.points=FALSE)
#better yet is to show the points as a period
pairs.panels(sat.act,pch=".")
#show many variables with 0 gap between scatterplots
# data(bfi)
# pairs.panels(bfi,show.points=FALSE,gap=0)

#plot raw data points and then the weighted correlations.
#output from statsBy
sb <- statsBy(sat.act,"education")
pairs.panels(sb$mean,wt=sb$n)  #report the weighted correlations
#compare with
pairs.panels(sb$mean) #unweighted correlations

#to color just some of the data points, see the example from outlier
```

---

pairwiseCount *Count number of pairwise cases for a data set with missing (NA) data and impute values.*

Description

When doing cor(x, use= "pairwise"), it is nice to know the number of cases for each pairwise correlation. This is particularly useful when doing SAPA type analyses. More importantly, when there are some missing pairs, it is useful to supply imputed values so that further analyses may be done. This is useful if using the Massively Missing Completely at Random (MMCAR) designs used by the SAPA project. The specific pairs missing may be identified by pairwiseZero. Summaries of the counts are given by pairwiseDescribe.

Usage

```
pairwiseCount(x, y = NULL,diagonal=TRUE)
pairwiseDescribe(x,y,diagonal=FALSE,...)
pairwiseZero(x,y=NULL, min=0, short=TRUE)
pairwiseImpute(keys,R,fix=FALSE)
pairwiseReport(x,y=NULL,cut=0,diagonal=FALSE,...)
pairwiseSample(x,y=NULL,diagonal=FALSE,size=100,...)
pairwiseCountBig(x,size=NULL)
pairwisePlot(x,y=NULL,upper=TRUE,diagonal=TRUE,labels=TRUE,show.legend=TRUE,n.legend=10,
colors=FALSE,gr=NULL,minlength=6,xlas=1,ylas=2,
main="Relative Frequencies",count=TRUE,...)

count.pairwise(x, y = NULL,diagonal=TRUE) #deprecated
```

Arguments

x	An input matrix, typically a data matrix ready to be correlated.

y	An optional second input matrix
diagonal	if TRUE, then report the diagonal, else fill the diagonals with NA
...	Other parameters to pass to describe
min	Count the number of item pairs with <= min entries
short	Show the table of the item pairs that have entries <= min
keys	A keys.list specifying which items belong to which scale.
R	A correlation matrix to be described or imputed
cut	Report the item pairs and numbers with cell sizes less than cut
fix	If TRUE, then replace all NA correlations with the mean correlation for that within or between scale
upper	Should the upper off diagonal matrix be drawn, or left blank?
labels	if NULL, use column and row names, otherwise use labels
show.legend	A legend (key) to the colors is shown on the right hand side
n.legend	How many categories should be labelled in the legend?
colors	Defaults to FALSE and will use a grey scale. colors=TRUE use colors \ from the colorRampPalette from red through white to blue
minlength	If not NULL, then the maximum number of characters to use in row/column labels
xlas	Orientation of the x axis labels (1 = horizontal, 0, parallel to axis, 2 perpendicular to axis)
ylas	Orientation of the y axis labels (1 = horizontal, 0, parallel to axis, 2 perpendicular to axis)
main	A title. Defaults to "Relative Frequencies"
gr	A color gradient: e.g., gr <- colorRampPalette(c("#B52127", "white", "#2171B5")) will produce slightly more pleasing (to some) colors. See next to last example of corPlot.
count	Should we count the number of pairwise observations using pairwiseCount, or just plot the counts for a matrix?
size	Sample size of the number of variables to sample in pairwiseSample

Details

When using Massively Missing Completely at Random (MMCAR) designs used by the SAPA project, it is important to count the number of pairwise observations (pairwiseCount). If there are pairs with 1 or fewer observations, these will produce NA values for correlations making subsequent factor analyses fa or reliability analsyes omega or scoreOverlap impossible.

pairwiseCountBig may be used to count the cells of large data sets. It is analogous to bigCor and returns the cell sizes for each pair of correlations.

In order to identify item pairs with counts less than a certain value pairwiseReport reports the names of those pairs with fewer than 'cut' observations. By default, it just reports the number of offending items. With short=FALSE, the print will give the items with n.obs < cut. Even more detail is available in the returned objects.

The specific pairs that have values <= n min in any particular table of the paiwise counts may be given by pairwiseZero.

To remedy the problem of missing correlations, we impute the missing correlations using pairwiseImpute. The technique takes advantage of the scale based structure of SAPA items. Items within a scale (e.g. Letter Number Series similar to the ability items) are imputed to correlate with items from another scale (e.g., Matrix Reasoning) at the average of these two between scale inter-item mean correlations. The average correlations within and between scales are reported by pairwiseImpute and if the fix paremeter is specified, the imputed correlation matrix is returned.

Alternative methods of imputing these correlations are not yet implemented.

The time to count cell size varies linearly by the number of subjects and of the number of items squared. This becomes prohibitive for larger (big n items) data sets. pairwiseSample will take samples of size=size of these bigger data sets and then returns basic descriptive statistics of these counts, including mean, median, and the .05, .25, .5, .75 and .95 quantiles.

Value

result	= matrix of counts of pairwise observations (if pairwiseCount)
av.r	The average correlation value of the observed correlations within/between scales
count	The numer of observed correlations within/between each scale
percent	The percentage of complete data by scale
imputed	The original correlation matrix with NA values replaced by the mean correlation for items within/between the appropriate scale.

Author(s)

Maintainer: William Revelle <revelle@northwestern.edu>

Examples

```
x <- matrix(rnorm(900),ncol=6)
y <- matrix(rnorm(450),ncol=3)
x[x < 0] <- NA
y[y > 1] <- NA

pairwiseCount(x)
pairwiseCount(y)
pairwiseCount(x,y)
pairwiseCount(x,diagonal=FALSE)
pairwiseDescribe(x,quant=c(.1,.25,.5,.75,.9))

#examine the structure of the ability data set
if(require(psychTools)) {
keys <- list(ICAR16=colnames(psychTools::ability),reasoning =
  cs(reason.4,reason.16,reason.17,reason.19),
  letters=cs(letter.7, letter.33,letter.34,letter.58, letter.7),
  matrix=cs(matrix.45,matrix.46,matrix.47,matrix.55),
  rotate=cs(rotate.3,rotate.4,rotate.6,rotate.8))
 pairwiseImpute(keys,psychTools::ability)
```

```
}
```

---

parcels *Find miniscales (parcels) of size 2 or 3 from a set of items*

Description

Given a set of n items, form n/2 or n/3 mini scales or parcels of the most similar pairs or triplets of items. These may be used as the basis for subsequent scale construction or multivariate (e.g., factor) analysis.

Usage

```
parcels(x, size = 3, max = TRUE, flip=TRUE,congruence = FALSE)
keysort(keys)
```

Arguments

x	A matrix/dataframe of items or a correlation/covariance matrix of items
size	Form parcels of size 2 or size 3
flip	if flip=TRUE, negative correlations lead to at least one item being negatively scored
max	Should item correlation/covariance be adjusted for their maximum correlation
congruence	Should the correlations be converted to congruence coefficients?
keys	Sort a matrix of keys to reflect item order as much as possible

Details

Items used in measuring ability or other aspects of personality are typically not very reliable. One suggestion has been to form items into homogeneous item composites (HICs), Factorially Homogeneous Item Dimensions (FHIDs) or mini scales (parcels). Parcelling may be done rationally, factorially, or empirically based upon the structure of the correlation/covariance matrix. `link{parcels}` facilitates the finding of parcels by forming a keys matrix suitable for using in `score.items`. These keys represent the n/2 most similar pairs or the n/3 most similar triplets.

The algorithm is straightforward: For size = 2, the correlation matrix is searched for the highest correlation. These two items form the first parcel and are dropped from the matrix. The procedure is repeated until there are no more pairs to form.

For size=3, the three items with the greatest sum of variances and covariances with each other is found. This triplet is the first parcel. All three items are removed and the procedure then identifies the next most similar triplet. The procedure repeats until n/3 parcels are identified.

Value

keys	A matrix of scoring keys to be used to form mini scales (parcels) These will be in order of importance, that is, the first parcel (P1) will reflect the most similar pair or triplet. The keys may also be sorted by average item order by using the keysort function.

Author(s)

William Revelle

References

Cattell, R. B. (1956). Validation and intensification of the sixteen personality factor questionnaire. Journal of Clinical Psychology , 12 (3), 205 -214.

See Also

scoreItems to score the parcels or iclust for an alternative way of forming item clusters.

Examples

```
parcels(Thurstone)
keys <- parcels(bfi)
keys <- keysort(keys)
scoreItems(keys,bfi)
```

---

partial.r — *Find the partial correlations for a set (x) of variables with set (y) removed.*

Description

A straightforward application of matrix algebra to remove the effect of the variables in the y set from the x set. Input may be either a data matrix or a correlation matrix. Variables in x and y are specified by location. If x and y are not specified, then the effect of all variables are partialled from all the other correlations. May also be done using formula input which is more convenient when comparing results to regression models. Also has the option to find part (aka semi-partial) correlations.

Usage

```
partial.r(data, x, y, use="pairwise",method="pearson",part=FALSE)
```

Arguments

data	A data or correlation matrix
x	The variable names or locations associated with the X set (or formula input)
y	The variable names or locations associated with the Y set to be partialled from the X set
use	How should we treat missing data? The default is pairwise complete.
method	Which method of correlation should we use, the default is pearson.
part	Find the part correlation (aka semi-partial) , defaults to finding partial correlations

Details

There are two ways to use partial.r. One is to find the complete partial correlation matrix (that is, partial all the other variables out of each variable). This may be done by simply specifying the raw data or correlation matrix. (In the case of raw data, correlations will be found according to use and method.) In this case, just specify the data matrix.

Alternatively, if we think of the data as an X matrix and Y matrix, then (D = X + Y) with correlations R. Then the partial correlations of the X predictors with the Y variables partialled out are just the last column of R^(-1). See the Tal.Or example below.

The second usage is to partial a set of variables(y) out of another set (x). It is sometimes convenient to partial the effect of a number of variables (e.g., sex, age, education) out of the correlations of another set of variables. This could be done laboriously by finding the residuals of various multiple correlations, and then correlating these residuals. The matrix algebra alternative is to do it directly. To find the confidence intervals and "significance" of the correlations, use the corr.p function with n = n - s where s is the number of covariates.

A perhaps easier format is to use formula input compatible with that used in lmCor. If using formula input,we specify X and Y with the partialled variables specified by subtraction. That is X ~ Y - z, This is useful in the case of multiple regression using which uses this notation.

Following a thoughtful request from Fransisco Wilheim, I just find the correlations of the variables specified in the call (previously I had found the entire correlation matrix, which is a waste of time and breaks if some variables are non-numeric).)

In the case of non-positive definite matrices, find the Pinv (pseudo inverse) of the matrix.

Value

The matrix of partial correlations.

Author(s)

William Revelle

References

Revelle, W. (in prep) An introduction to psychometric theory with applications in R. To be published by Springer. (working draft available at https://personality-project.org/r/book/

See Also

lmCor for a similar application for regression. lowerMat to neatly show a correlation matrix, and corr.p to find the confidence intervals of a correlation.

Examples

```
jen <- make.hierarchical()    #make up a correlation matrix
lowerMat(jen[1:5,1:5])
par.r <- partial.r(jen,c(1,3,5),c(2,4))
lowerMat(par.r)
#or
R <- jen[1:5,1:5]
```

```
par.r <- partial.r(R, y = cs(V2,V4))
lowerMat(par.r)
cp <- corr.p(par.r,n=98)  #assumes the jen data based upon n =100.
print(cp,short=FALSE)  #show the confidence intervals as well
 #partial all from all correlations.
lowerMat(partial.r(jen))


#Consider the Tal.Or data set.
lowerCor(Tal.Or)
#partial gender and age from these relations (they hardly change)
partial.r(Tal.Or,1:4,cs(gender,age))
#find the partial correlations between the first three variables and the DV (reaction)
round(partial.r(Tal.Or[1:4])[4,1:3],2) #The partial correlations with the criterion

#Consider the eminence data set from Del Giudice.
if(require("psychTools")) {
data(eminence)
partial.r(reputation ~ works + citations - birth.year, data=eminence)
#now do a part correlation
partial.r(reputation ~ works + citations - birth.year, data=eminence, part=TRUE)
}
```

---

phi *Find the phi coefficient of correlation between two dichotomous variables*

Description

Given a 1 x 4 vector or a 2 x 2 matrix of frequencies, find the phi coefficient of correlation. Typical use is in the case of predicting a dichotomous criterion from a dichotomous predictor.

Usage

```
phi(t, digits = 2)
```

Arguments

t	a 1 x 4 vector or a 2 x 2 matrix
digits	round the result to digits

Details

In many prediction situations, a dichotomous predictor (accept/reject) is validated against a dichotomous criterion (success/failure). Although a polychoric correlation estimates the underlying Pearson correlation as if the predictor and criteria were continuous and bivariate normal variables,

and the tetrachoric correlation if both x and y are assumed to dichotomized normal distributions, the phi coefficient is the Pearson applied to a matrix of 0's and 1s.

The phi coefficient was first reported by Yule (1912), but should not be confused with the Yule Q coefficient.

For a very useful discussion of various measures of association given a 2 x 2 table, and why one should probably prefer the Yule Q coefficient, see Warren (2008).

Given a two x two table of counts

a	b	a+b (R1)
c	d	c+d (R2)
a+c(C1)	b+d (C2)	a+b+c+d (N)

convert all counts to fractions of the total and then
Phi = [a- (a+b)*(a+c)]/sqrt((a+b)(c+d)(a+c)(b+d)) =
(a - R1 * C1)/sqrt(R1 * R2 * C1 * C2)

This is in contrast to the Yule coefficient, Q, where
Q = (ad - bc)/(ad+bc) which is the same as
[a- (a+b)*(a+c)]/(ad+bc)

Since the phi coefficient is just a Pearson correlation applied to dichotomous data, to find a matrix of phis from a data set involves just finding the correlations using cor or lowerCor or corr.test.

Value

phi coefficient of correlation

Author(s)

William Revelle with modifications by Leo Gurtler

References

Warrens, Matthijs (2008), On Association Coefficients for 2x2 Tables and Properties That Do Not Depend on the Marginal Distributions. Psychometrika, 73, 777-789.

Yule, G.U. (1912). On the methods of measuring the association between two attributes. Journal of the Royal Statistical Society, 75, 579-652.

See Also

phi2tetra, AUC, Yule, Yule.inv Yule2phi, comorbidity, tetrachoric and polychoric

Examples

```
phi(c(30,20,20,30))
phi(c(40,10,10,40))
x <- matrix(c(40,5,20,20),ncol=2)
phi(x)
```

phi.demo *A simple demonstration of the Pearson, phi, and polychoric corelation*

Description

A not very interesting demo of what happens if bivariate continuous data are dichotomized. Bascially a demo of r, phi, and polychor.

Usage

```
phi.demo(n=1000,r=.6, cuts=c(-2,-1,0,1,2))
```

Arguments

n	number of cases to simulate
r	correlation between latent and observed
cuts	form dichotomized variables at the value of cuts

Details

A demonstration of the problem of different base rates on the phi correlation, and how these are partially solved by using the polychoric correlation. Not one of my more interesting demonstrations. See https://personality-project.org/r/simulating-personality.html and https://personality-project.org/r/r.datageneration.html for better demonstrations of data generation.

Value

a matrix of correlations and a graphic plot. The items above the diagonal are the tetrachoric correlations, below the diagonal are raw correlations.

Author(s)

William Revelle

References

https://personality-project.org/r/simulating-personality.html and https://personality-project.org/r/r.datageneration.html for better demonstrations of data generation.

See Also

VSS.simulate,item.sim

Examples

```
#demo <- phi.demo() #compare the phi (lower off diagonal and polychoric correlations
# (upper off diagonal)
#show the result from tetrachoric  which corrects for zero entries by default
#round(demo$tetrachoric$rho,2)
#show the result from phi2poly
#tetrachorics above the diagonal, phi below the diagonal
#round(demo$phis,2)
```

---

phi2tetra *Convert a phi coefficient to a tetrachoric correlation*

Description

Given a phi coefficient (a Pearson r calculated on two dichotomous variables), and the marginal frequencies (in percentages), what is the corresponding estimate of the tetrachoric correlation?

Given a two x two table of counts

a	b
c	d

The phi coefficient is (a - (a+b)*(a+c))/sqrt((a+b)(a+c)(b+d)(c+c)).

This function reproduces the cell entries for specified marginals and then calls the tetrachoric function. (Which was originally based upon John Fox's polychor function.) The phi2poly name will become deprecated in the future.

Usage

```
phi2tetra(ph,m,n=NULL,correct=TRUE)
phi2poly(ph,cp,cc,n=NULL,correct=TRUE) #deprecated
```

Arguments

ph	phi
m	a vector of the selection ratio and probability of criterion. In the case where ph is a matrix, m is a vector of the frequencies of the selected cases
correct	When finding tetrachoric correlations, should we correct for continuity for small marginals. See tetrachoric for a discussion.
n	If the marginals are given as frequencies, what was the total number of cases?
cp	probability of the predictor – the so called selection ratio
cc	probability of the criterion – the so called success rate.

Details

used to require the mvtnorm package but this has been replaced with mnormt

Value

a tetrachoric correlation

Author(s)

William Revelle

See Also

tetrachoric, Yule2phi.matrix, phi2poly.matrix

Examples

```
 phi2tetra(.3,c(.5,.5))
#phi2poly(.3,.3,.7)
```

Pinv — *Compute the Moore-Penrose Pseudo Inverse of a matrix and other tools*

Description

Given a matrix of less than full rank, the conventional inverse function will fail. The pseudoinverse or generalized inverse resolves this problem by using just the postive values of the singular value decomposition d matrix. This is an adaptation of the ginv function from MASS and the pinv function from pracma. Includes several other simple matrix tools such as matSqrt and invMatSqrt which can be used for “whitening" a matrix.

Usage

```
Pinv(R, tol = sqrt(.Machine$double.eps))
matSqrt(x) #take the sqrt of a matrix
invMatSqrt(x) #find the sqrt of the inverse matrix
```

Arguments

R	A correlation or covariance matrix to analyze
x	A correlation or covariance matrix to analyze
tol	A very small number. Reject values with eigen values less than tolerance

Details

The singular value decomposition of a matrix X is UdV where for full rank matrices, d is the vector of eigen values and U and V are the matrices of eigen vectors. The inverse is just U(1/d) U. If the matrix is less than full rank, many of the d values are effectively zero (at the limit of computational accuracy.) Thus, to solve matrix equations with matrices of less than full rank (e.g. the schmid Schmid-Leiman solution), we need to find the generalized inverse.

matSqrt and invMatSqrt will find the square root of a matrix or the square root of the Pinv (Penrose pseudo inverse) of a matrix.

invMatSqrt can be used to find the "whitening" matrix of a correlation or covariances such that for a correlation matrix, R, W= invMatSqrt(R):

$$W = R^{-.5}$$

and

$$WRW' = I.$$

Value

The generalized inverse, or the sqrt of the generalized inverse, or the sqrt of the Penrose inverse.

Note

Adapted from the ginv function in MASS and the pinv function in pracma. Installed here to avoid loading those packages.

Author(s)

William Revelle

References

Agnan Kessy, Alex Lewin and Korbinian Strimmer (2018) Optimal Whitening and Decorrelation, The American Statistician, 72:4, 309-314.

Marco Del Giudice (2024) Data matrix disattenuation: a simple, effective method for correcting measurement error in multivariate datasets. Preprint: PsyArXiv

Venables, W. N. and Ripley, B. D. (1999) Modern Applied Statistics with S-PLUS. Third Edition. Springer. p.100.Optimal

See Also

schmid, faCor

Examples

```
round(Pinv(Thurstone) %*% Thurstone,2)  #an identity matrix


W <- invMatSqrt(Thurstone) #find the whitening of Thurstone
round(W %*% Thurstone %*% W,2)
```

plot.psych *Plotting functions for the psych package of class "psych"*

Description

Combines several plotting functions into one for objects of class "psych". This can be used to plot the results of fa, irt.fa, VSS, ICLUST, omega, factor.pa, or principal.

Usage

```
## S3 method for class 'psych'
plot(x,labels=NULL,...)
## S3 method for class 'irt'
plot(x,xlab,ylab,main,D,type=c("ICC","IIC","test"),cut=.3,labels=NULL,
 keys=NULL, xlim,ylim,y2lab,lncol="black",...)
## S3 method for class 'poly'
plot(x,D,xlab,ylab,xlim,ylim,main,type=c("ICC","IIC","test"),cut=.3,labels,
     keys=NULL,y2lab,lncol="black",...)
## S3 method for class 'residuals'
plot(x,main,type=c("qq","chi","hist","cor"),std, bad=4,
     numbers=TRUE, upper=FALSE,diag=FALSE,...)
```

Arguments

x	The object to plot
labels	Variable labels
xlab	Label for the x axis – defaults to Latent Trait
ylab	Label for the y axis
xlim	The limits for the x axis
ylim	Specify the limits for the y axis
main	Main title for graph
type	"ICC" plots items, "IIC" plots item information, "test" plots test information, defaults to IIC.,"qq" does a quantile plot,"chi" plots chi square distributions,"hist" shows the histogram,"cor" does a corPlot of the residuals.
D	The discrimination parameter
cut	Only plot item responses with discrimiantion greater than cut
keys	Used in plotting irt results from irt.fa.
y2lab	ylab for test reliability, defaults to "reliability"
bad	label the most 1.. bad items in residuals
numbers	if using the cor option in plot residuals, show the numeric values
upper	if using the cor option in plot residuals, show the upper off diagonal values

diag	if using the cor option in plot residuals, show the diagonal values
std	Standardize the resduals?
lncol	The color of the lines in the IRT plots. Defaults to all being black, but it is possible to specify lncol as a vector of colors to be used.
...	other calls to plot

Details

Passes the appropriate values to plot. For plotting the results of irt.fa, there are three options: type = "IIC" (default) will plot the item characteristic respone function. type = "IIC" will plot the item information function, and type= "test" will plot the test information function.

Note that plotting an irt result will call either plot.irt or plot.poly depending upon the type of data that were used in the original irt.fa call.

These are calls to the generic plot function that are intercepted for objects of type "psych". More precise plotting control is available in the separate plot functions. plot may be used for psych objects returned from fa, irt.fa, ICLUST, omega, principal as well as plot.reliability .

A "jiggle" parameter is available in the fa.plot function (called from plot.psych when the type is a factor or cluster. If jiggle=TRUE, then the points are jittered slightly (controlled by amount) before plotting. This option is useful when plotting items with identical factor loadings (e.g., when comparing hypothetical models).

Objects from irt.fa are plotted according to "type" (Item informations, item characteristics, or test information). In addition, plots for selected items may be done if using the keys matrix. Plots of irt information return three invisible objects, a summary of information for each item at levels of the trait, the average area under the curve (the average information) for each item as well as where the item is most informative.

If plotting multiple factor solutions in plot.poly, then main can be a vector of names, one for each factor. The default is to give main + the factor number.

It is also possible to create irt like plots based upon just a scoring key and item difficulties, or from a factor analysis and item difficulties. These are not true IRT type analyses, in that the parameters are not estimated from the data, but are rather indications of item location and discrimination for arbitrary sets of items. To do this, find irt.stats.like and then plot the results.

plot.residuals allows the user to graphically examine the residuals of models formed by fa, irt.fa, omega, as well as principal and display them in a number of ways. "qq" will show quantiles of standardized or unstandardized residuals, "chi" will show quantiles of the squared standardized or unstandardized residuals plotted against the expected chi square values, "hist" will draw the histogram of the raw or standardized residuals, and "cor" will show a corPlot of the residual correlations.

Value

Graphic output for factor analysis, cluster analysis and item response analysis.

Note

More precise plotting control is available in the separate plot functions.

Author(s)

William Revelle

See Also

VSS.plot and fa.plot, cluster.plot, fa, irt.fa, VSS, ICLUST, omega, principal or plot.reliability

Examples

```
test.data <- Harman74.cor$cov
f4 <- fa(test.data,4)
plot(f4)
plot(resid(f4))
plot(resid(f4),main="Residuals from a 4 factor solution")
#not run
#data(bfi)
#e.irt <- irt.fa(bfi[11:15])  #just the extraversion items
#plot(e.irt)   #the information curves
#
ic <- iclust(test.data,3)   #shows hierarchical structure
plot(ic)                    #plots loadings
#
```

---

polar — *Convert Cartesian factor loadings into polar coordinates*

Description

Factor and cluster analysis output typically presents item by factor correlations (loadings). Tables of factor loadings are frequently sorted by the size of loadings. This style of presentation tends to make it difficult to notice the pattern of loadings on other, secondary, dimensions. By converting to polar coordinates, it is easier to see the pattern of the secondary loadings.

Usage

```
polar(f, sort = TRUE)
```

Arguments

f	A matrix of loadings or the output from a factor or cluster analysis program
sort	sort=TRUE: sort items by the angle of the items on the first pair of factors.

Details

Although many uses of factor analysis/cluster analysis assume a simple structure where items have one and only one large loading, some domains such as personality or affect items have a more complex structure and some items have high loadings on two factors. (These items are said to have complexity 2, see VSS). By expressing the factor loadings in polar coordinates, this structure is more readily perceived.

For each pair of factors, item loadings are converted to an angle with the first factor, and a vector length corresponding to the amount of variance in the item shared with the two factors.

For a two dimensional structure, this will lead to a column of angles and a column of vector lengths. For n factors, this leads to n* (n-1)/2 columns of angles and an equivalent number of vector lengths.

Value

polar	A data frame of polar coordinates

Author(s)

William Revelle

References

Rafaeli, E. & Revelle, W. (2006). A premature consensus: Are happiness and sadness truly opposite affects? Motivation and Emotion. \

Hofstee, W. K. B., de Raad, B., & Goldberg, L. R. (1992). Integration of the big five and circumplex approaches to trait structure. Journal of Personality and Social Psychology, 63, 146-163.

See Also

ICLUST, cluster.plot, circ.tests, fa

Examples

```
circ.data <- circ.sim(24,500)
circ.fa <- fa(circ.data,2)
circ.polar <- round(polar(circ.fa),2)
circ.polar
#compare to the graphic
cluster.plot(circ.fa)
```

polychor.matrix *Phi or Yule coefficient matrix to polychoric coefficient matrix*

Description

A set of deprecated functions that have replaced by Yule2tetra and Yule2phi.

Some older correlation matrices were reported as matrices of Phi or of Yule correlations. That is, correlations were found from the two by two table of counts:

a	b
c	d

Yule Q is (ad - bc)/(ad+bc).

With marginal frequencies of a+b, c+d, a+c, b+d.

Given a square matrix of such correlations, and the proportions for each variable that are in the a + b cells, it is possible to reconvert each correlation into a two by two table and then estimate the corresponding polychoric correlation (using John Fox's polychor function.

Usage

```
Yule2poly.matrix(x, v)  #deprectated
phi2poly.matrix(x, v)     #deprectated
Yule2phi.matrix(x, v)     #deprectated
```

Arguments

x	a matrix of phi or Yule coefficients
v	A vector of marginal frequencies

Details

These functions call Yule2poly, Yule2phi or phi2poly for each cell of the matrix. See those functions for more details. See phi.demo for an example.

Value

A matrix of correlations

Author(s)

William Revelle

Examples

```
#demo <- phi.demo()
#compare the phi (lower off diagonal and polychoric correlations (upper off diagonal)
#show the result from poly.mat
#round(demo$tetrachoric$rho,2)
#show the result from phi2poly
#tetrachorics above the diagonal, phi below the diagonal
#round(demo$phis,2)
```

`predict.psych` *Prediction function for factor analysis, principal components (pca), bestScales*

Description

Finds predicted factor/component scores from a factor analysis or principal components analysis (pca) of data set A predicted to data set B. Predicted factor scores use the weights matrix used to find estimated factor scores, predicted components use the loadings matrix. Scores are either standardized with respect to the prediction sample or based upon the original data. Predicted scores from a bestScales model are based upon the statistics from the original sample.

Usage

```
## S3 method for class 'psych'
predict(object, data,old.data,options=NULL,missing=FALSE,impute="none",...)
```

Arguments

`object`	the result of a factor analysis, principal components analysis (pca) or bestScales of data set A
`data`	Data set B, of the same number of variables as data set A.
`old.data`	if specified, the data set B will be standardized in terms of values from the old data. This is probably the preferred option. This is done automatically if object is from `bestScales`
`options`	scoring options for bestScales objects ("best.keys","weights","optimal.keys","optimal.weights")
`missing`	If missing=FALSE, cases with missing data are given NA scores, otherwise they are given the values based upon the wts x complete data
`impute`	Should missing cases be replaced by "means", "medians" or treated as missing ("none" is the default
`...`	More options to pass to predictions

Value

Predicted factor/components/criteria scores. If predicting from either fa or pca,the scores are based upon standardized items where the standardization is either that of the original data (old.data) or of the prediction set. This latter case can lead to confusion if just a small number of predicted scores are found.

If the object is from bestScales, unit weighted scales are found (by default) using the best.keys and the predicted scores are then put into the metric of the means and standard deviations of the derivation sample. Other scoring key options may be specified using the "options" parameter. Possible values are best.keys","weights","optimal.keys","optimal.weights". See bestScales for details.

By default, predicted scores are found by the matrix product of the standardized data with the factor or regression weights. If missing is TRUE, then the predicted scores are the mean of the standardized data x weights for those data points that are not NA.

Note

Thanks to Reinhold Hatzinger for the suggestion and request and to Sarah McDougald for the bestScales prediction.

Author(s)

William Revelle

See Also

fa, principal, bestScales

Examples

```
set.seed(42)
x <- sim.item(12,500)$item
f2 <- fa(x[1:250,],2,scores="regression")  # a two factor solution
p2 <- principal(x[1:250,],2,scores=TRUE)  # a two component solution
round(cor(f2$scores,p2$scores),2)  #correlate the components and factors from the A set
#find the predicted scores (The B set)
pf2 <- predict(f2,x[251:500,],x[1:250,])

  #use the original data for standardization values
pp2 <- predict(p2,x[251:500,],x[1:250,])
 #standardized based upon the first set
round(cor(pf2,pp2),2)   #find the correlations in the B set
#test how well these predicted scores match the factor scores from the second set
fp2 <- fa(x[251:500,],2,scores=TRUE)
round(cor(fp2$scores,pf2),2)

pf2.n <- predict(f2,x[251:500,])  #Standardized based upon the new data set
round(cor(fp2$scores,pf2.n))
   #predict factors of set two from factors of set 1, factor order is arbitrary


#note that the signs of the factors in the second set are arbitrary
```

```
#predictions from bestScales
#the derivation sample
bs <- bestScales(bfi[1:1400,], cs(gender,education,age),folds=10,p.keyed=.5)
pred <- predict(bs,bfi[1401:2800,]) #The prediction sample
cor2(pred,bfi[1401:2800,26:28] ) #the validity of the prediction
summary(bs) #compare with bestScales cross validations
```

---

predicted.validity *Find the predicted validities of a set of scales based on item statistics*

Description

The validity of a scale varies as a function of the number of items in the scale, their average intercorrelation, and their average validity. The asymptotic limit of a scales validity for any particular criterion is just the average validity divided by the square root of the average within scale item correlation. predicted.validity will find the predicted validity for a set of scales (defined by a keys.list) and the average item validity for various criteria.

The function will find (and report) scale reliabilities (using reliability) and average item validities (using item.validity)

Usage

```
predicted.validity(x, criteria, keys, scale.rel = NULL, item.val = NULL)
item.validity(x,criteria,keys)
validityItem(x,criteria,keys)
```

Arguments

x	A data set
criteria	Variables to predict from the scales
keys	A keys.list that defines the scales
scale.rel	If not specified, these will be found. Otherwise, this is the output from reliability.
item.val	If not specified, the average item validities for each scale will be found. Otherwise use the output from item.validity

Details

When predicting criteria from a set of items formed into scales, the validity of the scale (that is, the correlations of the scale with each criteria) is a function of the average item validity (r_y), the average intercorrelation of the items in the scale (r_x), and the number of items in the scale (n). The limit of validity is r_y/sqrt(r_x).

Criteria will differ in their predictability from a set of scales. These asymptotic values may be used to help the decision on which scales to develop further.

Value

`predicted` The predicted validities given the scales specified

`item.validities` The average item validities for each scale with each criterion

`scale.reliabilities` The various statistics reported by the `reliability` function

`asymptotic` A matrix of the asymptotic validities

Author(s)

William Revelle

References

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at https://personality-project.org/r/book/

Revelle, W. and Condon, D.M. (2019) Reliability from alpha to omega: A tutorial. Psychological Assessment, 31, 12, 1395-1411. https://doi.org/10.1037/pas0000754. https://osf.io/preprints/psyarxiv/2y3w9 Preprint available from PsyArxiv

See Also

`reliability`, `scoreItems`, `scoreFast`

Examples

```
pred.bfi <- predicted.validity(bfi[,1:25], bfi[,26:28], bfi.keys)
pred.bfi
```

---

principal — *Principal components analysis (PCA)*

Description

Does an eigen value decomposition and returns eigen values, loadings, and degree of fit for a specified number of components. Basically it is just doing a principal components analysis (PCA) for n principal components of either a correlation or covariance matrix. Can show the residual correlations as well. The quality of reduction in the squared correlations is reported by comparing residual correlations to original correlations. Unlike princomp, this returns a subset of just the best nfactors. The eigen vectors are rescaled by the sqrt of the eigen values to produce the component loadings more typical in factor analysis.

Usage

```
principal(r, nfactors = 1, residuals = FALSE,rotate="varimax",n.obs=NA, covar=FALSE,
 scores=TRUE,missing=FALSE,impute="median",oblique.scores=TRUE,method="regression",
 use ="pairwise",cor="cor",correct=.5,weight=NULL,n.rotations=1,
hyper=.15,...)
```

Arguments

r	a correlation matrix. If a raw data matrix is used, the correlations will be found using pairwise deletions for missing values.
nfactors	Number of components to extract
residuals	FALSE, do not show residuals, TRUE, report residuals
rotate	"none", "varimax", "quartimax", "promax", "oblimin", "simplimax", and "cluster" are possible rotations/transformations of the solution. See fa for all rotations avaiable.
n.obs	Number of observations used to find the correlation matrix if using a correlation matrix. Used for finding the goodness of fit statistics.
covar	If false, find the correlation matrix from the raw data or convert to a correlation matrix if given a square matrix as input.
scores	If TRUE, find component scores
missing	if scores are TRUE, and missing=TRUE, then impute missing values using either the median or the mean
impute	"median" or "mean" values are used to replace missing values
oblique.scores	If TRUE (default), then the component scores are based upon the structure matrix. If FALSE, upon the pattern matrix.
method	Which way of finding component scores should be used. The default is "regression"
weight	If not NULL, a vector of length n.obs that contains weights for each observation. The NULL case is equivalent to all cases being weighted 1.
use	How to treat missing data, use="pairwise" is the default". See cor for other options.
cor	How to find the correlations: "cor" is Pearson", "cov" is covariance, "tet" is tetrachoric, "poly" is polychoric, "mixed" uses mixedCor for a mixture of tetrachorics, polychorics, Pearsons, biserials, and polyserials, Yuleb is Yulebonett, Yuleq and YuleY are the obvious Yule coefficients as appropriate
correct	When doing tetrachoric, polycoric, or mixed cor, how should we treat empty cells. (See the discussion in the help for tetrachoric.)
n.rotations	Number of random restarts for rotations
hyper	Count number of (absolute) loadings less than hyper.
...	other parameters to pass to functions such as factor.scores or the various rotation functions.

Details

Useful for those cases where the correlation matrix is improper (perhaps because of SAPA techniques).

There are a number of data reduction techniques including principal components analysis (PCA) and factor analysis (EFA). Both PC and FA attempt to approximate a given correlation or covariance matrix of rank n with matrix of lower rank (p). ${}_nR_n \approx_n F_{kk}F'_n + U^2$ where k is much less than n. For principal components, the item uniqueness is assumed to be zero and all elements of

the correlation or covariance matrix are fitted. That is, ${}_nR_n \approx_n F_{kk}F'_n$ The primary empirical difference between a components versus a factor model is the treatment of the variances for each item. Philosophically, components are weighted composites of observed variables while in the factor model, variables are weighted composites of the factors. As the number of items increases, the difference between the two models gets smaller. Factor loadings are the asymptotic component loadings as the number of items gets larger.

For a n x n correlation matrix, the n principal components completely reproduce the correlation matrix. However, if just the first k principal components are extracted, this is the best k dimensional approximation of the matrix.

It is important to recognize that rotated principal components are not principal components (the axes associated with the eigen value decomposition) but are merely components. To point this out, unrotated principal components are labelled as PCi, while rotated PCs are now labeled as RCi (for rotated components) and obliquely transformed components as TCi (for transformed components). (Thanks to Ulrike Gromping for this suggestion.)

Rotations and transformations are either part of psych (Promax and cluster), of base R (varimax), or of GPArotation (simplimax, quartimax, oblimin, etc.).

Of the various rotation/transformation options, varimax, Varimax, quartimax, bentlerT, geominT, and bifactor do orthogonal rotations. Promax transforms obliquely with a target matix equal to the varimax solution. oblimin, quartimin, simplimax, bentlerQ, geominQ and biquartimin are oblique transformations. Most of these are just calls to the GPArotation package. The "cluster" option does a targeted rotation to a structure defined by the cluster representation of a varimax solution. With the optional "keys" parameter, the "target" option will rotate to a target supplied as a keys matrix. (See `target.rot`.)

The rotation matrix (rot.mat) is returned from all of these options. This is the inverse of the Th (theta?) object returned by the GPArotation package. The correlations of the factors may be found by $\Phi = \theta'\theta$

Some of the statistics reported are more appropriate for (maximum likelihood) factor analysis rather than principal components analysis, and are reported to allow comparisons with these other models.

Although for items, it is typical to find component scores by scoring the salient items (using, e.g., `scoreItems`) component scores are found by regression where the regression weights are $R^{-1}\lambda$ where λ is the matrix of component loadings. The regression approach is done to be parallel with the factor analysis function `fa`. The regression weights are found from the inverse of the correlation matrix times the component loadings. This has the result that the component scores are standard scores (mean=0, sd = 1) of the standardized input. A comparison to the scores from `princomp` shows this difference. princomp does not, by default, standardize the data matrix, nor are the components themselves standardized. The regression weights are found from the Structure matrix, not the Pattern matrix. If the scores are found with the covar option = TRUE, then the scores are not standardized but are just mean centered.

Jolliffe (2002) discusses why the interpretation of rotated components is complicated. Rencher (1992) discourages the use of rotated components. The approach used here is consistent with the factor analytic tradition. The correlations of the items with the component scores closely matches (as it should) the component loadings (as reported in the structure matrix).

The output from the print.psych function displays the component loadings (from the pattern matrix), the h2 (communalities) the u2 (the uniquenesses), com (the complexity of the component loadings for that variable (see below). In the case of an orthogonal solution, h2 is merely the row sum of the squared component loadings. But for an oblique solution, it is the row sum of

the (squared) orthogonal component loadings (remember, that rotations or transformations do not change the communality). This information is returned (invisibly) from the print function as the object Vaccounted.

Value

values	Eigen Values of all components – useful for a scree plot
rotation	which rotation was requested?
n.obs	number of observations specified or found
communality	Communality estimates for each item. These are merely the sum of squared factor loadings for that item.
complexity	Hoffman's index of complexity for each item. This is just $\frac{(\Sigma a_i^2)^2}{\Sigma a_i^4}$ where a_i is the factor loading on the ith factor. From Hofmann (1978), MBR. See also Pettersson and Turkheimer (2010).
loadings	A standard loading matrix of class "loadings"
fit	Fit of the model to the correlation matrix
fit.off	how well are the off diagonal elements reproduced?
residual	Residual matrix – if requested
dof	Degrees of Freedom for this model. This is the number of observed correlations minus the number of independent parameters (number of items * number of factors - nf*(nf-1)/2. That is, dof = niI * (ni-1)/2 - ni * nf + nf*(nf-1)/2.
objective	value of the function that is minimized by maximum likelihood procedures. This is reported for comparison purposes and as a way to estimate chi square goodness of fit. The objective function is $f = (trace((FF' + U2)^{-1}R) - log(\|(FF' + U2)^{-1}R\|) - n.items$. Because components do not minimize the off diagonal, this fit will be not as good as for factor analysis. It is included merely for comparison purposes.
STATISTIC	If the number of observations is specified or found, this is a chi square based upon the objective function, f. Using the formula from `factanal`: $\chi^2 = (n.obs - 1 - (2 * p + 5)/6 - (2 * factors)/3)) * f$
PVAL	If n.obs > 0, then what is the probability of observing a chisquare this large or larger?
Phi	If oblique rotations (using oblimin from the GPArotation package) are requested, what is the interfactor correlation.
scores	If scores=TRUE, then estimates of the factor scores are reported
weights	The beta weights to find the principal components from the data
R2	The multiple R square between the factors and factor score estimates, if they were to be found. (From Grice, 2001) For components, these are of course 1.0.
valid	The correlations of the component score estimates with the components, if they were to be found and unit weights were used. (So called course coding).
r.scores	The correlation of the component scores. (Since components are just weighted linear sums of the items, this is the same as Phi).
rot.mat	The rotation matrix used to produce the rotated component loadings.

Note

By default, the accuracy of the varimax rotation function seems to be less than the Varimax function. This can be enhanced by specifying eps=1e-14 in the call to principal if using varimax rotation. Furthermore, note that Varimax by default does not apply the Kaiser normalization, but varimax does. Gottfried Helms compared these two rotations with those produced by SPSS and found identical values if using the appropriate options. (See the last two examples.)

The ability to use different kinds of correlations was added in version 1.9.12.31 to be compatible with the options in fa.

The ability to do random restarts for rotations or transformations was added in 2.6.4. See the discussion in fa about why restarts are important.

Author(s)

William Revelle

References

Grice, James W. (2001), Computing and evaluating factor scores. Psychological Methods, 6, 430-450

Jolliffe, I. (2002) Principal Component Analysis (2nd ed). Springer.

Rencher, A. C. (1992) Interpretation of Canonical Discriminant Functions, Canonical Variates, and Principal Components, the American Statistician, (46) 217-225.

Revelle, W. An introduction to psychometric theory with applications in R (in prep) Springer. Draft chapters available at https://personality-project.org/r/book/

See Also

VSS (to test for the number of components or factors to extract), VSS.scree and fa.parallel to show a scree plot and compare it with random resamplings of the data), factor2cluster (for course coding keys), fa (for factor analysis), factor.congruence (to compare solutions), predict.psych to find factor/component scores for a new data set based upon the weights from an original data set.

Examples

```
#Four principal components of the Harman 24 variable problem
#compare to a four factor principal axes solution using factor.congruence
pc <- principal(Harman74.cor$cov,4,rotate="varimax")
mr <- fa(Harman74.cor$cov,4,rotate="varimax")  #minres factor analysis
pa <- fa(Harman74.cor$cov,4,rotate="varimax",fm="pa")  # principal axis factor analysis
round(factor.congruence(list(pc,mr,pa)),2)

pc2 <- principal(Harman.5,2,rotate="varimax")
pc2
round(cor(Harman.5,pc2$scores),2)  #compare these correlations to the loadings
#now do it for unstandardized scores, and transform obliquely
pc2o <- principal(Harman.5,2,rotate="promax",covar=TRUE)
pc2o
```

```
round(cov(Harman.5,pc2o$scores),2)
pc2o$Structure    #this matches the covariances with the scores
biplot(pc2,main="Biplot of the Harman.5 socio-economic variables",labels=paste0(1:12))

#For comparison with SPSS  (contributed by Gottfried Helms)
pc2v <- principal(iris[1:4],2,rotate="varimax",normalize=FALSE,eps=1e-14)
print(pc2v,digits=7)
pc2V <- principal(iris[1:4],2,rotate="Varimax",eps=1e-7)
p <- print(pc2V,digits=7)
round(p$Vaccounted,2)  # the amount of variance accounted for is returned as an object of print
```

`print.psych` *Print and summary functions for the psych class*

Description

Give limited output (print) or somewhat more detailed (summary) for most of the functions in psych.

Usage

```
## S3 method for class 'psych'
print(x,digits=2,all=FALSE,cut=NULL,sort=FALSE,short=TRUE,lower=TRUE,signif=NULL,...)

## S3 method for class 'psych'
summary(object,digits=2,items=FALSE,...)
```

Arguments

`x`	Output from a psych function (e.g., factor.pa, omega,ICLUST, score.items, cluster.cor
`object`	Output from a psych function
`items`	items=TRUE (default) does not print the item whole correlations
`digits`	Number of digits to use in printing
`all`	if all=TRUE, then the object is declassed and all output from the function is printed
`cut`	Cluster loadings < cut will not be printed. For the factor analysis functions (fa and factor.pa etc.), cut defaults to 0, for ICLUST to .3, for omega to .2.
`sort`	Cluster loadings are in sorted order
`short`	Controls how much to print
`lower`	For square matrices, just print the lower half of the matrix
`signif`	If not NULL, a numeric value, show just signif number of leading digits for describe output
`...`	More options to pass to summary and print

Details

Most of the psych functions produce too much output. print.psych and summary.psych use generic methods for printing just the highlights. To see what else is available, ask for the structure of the particular object: (str(theobject)).

Alternatively, to get complete output, unclass(theobject) and then print it. This may be done by using the all=TRUE option.

As an added feature, if the promax function is applied to a factanal loadings matrix, the normal output just provides the rotation matrix. print.psych will provide the factor correlations. (Following a suggestion by John Fox and Uli Keller to the R-help list). The alternative is to just use the Promax function directly on the factanal object.

Value

Various psych functions produce copious output. This is a way to summarize the most important parts of the output of the score.items, cluster.scores, and ICLUST functions. See those (`score.items`, `cluster.cor`, `cluster.loadings`, or `ICLUST`) for details on what is produced.

The signf option is available for the output from `describe` to adjust the number of digits shown for all columns. This is slightly different from what happens if you specify digits, which rounds all output to the number of digits. print(x,signif=3) will print just the 3 largest digits of x, which will frequently result in scientific notation for any column where that would be appropriate for at least one row.

Note

See `score.items`, `cluster.cor`, `cluster.loadings`, or `ICLUST`for details on what is printed.

Author(s)

William Revelle

Examples

```
data(bfi)
 keys.list <- list(agree=c(-1,2:5),conscientious=c(6:8,-9,-10),
 extraversion=c(-11,-12,13:15),neuroticism=c(16:20),openness = c(21,-22,23,24,-25))
 keys <- make.keys(25,keys.list,item.labels=colnames(bfi[1:25]))
 scores <- scoreItems(keys,bfi[1:25])
 scores
 summary(scores)
```

`Promax` *Perform Procustes,bifactor, promax or targeted rotations and return the inter factor angles.*

Description

The bifactor rotation implements the rotation introduced by Jennrich and Bentler (2011) by calling GPForth in the GPArotation package. promax is an oblique rotation function introduced by Hendrickson and White (1964) and implemented in the promax function in the stats package. Unfortunately, promax does not report the inter factor correlations. Promax does. TargetQ does a target rotation with elements that can be missing (NA), or numeric (e.g., 0, 1). It uses the GPArotation package. target.rot does general target rotations to an arbitrary target matrix. The default target rotation is for an independent cluster solution. equamax facilitates the call to GPArotation to do an equamax rotation. Equamax, although available as a specific option within GPArotation is easier to call by name if using equamax. The varimin rotation suggested by Ertl (2013) is implemented by appropriate calls to GPArotation.

Usage

```
faRotate(loadings,rotate="oblimin",...)
bifactor(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
biquartimin(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000)
TargetQ(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000,Target=NULL)
TargetT(L, Tmat=diag(ncol(L)), normalize=FALSE, eps=1e-5, maxit=1000,Target=NULL)
Promax(x,m=4, normalize=FALSE, pro.m = 4,Tmat=NULL,...)
Procrustes(L,Target) #adapted from Niels Waler
target.rot(x,keys=NULL)
varimin(L, Tmat = diag(ncol(L)), normalize = FALSE, eps = 1e-05, maxit = 1000)
vgQ.bimin(L)   #called by bifactor
vgQ.targetQ(L,Target=NULL)  #called by TargetQ
vgQ.varimin(L)  #called by varimin
equamax(L, Tmat=diag(ncol(L)), eps=1e-5, maxit=1000)
```

Arguments

`x`	A loadings matrix
`L`	A loadings matrix
`loadings`	A loadings matrix
`rotate`	Which rotation should be used?
`m`	the power to which to raise the varimax loadings (for Promax)
`pro.m`	the power to which to raise the various loadings in Promax.
`keys`	An arbitrary target matrix, can be composed of any weights, but probably -1,0, 1 weights. If missing, the target is the independent cluster structure determined by assigning every item to it's highest loaded factor.
`Target`	A matrix of values (mainly 0s, some 1s, some NAs) to which the matrix is transformed.
`Tmat`	An initial rotation matrix
`normalize`	parameter passed to optimization routine (GPForth in the GPArotation package and Promax)
`eps`	parameter passed to optimization routine (GPForth in the GPArotation package)

`maxit`	parameter passed to optimization routine (GPForth in the GPArotation package)
`...`	Other parameters to pass (e.g. to faRotate) include a Target list or matrix

Details

The two most useful of these functions is probably biquartimin which implements the oblique bifactor rotation introduced by Jennrich and Bentler (2011). The second is TargetQ which allows for missing NA values in the target. Next best is the orthogonal case, bifactor. None of these seem to be implemented in GPArotation (yet).

TargetT is an orthogonal target rotation function which allows for missing NA values in the target.

faRotate is merely a convenient way to call the various GPArotation functions as well as the additional ones added here. For multiple iterations of rotations see `faRotations`.

The difference between biquartimin and bifactor is just that the latter is the orthogonal case which is documented in Jennrich and Bentler (2011). It seems as if these two functions are sensitive to the starting values and random restarts (modifying T) might be called for.

bifactor output for the 24 cognitive variable of Holzinger matches that of Jennrich and Bentler as does output for the Chen et al. problem when fm="mle" is used and the Jennrich and Bentler solution is rescaled from covariances to correlations.

Promax is a very direct adaptation of the stats::promax function. The addition is that it will return the interfactor correlations as well as the loadings and rotation matrix.

varimin implements the varimin criterion proposed by Suitbert Ertl (2013). Rather than maximize the varimax criterion, it minimizes it. For a discussion of the benefits of this procedure, consult Ertel (2013).

In addition, these functions will take output from either the factanal, `fa` or earlier (`factor.pa`, `factor.minres` or `principal`) functions and select just the loadings matrix for analysis.

equamax is just a call to GPArotation's cFT function (for the Crawford Ferguson family of rotations.

TargetQ implements Michael Browne's algorithm and allows specification of NA values. The Target input is a list (see examples). It is interesting to note how powerful specifying what a factor isn't works in defining a factor. That is, by specifying the pattern of 0s and letting most other elements be NA, the factor structure is still clearly defined.

The target.rot function is an adaptation of a function of Michael Browne's to do rotations to arbitrary target matrices. Suggested by Pat Shrout.

The default for target.rot is to rotate to an independent cluster structure (every items is assigned to a group with its highest loading.)

target.rot will not handle targets that have linear dependencies (e.g., a pure bifactor model where there is a g loading and a group factor for all variables).

Value

`loadings`	Oblique factor loadings
`rotmat`	The rotation matrix applied to the original loadings to produce the promax solution or the targeted matrix
`Phi`	The interfactor correlation matrix

Note

Promax is direct adaptation of the stats:promax function following suggestions to the R-help list by Ulrich Keller and John Fox. Further modified to do targeted rotation similar to a function of Michael Browne.

varimin is a direct application of the GPArotation GPForth function modified to do varimin.

Note

The Target for TargetT can be a matrix, but for TargetQ must be a list. This seems to be a feature of GPArotation.

Author(s)

William Revelle

References

Ertel, S. (2013). Factor analysis: healing an ailing model. Universitatsverlag Gottingen.

Hendrickson, A. E. and White, P. O, 1964, British Journal of Statistical Psychology, 17, 65-70.

Jennrich, Robert and Bentler, Peter (2011) Exploratory Bi-Factor Analysis. Psychometrika, 1-13

See Also

promax, fa, or principal for examples of data analysis and Holzinger or Bechtoldt for examples of bifactor data. factor.rotate for 'hand rotation'.

Examples

```
jen <- sim.hierarchical()
f3 <- fa(jen,3,rotate="varimax")
f3   #not a very clean solution
Promax(f3)  #this obliquely rotates, but from the varimax target
target.rot(f3)  #this obliquely rotates to wards a simple structure target

#compare this rotation with the solution from a targeted rotation aimed for
#an independent cluster solution
#now try a bifactor solution
fb <-fa(jen,3,rotate="bifactor")
fq <- fa(jen,3,rotate="biquartimin")
#Suitbert Ertel has suggested varimin
fm <-  fa(jen,3,rotate="varimin") #the Ertel varimin
fn <- fa(jen,3,rotate="none")  #just the unrotated factors
#compare them
factor.congruence(list(f3,fb,fq,fm,fn))
# compare an oblimin with a target rotation using the Browne algorithm
 #note that we are changing the factor #order (this is for demonstration only)
 Targ <- make.keys(9,list(f1=1:3,f2=7:9,f3=4:6))
 Targ <- scrub(Targ,isvalue=1)  #fix the 0s, allow the NAs to be estimated
 Targ <- list(Targ)  #input must be a list
#show the target
```

```
 Targ
 fa(Thurstone,3,rotate="TargetQ",Target=Targ)  #targeted oblique rotation
#compare with oblimin
f3 <- fa(Thurstone,3)
#now try a targeted orthogonal rotation
Targ <- make.keys(9,list(f1=1:3,f2=7:9,f3=4:6))
faRotate(f3$loadings,rotate="TargetT",Target=list(Targ)) #orthogonal
```

---

psych.misc *Miscellaneous helper functions for the psych package*

Description

This is a set of minor, if not trivial, helper functions. lowerCor finds the correlation of x variables and then prints them using lowerMat which is a trivial, but useful, function to round off and print the lower triangle of a matrix. reflect reflects the output of a factor analysis or principal components analysis so that one or more factors is reflected. (Requested by Alexander Weiss.) progressBar prints out ... as a calling routine (e.g., tetrachoric) works through a tedious calculation. shannon finds the Shannon index (H) of diversity or of information. test.all tests all the examples in a package. best.items sorts a factor matrix for absolute values and displays the expanded items names. fa.lookup returns sorted factor analysis output with item labels. cor2 correlates two data.frames (of equal length). levels2numeric and char2numeric convert dataframe columns that are categorical/levels to numeric values.

Usage

```
psych.misc()
vech(R,lower=TRUE,diagonal=FALSE)
lowerCor(x,digits=2,use="pairwise",method="pearson",minlength=5,cor="cor",show=TRUE)
cor2(x,y,digits=2,use="pairwise",method="pearson",cor="cor",show=TRUE,pval=FALSE)
lowerMat(R, digits = 2,minlength=5)
matMult(x,y) #multiply two matrices with missing data

tableF(x,y)
reflect(f,flip=NULL)
progressBar(value,max,label=NULL)
shannon(x,correct=FALSE,base=2)
test.all(pl,package="psych",dependencies
      = c("Depends", "Imports", "LinkingTo"),find=FALSE,skip=NULL)
 levels2numeric(x)
 char2numeric(x,flag=TRUE)
 nchar2numeric(x,flag=TRUE)
 isCorrelation(x,na.rm=FALSE) #test if an object is a symmetric matrix
    # with diagonals of 1 and  all values between -1 and 1
 isCovariance(x)  #test if an object is a symmetric matrix
 fromTo(data,from,to=NULL) #convert character names to locations as specified in colnames
```

```
#of data
cs(...) #convert a list of text words to  character vector
acs(...) #convert a list of text words to  a single string
SAPAfy(x,y)  #sample y columns from x, replace other columns with NA
```

Arguments

R	A rectangular matrix or data frame (probably a correlation matrix)
x	A data matrix or data frame or a vector depending upon the function.
y	A data matrix or data frame or a vector
lower	if TRUE, show the lower off diagonal, else the upper off diagonal
diagonal	if TRUE, include the diagonal, don't include diagonal
f	The object returned from either a factor analysis (fa) or a principal components analysis (principal)
digits	round to digits
minlength	Abbreviate to minlength in lowerCor
show	Display the correlations from lowerCor or cor2 (default is TRUE)
pval	Calculate the probability values associated with the correlations.
use	Should pairwise deletion be done, or one of the other options to cor
na.rm	Should we check for NA on the diagonal of a correlation matices
method	"pearson", "kendall", "spearman"
cor	defaults to the normal cor function, but can also do tetrachoric, polychoric or covariances (cov)
value	the current value of some looping variable
max	The maximum value the loop will achieve
label	what function is looping
flip	The factor or components to be reversed keyed (by factor number)
flag	flag=TRUE in char2numeric will flag (with *) those variables that had been numeric. This changes the variable name. flag = FALSE does not mark those variables.
correct	Correct for the maximum possible information in this item
base	What is the base for the log function (default=2, e implies base = exp(1))
pl	The name of a package (or list of packages) to be activated and then have all the examples tested.
package	Find the dependencies for this package, e.g., psych
dependencies	Which type of dependency to examine?
find	Look up the dependencies, and then test all of their examples
skip	Do not test these dependencies
data	A dataframe or matrix to choose from
from	select from column with name from to column with name to
to	select from column from to column to
...	Any string of legitimate objects

Details

lowerCor prints out the lower off diagonal matrix rounded to digits with column names abbreviated to digits + 3 characters, but also returns the full and unrounded matrix. By default, it uses pairwise deletion of variables. It in turn calls

lowerMat which does the pretty printing.

It is important to remember to not call lowerCor when all you need is lowerMat!

cs is a direct copy of the Cs function in the Hmisc package by Frank Harrell. Added to psych to avoid the overhead of the Hmisc package.

Value

tableF is fast alternative to the table function for creating two way tables of numeric variables. It does not have any of the elegant checks of the table function and thus is much faster. Used in the tetrachoric and polychoric functions to maximize speed.

lowerCor Finds and prints (using lowerMat) the lower diagonal correlation matrix but returns (invisibly) the full correlation matrix found with the use, method and cor parameters. The default values are for pairwise deletion of variables, "pearson" correlations. Specify cor="tetrachoric", "polychoric", or "mixed" for alternatives. By default, digits is set to print to 2 decimal places. By default, it will change character variables to numeric and flag them.

lowerMatShows the lower triangle of a matrix, rounded to digits with titles abbreviated to digits + 3

progressBar Display a series of dots as we progress through a slow loop (removed from anything using multicores).

tableF (for tableFast) is a cut down version of table that does no error checking, nor returns pretty output, but is significantly faster than table. It will just work on two integer vectors. This is used in polychoric an tetrachoric for about a 50% speed improvement for large problems.

shannon finds Shannon's H index of information. Used for estimating the complexity or diversity of the distribution of responses in a vector or matrix.

$$H = -\sum p_i log(p_i)$$

test.all allows one to test all the examples in specified package. This allows us to make sure that those examples work when other packages (e.g., psych) are also loaded. This is used when developing revisions to the psych package to make sure the the other packages work. Some packages will not work and/or crash the system (e.g., DeducerPlugInScaling requires Java and even with Java, crashes when loaded, even if psych is not there!). Alternatively, if testing a long list of dependencies, you can skip the first part by specifying them by name.

cor2 will find and display the correlations between two sets of variables, rounded to digits, using the other options. If x is a list of multiple sets (two or more), then all sets are correlated. If pval is TRUE, will return the probability values (which can then be displayed by calling corPlot.)

levels2numericconverts character data with levels to numeric data. Used in the SAPA analyses where we code some variables, (e.g., gender, education) with character codes to help in the documentation of files, but want to do analyses of correlations with other categorical variables.

char2numericconverts character data with levels to numeric data. Used for cases when data from questionnaires include the response categories rathere than numeric data. Unless the levels of the

data are in meaningful order, the numeric results are not useful. Most useful if doing polychoric analyses. Note this is not suitable for recoding numeric data stored as characters, for it will force them to levels first. See nchar2numeric. Note that for very small data sets, because the recoding done by char2numeric is column wise, it might result in different numerical results for the same characters in different columns.

Problems with char2numeric can be solved by using the psychTools::recode function.

nchar2numericconverts numbers coded as characters (quoted) to numeric without forcing them to factors first.

fromTo selects the columns in data from to (see the examples)

cs concatenates strings without the need to identify variables by " ".

See Also

corr.test to find correlations, count the pairwise occurrences, and to give significance tests for each correlation. r.test for a number of tests of correlations, including tests of the difference between correlations. lowerUpper will display the differences between two matrices.

Examples

```
if(require(psychTools)) {

lowerMat(Thurstone)
lb <- lowerCor(bfi[1:10])  #finds and prints the lower correlation matrix,
  # returns the square matrix.
lowerCor(psychTools::ability[,1:5]) #the Pearson correlations, compare with
lowerCor(psychTools::ability[,1:5],cor="tetra")
#fiml <- corFiml(bfi[1:10])     #FIML correlations require lavaan package
#lowerMat(fiml)  #to get pretty output
f3 <- fa(Thurstone,3)
f3r <- reflect(f3,2)  #reflect the second factor
#find the complexity of the response patterns of the iqitems.
round(shannon(psychTools::iqitems),2)
#test.all('BinNor')  #Does the BinNor package work when we are using other packages
bestItems(lb,"A3",cut=.1,dictionary=bfi.dictionary[1:2])
#to make this a latex table
#df2latex(bestItems(lb,2,cut=.2))
}
data(bfi.dictionary)
f2 <- fa(bfi[1:10],2)
fa.lookup(f2,bfi.dictionary)

sa1 <-sat.act[1:2]
sa2 <- sat.act[3:4]
sa3 <- sat.act[5:6]
cor2(sa1,sa2)
cor2(list(sa1,sa2))  #show within set and between set cors
cor2(list(sa1,sa2,sa3))
lowerCor(fromTo(sat.act,"ACT","SATQ")) #show some correlations
vect <- cs(ACT,SATQ)  #skip the quotes
vect   #they are in this vector
```

```
#to combine longer terms
vect <- cs("Here is a longish",vector, that, we ,"want to combine", into, several)
vect
temp <- acs("Here is a longish",vector, that, we ,"want to combine", into, one)
temp
lowerCor(fromTo(sat.act,cs(ACT,SATQ)))
lowerCor(fromTo(bfi,cs(A3,C4)),cor="poly") #compare with
lowerCor(fromTo(bfi,cs(A3,C4)))


set.seed(42)
temp <- SAPAfy(bfi[1:10],3)  #30 percent sample   from bfi
f2 <- fa(bfi[1:10],2)
 f2s <- fa(temp,2,missing=TRUE)
 fa.congruence(f2s,f2)  #the two factor structure are almost identical
 #although the scores are not identical
cor2(f2$scores, f2s$scores)
```

`r.test` *Tests of significance for correlations*

Description

Tests the significance of a single correlation, the difference between two independent correlations, the difference between two dependent correlations sharing one variable (Williams's Test), or the difference between two dependent correlations with different variables (Steiger Tests).

Usage

```
r.test(n, r12, r34 = NULL, r23 = NULL, r13 = NULL, r14 = NULL, r24 = NULL,
          n2 = NULL,pooled=TRUE, twotailed = TRUE)
```

Arguments

`n`	Sample size of first group
`r12`	Correlation to be tested
`r34`	Test if this correlation is different from r12, if r23 is specified, but r13 is not, then r34 becomes r13
`r23`	if ra = r(12) and rb = r(13) then test for differences of dependent correlations given r23
`r13`	implies ra =r(12) and rb =r(34) test for difference of dependent correlations
`r14`	implies ra =r(12) and rb =r(34)
`r24`	ra =r(12) and rb =r(34)
`n2`	n2 is specified in the case of two independent correlations. n2 defaults to n if if not specified
`pooled`	use pooled estimates of correlations
`twotailed`	should a twotailed or one tailed test be used

Details

Depending upon the input, one of four different tests of correlations is done. 1) For a sample size n, find the t value for a single correlation where

$$t = \frac{r * \sqrt{(n-2)}}{\sqrt{(1-r^2)}}$$

and

$$se = \sqrt{\frac{1-r^2}{n-2}})$$

.

2) For sample sizes of n and n2 (n2 = n if not specified) find the z of the difference between the z transformed correlations divided by the standard error of the difference of two z scores:

$$z = \frac{z_1 - z_2}{\sqrt{\frac{1}{(n_1-3)+(n_2-3)}}}$$

.

3) For sample size n, and correlations r12, r13 and r23 test for the difference of two dependent correlations (r12 vs r13).

4) For sample size n, test for the difference between two dependent correlations involving different variables.

Consider the correlations from Steiger (1980), Table 1: Because these all from the same subjects, any tests must be of dependent correlations. For dependent correlations, it is necessary to specify at least 3 correlations (e.g., r12, r13, r23)

Variable	M1	F1	V1	M2	F2	V2
M1 1.00						
F1	.10	1.00				
V1	.40	.50	1.00			
M2	.70	.05	.50	1.00		
F2	.05	.70	.50	.50	1.00	
V2	.45	.50	.80	.50	.60	1.00

For clarity, correlations may be specified by value. If specified by location and if doing the test of dependent correlations, if three correlations are specified, they are assumed to be in the order r12, r13, r23.

Consider the examples from Steiger:

Case A: where Masculinity at time 1 (M1) correlates with Verbal Ability .5 (r12), femininity at time 1 (F1) correlates with Verbal ability r13 =.4, and M1 correlates with F1 (r23= .1). Then, given the correlations: r12 = .4, r13 = .5, and r23 = .1, t = -.89 for n =103, i.e., r.test(n=103, r12=.4, r13=.5,r23=.1)

Case B: Test whether correlation between two variables (e.g., F and V) is the same over time (e.g. F1V1 = F2V2)

r.test(n = 103, r12 = 0.5, r34 = 0.6, r23 = 0.5, r13 = 0.7, r14 = 0.5, r24 = 0.8)

Value

test	Label of test done
z	z value for tests 2 or 4
t	t value for tests 1 and 3
p	probability value of z or t

Note

Steiger specifically rejects using the Hotelling T test to test the difference between correlated correlations. Instead, he recommends Williams' test. (See also Dunn and Clark, 1971). These tests follow Steiger's advice. The test of two independent correlations is just a z test of the difference of the Fisher's z transformed correlations divided by the standard error of the difference. (See Cohen et al, p 49).

One of the beautiful features of R is what works on single value works on vectors and matrices. Thus, r.test can be used to test the pairwise diference of all the elements of a correlation matrix. See the last example.

By default, the probabilities are reported to 2 decimal places. This will, of course, sometimes lead to statements such as p < .1 when in fact p < .1001 or even more precisely p < .1000759. To achieve the higher precision, use a print statement with the preferred number of digits. See the next to last set of examples (courtesy of Julia Rohrer).

Author(s)

William Revelle

References

Cohen, J. and Cohen, P. and West, S.G. and Aiken, L.S. (2003) Applied multiple regression/correlation analysis for the behavioral sciences, L.Erlbaum Associates, Mahwah, N.J.

Olkin, I. and Finn, J. D. (1995). Correlations redux. Psychological Bulletin, 118(1):155-164.

Steiger, J.H. (1980), Tests for comparing elements of a correlation matrix, Psychological Bulletin, 87, 245-251.

Williams, E.J. (1959) Regression analysis. Wiley, New York, 1959.

See Also

See also corTest which tests all the elements of a correlation matrix, and cortest.mat to compare two matrices of correlations. r.test extends the tests in paired.r,r.con

Examples

```
n <- 30
r <- seq(0,.9,.1)
rc <- matrix(r.con(r,n),ncol=2)
test <- r.test(n,r)
r.rc <- data.frame(r=r,z=fisherz(r),lower=rc[,1],upper=rc[,2],t=test$t,p=test$p)
round(r.rc,2)
```

```
r.test(50,r)
r.test(30,.4,.6)       #test the difference between two independent correlations
r.test(103,.4,.5,.1)   #Steiger case A of dependent correlations
r.test(n=103, r12=.4, r13=.5,r23=.1)
#for complicated tests, it is probably better to specify correlations by name
r.test(n=103,r12=.5,r34=.6,r13=.7,r23=.5,r14=.5,r24=.8)   #steiger Case B


##By default, the precision of p values is 2 decimals
#Consider three different precisions shown by varying the requested number of digits
r12 =  0.693458895410494
r23 =  0.988475791500198
r13 =  0.695966022434845
print(r.test(n =  5105 ,  r12 =  r12 ,  r23 = r23 , r13 =  r13 )) #probability < 0.1
print(r.test(n =  5105 ,  r12 =  r12, r23 = r23 , r13 =  r13 ),digits=4) #p < 0.1001
print(r.test(n =  5105 , r12 =  r12, r23 = r23 , r13 =  r13 ),digits=8) #p< <0.1000759



#an example of how to compare the elements of two matrices
R1 <- lowerCor(bfi[1:200,1:5])  #find one set of Correlations
R2 <- lowerCor(bfi[201:400,1:5]) #and now another set sampled
#from the same population
test <- r.test(n=200, r12 = R1, r34 = R2)
round(lowerUpper(R1,R2,diff=TRUE),digits=2)  #show the differences between correlations
#lowerMat(test$p)  #show the p values of the difference between the two matrices
adjusted <- p.adjust(test$p[upper.tri(test$p)])
both <- test$p
both[upper.tri(both)] <- adjusted
round(both,digits=2)  #The lower off diagonal are the raw ps, the upper the adjusted ps
```

`rangeCorrection` *Correct correlations for restriction of range. (Thorndike Case 2)*

Description

In applied settings, it is typical to find a correlation between a predictor and some criterion. Unfortunately, if the predictor is used to choose the subjects, the range of the predictor is seriously reduced. This restricts the observed correlation to be less than would be observed in the full range of the predictor. A correction for this problem is well known as Thorndike Case 2:

Let R the unrestricted correlaton, r the restricted correlation, S the unrestricted standard deviation, s the restricted standard deviation, then

R = (rS/s)/ sqrt(1-r^2 + r^2(S^2/s^2)).

Several other cases of restriction were also considered by Thorndike and are implemented in `rangeCorrection`.

Usage

```
rangeCorrection(r,sdu,sdr,sdxu=NULL,sdxr=NULL,case=2)
```

Arguments

r	The observed correlation
sdu	The unrestricted standard deviation)
sdr	The restricted standard deviation
sdxu	Unrestricted standard deviation for case 4
sdxr	Restricted standard deviation for case 4
case	Which of the four Thurstone/Stauffer cases to use

Details

When participants in a study are selected on one variable, that will reduce the variance of that variable and the resulting correlation. Thorndike (1949) considered four cases of range restriction. Others have continued this discussion but have changed the case numbers.

Can be used to find correlations in a restricted sample as well as the unrestricted sample. Not the same as the correction to reliability for restriction of range.

Value

The corrected correlation.

Author(s)

William Revelle

References

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at https://personality-project.org/r/book/

Stauffer, Joseph and Mendoza, Jorge. (2001) The proper sequence for correcting correlation coefficients for range restriction and unreliability. Psychometrika, 66, 63-68.

See Also

cRRr in the psychometric package.

Examples

```
rangeCorrection(.33,100.32,48.19) #example from Revelle (in prep) Chapter 4.
```

reliability *Reports 7 different estimates of scale reliabity including alpha, omega, split half*

Description

Revelle and Condon, (2019) reviewed the problem of reliability in a tutorial meant to useful to the theoretician as well as the practitioner. Although there are a number of functions in psych for estimating reliability of single scales, (e.g. alpha and omega), for split half reliability splitHalf or for finding test-retest reliability testRetest or multilevel reliability mlr, the reliability function combines several of these functions to report these recommended measures for multiple scales.

To quote from Revelle and Condon (2019) "Reliability is a fundamental problem for measurement in all of science for '(a)ll measurement is befuddled by error' (p 294 McNemar, 1946). Perhaps because psychological measures are more befuddled than those of the other natural sciences, psychologists have long studied the problem of reliability.

"Issues of reliability are fundamental to understanding how correlations between observed variables are (attenuated) underestimates of the relationships between the underlying constructs, how observed estimates of a person's score are biased estimates of their latent score, and how to estimate the confidence intervals around any particular measurement. Understanding the many ways to estimate reliability as well as the ways to use these estimates allows one to better assess individuals and to evaluate selection and prediction techniques. This is not just a problem for measurement specialists but for all who want to make theoretical inferences from observed data.

" It is no longer acceptable to report one coefficient that is only correct if all items are exactly equally good measures of a construct. Researchers are encouraged to report at least two coefficients (e.g., omega_h and omega_t) and then discuss why each is appropriate for the inference that is being made. They are discouraged from reporting just alpha unless they can justify the assumptions implicit in using it (i.e., tau equivalence and unidimensionality)." Here we make it easy to do so.

Although the alpha and omega functions will find reliability estimates for a single scale, and scoreItems and scoreOverlap will find alpha for multiple scales, it sometimes is convenient to call omega and splitHalf for multiple scales. reliability takes a keys list (suitable for scoreItems) and then finds hierarchical and total omega as well as split half reliabilities for each separate scale. It also reports the unidim statistic from the unidim function.

plot.reliability takes the output of reliability and displays it as a dot chart showing the values of both omegas (ω_h and ω_t as well as α, u (the unidim statistic) and the distributions of split half reliabilities.

Usage

```
reliability(keys=NULL, items, nfactors = 2,n.obs=NA,
   split = TRUE,
   raw=TRUE,
    plot=FALSE,plot.sum=TRUE,
    hist=FALSE,
    n.sample=10000,
    brute=FALSE,
```

```
    check.keys=TRUE,
    covar=FALSE,
    fm="minres",
    rotate="oblimin",
    weighted="min")
## S3 method for class 'reliability'
plot(x,omega=TRUE,alpha=TRUE,split=TRUE,uni=TRUE,add=FALSE,
xlim=NULL, main=NULL,...)
```

Arguments

keys	A list of items to be scored (may be taken from a keys.list for scoreItems. This list may contain one or more keys.) If keys are not specified, then all the items are used.
items	The matrix or data.frame of items to be scored. Can be substantially greater than the items included in keys. For just those items in each key are scored.
nfactors	Omega is not well defined for two factors, but for small sets of items, two is the better choice. For larger number of items per scale, 3 is probably preferable.
n.obs	Number of observations, if given a correlation matrix. Needed for CFI
split	By default, find splitHalf reliabilities as well as the omega statistics. When plotting, split implies that raw was called in reliability.
plot	By default, suppress the omega plots for each scale.
plot.sum	Show the summary plot of the various estimates.
raw	If TRUE, return a list of all the possible splits (up to n.samples). Useful for graphic display.
hist	If TRUE then split and raw are forced to TRUE and the histograms of the split half values are drawn. (Otherwise, just return the values for later plotting)
n.sample	Normally defaults to 10,000. This means that for up to 16 item tests, all possible splits are found. choose(n,n/2)/2 explodes above that, eg. for all splits of the epi E scale requires 1,352,078 splits or 23.4 seconds on a MacBook Pro with a 2.4GHZ 8 core Intel Core I9. Can be done, but do you want to do so?
brute	Do all possible splits rather than sampling. (see splitHalf for details)
check.keys	By default, check that the keys are signe in the direction of the loadings on the first PCA.
covar	Should we use covariances rather than correlations?
fm	The factor extraction method to use. By default this is minres but could be "minrank" or "mle".
rotate	How should the factors be rotated. Defaults to oblimin, however cluster might be better.
weighted	The weight parameter in iclust. Defaults to "min"
x	The object returned from reliability
omega	Add in the values of omega_h and omega_t
uni	Show the unidimensionality value from unidim.

alpha	Add the value of alpha
add	Allows us to merge this figure with other ones
main	Defaults to "Split half distributions + omega, alpha"
xlim	The xlim of the plot
...	Other graphical parameters

Details

reliability is basically just a wrapper for omegah, unidim and splitHalf. Revelle and Condon (2019) recommended reporting at least three reliability statistics for any scale, here we make it easy to do.

If the hist option is set to true, histgrams and density plots of the split half values for each test are also shown. The output from reliability can be passed to error.dots to show the reliability statistics for multiple scales graphically. It is however more useful to just call the plot.reliability function to show the basic information.

For detailed analysis of any one scale, it is recommended to do a complete omega analysis, perhaps combined with a splitHalf analysis. The reliability function is just meant for the case where the user has multiple scales (perhaps scored using scoreItems) and then wants to get more complete reliability information for all of the scales.

Following a suggestion, the ability to not bother with keys and just do omega and split half and draw the results has been added. Either specify that keys=NULL, or just specify the items to use. (See the first example.)

plot.reliability provides a dot chart summary of the distributions of the split half values, as well as the estimates of omega and alpha and unidimensionality. It can also be called by just issuing a plot command.

Value

omega_h	Omega_h is the (model based) hierarchical estimate of the general factor saturation of a scale.
alpha	The conventional alpha statistic (which is not model based)
omega.tot	A model based estimate of the total reliability of a scale
Uni	An experimental estimate of unidimensionality (from unidim)
cong	How well does the average r of the correlations reproduce the matrix? A measure of congeneric fit
tau	How well does a single factor reproduce the correlation matrix. A measure of tau equivalance.
max.split	The greatest split half reliability of the scale. Found by finding all possible splits (if this is < 15,000) or sampled from 15,000 possible splits.
min.split	The lowest split half reliability of the scale. An estimate of beta (see iclust).
mean.r	The average correlation of the items in the scale
med.r	The median correlation of the items. If this differs from the mean, that is a sign of poor scale.

splits	A list of the split half values for all possible splits.
Beta	The beta statistic from ICLUST – forced to a one cluster solution.
CFI	The comparative fit index
EVR	The ratio of the first to second factor eigen value
MAP	The Minimum Average partial

Note

For much more information on reliability, see the help pages for omega, iclust and unidim, as well as the Revelle and Condon (2019) tutorial or the Revelle (in prep) chapter on reliability.

For some rare cases in some simulations, check.keys=FALSE gets a better minimum split half.

Author(s)

William Revelle

References

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at https://personality-project.org/r/book/

Revelle, W. and Condon, D.M. (2019) Reliability from alpha to omega: A tutorial. Psychological Assessment, 31, 12, 1395-1411. doi:10.1037/pas0000754. https://osf.io/preprints/psyarxiv/2y3w9 Preprint available from PsyArxiv

Revelle, W. and Condon, D.M (2025) Using undim rather than omega in estimating undimensionality. Psychological Methods (Advanced online publication) doi:10.1037/met0000729 (Open source)

See Also

omega to find more complete output for the various omega analyses, splitHalf to show more detail on split half estimates, scoreItems to find scores on multiple scales using unit weights, testRetest to find test retest reliabilities, mlr to find multilevel reliabilities.

predicted.validity will call reliability and item.validity to use the average r information to find the asymptotic validity of a set of scales for a set of criteria.

Examples

```
if(require(psychTools)) {
reliability(psychTools::ability) #an example of finding reliability for all items
rel <- reliability(psychTools::ability.keys,psychTools::ability)  #use keys to select scales
R <- cor(psychTools::ability,use="pairwise") #find the correlations to test
rel.R <- reliability(psychTools::ability.keys,R) #this should be the same as rel
plot(rel.R) #versus all and subsets
all.equal(rel.R$result.df,rel.R$result.df ) #should be TRUE
}
reliability(bfi.keys,bfi)  #reliability when some items are keyed negatively

## Not run:
#this takes a few seconds but shows nice graphic displays
```

```
if(require(psychTools)) {
spi.rel <- reliability(psychTools::spi.keys,psychTools::spi,hist=TRUE) #graph them
spi.rel #show them

#plot them using plot.reliability
plot(spi.rel)  #draw the density distrbutions

plot(spi.rel,split=FALSE) #don't draw the split half density distribution
plot(spi.rel,omega=FALSE) # don't add omega values to the diagram
#or do this without the densities

#plot the first three values in a dot chart
error.dots(spi.rel$result.df[,1],sort=FALSE, xlim=c(.3,1),head=16,tail=16,
          main = expression(paste(omega[h], ~~~~ alpha,~~~~ omega[t])))
  #plot the omega_h values
error.dots(spi.rel$result.df[,2],sort=FALSE,pch=2,xlim=c(.3,1),head=16,tail=16,
       main="",labels="",add=TRUE)#add the alpha values
error.dots(spi.rel$result.df[,3],sort=FALSE, xlim=c(.3,1),head=16,tail=16,
       pch=3,labels="", main="",add=TRUE) #and the omega_t values

#or, show the smallest and greatest split half, as well as alpha
error.dots(spi.rel$result.df[,4],sort=FALSE, xlim=c(.3,1),head=16,tail=16,
          main = expression(paste(beta, ~~~~ alpha,~~~~  glb)))
error.dots(spi.rel$result.df[,5],sort=FALSE,pch=5,xlim=c(.3,1),head=16,tail=16,
       main="",labels="",add=TRUE)#add the GLB values
error.dots(spi.rel$result.df[,2],sort=FALSE,pch=2,xlim=c(.3,1),head=16,tail=16,
       main="",labels="",add=TRUE)#add the alpha values

}

## End(Not run)
```

`rescale` *Function to convert scores to "conventional " metrics*

Description

Psychologists frequently report data in terms of transformed scales such as "IQ" (mean=100, sd=15, "SAT/GRE" (mean=500, sd=100), "ACT" (mean=18, sd=6), "T-scores" (mean=50, sd=10), or "Stanines" (mean=5, sd=2). The `rescale` function converts the data to standard scores and then rescales to the specified mean(s) and standard deviation(s).

Usage

```
rescale(x, mean = 100, sd = 15,df=TRUE)
```

Arguments

`x`	A matrix or data frame
`mean`	Desired mean of the rescaled scores- may be a vector

sd	Desired standard deviation of the rescaled scores
df	if TRUE, returns a data frame, otherwise a matrix

Value

A data.frame (default) or matrix of rescaled scores.

Author(s)

William Revelle

See Also

See Also scale

Examples

```
T <- rescale(attitude,50,10) #all put on same scale
describe(T)
T1 <- rescale(attitude,seq(0,300,50),seq(10,70,10)) #different means and sigmas
describe(T1)
```

---

residuals.psych *Extract residuals from various psych objects*

Description

Residuals in the various psych functions are extracted and then may be "pretty" printed.

Usage

```
## S3 method for class 'psych'
residuals(object,diag=TRUE,...)
## S3 method for class 'psych'
resid(object,diag=TRUE,...)
```

Arguments

object	The object returned by a psych function.
diag	if FALSE, then convert the diagonal of the residuals to NA
...	Other parameters to be passed to residual (ignored but required by the generic function)

Details

Currently implemented for fa, principal, omega, irt.fa, and fa.extension.

Value

residuals: a matrix of residual estimates

Author(s)

William Revelle

Examples

```
f3 <- fa(Thurstone,3)
residuals(f3)
sum(residuals(f3)^2) #include diagonal
sum(residuals(f3,diag=FALSE)^2,na.rm=TRUE) #drop diagonal
```

---

reverse.code *Reverse the coding of selected items prior to scale analysis*

Description

Some IRT functions require all items to be coded in the same direction. Some data sets have items that need to be reverse coded (e.g., 6 -> 1, 1 -> 6). reverse.code will flip items based upon a keys vector of 1s and -1s. Reversed items are subtracted from the item max + item min. These may be specified or may be calculated.

Usage

```
reverse.code(keys, items, mini = NULL, maxi = NULL)
```

Arguments

keys	A vector of 1s and -1s. -1 implies reverse the item
items	A data set of items
mini	if NULL, the empirical minimum for each item. Otherwise, a vector of minima
maxi	f NULL, the empirical maximum for each item. Otherwise, a vector of maxima

Details

Not a very complicated function, but useful in the case that items need to be reversed prior to using IRT functions from the ltm or eRM packages. Most psych functions do not require reversing prior to analysis, but will do so within the function.

Value

The corrected items.

Examples

```
original <- matrix(sample(6,50,replace=TRUE),10,5)
keys <- c(1,1,-1,-1,1)  #reverse the 3rd and 4th items
new <- reverse.code(keys,original,mini=rep(1,5),maxi=rep(6,5))
original[1:3,]
new[1:3,]
```

RMSEA *Root Mean Squared Error of Approximation from chisq, df, and n*

Description

Find the RMSEA from model chi square, degrees of freedom and number of observations. Show confidence intervals.

Usage

```
RMSEA(chisq, dof, n.obs, alpha = 0.1)
```

Arguments

chisq	The Chi square statistic from an analysis
dof	Degrees of freedom of the model
n.obs	Number of observations
alpha	alpha level for confidence intervals

Details

RMSEA is just a chisquare adjusted by its degrees of freedom and the sample size. $\sqrt{chisq/(dof*(nobs))-1/(n.obs-1)}$. It is given in most of the appropriate functions (e.g.,fa) and given here for completeness.

Value

RMSEA	the estimated value
RMSEA-L	the lower bound
RMSEA-U	the upper bound

Author(s)

William Revelle

References

Steiger, J. H.,and Lind, J. C. (1980). Statistically based tests for the number of common factors. Paper presented at the Annual Meeting of the Psychometric Society, Iowa City, IA.

See Also

fa, omega , esem

RV *Three measures of the correlations between sets of variables*

Description

How to measure the the correlation between two clusters or groups of variables x and y from the same data set is a recurring problem. Perhaps the most obvious is simply the unweighted correlation Ru.

Consider the matrix M composed of four submatrices

$$\begin{array}{rcc} & \text{Rx} & \text{Rxy} \\ \text{M} = & \text{Rxy} & \text{Ry} \end{array}$$

The unit weighted correlation, Ru is merely

$Ru = \frac{\Sigma r_{xy}}{\sqrt{\Sigma r_x \Sigma r_y}}$

A second is the Set correlation (also found in lmCor) by Cohen 1982) which is

$Rset = 1 - \frac{det(m)}{det(x)*det(y)}$

Where m is the full matrix (x+y)by (x+y). and det represents the determinant.

A third approach (the RV coeffiecent) was introduced by Escoufier (1970) and Robert and Escoufier (1976).

$RV = \frac{tr(xy(xy)')}{\sqrt{(tr(xx')*tr(yy'))}}$.

Where tr is the trace operator. (The sum of the diagonals).

The analysis can be done from the raw data or from correlation or covariance matrices. From the raw data, just specify the x and y variables. If using correlation/covariance matrixes, the xy matrix must be specified as well.

If using raw data, just specify the x and y columns and the data file.

Usage

```
RV(x, y, xy = NULL, data=NULL, cor = "cor",correct=0)
```

Arguments

x	Columns of the data matrix of n rows and p columns, (if data is specified) or a p x p correlation matrix.
y	Columns of a raw data matrix of n rows and q columns, or a q * q correlation matrix.

xy	A p x q correlation or covariance matrix, if not using the raw data.
data	A matrix or data frame containing the raw data.
cor	If xy is NULL, find the p x p correlations or covariances from x, and the q x q correlations from y as well as the p x q covariance/correlation matrix.. Options are "cor" (for Pearson), "spearman" , "cov" for covariances, "tet" for tetrachoric, or "poly" for polychoric correlation.
correct	The correction for continuity if desired.

Details

If using raw data, just specify the columns in x and y. If using a correlation matrix or covariance matrix, the inter corrlations/covariances) are specified in xy. The results match those of the RV function from matrixCalculations and the coeffRV function from factoMineR.

Value

RV	The RV statistic
Rset	Cohen's set correlation
Ru	The unit weighted correlation between x and y.
Rx	The correlation matrix of the x variables.
Ry	The correlation matrix of the y variables.
Rxy	The intercorrelations of x and y.

Author(s)

William Revelle

References

P. Robert and Y. Escoufier, 1976, A Unifying Tool for Linear Multivariate Statistical Methods: The RV- Coefficient. Journal of the Royal Statistical Society. Series C (Applied Statistics), Volume 25, pp. 257-265.

J. Cohen (1982) Set correlation as a general multivariate data-analytic method. Multivariate Behavioral Research, 17(3):301-341.

See Also

lmCor for unit weighted correlations.

Examples

```
#from raw data
RV (attitude[1:3],attitude[4:7])  #find the correlations
RV (attitude[1:3],attitude[4:7],cor="cov")

#find the correlations
R <- cor(attitude)
r1 <- R[1:3,1:3]
```

```
r2 <- R[4:7,4:7]
r12 <- R[1:3,4:7]
RV(r1,r2,r12)

#or find the covariances
C <- cov(attitude)
c1 <- C[1:3,1:3]
c2 <- C[4:7,4:7]
c12 <- C[1:3,4:7]
RV(c1, c2, c12)
```

`sat.act` *3 Measures of ability: SATV, SATQ, ACT*

Description

Self reported scores on the SAT Verbal, SAT Quantitative and ACT were collected as part of the Synthetic Aperture Personality Assessment (SAPA) web based personality assessment project. Age, gender, and education are also reported. The data from 700 subjects are included here as a demonstration set for correlation and analysis.

Usage

```
data(sat.act)
```

Format

A data frame with 700 observations on the following 6 variables.

`gender` males = 1, females = 2

`education` self reported education 1 = high school ... 5 = graduate work

`age` age

`ACT` ACT composite scores may range from 1 - 36. National norms have a mean of 20.

`SATV` SAT Verbal scores may range from 200 - 800.

`SATQ` SAT Quantitative scores may range from 200 - 800

Details

These items were collected as part of the SAPA project (https://www.sapa-project.org/) to develop online measures of ability (Revelle, Wilt and Rosenthal, 2009). The score means are higher than national norms suggesting both self selection for people taking on line personality and ability tests and a self reporting bias in scores.

See also the iq.items data set.

Source

https://personality-project.org/

References

Revelle, William, Wilt, Joshua, and Rosenthal, Allen (2009) Personality and Cognition: The Personality-Cognition Link. In Gruszka, Alexandra and Matthews, Gerald and Szymura, Blazej (Eds.) Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control, Springer.

Examples

```
data(sat.act)
describe(sat.act)
pairs.panels(sat.act)
```

`scaling.fits` *Test the adequacy of simple choice, logistic, or Thurstonian scaling.*

Description

Given a matrix of choices and a vector of scale values, how well do the scale values capture the choices? That is, what is size of the squared residuals given the model versus the size of the squared choice values?

Usage

```
scaling.fits(model, data, test = "logit", digits = 2, rowwise = TRUE)
```

Arguments

`model`	A vector of scale values
`data`	A matrix or dataframe of choice frequencies
`test`	"choice", "logistic", "normal"
`digits`	Precision of answer
`rowwise`	Are the choices ordered by column over row (TRUE) or row over column False)

Details

How well does a model fit the data is the classic problem of all of statistics. One fit statistic for scaling is the just the size of the residual matrix compared to the original estimates.

Value

`GF`	Goodness of fit of the model
`original`	Sum of squares for original data
`resid`	Sum of squares for residuals given the data and the model
`residual`	Residual matrix

Note

Mainly for demonstration purposes for a course on psychometrics

Author(s)

William Revelle

References

Revelle, W. (in preparation) Introduction to psychometric theory with applications in R, Springer. https://personality-project.org/r/book/

See Also

thurstone, vegetables

---

scatterHist *Draw a scatter plot with associated X and Y histograms, densities and correlation*

Description

Draw a X Y scatter plot with associated X and Y histograms with estimated densities. Will also draw density plots by groups, as well as distribution ellipses by group. Partly a demonstration of the use of layout. Also includes lowess smooth or linear model slope, as well as correlation and Mahalanobis distances

Usage

```
scatterHist(x,y=NULL,smooth=TRUE,ab=FALSE, correl=TRUE,data=NULL, density=TRUE,means=TRUE,
   ellipse=TRUE,digits=2,method="pearson",cex.cor=1,cex.point=1,
   title="Scatter plot + density",
  xlab=NULL,ylab=NULL,smoother=FALSE,nrpoints=0,xlab.hist=NULL,ylab.hist=NULL,grid=FALSE,
   xlim=NULL,ylim=NULL,x.breaks=11,y.breaks=11,
   x.space=0,y.space=0,freq=TRUE,x.axes=TRUE,y.axes=TRUE,size=c(1,2),
   col=c("blue","red","black"),legend=NULL,transparency=.5,pch=21, show.d=TRUE,
   x.arrow=NULL,y.arrow=NULL,d.arrow=FALSE,cex.arrow=1,
      line.col="blue",alpha = .05,ci=TRUE, ci.col="light blue",...)

scatter.hist(x,y=NULL,smooth=TRUE,ab=FALSE, correl=TRUE,data=NULL,density=TRUE,
  means=TRUE, ellipse=TRUE,digits=2,method="pearson",cex.cor=1,cex.point=1,
  title="Scatter plot + density",
 xlab=NULL,ylab=NULL,smoother=FALSE,nrpoints=0,xlab.hist=NULL,ylab.hist=NULL,grid=FALSE,
  xlim=NULL,ylim=NULL,x.breaks=11,y.breaks=11,
  x.space=0,y.space=0,freq=TRUE,x.axes=TRUE,y.axes=TRUE,size=c(1,2),
  col=c("blue","red","black"),legend=NULL,transparency=.5,pch=21, show.d=TRUE,
   x.arrow=NULL,y.arrow=NULL,d.arrow=FALSE,cex.arrow=1,
      line.col="blue",alpha = .05,ci=TRUE, ci.col="light blue",...)
```

Arguments

x	The X vector, or the first column of a data.frame or matrix. Can be specified using formula input.
y	The Y vector, of if X is a data.frame or matrix, the second column of X
smooth	if TRUE, then add a loess smooth to the plot
ab	if TRUE, then show the best fitting linear fit
correl	TRUE: Show the correlation
data	if using formula input, the data must be specified
density	TRUE: Show the estimated densities
means	TRUE: If TRUE, show the means for the distributions.
ellipse	TRUE: draw 1 and 2 sigma ellipses and smooth
digits	How many digits to use if showing the correlation
method	Which method to use for correlation ("pearson","spearman","kendall") defaults to "pearson"
smoother	if TRUE, use smoothScatter instead of plot. Nice for large N.
nrpoints	If using smoothScatter, show nrpoints as dots. Defaults to 0
grid	If TRUE, show a grid for the scatter plot.
cex.cor	Adjustment for the size of the correlation
cex.point	Adjustment for the size of the data points
xlab	Label for the x axis
ylab	Label for the y axis
xlim	Allow specification for limits of x axis, although this seems to just work for the scatter plots.
ylim	Allow specification for limits of y axis
x.breaks	Number of breaks to suggest to the x axis histogram.
y.breaks	Number of breaks to suggest to the y axis histogram.
x.space	space between bars
y.space	Space between y bars
freq	Show frequency counts, otherwise show density counts
x.axes	Show the x axis for the x histogram
y.axes	Show the y axis for the y histogram
size	The sizes of the ellipses (in sd units). Defaults to 1,2
col	Colors to use when showing groups
transparency	Amount of transparency in the density plots
legend	Where to put a legend c("topleft","topright","top","left","right")
pch	Base plot character (each group is one more)
xlab.hist	Not currently available
ylab.hist	Label for y axis histogram. Not currently available.

title	An optional title
show.d	If TRUE, show the distances between the groups
d.arrow	If TRUE, draw an arrow between the two centroids
x.arrow	optional lable for the arrow connecting the two groups for the x axis
y.arrow	optional lable for the arrow connecting the two groups for the y axis
cex.arrow	cex control for the label size of the arrows.
line.col	color of the lowess or lm fit line
alpha	When drawing confidence intervals, what is the margin of error
ci	Draw confidence intervals
ci.col	color of confidence intervals
...	Other parameters for graphics

Details

Just a straightforward application of layout and barplot, with some tricks taken from pairs.panels. The various options allow for correlation ellipses (1 and 2 sigma from the mean), lowess smooths, linear fits, density curves on the histograms, and the value of the correlation. ellipse = TRUE implies smooth = TRUE. The grid option provides a background grid to the scatterplot.

If using grouping variables, will draw ellipses (defaults to 1 sd) around each centroid. This is useful when demonstrating Mahalanobis distances.

Formula input allows specification of grouping variables as well.)

For plotting data for two groups, Mahalobnis differences between the groups may be shown by drawing an arrow between the two centroids. This is a bit messy and it is useful to use pch="." in this case.

Note

Originally adapted from Addicted to R example 78. Modified following some nice suggestions from Jared Smith. Substantial revisions in 2021 to allow for a clearer demonstration of group differences.

Author(s)

William Revelle

See Also

pairs.panels for multiple plots, multi.hist for multiple histograms and histBy for single variables with multiple groups. Perhaps the best example is found in the psychTools::GERAS data set.

Examples

```
data(sat.act)
with(sat.act,scatterHist(SATV,SATQ))
scatterHist(SATV ~ SATQ,data=sat.act)  #formula input

#or for something a bit more splashy
scatterHist(sat.act[5:6],pch=(19+sat.act$gender),col=c("blue","red")[sat.act$gender],grid=TRUE)
#better yet
scatterHist(SATV ~ SATQ + gender,data=sat.act) #formula input with a grouping variable

#If using a factor for grouping, we must first convert it to numeric
iris.1 <- char2numeric(iris,flag=FALSE)
scatterHist(Sepal.Width ~ Petal.Length + Species,data=iris.1, show.d=FALSE,
   main="Fisher's Iris example") #see pairs.panels
```

---

schmid *Apply the Schmid Leiman transformation to a correlation matrix*

Description

One way to find omega is to do a factor analysis of the original data set, rotate the factors obliquely, do a Schmid Leiman transformation, and then find omega. Here is the code for Schmid Leiman. The S-L transform takes a factor or PC solution, transforms it to an oblique solution, factors the oblique solution to find a higher order (g) factor, and then residualizes g out of the the group factors.

Usage

```
schmid(model, nfactors = 3, fm = "minres",digits=2,rotate="oblimin",
            n.obs=NA,option="equal",Phi=NULL,covar=FALSE,two.ok=FALSE,...)
```

Arguments

model	A correlation matrix
nfactors	Number of factors to extract
fm	the default is to do minres. fm="pa" for principal axes, fm="pc" for principal components, fm = "minres" for minimum residual (OLS), pc="ml" for maximum likelihood
digits	if digits not equal NULL, rounds to digits
rotate	The default, oblimin, produces somewhat more correlated factors than the alternative, simplimax. Other options include Promax (not Kaiser normalized) or promax (Promax with Kaiser normalization). See fa for possible oblique rotations.
n.obs	Number of observations, used to find fit statistics if specified. Will be calculated if input is raw data
option	When asking for just two group factors, option can be for "equal", "first" or "second"

`Phi`	If Phi is specified, then the analysis is done on a pattern matrix with the associated factor intercorrelation (Phi) matrix. This allows for reanalysess of published results
`covar`	Defaults to FALSE and finds correlations. If set to TRUE, then do the calculations on the unstandardized variables.
`two.ok`	If TRUE, do not give a warning about three factors being required.
`...`	Allows additional parameters to be passed to the factoring routines

Details

Schmid Leiman orthogonalizations are typical in the ability domain, but are not seen as often in the non-cognitive personality domain. S-L is one way of finding the loadings of items on the general factor for estimating omega.

A typical example would be in the study of anxiety and depression. A general neuroticism factor (g) accounts for much of the variance, but smaller group factors of tense anxiety, panic disorder, depression, etc. also need to be considerd.

An alternative model is to consider hierarchical cluster analysis techniques such as `ICLUST`.

Requires the GPArotation package.

Although 3 factors are the minimum number necessary to define the solution uniquely, it is occasionally useful to allow for a two factor solution. There are three possible options for this condition: setting the general factor loadings between the two lower order factors to be "equal" which will be the sqrt(oblique correlations between the factors) or to "first" or "second" in which case the general factor is equated with either the first or second group factor. A message is issued suggesting that the model is not really well defined.

A diagnostic tool for testing the appropriateness of a hierarchical model is p2 which is the percent of the common variance for each variable that is general factor variance. In general, p2 should not have much variance.

An alternative approach is to use the `directSl` function to do a direct Schmid Leiman transformation (code adapted from Niels Waller).

Value

`sl`	loadings on g + nfactors group factors, communalities, uniqueness, percent of g2 of h2
`orthog`	original orthogonal factor loadings
`oblique`	oblique factor loadings
`phi`	correlations among the transformed factors
`gload`	loadings of the lower order factors on g
...	

Author(s)

William Revelle

References

https://personality-project.org/r/r.omega.html gives an example taken from Jensen and Weng, 1994 of a S-L transformation.

See Also

omega, directSl, omega.graph, fa.graph, ICLUST,VSS

Examples

```
jen <- sim.hierarchical()  #create a hierarchical demo
if(!require(GPArotation)) {
   message("I am sorry, you must have GPArotation installed to use schmid.")} else {
   p.jen <- schmid(jen,digits=2)   #use the oblimin rotation
p.jen <- schmid(jen,rotate="promax") #use the promax rotation
}
```

Schmid *12 variables created by Schmid and Leiman to show the Schmid-Leiman Transformation*

Description

John Schmid and John M. Leiman (1957) discuss how to transform a hierarchical factor structure to a bifactor structure. Schmid contains the example 12 x 12 correlation matrix. schmid.leiman is a 12 x 12 correlation matrix with communalities on the diagonal. This can be used to show the effect of correcting for attenuation. Two additional data sets are taken from Chen et al. (2006).

Usage

```
data(Schmid)
```

Details

Two artificial correlation matrices from Schmid and Leiman (1957). One real and one artificial covariance matrices from Chen et al. (2006).

- Schmid: a 12 x 12 artificial correlation matrix created to show the Schmid-Leiman transformation.
- schmid.leiman: A 12 x 12 matrix with communalities on the diagonal. Treating this as a covariance matrix shows the 6 x 6 factor solution

- Chen: An 18 x 18 covariance matrix of health related quality of life items from Chen et al. (2006). Number of observations = 403. The first item is a measure of the quality of life. The remaining 17 items form four subfactors: The items are (a) Cognition subscale: “Have difficulty reasoning and solving problems?" “React slowly to things that were said or done?"; “Become confused and start several actions at a time?" “Forget where you put things or appointments?"; “Have difficulty concentrating?" (b) Vitality subscale: “Feel tired?" “Have enough energy to do the things you want?" (R) “Feel worn out?" ; “Feel full of pep?" (R). (c) Mental health subscale: “Feel calm and peaceful?"(R) “Feel downhearted and blue?"; “Feel very happy"(R) ; “Feel very nervous?" ; “Feel so down in the dumps nothing could cheer you up? (d) Disease worry subscale: “Were you afraid because of your health?"; “Were you frustrated about your health?"; “Was your health a worry in your life?" .
- West: A 16 x 16 artificial covariance matrix from Chen et al. (2006).

Source

John Schmid Jr. and John. M. Leiman (1957), The development of hierarchical factor solutions.Psychometrika, 22, 83-90.

F.F. Chen, S.G. West, and K.H. Sousa.(2006) A comparison of bifactor and second-order models of quality of life. Multivariate Behavioral Research, 41(2):189-225, 2006.

References

Y.-F. Yung, D.Thissen, and L.D. McLeod. (1999) On the relationship between the higher-order factor model and the hierarchical factor model. Psychometrika, 64(2):113-128, 1999.

Examples

```
data(Schmid)
cor.plot(Schmid,TRUE)
print(fa(Schmid,6,rotate="oblimin"),cut=0)  #shows an oblique solution
round(cov2cor(schmid.leiman),2)
cor.plot(cov2cor(West),TRUE)
#do not use the first model, just showing the names
model <- 'Cognition =~ dif_reasoning +slow_react + confused + forgetful + dif_concent
          Vitality =~ tired + energetic.R + worn_out      +    peppy.R
          MentalHealth =~ calm.R  + blue+ happy.R  + nervous+ down
          DiseaseWorry =~ afraid + frustrated+ worried'
#model with relative short names
rownames(Chen) <- colnames(Chen)
model <- 'Cognition =~ diff  + slo + con + for + dcon
          Vitality =~ tired + ener + worn + pep
           MentalHealth =~ calm + blue+ hap  + nerv + down
           DiseaseWorry =~ afr + frust + wor'
 #no g
chenwest <- CFA(model, Chen[-1,-1])
#bifactor
chenwest.bi <- CFA.bifactor(model,Chen[-1,-1], all=TRUE)
om <- omega(Chen[-1,-1],4, covar=TRUE)
fa.congruence(chenwest.bi,om) #bifactor agrees with sl from omega
```

score.alpha *Score scales and find Cronbach's alpha as well as associated statistics*

Description

Given a matrix or data.frame of k keys for m items (-1, 0, 1), and a matrix or data.frame of items scores for m items and n people, find the sum scores or average scores for each person and each scale. In addition, report Cronbach's alpha, the average r, the scale intercorrelations, and the item by scale correlations. (Superseded by scoreItems).

Usage

```
score.alpha(keys, items, labels = NULL, totals=TRUE,digits = 2) #deprecated
```

Arguments

keys	A matrix or dataframe of -1, 0, or 1 weights for each item on each scale
items	Data frame or matrix of raw item scores
labels	column names for the resulting scales
totals	Find sum scores (default) or average score
digits	Number of digits for answer (default =2)

Details

This function has been replaced with scoreItems (for multiple scales) and alpha for single scales.

The process of finding sum or average scores for a set of scales given a larger set of items is a typical problem in psychometric research. Although the structure of scales can be determined from the item intercorrelations, to find scale means, variances, and do further analyses, it is typical to find the sum or the average scale score.

Various estimates of scale reliability include "Cronbach's alpha", and the average interitem correlation. For k = number of items in a scale, and av.r = average correlation between items in the scale, alpha = k * av.r/(1+ (k-1)*av.r). Thus, alpha is an increasing function of test length as well as the test homeogeneity.

Alpha is a poor estimate of the general factor saturation of a test (see Zinbarg et al., 2005) for it can seriously overestimate the size of a general factor, and a better but not perfect estimate of total test reliability because it underestimates total reliability. None the less, it is a useful statistic to report.

Value

scores	Sum or average scores for each subject on the k scales
alpha	Cronbach's coefficient alpha. A simple (but non-optimal) measure of the internal consistency of a test. See also beta and omega.
av.r	The average correlation within a scale, also known as alpha 1 is a useful index of the internal consistency of a domain.

n.items	Number of items on each scale
cor	The intercorrelation of all the scales
item.cor	The correlation of each item with each scale. Because this is not corrected for item overlap, it will overestimate the amount that an item correlates with the other items in a scale.

Author(s)

William Revelle

References

An introduction to psychometric theory with applications in R (in preparation). https://personality-project.org/r/book/

See Also

scoreItems, alpha, correct.cor, cluster.loadings, omega

Examples

```
y <- attitude     #from the datasets package
keys <- matrix(c(rep(1,7),rep(1,4),rep(0,7),rep(-1,3)),ncol=3)
labels <- c("first","second","third")
x <- score.alpha(keys,y,labels) #deprecated
```

---

score.multiple.choice *Score multiple choice items and provide basic test statistics*

Description

Ability tests are typically multiple choice with one right answer. score.multiple.choice takes a scoring key and a data matrix (or data.frame) and finds total or average number right for each participant. Basic test statistics (alpha, average r, item means, item-whole correlations) are also reported.

Usage

```
score.multiple.choice(key, data, score = TRUE, totals = FALSE, ilabels = NULL,
      missing = TRUE, impute = "median", digits = 2,short=TRUE,skew=FALSE)
```

Arguments

key	A vector of the correct item alternatives
data	a matrix or data frame of items to be scored.
score	score=FALSE, just convert to right (1) or wrong (0). score=TRUE, find the totals or average scores and do item analysis
totals	total=FALSE: find the average number correct total=TRUE: find the total number correct
ilabels	item labels
missing	missing=TRUE: missing values are replaced with means or medians missing=FALSE missing values are not scored
impute	impute="median", replace missing items with the median score impute="mean": replace missing values with the item mean
digits	How many digits of output
short	short=TRUE, just report the item statistics, short=FALSE, report item statistics and subject scores as well
skew	Should the skews and kurtosi of the raw data be reported? Defaults to FALSE because what is the meaning of skew for a multiple choice item?

Details

Basically combines score.items with a conversion from multiple choice to right/wrong.

The item-whole correlation is inflated because of item overlap.

The example data set is taken from the Synthetic Aperture Personality Assessment personality and ability test at https://www.sapa-project.org/.

Value

scores	Subject scores on one scale
missing	Number of missing items for each subject
item.stats	scoring key, response frequencies, item whole correlations, n subjects scored, mean, sd, skew, kurtosis and se for each item
alpha	Cronbach's coefficient alpha
av.r	Average interitem correlation

Author(s)

William Revelle

See Also

score.items, omega

Examples

```
if(require(psychTools)) {
data(psychTools::iqitems)
iq.keys <- c(4,4,4, 6,6,3,4,4,  5,2,2,4,  3,2,6,7)
score.multiple.choice(iq.keys,psychTools::iqitems)
#just convert the items to true or false
iq.tf <- score.multiple.choice(iq.keys,psychTools::iqitems,score=FALSE)
describe(iq.tf)  #compare to previous results
}
```

---

scoreIrt *Find Item Response Theory (IRT) based scores for dichotomous or polytomous items*

Description

irt.fa finds Item Response Theory (IRT) parameters through factor analysis of the tetrachoric or polychoric correlations of dichtomous or polytomous items. scoreIrt uses these parameter estimates of discrimination and location to find IRT based scores for the responses. As many factors as found for the correlation matrix will be scored. scoreIrt.2pl will score lists of scales.

Usage

```
scoreIrt(stats=NULL, items, keys=NULL,cut = 0.3,bounds=c(-4,4),mod="logistic")
#scoreIrt is a wrapper for
#     score.irt.2 and score.irt.poly
score.irt.2(stats, items,keys=NULL,cut = 0.3,bounds=c(-4,4),mod="logistic")
scoreIrt.1pl(keys.list,items,correct=.5,messages=FALSE,cut=.3,bounds=c(-4,4),
     mod="logistic")  #Rasch like scaling
scoreIrt.2pl(itemLists,items,correct=.5,messages=FALSE,cut=.3,bounds=c(-4,4),
   mod="logistic")  #2 pl scoring



 #the higher order call just calls one of the next two
  #for dichotomous items
score.irt.2(stats, items,keys=NULL,cut = 0.3,bounds=c(-4,4),mod="logistic")
  #for polytomous items
score.irt.poly(stats, items, keys=NULL, cut = 0.3,bounds=c(-4,4),mod="logistic")
    #to create irt like statistics for plotting
irt.stats.like(items,stats,keys=NULL,cut=.3)
make.irt.stats(difficulty,discrimination)

irt.tau(x)    #find the tau values for the x object
irt.se(stats,scores=0,D=1.702)

#the next is an alias for scoreIrt and has been deprecated.
```

```
score.irt(stats=NULL, items, keys=NULL,cut = 0.3,bounds=c(-4,4),mod="logistic")
```

Arguments

stats	Output from irt.fa is used for parameter estimates of location and discrimination. Stats may also be the output from a normal factor analysis (fa). If stats is a data.frame of discrimination and thresholds from some other data set, these values will be used. See the last example.
items	The raw data, may be either dichotomous or polytomous.
itemLists	a list of items to be factored and scored for each scale, can be a keys.list as used in scoreItems or scoreIrt.1pl
keys.list	A list of items to be scored with keying direction (see example)
keys	A keys matrix of which items should be scored for each factor
cut	Only items with discrimination values > cut will be used for scoring.
x	The raw data to be used to find the tau parameter in irt.tau
bounds	The lower and upper estimates for the fitting function
mod	Should a logistic or normal model be used in estimating the scores?
correct	What value should be used for continuity correction when finding the tetrachoric or polychoric correlations when using irt.fa
messages	Should messages be suppressed when running multiple scales?
scores	A single score or a vector of scores to find standard errors
D	The scaling function for the test information statistic used in irt.se
difficulty	The difficulties for each item in a polytomous scoring
discrimination	The item discrimin

Details

Although there are more elegant ways of finding subject scores given a set of item locations (difficulties) and discriminations, simply finding that value of theta θ that best fits the equation $P(x|\theta) = 1/(1 + exp(\beta(\delta - \theta))$ for a score vector X, and location δ and discrimination β provides more information than just total or average scores. With complete data, total scores and irt estimates are almost perfectly correlated. However, the irt estimates provide much more information in the case of missing data.

The bounds parameter sets the lower and upper limits to the estimate. This is relevant for the case of a subject who gives just the lowest score on every item, or just the top score on every item. Formerly (prior to 1.6.12) this was done by estimating these scores by finding the probability of missing every item taken, converting this to a quantile score based upon the normal distribution, and then assigning a z value equivalent to 1/2 of that quantile. Similarly, if a person gets all the items they take correct, their score is defined as the quantile of the z equivalent to the probability of getting all of the items correct, and then moving up the distribution half way. If these estimates exceed either the upper or lower bounds, they are adjusted to those boundaries.

As of 1.6.9, the procedure is very different. We now assume that all items are bounded with one passed item that is easier than all items given, and one failed item that is harder than any item given. This produces much cleaner results.

There are several more elegant packages in R that provide Full Information Maximum Likeliood IRT based estimates. In particular, the MIRT package seems especially good. The ltm package give equivalent estimates to MIRT for dichotomous data but produces unstable estimates for polytomous data and should be avoided.

Although the scoreIrt estimates are are not FIML based they seem to correlated with the MIRT estiamtes with values exceeding .99. Indeed, based upon very limited simulations there are some small hints that the solutions match the true score estimates slightly better than do the MIRT estimates. `scoreIrt` seems to do a good job of recovering the basic structure.

If trying to use item parameters from a different data set (e.g. some standardization sample), specify the stats as a data frame with the first column representing the item discriminations, and the next columns the item difficulties. See the last example.

The two wrapper functions `scoreIrt.1pl` and `scoreIrt.2pl` are very fast and are meant for scoring one or many scales at a time with a one factor model (`scoreIrt.2pl`) or just Rasch like scoring. Just specify the scoring direction for a number of scales (`scoreIrt.1pl`) or just items to score for a number of scales `scoreIrt.2pl`. `scoreIrt.2pl` will then apply `irt.fa` to the items for each scale separately, and then find the 2pl scores.

The keys.list is a list of items to score for each scale. Preceding the item name with a negative sign will reverse score that item (relevant for `scoreIrt.1pl`. Alternatively, a keys matrix can be created using `make.keys`. The keys matrix is a matrix of 1s, 0s, and -1s reflecting whether an item should be scored or not scored for a particular factor. See `scoreItems` or `make.keys` for details. The default case is to score all items with absolute discriminations > cut.

If one wants to score scales taking advantage of differences in item location but not do a full IRT analysis, then find the item difficulties from the raw data using `irt.tau` or combine this information with a scoring keys matrix (see `scoreItems` and `make.keys` and create quasi-IRT statistics using `irt.stats.like`. This is the equivalent of doing a quasi-Rasch model, in that all items are assumed to be equally discriminating. In this case, tau values may be found first (using `irt.tau` or just found before doing the scoring. This is all done for you inside of `scoreIrt.1pl`.

Such irt based scores are particularly useful if finding scales based upon massively missing data (e.g., the SAPA data sets). Even without doing the full irt analysis, we can take into account different item difficulties.

David Condon has added a very nice function to do 2PL analysis for a number of scales at one time. `scoreIrt.2pl` takes the raw data file and a list of items to score for each of multiple scales. These are then factored (currently just one factor for each scale) and the loadings and difficulties are used for scoring.

There are conventionally two different metrics and models that are used. The logistic metric and model and the normal metric and model. These are chosen using the mod parameter.

`irt.se` finds the standard errors for scores with a particular value. These are based upon the information curves calculated by `irt.fa` and are not based upon the particular score of a particular subject.

Value

`scores`	A data frame of theta estimates, total scores based upon raw sums, and estimates of fit.
`tau`	Returned by irt.tau: A data frame of the tau values for an object of dichotomous or polytomous items. Found without bothering to find the correlations.

Note

It is very important to note that when using irt.fa to find the discriminations, to set the sort option to be FALSE. This is now the default. Otherwise, the discriminations will not match the item order.

Always under development. Suggestions for improvement are most appreciated.

scoreIrt is just a wrapper to score.irt.poly and score.irt.2. The previous version had score.irt which is now deprecated as I try to move to camelCase.

scoreIrt.2pl is a wrapper for irt.fa and scoreIrt. It was originally developed by David Condon.

Author(s)

William Revelle, David Condon

References

Kamata, Akihito and Bauer, Daniel J. (2008) A Note on the Relation Between Factor Analytic and Item Response Theory Models Structural Equation Modeling, 15 (1) 136-153.

McDonald, Roderick P. (1999) Test theory: A unified treatment. L. Erlbaum Associates.

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at https://personality-project.org/r/book/

See Also

irt.fa for finding the parameters. For more conventional scoring algorithms see scoreItems. irt.responses will plot the empirical response patterns for the alternative response choices for multiple choice items. For more conventional IRT estimations, see the ltm package.

Examples

```
  #not run in the interest of time, but worth doing
d9 <- sim.irt(9,1000,-2.5,2.5,mod="normal") #dichotomous items
test <- irt.fa(d9$items)
scores <- scoreIrt(test,d9$items)
scores.df <- data.frame(scores,true=d9$theta) #combine the estimates with the true thetas.
pairs.panels(scores.df,pch=".",
main="Comparing IRT and classical with complete data")
#now show how to do this with a quasi-Rasch model
tau <- irt.tau(d9$items)
scores.rasch <- scoreIrt(tau,d9$items,key=rep(1,9))
scores.dfr<- data.frame(scores.df,scores.rasch) #almost identical to 2PL model!
pairs.panels(scores.dfr)
#with all the data, why bother ?

#now delete some of the data
d9$items[1:333,1:3] <- NA
d9$items[334:666,4:6] <- NA
d9$items[667:1000,7:9] <- NA
scores <- scoreIrt(test,d9$items)
scores.df <- data.frame(scores,true=d9$theta) #combine the estimates with the true thetas.
pairs.panels(scores.df, pch=".",
```

```
main="Comparing IRT and classical with random missing data")
 #with missing data, the theta estimates are noticably better.
#now show how to do this with a quasi-Rasch model
tau <- irt.tau(d9$items)
scores.rasch <- scoreIrt(tau,d9$items,key=rep(1,9))
scores.dfr <- data.frame(scores.df,rasch = scores.rasch)
pairs.panels(scores.dfr)  #rasch is actually better!



v9 <- sim.irt(9,1000,-2.,2.,mod="normal") #dichotomous items
items <- v9$items
test <- irt.fa(items)
total <- rowSums(items)
ord <- order(total)
items <- items[ord,]


#now delete some of the data - note that they are ordered by score
items[1:333,5:9] <- NA
items[334:666,3:7] <- NA
items[667:1000,1:4] <- NA
items[990:995,1:9] <- NA   #the case of terrible data
items[996:998,] <- 0   #all wrong
items[999:1000] <- 1   #all right
scores <- scoreIrt(test,items)
unitweighted <- scoreIrt(items=items,keys=rep(1,9)) #each item has a discrimination of 1
#combine the estimates with the true thetas.
scores.df <- data.frame(v9$theta[ord],scores,unitweighted)

colnames(scores.df) <- c("True theta","irt theta","total","fit","rasch","total","fit")
pairs.panels(scores.df,pch=".",main="Comparing IRT and classical with missing data")
 #with missing data, the theta estimates are noticably better estimates
 #of the generating theta than using the empirically derived factor loading weights

#now show the ability to score multiple scales using keys
if(require(psychTools)) {
ab.tau <- irt.tau(psychTools::ability)  #first find the tau values
ab.keys <- make.keys(psychTools::ability,list(g=1:16,reason=1:4,
letter=5:8,matrix=9:12,rotate=13:16))
#ab.scores <- scoreIrt(stats=ab.tau, items = psychTools::ability, keys = ab.keys)
}
#and now do it for polytomous items using 2pl
bfi.scores <- scoreIrt.2pl(bfi.keys,bfi[1:25])
#compare with classical unit weighting by using scoreItems
#not run in the interests of time
#bfi.unit <- scoreItems(bfi.keys,bfi[1:25])
#bfi.df <- data.frame(bfi.scores,bfi.unit$scores)
#pairs.panels(bfi.df,pch=".")


bfi.irt <- scoreIrt(items=bfi[16:20])  #find irt based N scores
```

```
#Specify item difficulties and discriminations from a different data set.
stats <- structure(list(MR1 = c(1.4, 1.3, 1.3, 0.8, 0.7), difficulty.1 = c(-1.2,
-2, -1.5, -1.2, -0.9), difficulty.2 = c(-0.1, -0.8, -0.4, -0.3,
-0.1), difficulty.3 = c(0.6, -0.2, 0.2, 0.2, 0.3), difficulty.4 = c(1.5,
0.9, 1.1, 1, 1), difficulty.5 = c(2.5, 2.1, 2.2, 1.7, 1.6)), row.names = c("N1",
"N2", "N3", "N4", "N5"), class = "data.frame")


stats #show them
bfi.new <-scoreIrt(stats,bfi[16:20])
bfi.irt <- scoreIrt(items=bfi[16:20])
cor2(bfi.new,bfi.irt)
newstats <- stats
newstats[2:6] <-stats[2:6 ] + 1  #change the difficulties
bfi.harder <- scoreIrt(newstats,bfi[16:20])
pooled <- cbind(bfi.irt,bfi.new,bfi.harder)
describe(pooled)  #note that the mean scores have changed
lowerCor(pooled) #and although the unit weighted scale are identical,
# the irt scales differ by changing the difficulties
```

---

scoreItems *Score item composite scales and find Cronbach's alpha, Guttman lambda 6 and item whole correlations*

Description

Given a data.frame or matrix of n items and N observations and a list of the direction to score them (a keys.list with k keys) find the sum scores or average scores for each person and each scale. In addition, report Cronbach's alpha, Guttman's Lambda 6, the average r, the scale intercorrelations, and the item by scale correlations (raw and corrected for item overlap). Replace missing values with the item median or mean if desired. Items may be keyed 1 (score it), -1) (reverse score it), or 0 (do not score it). Negatively keyed items will be reverse scored. Although prior versions used a keys matrix, it is now recommended to just use a list of scoring keys. See `make.keys` for a convenient way to make the keys file. If the input is a square matrix, then it is assumed that the input is a covariance or correlation matix and scores are not found, but the item statistics are reported. (Similar functionality to `cluster.cor`). `response.frequencies` reports the frequency of item endorsements fore each response category for polytomous or multiple choice items. `scoreFast` and `scoreVeryFast` just find sums/mean scores and do not report reliabilities. Much faster for large data sets.

Usage

```
scoreItems(keys, items, totals = FALSE, ilabels = NULL,missing=TRUE, impute="median",
      delete=TRUE, min = NULL, max = NULL, digits = 2,n.obs=NULL,select=TRUE)
scoreFast(keys, items, totals = FALSE, ilabels = NULL,missing=TRUE, impute="none",
       delete=TRUE, min = NULL, max = NULL,count.responses=FALSE, digits = 2)
```

```
scoreVeryFast(keys,items,totals=FALSE, min=NULL,max=NULL,count.responses=FALSE)

responseFrequency(items,max=10,uniqueitems=NULL)
removeMissing(x,max.miss =0)

score.items(keys, items, totals = FALSE, ilabels = NULL,missing=TRUE, impute="median",
      delete=TRUE, min = NULL, max = NULL, digits = 2,select=TRUE) #deprecated
response.frequencies(items,max=10,uniqueitems=NULL) #deprecated
```

Arguments

keys	A list of scoring keys or a matrix or dataframe of -1, 0, or 1 weights for each item on each scale which may be created by hand, or by using make.keys. Just using a list of scoring keys (see example) is probably more convenient.
items	Matrix or dataframe of raw item scores
totals	if TRUE find total scores, if FALSE (default), find average scores
ilabels	a vector of item labels.
missing	missing = TRUE is the normal case and data are imputed according to the impute option. missing=FALSE, only complete cases are scored.
impute	impute="median" replaces missing values with the item medians, impute = "mean" replaces values with the mean response. impute="none" the subject's scores are based upon the average of the keyed, but non missing scores. impute = "none" is probably more appropriate for a large number of missing cases (e.g., SAPA data).
delete	if delete=TRUE, automatically delete items with no variance (and issue a warning)
min	May be specified as minimum item score allowed, else will be calculated from data. min and max should be specified if items differ in their possible minima or maxima. See notes for details.
max	May be specified as maximum item score allowed, else will be calculated from data. Alternatively, in response frequencies, it is maximum number of alternative responses to count.
uniqueitems	If specified, the set of possible unique response categories
digits	Number of digits to report for mean scores
n.obs	If scoring from a correlation matrix, specify the number of subjects allows for the calculation of the confidence intervals for alpha.
select	By default, just find the statistics of those items that are included in scoring keys. This allows scoring of data sets that have bad data for some items that are not included in the scoring keys. This also speeds up the scoring of small subsets of item from larger data sets.
count.responses	If TRUE, report the number of items/scale answered for each subject.
x	The object returned by scoreItems
max.miss	Replace all scales with missing > max.miss with NA

Details

The process of finding sum or average scores for a set of scales given a larger set of items is a typical problem in applied psychometrics and in psychometric research. Although the structure of scales can be determined from the item intercorrelations, to find scale means, variances, and do further analyses, it is typical to find scores based upon the sum or the average item score. For some strange reason, personality scale scores are typically given as totals, but attitude scores as averages. The default for scoreItems is the average as it would seem to make more sense to report scale scores in the metric of the item.

When scoring more than one scale, it is convenient to have a list of the items on each scale and the direction to score the items. This may be converted to a keys.matrix using `make.keys` or may be entered as a keys.list directly.

Various estimates of scale reliability include "Cronbach's alpha", Guttman's Lambda 6, and the average interitem correlation. For k = number of items in a scale, and av.r = average correlation between items in the scale, alpha = k * av.r/(1+ (k-1)*av.r). Thus, alpha is an increasing function of test length as well as the test homeogeneity.

Surprisingly, more than a century after Spearman (1904) introduced the concept of reliability to psychologists, there are still multiple approaches for measuring it. Although very popular, Cronbach's α (1951) underestimates the reliability of a test and over estimates the first factor saturation.

α (Cronbach, 1951) is the same as Guttman's λ_3 (Guttman, 1945) and may be found by

$$\lambda_3 = \frac{n}{n-1}\Big(1 - \frac{tr(\vec{V})_x}{V_x}\Big) = \frac{n}{n-1}\frac{V_x - tr(\vec{V}_x)}{V_x} = \alpha$$

Perhaps because it is so easy to calculate and is available in most commercial programs, alpha is without doubt the most frequently reported measure of internal consistency reliability. Alpha is the mean of all possible spit half reliabilities (corrected for test length). For a unifactorial test, it is a reasonable estimate of the first factor saturation, although if the test has any microstructure (i.e., if it is "lumpy") coefficients β (Revelle, 1979; see `ICLUST`) and ω_h (see `omega`) (McDonald, 1999; Revelle and Zinbarg, 2009) are more appropriate estimates of the general factor saturation. ω_t (see `omega`) is a better estimate of the reliability of the total test.

Guttman's Lambda 6 (G6) considers the amount of variance in each item that can be accounted for the linear regression of all of the other items (the squared multiple correlation or smc), or more precisely, the variance of the errors, e_j^2, and is

$$\lambda_6 = 1 - \frac{\sum e_j^2}{V_x} = 1 - \frac{\sum(1 - r_{smc}^2)}{V_x}.$$

The squared multiple correlation is a lower bound for the item communality and as the number of items increases, becomes a better estimate.

G6 is also sensitive to lumpyness in the test and should not be taken as a measure of unifactorial structure. For lumpy tests, it will be greater than alpha. For tests with equal item loadings, alpha > G6, but if the loadings are unequal or if there is a general factor, G6 > alpha. Although it is normal when scoring just a single scale to calculate G6 from just those items within the scale, logically it is appropriate to estimate an item reliability from all items available. This is done here and is labeled as G6* to identify the subtle difference.

Alpha and G6* are both positive functions of the number of items in a test as well as the average intercorrelation of the items in the test. When calculated from the item variances and total test

variance, as is done here, raw alpha is sensitive to differences in the item variances. Standardized alpha is based upon the correlations rather than the covariances. alpha is a generalization of an earlier estimate of reliability for tests with dichotomous items developed by Kuder and Richardson, known as KR20, and a shortcut approximation, KR21. (See Revelle, in prep; Revelle and Condon, 2018, 2019).

A useful index is the ratio of reliable variance to unreliable variance and is known as the Signal/Noise ratio. This is just

$$s/n = \frac{n\bar{r}}{1 - n\bar{r}}$$

(Cronbach and Gleser, 1964; Revelle and Condon (2019)).

Standard errors for unstandardized alpha are reported using the formula from Duhachek and Iacobucci (2005).

More complete reliability analyses of a single scale can be done using the `omega` function which finds ω_h and ω_t based upon a hierarchical factor analysis. Alternative estimates of the Greatest Lower Bound for the reliability are found in the `guttman` function.

Alpha is a poor estimate of the general factor saturation of a test (see Revelle and Zinbarg, 2009; Zinbarg et al., 2005) for it can seriously overestimate the size of a general factor, and a better but not perfect estimate of total test reliability because it underestimates total reliability. None the less, it is a common statistic to report. In general, the use of alpha should be discouraged and the use of more appropriate estimates (ω_h and ω_t) should be encouraged.

Correlations between scales are attenuated by a lack of reliability. Correcting correlations for reliability (by dividing by the square roots of the reliabilities of each scale) sometimes help show structure. This is done in the scale intercorrelation matrix with raw correlations below the diagonal and unattenuated correlation above the diagonal.

There are several alternative ways to treat missing values. By default, missing values are replaced with the corresponding median value for that item. Means can be used instead (impute="mean"), or subjects with missing data can just be dropped (missing = FALSE). For data with a great deal of missingness, yet another option is to just find the average of the available responses (impute="none"). This is useful for findings means for scales for the SAPA project (see https://www.sapa-project.org/) where most scales are estimated from random sub samples of the items from the scale. In this case, the alpha reliabilities are seriously overinflated because they are based upon the total number of items in each scale. The "alpha observed" values are based upon the average number of items answered in each scale using the standard form for alpha a function of inter-item correlation and number of items.

Using the impute="none" option as well as asking for totals (totals="TRUE") will be done, although a warning will be issued because scores will now reflect the number of items responded to much more than the actual pattern of responses.

The number of missing responses for each person for each scale is reported in the missing object. One possibility is to drop scores just for those scales with missing responses. This may be done adding the code:

scores$scores[scores$missing >0] <- NA

Or by calling the removeMissing function which does the same thing.

This is shown in the last example.

Note that the default for scoreItems is to impute missing items with their median, but the default for scoreFAst is to not impute but must return the scale scores based upon the mean or total value for the items scored.

By default, scoreItems will drop those items with no variance. This changes the reliability calculations (number of items is reduced), and it mean that those items are not used in finding scores. Using the delete=FALSE option, these variables will not be dropped and their scores will be found. Various warnings are issued about the SMC not being correct. What do you you expect when there is no variance for an item?

scoreItems can be applied to correlation matrices to find just the reliability statistics. This will be done automatically if the items matrix is symmetric.

scoreFast just finds the scores (with or without imputation) and does not report other statistics. It is much faster! scoreVeryFast is even more stripped down, no imputation, just scores based upon the observed data. No statistics.

Value

scores	Sum or average scores for each subject on the k scales
alpha	Cronbach's coefficient alpha. A simple (but non-optimal) measure of the internal consistency of a test. See also beta and omega. Set to 1 for scales of length 1.
av.r	The average correlation within a scale, also known as alpha 1, is a useful index of the internal consistency of a domain. Set to 1 for scales with 1 item.
G6	Guttman's Lambda 6 measure of reliability
G6*	A generalization of Guttman's Lambda 6 measure of reliability using all the items to find the smc.
n.items	Number of items on each scale
item.cor	The correlation of each item with each scale. Because this is not corrected for item overlap, it will overestimate the amount that an item correlates with the other items in a scale.
cor	The intercorrelation of all the scales based upon the interitem correlations (see note for why these differ from the correlations of the observed scales themselves).
corrected	The correlations of all scales (below the diagonal), alpha on the diagonal, and the unattenuated correlations (above the diagonal)
item.corrected	The item by scale correlations for each item, corrected for item overlap by replacing the item variance with the smc for that item
response.freq	The response frequency (based upon number of non-missing responses) for each alternative.
missing	How many items were not answered for each scale
num.ob.item	The average number of items with responses on a scale. Used in calculating the alpha.observed– relevant for SAPA type data structures.

Note

It is important to recognize in the case of massively missing data (e.g., data from a Synthetic Aperture Personality Assessment (https://www.sapa-project.org/) or the International Cognitive Ability Resources (https://icar-project.org)) study where perhaps only 10-50% of the items per scale are given to any one subject)) that the number of items per scale, and hence standardized alpha, is not the nominal value and hence alpha of the observed scales will be overestimated. For this case (impute="none"), an additional alpha (alpha.ob) is reported.

More importantly in this case of massively missing data, there is a difference between the correlations of the composite scales based upon the correlations of the items and the correlations of the scored scales based upon the observed data. That is, the cor object will have correlations as if all items had been given, while the correlation of the scores object will reflect the actual correlation of the scores. For SAPA data, it is recommended to use the cor object. Confidence of these correlations may be found using the cor.ci function.

Further note that the inter-scale correlations are based upon the correlations of scales formed from the covariance matrix of the items. This will differ from the correlation of scales based upon the correlation of the items. Thus, although scoreItems will produce reliabilities and intercorrelations from either the raw data or from a correlation matrix, these values will differ slightly. In addition, with a great deal of missing data, the scale intercorrelations will differ from the correlations of the scores produced, for the latter will be attenuated.

An alternative to classical test theory scoring is to use scoreIrt or irt.CFA to find score estimates based upon Item Response Theory. This is particularly useful in the case of SAPA data which tend to be massively missing. It is also useful to find scores based upon polytomous items following a factor analysis of the polychoric correlation matrix (see irt.fa and irt.CFA). However, remember that this only makes sense if the items are unidimensional. That is to say, if forming item composites from (e.g., bestScales), that are empirically derived, they will necessarily have a clear factor structure and the IRT based scoring does not make sense.

For intentionally multiple dimensional sets of items, the irt.CFA function can be used for multiple scales.

When reverse scoring items from a set where items differ in their possible minima or maxima, it is important to specify the min and max values. Items are reversed by subtracting them from max + min. Thus, if items range from 1 to 6, items are reversed by subtracting them from 7. But, if the data set includes other variables, (say an id field) that far exceeds the item min or max, then the max id will incorrectly be used to reverse key. min and max can either be single values, or vectors for all items. Compare two examples of scoring the bfi data set.

If scales are formed with overlapping items, then the correlations of the scales will be seriously inflated. scoreOverlap will adjust the correlations for this overlap.

Yet another possibility for scoring large data sets is to ignore all the reliability calculations and just find the scores. This may be done using scoreFast or scoreVeryFast. These two functions just find mean scores (scoreVeryFast without imputation) or will do imputation if desired scoreFast. For 200K cases on 1000 variables with 11 scales, scoreVeryFast took 4.7 seconds on a Mac PowerBook with a 2.8GHZ Intel I7 and scoreFast took 23.2 seconds. scoreIrt.1pl for the same problem took xxx with options("mc.cores"=1) (not parallel processing) and 1259 seconds with options("mc.cores"=NULL) (implying 2 cores) and with four cores was very slow (probably too much parallel processing).

Yet one more possibility is to find scores based upon a matrix of weights (e.g. zero order correlations, beta weights, or factor weights.) In this case, scores are simply the product of the weights

times the (standardized) items. If using coefficients from a regression analysis (lm), a column of 1's is added to the data and labeled "(Intercept)" and this is used in the calculation. This is done by scoreWtd.

Author(s)

William Revelle

References

Cronbach, L.J. and Gleser G.C. (1964)The signal/noise ratio in the comparison of reliability coefficients. Educational and Psychological Measurement, 24 (3) 467-480.

Duhachek, A. and Iacobucci, D. (2004). Alpha's standard error (ase): An accurate and precise confidence interval estimate. Journal of Applied Psychology, 89(5):792-808.

McDonald, R. P. (1999). Test theory: A unified treatment. L. Erlbaum Associates, Mahwah, N.J.

Revelle, W. (in preparation) An introduction to psychometric theory with applications in R. https://personality-project.org/r/book/

Revelle, W. and Condon, D.C. Reliability. In Irwing, P., Booth, T. and Hughes, D. (Eds). the Wiley-Blackwell Handbook of Psychometric Testing (2018).

Revelle, W. and Condon, D.C. (2019) Reliability: from alpha to omega. Psychological Assessment, 31, 1395-1411.

Revelle W. and R.E. Zinbarg. (2009) Coefficients alpha, beta, omega and the glb: comments on Sijtsma. Psychometrika, 74(1):145-154.

Zinbarg, R. E., Revelle, W., Yovel, I. and Li, W. (2005) Cronbach's alpha, Revelle's beta, and McDonald's omega h, Their relations with each other and two alternative conceptualizations of reliability, Psychometrika, 70, 123-133.

See Also

make.keys for a convenient way to create the keys file, score.multiple.choice for multiple choice items,
alpha, correct.cor, cluster.cor , cluster.loadings, omega, guttman for item/scale analysis.

If scales are formed from overlapping sets of items, their correlations will be inflated. This is corrected for when using the scoreOverlap function which, although it will not produce scores, will report scale intercorrelations corrected for item overlap.

In addition, the irt.fa function provides an alternative way of examining the structure of a test and emphasizes item response theory approaches to the information returned by each item and the total test. Associated with these IRT parameters is the scoreIrt function for finding IRT based scores as well as irt.responses to show response curves for the alternatives in a multiple choice test.

scoreIrt will find both IRT based estimates as well as average item response scores. These latter correlate perfectly with those found by scoreItems. If using a keys matrix, the score.irt results are based upon the item difficulties with the assumption that all items are equally discriminating (effectively a Rasch model). These scores are probably most useful in the case of massively missing data because they can take into account the item difficulties.

scoreIrt.1pl finds the item difficulty parameters and then applies a 1 parameter (Rasch like) model. It chooses items based upon a keys.list.

Examples

```
#see  the example including the bfi data set
data(bfi)
keys.list <- list(agree=c("-A1","A2","A3","A4","A5"),
 conscientious=c("C1","C2","C3","-C4","-C5"),extraversion=c("-E1","-E2","E3","E4","E5"),
  neuroticism=c("N1","N2","N3","N4","N5"), openness = c("O1","-O2","O3","O4","-O5"))
  keys <- make.keys(bfi,keys.list)  #no longer necessary
 scores <- scoreItems(keys,bfi,min=1,max=6)  #using a keys matrix
 scores <- scoreItems(keys.list,bfi,min=1,max=6)  # or just use the keys.list
 summary(scores)
 #to get the response frequencies, we need to not use the age variable
 scores <- scoreItems(keys[1:25,],bfi[1:25]) #we do not need to specify min or
 #max if  there are no values (such as age) outside the normal item range.
 scores
 #The scores themselves are available in the scores$scores object.  I.e.,
 describe(scores$scores)


 #compare this output to that for the impute="none" option for SAPA type data
 #first make many of the items missing in a missing pattern way
 missing.bfi <- bfi
 missing.bfi[1:1000,3:8] <- NA
 missing.bfi[1001:2000,c(1:2,9:10)] <- NA
 scores <- scoreItems(keys.list,missing.bfi,impute="none",min=1,max=6)
 scores
 describe(scores$scores)  #the actual scores themselves

 #If we want to delete scales scores for people who did not answer some items for one
 #(or more) scales, we can do the following:

  scores <- scoreItems(keys.list,missing.bfi,totals=TRUE,min=1,max=6) #find total scores
  describe(scores$scores) #note that missing data were replaced with median for the item
  scores$scores[scores$missing > 0] <- NA  #get rid of cases with missing data
  describe(scores$scores)

  #or
  fixed <- removeMissing(scores)   #does the same thing
describe(fixed)
```

---

scoreOverlap *Find correlations of composite variables (corrected for overlap) from a larger matrix.*

Description

Given a n x c cluster definition matrix of -1s, 0s, and 1s (the keys) , and a n x n correlation matrix, or an N x n data matrix, find the correlations of the composite clusters. The keys matrix can be entered by hand, copied from the clipboard (read.clipboard), or taken as output from the

factor2cluster or make.keys functions. Similar functionality to scoreItems which also gives item by cluster correlations. scoreBy does this for individual subjects after a call to statsBy.

Usage

```
scoreOverlap(keys, r, correct = TRUE, SMC = TRUE, av.r = TRUE, item.smc = NULL,
     impute = TRUE,select=TRUE, scores=FALSE,  min=NULL,max=NULL)
scoreBy(keys,stats, correct = TRUE, SMC = TRUE, av.r = TRUE, item.smc = NULL,
     impute = TRUE,select=TRUE,min.n=3,smooth=FALSE)
cluster.cor(keys, r.mat, correct = TRUE,SMC=TRUE,item.smc=NULL,impute=TRUE)
```

Arguments

keys	A list of scale/cluster keys, or a matrix of cluster keys
r.mat	A correlation matrix
r	Either a correlation matrix or a raw data matrix
stats	The output from statsBy. Needed for scoreBy.
correct	TRUE shows both raw and corrected for attenuation correlations
SMC	Should squared multiple correlations be used as communality estimates for the correlation matrix?
item.smc	the smcs of the items may be passed into the function for speed, or calculated if SMC=TRUE
impute	if TRUE and scores=FALSE, impute missing scale correlations based upon the average interitem correlation, otherwise return NA. If scores=TRUE, impute missing data by means, medians, or none.
av.r	Should the average r be used in correcting for overlap? smcs otherwise.
select	By default, just find statistics for items included in the scoring keys. This allows for finding scores from matrices with bad items if they are not included in the set of scoring keys.
min.n	The minimum number of pairwise observations needed to define a correlation (in scoreBy)
smooth	If the matrices used in scoreBy are not positive semi-definite, should we smooth them?
scores	Find the raw average scores for each subject. Same functionality as scoreItems
min	May be specified as minimum item score allowed, else will be calculated from data. min and max should be specified if items differ in their possible minima or maxima. See notes for details.
max	May be specified as maximum item score allowed, else will be calculated from data.

Details

These are three of the functions used in the SAPA (https://www.sapa-project.org/) procedures to form synthetic correlation matrices. Given any correlation matrix of items, it is easy to find the correlation matrix of scales made up of those items. This can also be done from the original data

matrix or from the correlation matrix using scoreItems which is probably preferred unless the keys are overlapping. It is important to remember with SAPA data, that scale correlations should be found from the item correlations, not the raw data.

In the case of overlapping keys, (items being scored on multiple scales), scoreOverlap will adjust for this overlap by replacing the overlapping covariances (which are variances when overlapping) with the corresponding best estimate of an item's "true" variance using either the average correlation or the smc estimate for that item. This parallels the operation done when finding alpha reliability. This is similar to ideas suggested by Cureton (1966) and Bashaw and Anderson (1966) but uses the smc or the average interitem correlation (default).

A typical use in the SAPA project is to form item composites by clustering or factoring (see fa, ICLUST, principal), extract the clusters from these results (factor2cluster), and then form the composite correlation matrix using cluster.cor. The variables in this reduced matrix may then be used in multiple correlation procedures using lmCor.

The original correlation is pre and post multiplied by the (transpose) of the keys matrix.

If some correlations are missing from the original matrix this will lead to missing values (NA) for scale intercorrelations based upon those lower level correlations. If impute=TRUE (the default), a warning is issued and the correlations are imputed based upon the average correlations of the non-missing elements of each scale.

Because the alpha estimate of reliability is based upon the correlations of the items rather than upon the covariances, this estimate of alpha is sometimes called "standardized alpha". If the raw items are available, it is useful to compare standardized alpha with the raw alpha found using scoreItems. They will differ substantially only if the items differ a great deal in their variances.

The MIMS matrix reflects the average correlations of the Multiple Items within each of the Multiple Scales. This has been suggested as a measure of scale validity. These correlations are adjusted for item overlap.

scoreOverlap answers an important question when developing scales and related subscales, or when comparing alternative versions of scales. For by removing the effect of item overlap, it gives a better estimate the relationship between the latent variables estimated by the observed sum (mean) scores.

scoreBy finds the within subject correlations after preprocessing with statsBy. This is useful if doing an ESM study with multiple occasions for each subject. It also makes it possible to find the correlations for subsets of subjects. See the example. Note that it likely that for ESM data with a high level of missingness that the correlation matrices will not be positive-semi-definite. This can lead to composite score correlations that exceed 1. Smoothing will resolve this problem.

scoreBy is useful when examining multi-level models where we want to examine the correlations within subjects (e.g., for ESM data) or within groups of subjects (when examining the stability of correlational structures by subgroups). For both cases the data must be processed first by statsBy. To find the variances of the scales it is necessary to use the cor="cov" option in statsBy.

Starting in version 2.4.1, the scores option has been added. This will score the data (as does scoreItems) if it is not a correlation matrix but does not have all the options of scoreItems. Obviously, while the correlations can be corrected for item ovelap, the raw scores can not be so corrected.

Value

cor the (raw) correlation matrix of the clusters

sd	standard deviation of the cluster scores
corrected	raw correlations below the diagonal, alphas on diagonal, disattenuated above diagonal
alpha	The (standardized) alpha reliability of each scale.
G6	Guttman's Lambda 6 reliability estimate is based upon the smcs for each item in a scale. G6 uses the smc based upon the entire item domain.
av.r	The average inter item correlation within a scale
size	How many items are in each cluster?
MIMS	A matrix of the average correlations, corrected for overlap, between the various scales.

Note

See SAPA Revelle, W., Wilt, J., and Rosenthal, A. (2010) Personality and Cognition: The Personality-Cognition Link. In Gruszka, A. and Matthews, G. and Szymura, B. (Eds.) Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control, Springer.

The second example uses the msq data set of 72 measures of motivational state to examine the overlap between four lower level scales and two higher level scales.

Author(s)

Maintainer: William Revelle <revelle@northwestern.edu>

References

Bashaw, W. and Anderson Jr, H. E. (1967). A correction for replicated error in correlation coefficients. Psychometrika, 32(4):435-441.

Cureton, E. (1966). Corrected item-test correlations. Psychometrika, 31(1):93-96.

See Also

factor2cluster, mat.regress, alpha, and most importantly, scoreItems, which will do all of what cluster.cor does for most users. cluster.cor is an important helper function for iclust

Examples

```
if(require(psychTools)) {
#use the msq data set that shows the structure of energetic and tense arousal
small.msq <- psychTools::msq[ c("active", "energetic", "vigorous", "wakeful",
"wide.awake", "full.of.pep", "lively", "sleepy", "tired", "drowsy","intense",
"jittery", "fearful", "tense", "clutched.up", "quiet", "still",    "placid",
 "calm", "at.rest") ]
small.R <- cor(small.msq,use="pairwise")
keys.list <- list(
EA = c("active", "energetic", "vigorous", "wakeful", "wide.awake", "full.of.pep",
       "lively", "-sleepy", "-tired", "-drowsy"),
TA =c("intense", "jittery", "fearful", "tense", "clutched.up", "-quiet", "-still",
       "-placid", "-calm", "-at.rest") ,
```

```
high.EA = c("active", "energetic", "vigorous", "wakeful", "wide.awake", "full.of.pep",
       "lively"),
low.EA =c("sleepy", "tired", "drowsy"),
lowTA= c("quiet", "still", "placid", "calm", "at.rest"),
highTA = c("intense", "jittery", "fearful", "tense", "clutched.up")
   )

keys <- make.keys(small.R,keys.list)

adjusted.scales <- scoreOverlap(keys.list,small.R)
#compare with unadjusted
confounded.scales <- cluster.cor(keys,small.R)
summary(adjusted.scales)
#note that the EA and high and low EA and TA and high and low TA
# scale correlations are confounded
summary(confounded.scales)
}
bfi.stats <- statsBy(bfi,group="education",cors=TRUE ,cor="cov")
 #specify to find covariances
 bfi.plus.keys <- c(bfi.keys,gender="gender",age ="age")
bfi.by <- scoreBy(bfi.plus.keys,bfi.stats)
bfi.by$var   #to show the variances of each scale by groupl
round(bfi.by$cor.mat,2)  #the correlations by group
bfi.by$alpha
```

---

scoreWtd *Score items using regression or correlation based weights*

Description

Item weights from bestScales or lmCor are used to find weighted scale scores. In contrast to the unit weights used in scoreItems, scoreWtd will multiply the data by a set of weights to find scale scores. These weight may come from a regression (e.g., lm or lmCor) or may be the zero order correlation weights from bestScales.

Usage

```
scoreWtd(weights, items, std = TRUE, sums = FALSE, impute = "none")
```

Arguments

weights	This is just a matrix of weights to use for each item for each scale.
items	Matrix or dataframe of raw item scores
std	if TRUE, then find weighted standard scores else just use raw data
sums	By default, find the average item score. If sums = TRUE, then find the sum scores. This is useful for regression with an intercept term

impute — impute="median" replaces missing values with the item medians, impute = "mean" replaces values with the mean response. impute="none" the subject's scores are based upon the average of the keyed, but non missing scores. impute = "none" is probably more appropriate for a large number of missing cases (e.g., SAPA data).

Details

Although meant for finding correlation weighted scores using the weights from bestScales, it also possible to use alternative weight matrices, such as those returned by the coefficients in lm.

Value

A data frame of scores.

Author(s)

William Revelle

See Also

bestScales and lmCor

Examples

```
#find the weights from a regression model and then apply them to a new set
#derivation of weights from the first 20 cases
model.lm <- lm(rating ~ complaints + privileges + learning,data=attitude[1:20,])
#or use lmCor to find the coefficents
model <- lmCor(rating ~ complaints + privileges +learning,data=attitude[1:20,],std=FALSE)

 #Apply these to a different set of data (the last 10 cases)
  #note that the regression coefficients need to be a matrix
 scores.lm <- scoreWtd(as.matrix(model.lm$coefficients),attitude[21:30,],sums=TRUE,std=FALSE)
scores <- scoreWtd(model$coefficients,attitude[21:30,],sums=TRUE,std=FALSE)
describe(scores)
```

---

scrub — *A utility for basic data cleaning and recoding. Changes values outside of minimum and maximum limits to NA.*

Description

A tedious part of data analysis is addressing the problem of miscoded data that need to be converted to NA or some other value. For a given data.frame or matrix, scrub will set all values of columns from=from to to=to that are less than a set (vector) of min values or more than a vector of max values to NA. Can also be used to do basic recoding of data for all values=isvalue to newvalue. Will also recode continuus variables into fewer categories. Will convert Nan, -Inf and Inf to NA

The length of the where, isvalue, and newvalues must either match, or be 1.

Usage

```
scrub(x, where, min, max,isvalue,newvalue, cuts=NULL)
```

Arguments

x	a data frame or matrix
where	The variables to examine. (Can be by name or by column number)
min	a vector of minimum values that are acceptable
max	a vector of maximum values that are acceptable
isvalue	a vector of values to be converted to newvalue (one per variable)
newvalue	a vector of values to replace those that match isvalue
cuts	The lower,and upper boundaries for recoding

Details

Solves a tedious problem that can be done directly but that is sometimes awkward. Will either replace specified values with NA or will recode to values within a range.

Value

The corrected data frame.

Author(s)

William Revelle

See Also

reverse.code, rescale for other simple utilities.

Examples

```
data(attitude)
x <- scrub(attitude,isvalue=55) #make all occurrences of 55 NA
x1 <- scrub(attitude, where=c(4,5,6), isvalue =c(30,40,50),
     newvalue = c(930,940,950)) #will do this for the 4th, 5th, and 6th variables
x2 <- scrub(attitude, where=c(4,4,4), isvalue =c(30,40,50),
            newvalue = c(930,940,950)) #will just do it for the 4th column
new <- scrub(attitude,1:3,cuts= c(10,40,50,60,100))  #change many values to fewer
#get rid of a complicated set of cases and replace with missing values
y <- scrub(attitude,where=2:4,min=c(20,30,40),max= c(120,110,100),isvalue= c(32,43,54))
y1 <- scrub(attitude,where="learning",isvalue=55,newvalue=999) #change a column by name
y2 <- scrub(attitude,where="learning",min=45,newvalue=999) #change a column by name

y3 <- scrub(attitude,where="learning",isvalue=c(45,48),
    newvalue=999) #change a column by name look for multiple values in that column
y4 <- scrub(attitude,where="learning",isvalue=c(45,48),
      newvalue= c(999,-999)) #change values in one column to one of two different things
```

SD *Find the Standard deviation for a vector, matrix, or data.frame - do not return error if there are no cases*

Description

Find the standard deviation of a vector, matrix, or data.frame. In the latter two cases, return the sd of each column. Unlike the sd function, return NA if there are no observations rather than throw an error.

Usage

```
SD(x, na.rm = TRUE)   #deprecated
```

Arguments

x	a vector, data.frame, or matrix
na.rm	na.rm is assumed to be TRUE

Details

Finds the standard deviation of a vector, matrix, or data.frame. Returns NA if no cases.

Just an adaptation of the stats:sd function to return the functionality found in R < 2.7.0 or R >= 2.8.0 Because this problem seems to have been fixed, SD will be removed eventually.

Value

The standard deviation

Note

Until R 2.7.0, sd would return a NA rather than an error if no cases were observed. SD brings back that functionality. Although unusual, this condition will arise when analyzing data with high rates of missing values. This function will probably be removed as 2.7.0 becomes outdated.

Author(s)

William Revelle

See Also

These functions use SD rather than sd: describe.by, skew, kurtosi

Examples

```
data(attitude)
apply(attitude,2,sd) #all complete
attitude[,1] <- NA
SD(attitude) #missing a column
describe(attitude)
```

---

sim *Functions to simulate psychological/psychometric data.*

Description

A number of functions in the psych package will generate simulated data with particular structures. The three documented here are for basic factor analysis simulations. These functions include

sim for a factor simplex
sim.simplex for a data simplex
sim.minor to simulate major and minor factors.

All three of these, and most of the remaining simulation functions return the simulated model, the true scores that generate the data, and the observed data.

Other simulations aimed at special item structures include the following which are are listed here in order to help find them.

Factor based models:

sim.structural a general simulation of structural models.

sim.circ for a circumplex or simple structure for two factors
sim.congeneric for a one factor factor congeneric model.
sim.dichot to simulate dichotomous items
sim.hierarchical to create a hierarchical factor model
sim.omega to test various examples of omega,
sim.item a more general item simulation (including the ability to simulate dichototomous or polytommous items).

simCor to generate sample correlation matrices from a target matrix.

sim.parallel to compare the efficiency of various ways of deterimining the number of factors.

Item Response Theory based models

sim.rasch to create simulated rasch data
sim.irt to create general 1 to 4 parameter IRT data by calling one of the following:
sim.npl 1 to 4 parameter logistic IRT
sim.npn 1 to 4 paramater normal IRT
sim.poly to create polytomous ideas by calling
sim.poly.npn 1-4 parameter polytomous normal theory items
sim.poly.npl 1-4 parameter polytomous logistic items

sim.poly.ideal which creates data following an ideal point or unfolding model by calling sim.poly.ideal.npn 1-4 parameter polytomous normal theory ideal point model or sim.poly.ideal.npl 1-4 parameter polytomous logistic ideal point model.

sim.anova for ANOVA and lm simulations

sim.VSS.

Most of these functions are separately documented and are listed here for ease of the help function. See each function for more detailed help.

Usage

```
sim(fx=NULL,Phi=NULL,fy=NULL,alpha=.8,lambda = 0,n=0,mu=NULL,raw=TRUE,threshold=NULL,
       items = FALSE, low=-2,high=2,cat=5)
sim.simplex(nvar =12, alpha=.8,lambda=0,beta=1,mu=NULL, n=0,threshold=NULL)
sim.minor(nvar=12,nfact=3,n=0,g=NULL,fbig=NULL,fsmall = c(-.2,.2),n.small=NULL,
    Phi=NULL, bipolar=TRUE, threshold=NULL,
     items = FALSE, low=-2,high=2,cat=5)
```

Arguments

fx	The measurement model for x. If NULL, a 4 factor model is generated
Phi	The structure matrix of the latent variables
fy	The measurement model for y
mu	The means structure for the fx factors, defaults to 0,1,... 3. Set to 0 if using thresholds.
n	Number of cases to simulate. If n=0 or NULL, the population matrix is returned. If n > 0, the latent and observed data are returned.
raw	if raw=TRUE, raw data are returned as well.
threshold	If raw=TRUE (n >0) and threshold is not NULL, convert the data to binary values, cut at threshold.
items	TRUE if simulating items, FALSE if simulating scales
low	Restrict the item difficulties to range from low to high
high	Restrict the item difficulties to range from low to high
cat	Number of categories when creating binary (2) or polytomous items
nvar	Number of variables for a simplex structure
nfact	Number of large factors to simulate in sim.minor,number of group factors in sim.general,sim.omega
g	General factor correlations in sim.general and general factor loadings in sim.omega and sim.minor
alpha	the base correlation for an autoregressive simplex
lambda	the trait component of a State Trait Autoregressive Simplex
beta	Test reliability of a STARS simplex

fbig	Factor loadings for the main factors. Default is a simple structure with loadings sampled from (.8,.6) for nvar/nfact variables and 0 for the remaining. If fbig is specified, then each factor has loadings sampled from it.
bipolar	if TRUE, then positive and negative loadings are generated from fbig
fsmall	nvar/2 small factors are generated with loadings sampled from e.g. (-.2,0,.2)
n.small	If NULL, then generate nvar/2 small factors, else generate n.small small factors

Details

Simulation of data structures is a very useful tool in psychometric research and teaching. By knowing "truth" it is possible to see how well various algorithms can capture it. For a much longer discussion of the use of simulation in psychometrics, see the accompany vignettes.

The simulations documented here are the core set of functions. Others are documented in other help files.

sim simulates one (fx) or two (fx and fy) factor structures where both fx and fy respresent factor loadings of variables. The use of fy is particularly appropriate for simulating sem models. This is better documentated in the help for sim.structural.

The code for sim has been enhanced (10/21/23) to simulate items as well as continuous variables. This matches the code in sim.structural.

Perhaps the easist simulation to understand is just sim. A factor model (fx) and perhaps fy with intercorrelations between the two factor sets of Phi. This will produce a correlation matrix R = fx' phi fy. Factors can differ in their mean values by specifying mu.

With the addition of the treshold parameter, sim will generate dichotomous items where values < threshold become 0, greater become 1. If threshold is a vector less than nvar, then values are sampled from threshold. If the length of threshold = nvar, then each item is dichotomized at threshold.

With the adddition of the items=TRUE option, continuous observed variables are transformed into discrete categories.

The default values for sim.structure is to generate a 4 factor, 12 variable data set with a simplex structure between the factors. This, and the simplex of items (sim.simplex) can also be converted in a STARS model with an autoregressive component (alpha) and a stable trait component (lambda).

Two data structures that are particular challenges to exploratory factor analysis are the simplex structure and the presence of minor factors. Simplex structures, sim.simplex, will typically occur in developmental or learning contexts and have a correlation structure of r between adjacent variables and r^n for variables n apart. Although just one latent variable (r) needs to be estimated, the structure will have nvar-1 factors.

An alternative version of the simplex is the State-Trait-Auto Regressive Structure (STARS) which has both a simplex state structure, with autoregressive path alpha and a trait structure with path lambda. This simulated in sim.simplex by specifying a non-zero lambda value.

Many simulations of factor structures assume that except for the major factors, all residuals are normally distributed around 0. An alternative, and perhaps more realistic situation, is that the there are a few major (big) factors and many minor (small) factors. For a nice discussion of this situation, see Maccallum, Browne and Cai (2007). The challenge is thus to identify the major factors. sim.minor generates such structures. The structures generated can be thought of as having a major factor structure with some small correlated residuals. To make these simulations complete, the possibility of a general factor is considered. For simplicity, sim.minor allows one to specify a

set of loadings to be sampled from for g, fmajor and fminor. Alternatively, it is possible to specify the complete factor matrix. By default, the nvar/2 minor factors are given loadings of +/- .2. If the major factors are very big, this leads to impossible models. a warning to try again is issued.

Another structure worth considering is direct modeling of a general factor with several group factors. This is done using `sim.general`.

Although coefficient ω_h is a very useful indicator of the general factor saturation of a unifactorial test (one with perhaps several sub factors), it has problems with the case of multiple, independent factors. In this situation, one of the factors is labelled as "general" and the omega estimate is too large. This situation may be explored using the `sim.omega` function with general left as NULL. If there is a general factor, then results from `sim.omega` suggests that omega estimated either from EFA or from SEM does a pretty good job of identifying it but that the EFA approach using Schmid-Leiman transformation is somewhat more robust than the SEM approach.

The four irt simulations, `sim.rasch`, `sim.irt`, `sim.npl` and `sim.npn` simulate dichotomous items following the Item Response model. `sim.irt` just calls either `sim.npl` (for logistic models) or `sim.npn` (for normal models) depending upon the specification of the model.

The logistic model is

$$P(i,j) = \gamma + \frac{\zeta - \gamma}{1 + e^{\alpha(\delta - \theta)}}$$

where γ is the lower asymptote or guesssing parameter, ζ is the upper asymptote (normally 1), α is item discrimination and δ is item difficulty. For the 1 Paramater Logistic (Rasch) model, gamma=0, zeta=1, alpha=1 and item difficulty is the only free parameter to specify.

For the 2PL and 2PN models, a = α and d = δ are specified.
For the 3PL or 3PN models, items also differ in their guessing parameter c =γ.
For the 4PL and 4PN models, the upper asymptote, z= ζ is also specified.
(Graphics of these may be seen in the demonstrations for the `logistic` function.)

The normal model (irt.npn) calculates the probability using pnorm instead of the logistic function used in irt.npl, but the meaning of the parameters are otherwise the same. With the a = α parameter = 1.702 in the logistic model the two models are practically identical.

In parallel to the dichotomous IRT simulations are the poly versions which simulate polytomous item models. They have the additional parameter of how many categories to simulate. In addition, the `sim.poly.ideal` functions will simulate an ideal point or unfolding model in which the response probability varies by the distance from each subject's ideal point. Some have claimed that this is a more appropriate model of the responses to personality questionnaires. It will lead to simplex like structures which may be fit by a two factor model. The middle items form one factor, the extreme a bipolar factor.

By default, the theta parameter is created in each function as normally distributed with mean mu=0 and sd=1. In the case where you want to specify the theta to be equivalent from another simulation or fixed for a particular experimental condition, either take the theta object from the output of a previous simulation, or create it using whatever properties are desired.

The previous functions all assume one latent trait. Alternatively, we can simulate dichotomous or polytomous items with a particular structure using the sim.poly.mat function. This takes as input the population correlation matrix, the population marginals, and the sample size. It returns categorical items with the specified structure.

Other simulation functions in psych that are documented separately includ:

sim.structure A function to combine a measurement and structural model into one data matrix. Useful for understanding structural equation models. Combined with structure.diagram to see the proposed structure.

sim.congeneric A function to create congeneric items/tests for demonstrating classical test theory. This is just a special case of sim.structure.

sim.hierarchical A function to create data with a hierarchical (bifactor) structure. Compare this with directly simulating a bifactor model

sim.item A function to create items that either have a simple structure or a circumplex structure.

sim.circ Create data with a circumplex structure.

sim.dichot Create dichotomous item data with a simple or circumplex structure.

sim.minor Create a factor structure for nvar variables defined by nfact major factors and nvar/2 “minor" factors for n observations.

Although the standard factor model assumes that K major factors (K « nvar) will account for the correlations among the variables

$$R = FF' + U^2$$

where R is of rank P and F is a P x K matrix of factor coefficients and U is a diagonal matrix of uniquenesses. However, in many cases, particularly when working with items, there are many small factors (sometimes referred to as correlated residuals) that need to be considered as well. This leads to a data structure such that

$$R = FF' + MM' + U^2$$

where R is a P x P matrix of correlations, F is a P x K factor loading matrix, M is a P x P/2 matrix of minor factor loadings, and U is a diagonal matrix (P x P) of uniquenesses.

Such a correlation matrix will have a poor χ^2 value in terms of goodness of fit if just the K factors are extracted, even though for all intents and purposes, it is well fit.

sim.minor will generate such data sets with big factors with loadings of .6 to .8 and small factors with loadings of -.2 to .2. These may both be adjusted.

sim.parallel Create a number of simulated data sets using sim.minor to show how parallel analysis works. The general observation is that with the presence of minor factors, parallel analysis is probably best done with component eigen values rather than factor eigen values, even when using the factor model.

sim.anova Simulate a 3 way balanced ANOVA or linear model, with or without repeated measures. Useful for teaching research methods and generating teaching examples.

sim.multilevel To understand some of the basic concepts of multilevel modeling, it is useful to create multilevel structures. The correlations of aggregated data is sometimes called an ’ecological correlation’. That group level and individual level correlations are independent makes such inferences problematic. This simulation allows for demonstrations that correlations within groups do not imply, nor are implied by, correlations between group means.

Value

model	The model based correlation matrix (FF’)
reliability	Model based reliability

r	Observed (sample based) correlation matrix
theta	True scores generating the data
N	Sample size
fload	The factor model (F + S) generating the model

Note

The theta values are "true scores" and may be compared to the factor analytic based factor score estimates of the data. This is a useful way to understand factor indeterminancy.

Author(s)

William Revelle

References

MacCallum, Robert C. and Browne, Michael W. and Cai, Li (2007) Factor analysis models as approximations. Factor analysis at 100: Historical developments and future directions. (Cudeck and MacCallum Eds). Lawrence Erlbaum Associates Publishers.

Revelle, W. (in preparation) An Introduction to Psychometric Theory with applications in R. Springer. at https://personality-project.org/r/book/

See Also

See above

Examples

```
simplex <- sim.simplex() #create the default simplex structure
lowerMat(simplex) #the correlation matrix
#create a congeneric matrix
congeneric <- sim.congeneric()
lowerMat(congeneric)
R <- sim.hierarchical()
lowerMat(R)
#now simulate categorical items with the hierarchical factor structure.
#Let the items be dichotomous with varying item difficulties.
marginals = matrix(c(seq(.1,.9,.1),seq(.9,.1,-.1)),byrow=TRUE,nrow=2)
X <- sim.poly.mat(R=R,m=marginals,n=1000)
lowerCor(X) #show the raw correlations
#lowerMat(tetrachoric(X)$rho) # show the tetrachoric correlations (not run)
#generate a structure
fx <- matrix(c(.9,.8,.7,rep(0,6),c(.8,.7,.6)),ncol=2)
fy <- matrix(c(.6,.5,.4),ncol=1)

Phi <- matrix(c(1,0,.5,0,1,.4,0,0,0),ncol=3)
R <- sim.structure(fx,Phi,fy)
cor.plot(R$model) #show it graphically

simp <- sim.simplex()
```

```
#show the simplex structure using cor.plot
cor.plot(simp,colors=TRUE,main="A simplex structure")
#Show a STARS model
simp <- sim.simplex(alpha=.8,lambda=.4)
#show the simplex structure using cor.plot
cor.plot(simp,colors=TRUE,main="State Trait Auto Regressive Simplex" )

dichot.sim <- sim.irt()  #simulate 5 dichotomous items
poly.sim <- sim.poly(theta=dichot.sim$theta)  #simulate 5 polytomous items that correlate
  #with the dichotomous items

  #simulate a bifactor model (compare with sim.hierarchical)

fx <- matrix(c(rep(.5,9),.8,.7,.6,rep(0,9),.8,.7,.6,rep(0,9),.8,.7,.6), ncol=4)
fx   #show the structure
bi <- sim(fx=fx,n=1000)
lowerMat(bi$model)
```

`sim.anova` *Simulate a 3 way balanced ANOVA or linear model, with or without repeated measures.*

Description

For teaching basic statistics, it is useful to be able to generate examples suitable for analysis of variance or simple linear models. sim.anova will generate the design matrix of three independent variables (IV1, IV2, IV3) with an arbitrary number of levels and effect sizes for each main effect and interaction. IVs can be either continuous or categorical and can have linear or quadratic effects. Either a single dependent variable or multiple (within subject) dependent variables are generated according to the specified model. The repeated measures are assumed to be tau equivalent with a specified reliability.

Usage

```
sim.anova(es1 = 0, es2 = 0, es3 = 0, es12 = 0, es13 = 0,
    es23 = 0, es123 = 0, es11=0,es22=0, es33=0,n = 2,n1 = 2, n2 = 2, n3 = 2,
    within=NULL,r=.8,factors=TRUE,center = TRUE,std=TRUE)
```

Arguments

`es1`	Effect size of IV1
`es2`	Effect size of IV2
`es3`	Effect size of IV3
`es12`	Effect size of the IV1 x IV2 interaction
`es13`	Effect size of the IV1 x IV3 interaction
`es23`	Effect size of the IV2 x IV3 interaction
`es123`	Effect size of the IV1 x IV2 * IV3 interaction

es11	Effect size of the quadratric term of IV1
es22	Effect size of the quadratric term of IV2
es33	Effect size of the quadratric term of IV3
n	Sample size per cell (if all variables are categorical) or (if at least one variable is continuous), the total sample size
n1	Number of levels of IV1 (0) if continuous
n2	Number of levels of IV2
n3	Number of levels of IV3
within	if not NULL, then within should be a vector of the means of any repeated measures.
r	the correlation between the repeated measures (if they exist). This can be thought of as the reliablility of the measures.
factors	report the IVs as factors rather than numeric
center	center=TRUE provides orthogonal contrasts, center=FALSE adds the minimum value + 1 to all contrasts
std	Standardize the effect sizes by standardizing the IVs

Details

A simple simulation for teaching about ANOVA, regression and reliability. A variety of demonstrations of the relation between anova and lm can be shown.

The default is to produce categorical IVs (factors). For more than two levels of an IV, this will show the difference between the linear model and anova in terms of the comparisons made.

The within vector can be used to add congenerically equivalent dependent variables. These will have intercorrelations (reliabilities) of r and means as specified as values of within.

To demonstrate the effect of centered versus non-centering, make factors = center=FALSE. The default is to center the IVs. By not centering them, the lower order effects will be incorrect given the higher order interaction terms.

Value

y.df is a data.frame of the 3 IV values as well as the DV values.

IV1 ... IV3	Independent variables 1 ... 3
DV	If there is a single dependent variable
DV.1 ... DV.n	If within is specified, then the n within subject dependent variables

Author(s)

William Revelle

See Also

The general set of simulation functions in the psych package sim

Examples

```
set.seed(42)
data.df <- sim.anova(es1=1,es2=-.5,es13=1)  # two main effect and one interaction
psych::describe(data.df)
pairs.panels(data.df)   #show how the design variables are orthogonal
#
data.df <- char2numeric(data.df,flag=FALSE)

summary(lm(DV~IV1*IV2*IV3,data=data.df))

summary(aov(DV~IV1*IV2*IV3,data=data.df))
lmCor(DV~IV1*IV2*IV3,data=data.df, std=FALSE)
set.seed(42)
 #demonstrate the effect of not centering the data on the regression
data.df <- sim.anova(es1=1,es2=.5,es13=1,center=FALSE)  #
psych::describe(data.df)
#
#this one is incorrect, because the IVs are not centered
data.df <- char2numeric(data.df,flag=FALSE)
summary(lm(DV~IV1*IV2*IV3,data=data.df))
data.df <- char2numeric(data.df,flag=FALSE)

summary(aov(DV~IV1*IV2*IV3,data=data.df)) #compare with the lm model
#but lmCor by default zero centers which works
lmCor(DV~IV1*IV2*IV3,data=data.df)
#now examine multiple levels and quadratic terms
set.seed(42)
data.df <- sim.anova(es1=1,es13=1,n2=3,n3=4,es22=1)
summary(lm(DV~IV1*IV2*IV3,data=data.df))
summary(aov(DV~IV1*IV2*IV3,data=data.df))
pairs.panels(data.df)
#
data.df <- sim.anova(es1=1,es2=-.5,within=c(-1,0,1),n=10)
pairs.panels(data.df)
```

`sim.congeneric` *Simulate a congeneric data set with or without minor factors*

Description

Classical Test Theory (CTT) considers four or more tests to be congenerically equivalent if all tests may be expressed in terms of one factor and a residual error. Parallel tests are the special case where (usually two) tests have equal factor loadings. Tau equivalent tests have equal factor loadings but may have unequal errors. Congeneric tests may differ in both factor loading and error variances. Minor factors may be added as systematic but trivial disturbances

Usage

```
sim.congeneric(loads = c(0.8, 0.7, 0.6, 0.5),N = NULL,  err=NULL, short = TRUE,
         categorical=FALSE, low=-3,high=3,cuts=NULL,minor=FALSE,fsmall = c(-.2,.2))
```

Arguments

N	How many subjects to simulate. If NULL, return the population model
loads	A vector of factor loadings for the tests
err	A vector of error variances – if NULL then error = 1 - loading 2
short	short=TRUE: Just give the test correlations, short=FALSE, report observed test scores as well as the implied pattern matrix
categorical	continuous or categorical (discrete) variables.
low	values less than low are forced to low
high	values greater than high are forced to high
cuts	If specified, and categorical = TRUE, will cut the resulting continuous output at the value of cuts
minor	Should n/2 minor factors be added (see Maccallum and Tucker, 1991)
fsmall	nvar/2 small factors are generated with loadings sampled from fsmall e.g. (-.2,0,.2)

Details

When constructing examples for reliability analysis, it is convenient to simulate congeneric data structures. These are the most simple of item structures, having just one factor. Mainly used for a discussion of reliability theory as well as factor score estimates.

Maccallum and Tucker (1991) suggest that factor models should include minor factors, that at not random error but unspecifed by the basic model. This option has been added in November, 2022.

The implied covariance matrix is just pattern %*% t(pattern).

Value

model	The implied population correlation matrix if N=NULL or short=FALSE, otherwise the sample correlation matrix
pattern	The pattern matrix implied by the loadings and error variances
r	The sample correlation matrix for long output
observed	a matrix of test scores for n tests
latent	The latent trait and error scores

Author(s)

William Revelle

References

Revelle, W. (in prep) An introduction to psychometric theory with applications in R. To be published by Springer. (working draft available at https://personality-project.org/r/book/

MacCallum, R. C., & Tucker, L. R. (1991). Representing sources of error in the common-factormodel: Implications for theory and practice. Psychological Bulletin, 109(3), 502-511.

See Also

item.sim for other simulations, fa for an example of factor scores, irt.fa and polychoric for the treatment of item data with discrete values.

Examples

```
test <- sim.congeneric(c(.9,.8,.7,.6))   #just the population matrix
test <- sim.congeneric(c(.9,.8,.7,.6),N=100)   # a sample correlation matrix
test <- sim.congeneric(short=FALSE, N=100)
round(cor(test$observed),2) # show  a congeneric correlation matrix
f1=fa(test$observed,scores=TRUE)
round(cor(f1$scores,test$latent),2)
     #factor score estimates are correlated with but not equal to the factor scores
set.seed(42)
#500 responses to 4 discrete items
items <- sim.congeneric(N=500,short=FALSE,low=-2,high=2,categorical=TRUE)
d4 <- irt.fa(items$observed)  #item response analysis of congeneric measures
```

---

sim.hierarchical *Create a population or sample correlation matrix, perhaps with hierarchical structure.*

Description

Create a population orthogonal or hierarchical correlation matrix from a set of factor loadings and factor intercorrelations. Samples of size n may be then be drawn from this population. Return either the sample data, sample correlations, or population correlations. This is used to create sample data sets for instruction and demonstration.

Usage

```
sim.hierarchical(gload=NULL, fload=NULL, n = 0, raw = TRUE,mu = NULL,
   categorical=FALSE, low=-3,high=3,threshold=NULL,minor=FALSE, fsmall=c(-.2,.2) )
sim.bonds(nvar=9,nf=NULL, loads=c(0,0,.5,.6),validity=.8)

make.hierarchical(gload=NULL, fload=NULL, n = 0, raw = FALSE)  #deprecated
```

Arguments

gload	Loadings of group factors on a general factor. Defaults to c(.9,.8,.7)
fload	Loadings of items on the group factors. Defaults to matrix(c(.8,.7,.6,rep(0,9),.7,.6,.5,rep(0,9),.6,.5,.4), ncol=3)
n	Number of subjects to generate: N=0 => population values
raw	raw=TRUE, report the raw data, raw=FALSE, report the sample correlation matrix.
mu	means for the individual variables
low	lower cutoff for categorical data
categorical	If True, then create categorical data
threshold	If categorical is TRUE, and binary output is desired, what is the threshold to convert continuous scores into 0/1. May be a vector.
minor	Add in minor noise factors to the basic model
fsmall	If minor, what is the size of the small/noise factor loadings.
high	Upper cuttoff for categorical data
nvar	Number of variables to simulate
loads	A vector of loadings that will be sampled (rowwise) to define the factors
validity	The factor loadings of 'pure' measures of the factor.
nf	Number of factors to generate in sim.bonds

Details

Many personality and cognitive tests have a hierarchical factor structure. For demonstration purposes, it is useful to be able to create such matrices, either with population values, or sample values.

Given a matrix of item factor loadings (fload) and of loadings of these factors on a general factor (gload), we create a population correlation matrix by using the general factor law $R \approx F'\theta F + U^2$ where $\theta = g'g$

The default is to return population correlation matrices. Sample correlation matrices are generated if n > 0. Raw data are returned if raw = TRUE.

The default values for gload and fload create a data matrix discussed by Jensen and Weng, 1994.

In order to properly simulate polytomous items, the categorical option will round continuous scores into integers. To simulate dichotomous items, the threshold vector may be used to specify the value at which the continuous values are cut into 0/1. If the length of threshold is less than the number of variables, the vector will be sampled (with replacement) to fill it out.

Although written to create hierarchical structures, if the gload matrix is all 0, then a non-hierarchical structure will be generated. Varying the size of the gloading, we can simulate the amount of g-saturation on a test.

Yet another model is that of Godfrey H. Thomson (1916) who suggested that independent bonds could produce the same factor structure as a g factor model. This is simulated in `sim.bonds`. Compare the `omega` solutions for a `sim.hierarchical` with a `sim.bonds` model. Both produce reasonable values of omega, although the one was generated without a general factor.

Value

r	a matrix of correlations
model	The population correlation matrix
observed	The simulated data matrix with the defined structure
theta	The latent factor scores used to generate the data. Compare how these correlate with the observed data with the results from omega.
sl	The Schmid Leiman transformed factor loadings. These may be used to test factor scoring problem.

Author(s)

William Revelle

References

https://personality-project.org/r/r.omega.html
Jensen, A.R., Weng, L.J. (1994) What is a Good g? Intelligence, 18, 231-258.

Godfrey H. Thomson (1916) A hierarchy without a general factor, British Journal of Psychology, 8, 271-281.

See Also

omega, schmid, ICLUST, VSS for ways of analyzing these data. Also see sim.structure to simulate a variety of structural models (e.g., multiple correlated factor models).

Examples

```
gload <-  gload<-matrix(c(.9,.8,.7),nrow=3)    # a higher order factor matrix
fload <-matrix(c(                    #a lower order (oblique) factor matrix
           .8,0,0,
           .7,0,.0,
           .6,0,.0,
            0,.7,.0,
            0,.6,.0,
            0,.5,0,
            0,0,.6,
            0,0,.5,
            0,0,.4),   ncol=3,byrow=TRUE)

jensen <- sim.hierarchical(gload,fload)    #the test set used by omega
round(jensen,2)
set.seed(42) #for reproducible results
jensen <-  sim.hierarchical(n=10000,categorical =TRUE, threshold =c(-1,0,1))
           #use the same gload and fload values, but produce the data
#items have three levels of difficulty
#Compare factor scores using the sl model with those that generated the data
lowerCor(jensen$theta) #the correlations of the factors
fs <- factor.scores(jensen$observed, jensen$sl)  #find factor scores from the data
lowerCor(fs$scores) #these are now correlated
```

```
cor2(fs$scores,jensen$theta)  #correlation with the generating factors


#compare this to a simulation of the bonds model
set.seed(42)
R <- sim.bonds()
R$R

#simulate a non-hierarchical structure
fload <- matrix(c(c(c(.9,.8,.7,.6),rep(0,20)),c(c(.9,.8,.7,.6),rep(0,20)),
    c(c(.9,.8,.7,.6),rep(0,20)),c(c(c(.9,.8,.7,.6),rep(0,20)),c(.9,.8,.7,.6))),ncol=5)
gload <- matrix(rep(0,5))
five.factor <- sim.hierarchical(gload,fload,500,TRUE) #create sample data set
#do it again with a hierachical structure
gload <- matrix(rep(.7,5)  )
five.factor.g <- sim.hierarchical(gload,fload,500,TRUE) #create sample data set
#compare these two with omega
#not run
#om.5 <- omega(five.factor$observed,5)
#om.5g <- omega(five.factor.g$observed,5)
```

---

sim.irt *Functions to simulate psychological/psychometric data.*

Description

A number of functions in the psych package will generate simulated data with particular structures. These functions include sim for a factor simplex, and sim.simplex for a data simplex, sim.circ for a circumplex structure, sim.congeneric for a one factor factor congeneric model, sim.dichot to simulate dichotomous items, sim.hierarchical to create a hierarchical factor model, sim.item a more general item simulation, sim.minor to simulate major and minor factors, sim.omega to test various examples of omega, sim.parallel to compare the efficiency of various ways of deterimining the number of factors, sim.rasch to create simulated rasch data, sim.irt to create general 1 to 4 parameter IRT data by calling sim.npl 1 to 4 parameter logistic IRT or sim.npn 1 to 4 paramater normal IRT, sim.poly to create polytomous ideas by calling sim.poly.npn 1-4 parameter polytomous normal theory items or sim.poly.npl 1-4 parameter polytomous logistic items, and sim.poly.ideal which creates data following an ideal point or unfolding model by calling sim.poly.ideal.npn 1-4 parameter polytomous normal theory ideal point model or sim.poly.ideal.npl 1-4 parameter polytomous logistic ideal point model.

sim.structural a general simulation of structural models, and sim.anova for ANOVA and lm simulations, and sim.VSS. Some of these functions are separately documented and are listed here for ease of the help function. See each function for more detailed help.

Usage

```
sim.rasch(nvar = 5,n = 500, low=-3,high=3,d=NULL, a=1,mu=0,sd=1)
sim.irt(nvar = 5, n = 500, low=-3, high=3, a=NULL,c=0, z=1,d=NULL, mu=0,sd=1,
  mod="logistic",theta=NULL)
```

```
sim.npl(nvar = 5, n = 500, low=-3,high=3,a=NULL,c=0,z=1,d=NULL,mu=0,sd=1,theta=NULL)
sim.npn(nvar = 5, n = 500, low=-3,high=3,a=NULL,c=0,z=1,d=NULL,mu=0,sd=1,theta=NULL)
sim.poly(nvar = 5 ,n = 500,low=-2,high=2,a=NULL,c=0,z=1,d=NULL,
    mu=0,sd=1,cat=5,mod="logistic",theta=NULL)
sim.poly.npn(nvar = 5 ,n = 500,low=-2,high=2,a=NULL,c=0,z=1,d=NULL,  mu=0, sd=1,
     cat=5,theta=NULL)
sim.poly.npl(nvar = 5 ,n = 500,low=-2,high=2,a=NULL,c=0,z=1,d=NULL,  mu=0, sd=1,
cat=5,theta=NULL)
sim.poly.ideal(nvar = 5 ,n = 500,low=-2,high=2,a=NULL,c=0,z=1,d=NULL,
   mu=0,sd=1,cat=5,mod="logistic")
sim.poly.ideal.npn(nvar = 5,n = 500,low=-2,high=2,a=NULL,c=0,z=1,d=NULL, mu=0,sd=1,cat=5)
sim.poly.ideal.npl(nvar = 5,n = 500,low=-2,high=2,a=NULL,c=0,z=1,d=NULL,
      mu=0,sd=1,cat=5,theta=NULL)
sim.poly.mat(R,m,n)
```

Arguments

n	Number of cases to simulate
mu	The means for the items (if not 0)
nvar	Number of variables for a simplex structure
low	lower difficulty for sim.rasch or sim.irt
high	higher difficulty for sim.rasch or sim.irt
a	if not specified as a vector, the descrimination parameter a = α will be set to 1.0 for all items
d	if not specified as a vector, item difficulties (d = δ) will range from low to high
c	the gamma parameter: if not specified as a vector, the guessing asymptote is set to 0
z	the zeta parameter: if not specified as a vector, set to 1
sd	the standard deviation for the underlying latent variable in the irt simulations
mod	which IRT model to use, mod="logistic" simulates a logistic function, otherwise, a normal function
cat	Number of categories to simulate in sim.poly. If cat=2, then this is the same as simulating t/f items and sim.poly is functionally equivalent to sim.irt
theta	The underlying latent trait value for each simulated subject
R	A correlation matrix to be simulated using the sim.poly.mat function
m	The matrix of marginals for all the items

Details

Simulation of data structures is a very useful tool in psychometric research and teaching. By knowing “truth" it is possible to see how well various algorithms can capture it. For a much longer discussion of the use of simulation in psychometrics, see the accompany vignettes.

The simulations documented here are a miscellaneous set of functions that will be documented in other help files eventually.

The default values for sim.structure is to generate a 4 factor, 12 variable data set with a simplex structure between the factors. This, and the simplex of items (sim.simplex) can also be converted in a STARS model with an autoregressive component (alpha) and a stable trait component (lambda).

Two data structures that are particular challenges to exploratory factor analysis are the simplex structure and the presence of minor factors. Simplex structures sim.simplex will typically occur in developmental or learning contexts and have a correlation structure of r between adjacent variables and r^n for variables n apart. Although just one latent variable (r) needs to be estimated, the structure will have nvar-1 factors.

An alternative version of the simplex is the State-Trait-Auto Regressive Structure (STARS) which has both a simplex state structure, with autoregressive path alpha and a trait structure with path lambda. This simulated in sim.simplex by specifying a non-zero lambda value.

Many simulations of factor structures assume that except for the major factors, all residuals are normally distributed around 0. An alternative, and perhaps more realistic situation, is that the there are a few major (big) factors and many minor (small) factors. The challenge is thus to identify the major factors. sim.minor generates such structures. The structures generated can be thought of as havinga a major factor structure with some small correlated residuals. To make these simulations complete, the possibility of a general factor is considered. For simplicity, sim.minor allows one to specify a set of loadings to be sampled from for g, fmajor and fminor. Alternatively, it is possible to specify the complete factor matrix.

Another structure worth considering is direct modeling of a general factor with several group factors. This is done using sim.general.

Although coefficient ω is a very useful indicator of the general factor saturation of a unifactorial test (one with perhaps several sub factors), it has problems with the case of multiple, independent factors. In this situation, one of the factors is labelled as "general" and the omega estimate is too large. This situation may be explored using the sim.omega function with general left as NULL. If there is a general factor, then results from sim.omega suggests that omega estimated either from EFA or from SEM does a pretty good job of identifying it but that the EFA approach using Schmid-Leiman transformation is somewhat more robust than the SEM approach.

The four irt simulations, sim.rasch, sim.irt, sim.npl and sim.npn, simulate dichotomous items following the Item Response model. sim.irt just calls either sim.npl (for logistic models) or sim.npn (for normal models) depending upon the specification of the model.

The logistic model is

$$P(i,j) = \gamma + \frac{\zeta - \gamma}{1 + e^{\alpha(\delta - \theta)}}$$

where γ is the lower asymptote or guesssing parameter, ζ is the upper asymptote (normally 1), α is item discrimination and δ is item difficulty. For the 1 Paramater Logistic (Rasch) model, gamma=0, zeta=1, alpha=1 and item difficulty is the only free parameter to specify.

For the 2PL and 2PN models, a = α and d = δ are specified.
For the 3PL or 3PN models, items also differ in their guessing parameter c =γ.
For the 4PL and 4PN models, the upper asymptote, z= ζ is also specified.
(Graphics of these may be seen in the demonstrations for the logistic function.)

The normal model (irt.npn calculates the probability using pnorm instead of the logistic function used in irt.npl, but the meaning of the parameters are otherwise the same. With the a = α parameter = 1.702 in the logistic model the two models are practically identical.

In parallel to the dichotomous IRT simulations are the poly versions which simulate polytomous item models. They have the additional parameter of how many categories to simulate. In addition, the sim.poly.ideal functions will simulate an ideal point or unfolding model in which the response probability varies by the distance from each subject's ideal point. Some have claimed that this is a more appropriate model of the responses to personality questionnaires. It will lead to simplex like structures which may be fit by a two factor model. The middle items form one factor, the extreme a bipolar factor.

By default, the theta parameter is created in each function as normally distributed with mean mu=0 and sd=1. In the case where you want to specify the theta to be equivalent from another simulation or fixed for a particular experimental condition, either take the theta object from the output of a previous simulation, or create it using whatever properties are desired.

The previous functions all assume one latent trait. Alternatively, we can simulate dichotomous or polytomous items with a particular structure using the sim.poly.mat function. This takes as input the population correlation matrix, the population marginals, and the sample size. It returns categorical items with the specified structure.

Other simulation functions in psych are:

sim.structure A function to combine a measurement and structural model into one data matrix. Useful for understanding structural equation models. Combined with structure.diagram to see the proposed structure.

sim.congeneric A function to create congeneric items/tests for demonstrating classical test theory. This is just a special case of sim.structure.

sim.hierarchical A function to create data with a hierarchical (bifactor) structure.

sim.item A function to create items that either have a simple structure or a circumplex structure.

sim.circ Create data with a circumplex structure.

sim.dichot Create dichotomous item data with a simple or circumplex structure.

sim.minor Create a factor structure for nvar variables defined by nfact major factors and nvar/2 "minor" factors for n observations.

Although the standard factor model assumes that K major factors (K « nvar) will account for the correlations among the variables

$$R = FF' + U^2$$

where R is of rank P and F is a P x K matrix of factor coefficients and U is a diagonal matrix of uniquenesses. However, in many cases, particularly when working with items, there are many small factors (sometimes referred to as correlated residuals) that need to be considered as well. This leads to a data structure such that

$$R = FF' + MM' + U^2$$

where R is a P x P matrix of correlations, F is a P x K factor loading matrix, M is a P x P/2 matrix of minor factor loadings, and U is a diagonal matrix (P x P) of uniquenesses.

Such a correlation matrix will have a poor χ^2 value in terms of goodness of fit if just the K factors are extracted, even though for all intents and purposes, it is well fit.

sim.minor will generate such data sets with big factors with loadings of .6 to .8 and small factors with loadings of -.2 to .2. These may both be adjusted.

sim.parallel Create a number of simulated data sets using sim.minor to show how parallel analysis works. The general observation is that with the presence of minor factors, parallel analysis is probably best done with component eigen values rather than factor eigen values, even when using the factor model.

sim.anova Simulate a 3 way balanced ANOVA or linear model, with or without repeated measures. Useful for teaching research methods and generating teaching examples.

sim.multilevel To understand some of the basic concepts of multilevel modeling, it is useful to create multilevel structures. The correlations of aggregated data is sometimes called an 'ecological correlation'. That group level and individual level correlations are independent makes such inferences problematic. This simulation allows for demonstrations that correlations within groups do not imply, nor are implied by, correlations between group means.

Author(s)

William Revelle

References

Revelle, W. (in preparation) An Introduction to Psychometric Theory with applications in R. Springer. at https://personality-project.org/r/book/

See Also

See above

Examples

```
simplex <- sim.simplex() #create the default simplex structure
lowerMat(simplex) #the correlation matrix
#create a congeneric matrix
congeneric <- sim.congeneric()
lowerMat(congeneric)
R <- sim.hierarchical()
lowerMat(R)
#now simulate categorical items with the hierarchical factor structure.
#Let the items be dichotomous with varying item difficulties.
marginals = matrix(c(seq(.1,.9,.1),seq(.9,.1,-.1)),byrow=TRUE,nrow=2)
X <- sim.poly.mat(R=R,m=marginals,n=1000)
lowerCor(X) #show the raw correlations
#lowerMat(tetrachoric(X)$rho) # show the tetrachoric correlations (not run)
#generate a structure
fx <- 'f1 =~ .9*V1 + .8*V2 + .7*V3
      f2 =~ .8*V4 + .7*V5 + .6*V6'
#lavaan syntax to replace
#fx <- matrix(c(.9,.8,.7,rep(0,6),c(.8,.7,.6)),ncol=2)
fy <- 'f3 =~ .6*V7 + .5*V8 + .4*V9'
#fy <- as.matrix(c(.6,.5,.4))
Phi <- matrix(c(1,0,.5,0,1,.4,0,0,0),ncol=3)
R <- sim.structure(fx,Phi,fy)
cor.plot(R$model) #show it graphically
```

```
simp <- sim.simplex()
#show the simplex structure using cor.plot
cor.plot(simp,colors=TRUE,main="A simplex structure")
#Show a STARS model
simp <- sim.simplex(alpha=.8,lambda=.4)
#show the simplex structure using cor.plot
cor.plot(simp,colors=TRUE,main="State Trait Auto Regressive Simplex" )

dichot.sim <- sim.irt()  #simulate 5 dichotomous items
poly.sim <- sim.poly(theta=dichot.sim$theta)  #simulate 5 polytomous items that correlate
  #with the dichotomous items
```

`sim.item` *Generate simulated data structures for circumplex, spherical, or simple structure*

Description

Rotations of factor analysis and principal components analysis solutions typically try to represent correlation matrices as simple structured. An alternative structure, appealing to some, is a circumplex structure where the variables are uniformly spaced on the perimeter of a circle in a two dimensional space. Generating simple structure and circumplex data is straightforward, and is useful for exploring alternative solutions to affect and personality structure. A generalization to 3 dimensional (spherical) data is straightforward.

Usage

```
sim.item(nvar = 72, nsub = 500, circum = FALSE, xloading = 0.6, yloading = 0.6,
 gloading = 0, xbias = 0, ybias = 0, categorical = FALSE, low = -3, high = 3,
 truncate = FALSE, threshold=NULL)
sim.circ(nvar = 72, nsub = 500, circum = TRUE, xloading = 0.6, yloading = 0.6,
  gloading = 0, xbias = 0, ybias = 0, categorical = FALSE, low = -3, high = 3,
  truncate = FALSE, cutpoint = 0)
sim.dichot(nvar = 72, nsub = 500, circum = FALSE, xloading = 0.6, yloading = 0.6,
    gloading = 0, xbias = 0, ybias = 0, low = 0, high = 0)
item.dichot(nvar = 72, nsub = 500, circum = FALSE, xloading = 0.6, yloading = 0.6,
  gloading = 0, xbias = 0, ybias = 0, low = 0, high = 0)
sim.spherical(simple=FALSE, nx=7,ny=12 ,nsub = 500,  xloading =.55, yloading = .55,
   zloading=.55, gloading=0, xbias=0,  ybias = 0, zbias=0,categorical=FALSE,
   low=-3,high=3,truncate=FALSE, threshold=NULL)
con2cat(old,cuts=c(0,1,2,3),where)
```

Arguments

`nvar`	Number of variables to simulate
`nsub`	Number of subjects to simulate
`circum`	circum=TRUE is circumplex structure, FALSE is simple structure

simple	simple structure or spherical structure in sim.spherical
xloading	the average loading on the first dimension
yloading	Average loading on the second dimension
zloading	the average loading on the third dimension in sim.spherical
gloading	Average loading on a general factor (default=0)
xbias	To introduce skew, how far off center is the first dimension
ybias	To introduce skew on the second dimension
zbias	To introduce skew on the third dimension – if using sim.spherical
categorical	continuous or categorical variables.
low	values less than low are forced to low (or 0 in item.dichot)
high	values greater than high are forced to high (or 1 in item.dichot)
cutpoint	cut items at cutpoint (see threshold)
truncate	Change all values less than cutpoint to cutpoint.
threshold	A vector of cutpoints to conver continuous items to binary
nx	number of variables for the first factor in sim.spherical
y	
ny	number of variables for the second and third factors in sim.spherical
old	a matrix or data frame
cuts	Values of old to be used as cut points when converting continuous values to categorical values
where	Which columns of old should be converted to categorical variables. If missing, then all columns are converted.

Details

This simulation was originally developed to compare the effect of skew on the measurement of affect (see Rafaeli and Revelle, 2005). It has been extended to allow for a general simulation of affect or personality items with either a simple structure or a circumplex structure. Items can be continuous normally distributed, or broken down into n categories (e.g, -2, -1, 0, 1, 2). Items can be distorted by limiting them to these ranges, even though the items have a mean of (e.g., 1).

With the addition of a threshold parameter (replacing the previous cut parameter), each item will converted to a binary (0/1) value if the theta exceeds the threshold. If threshold is a vector with length less than nvar, then it will be filled out to length nvar by sampling with replacement.

The addition of item.dichot allows for testing structures with dichotomous items of different difficulty (endorsement) levels. Two factor data with either simple structure or circumplex structure are generated for two sets of items, one giving a score of 1 for all items greater than the low (easy) value, one giving a 1 for all items greater than the high (hard) value. The default values for low and high are 0. That is, all items are assumed to have a 50 percent endorsement rate. To examine the effect of item difficulty, low could be -1, high 1. This will lead to item endorsements of .84 for the easy and .16 for the hard. Within each set of difficulties, the first 1/4 are assigned to the first factor factor, the second to the second factor, the third to the first factor (but with negative loadings) and the fourth to the second factor (but with negative loadings).

It is useful to compare the results of sim.item with sim.hierarchical. sim.item will produce a general factor that runs through all the items as well as two orthogonal factors. This produces a data set that is hard to represent with standard rotation techniques. Extracting 3 factors without rotation and then rotating the 2nd and 3rd factors reproduces the correct solution. But simple oblique rotation of 3 factors, or an omega analysis do not capture the underlying structure. See the last example.

Yet another structure that might be appealing is fully complex data in three dimensions. That is, rather than having items representing the circumference of a circle, items can be structured to represent equally spaced three dimensional points on a sphere. sim.spherical produces such data.

Value

A data matrix of (nsub) subjects by (nvar) variables.

Author(s)

William Revelle

References

Variations of a routine used in Rafaeli and Revelle, 2006; Rafaeli, E. & Revelle, W. (2006). A premature consensus: Are happiness and sadness truly opposite affects? Motivation and Emotion. https://personality-project.org/revelle/publications/rafaeli.revelle.06.pdf

Acton, G. S. and Revelle, W. (2004) Evaluation of Ten Psychometric Criteria for Circumplex Structure. Methods of Psychological Research Online, Vol. 9, No. 1 (formerly (https://www.dgps.de/fachgruppen/methoden/mpr-online/issue22/mpr110_10.pdf) also at https://personality-project.org/revelle/publications/acton.revelle.mpr110_10.pdf

See Also

See also the implementation in this series of functions to generate numerous data structures. simCor, simulation.circ, circ.tests as well as other simulations (sim.structural sim.hierarchical)

Examples

```
round(cor(circ.sim(nvar=8,nsub=200)),2)
plot(fa(circ.sim(16,500),2)$loadings,main="Circumplex Structure") #circumplex structure
#
#
plot(fa(item.sim(16,500),2)$loadings,main="Simple Structure") #simple structure
#
cluster.plot(fa(item.dichot(16,low=0,high=1),2))

 set.seed(42)

 data <- mnormt::rmnorm(1000, c(0, 0), matrix(c(1, .5, .5, 1), 2, 2)) #continuous data
 new <- con2cat(data,c(-1.5,-.5,.5,1.5))  #discreet data
 polychoric(new)
#not run
#x12 <- sim.item(12,gloading=.6)
#f3 <- fa(x12,3,rotate="none")
```

```
#f3  #observe the general factor
#oblimin(f3$loadings[,2:3])  #show the 2nd and 3 factors.
#f3 <- fa(x12,3)   #now do it with oblimin rotation
#f3  # not what one naively expect.
```

`sim.multilevel` *Simulate multilevel data with specified within group and between group correlations*

Description

Multilevel data occur when observations are nested within groups. This can produce correlational structures that are sometimes difficult to understand. These two simulations allow for demonstrations that correlations within groups do not imply, nor are implied by, correlations between group means. The correlations of aggregated data is sometimes called an 'ecological correlation'. That group level and individual level correlations are independent makes such inferences problematic. Within individual data are simulated in sim.multi with a variety of possible within person structures.

Usage

```
sim.multi(n.obs=4,nvar = 2, nfact=2, ntrials=96, days=16, mu=0,sigma=1, fact=NULL,
loading=.9, phi=0,phi.i=NULL,beta.i=0,mu.i=0, sigma.i = 1,sin.i=0, cos.i=0, AR1=0,
f.i=NULL, plot=TRUE)
sim.multilevel(nvar = 9, ngroups = 4, ncases = 16, rwg, rbg, eta)
```

Arguments

`n.obs`	How many subjects should be simulated. Four allows for nice graphics, use more to examine structural properties.
`nvar`	How many variables are to be simulated?
`nfact`	How many factors are simulated,defaults to 2
`ntrials`	How many observations per subject (across time)
`days`	How many days do these observations reflect? This is relevant if we are adding sine and cosines to the model to model diurnal rhythms.
`mu`	The grand mean for each variable across subjects
`sigma`	The between person standard deviation
`fact`	if NULL, a two factor model is created with loadings of loading or zero in a simple structure form
`loading`	If fact is NULL, then we create a factor model with loading or zeros
`phi`	The between person factor intercorrelation
`phi.i`	The within person factor intercorrelations
`beta.i`	Within subject rate of change over trials

mu.i	The within subject mean for each subject
sigma.i	the within subject standard deviation
sin.i	To what extent should we diurnally vary by subject?
cos.i	This will specify the within subject diurnal phase (lag)
AR1	Auto regressive value implies error at time t +1 is partly a function of error at time t.
f.i	Factor loadings for each subject
plot	If TRUE, create a lattice plot for each subject
ngroups	The number of groups to simulate
ncases	The number of simulated cases
rwg	The within group correlational structure
rbg	The between group correlational structure
eta	The correlation of the data with the within data

Details

The basic concepts of the independence of within group and between group correlations is discussed very clearly by Pedhazur (1997) as well as by Bliese (2009). sim.multi generates within subject data to model the traditional two level structure of multilevel data.

This is meant to show how within subject data measured over ntrials can vary independently within and between subjects. Furthermore, several variables can correlate within subjects show a person by person factor structure.

Factor scores for n.obs subjects are created for nfact factors with loadings on nvar variables. A simple structure model is assumed, so that the loadings on nvar/fact are set to loading for each factor, the others are set to 0. Factors are allowed to correlate phi between subjects and phi.i for each subject. (Which can be different for each subject). Scores can change over time with a slope of beta.i and can vary diurnally as a function of sine and cosine of time (24 hours/day converted to radians). Error is added to every trial and can be related across trials with a lag of 1. Thus, if we set AR1=1, then the errors at time t = error + error at t -1. This will lead to auto correlations of about .5. (See autoR).

sim.multilevel merely simulates pooled correlations (mixtures of between group and within group correlations) to allow for a better understanding of the problems inherent in multi-level modeling.

Data (wg) are created with a particular within group structure (rwg). Independent data (bg) are also created with a between group structure (rbg). Note that although there are ncases rows to this data matrix, there are only ngroups independent cases. That is, every ngroups case is a repeat. The resulting data frame (xy) is a weighted sum of the wg and bg. This is the inverse procedure for estimating estimating rwg and rbg from an observed rxy which is done by the statsBy function.

$r_{xy} = \eta_{x_{within}} * \eta_{y_{within}} * r_{xy_{within}} + \eta_{x_{between}} * \eta_{y_{between}} * r_{xy_{between}}$

Value

x.df	A data frame for further analysis using statsBy including nvar variable values for each of n.obs subjects (id) for ntrials.

wg	A matrix (ncases * nvar) of simulated within group scores
bg	A matrix (ncases * nvar) of simulated between group scores
xy	A matrix ncases * (nvar +1) of pooled data

Author(s)

William Revelle

References

P. D. Bliese. Multilevel modeling in R (2.3) a brief introduction to R, the multilevel package and the nlme package, 2009.

Pedhazur, E.J. (1997) Multiple regression in behavioral research: explanation and prediction. Harcourt Brace.

Revelle, W. An introduction to psychometric theory with applications in R (in prep) Springer. Draft chapters available at https://personality-project.org/r/book/

See Also

statsBy for the decomposition of multi level data and withinBetween for an example data set.

Examples

```
#First, show a few results from sim.multi

x.df <- sim.multi()  #the default is 4 subjects for two variables
#                     over 16 days measured 6 times/day

#sb <- statsBy(x.df,group ="id",cors=TRUE)
#round(sb$within,2)  #show the within subject correlations

#get some parameters to simulate
data(withinBetween)
wb.stats <- statsBy(withinBetween,"Group")
rwg <- wb.stats$rwg
rbg <- wb.stats$rbg
eta <- rep(.5,9)

#simulate them.  Try this again to see how it changes
XY <- sim.multilevel(ncases=100,ngroups=10,rwg=rwg,rbg=rbg,eta=eta)
lowerCor(XY$wg)  #based upon 89 df
lowerCor(XY$bg)  #based upon 9 df   --
```

sim.omega *Further functions to simulate psychological/psychometric data.*

Description

A number of functions in the psych package will generate simulated data with particular structures. These functions include sim for a factor simplex, and sim.simplex for a data simplex, sim.circ for a circumplex structure, sim.congeneric for a one factor factor congeneric model, sim.dichot to simulate dichotomous items, sim.hierarchical to create a hierarchical factor model, sim.item a more general item simulation, sim.minor to simulate major and minor factors, sim.omega to test various examples of omega, sim.parallel to compare the efficiency of various ways of determining the number of factors. See the help pages for some more simulation functions here: sim.rasch to create simulated rasch data, sim.irt to create general 1 to 4 parameter IRT data by calling sim.npl 1 to 4 parameter logistic IRT or sim.npn 1 to 4 paramater normal IRT, sim.poly to create polytomous ideas by calling sim.poly.npn 1-4 parameter polytomous normal theory items or sim.poly.npl 1-4 parameter polytomous logistic items, and sim.poly.ideal which creates data following an ideal point or unfolding model by calling sim.poly.ideal.npn 1-4 parameter polytomous normal theory ideal point model or sim.poly.ideal.npl 1-4 parameter polytomous logistic ideal point model.

sim.structural a general simulation of structural models, and sim.anova for ANOVA and lm simulations, and sim.VSS. Some of these functions are separately documented and are listed here for ease of the help function. See each function for more detailed help.

Usage

```
sim.omega(nvar = 12, nfact = 3, n = 500, g = NULL, sem = FALSE, fbig = NULL,
     fsmall = c(-0.2, 0.2),bipolar = TRUE, om.fact = 3, flip = TRUE,
      option = "equal", ntrials = 10,threshold=NULL)
sim.parallel(ntrials=10,nvar = c(12,24,36,48),nfact = c(1,2,3,4,6),
            n = c(200,400))
```

Arguments

nvar	Number of variables
nfact	Number of group factors in sim.general, sim.omega
n	Number of cases to simulate. If n=0 or NULL, the population matrix is returned.
g	General factor correlations in sim.general and general factor loadings in sim.omega and sim.minor
sem	Should the sim.omega function do both an EFA omega as well as a CFA omega using the sem package?
fbig	Factor loadings for the main factors. Default is a simple structure with loadings sampled from (.8,.6) for nvar/nfact variables and 0 for the remaining. If fbig is specified, then each factor has loadings sampled from it.
fsmall	nvar/2 small factors are generated with loadings sampled from (-.2,0,.2)

bipolar	if TRUE, then positive and negative loadings are generated from fbig
om.fact	Number of factors to extract in omega
flip	In omega, should item signs be flipped if negative
option	In omega, for the case of two factors, how to weight them?
threshold	If raw=TRUE (n >0) and threshold is not NULL, convert the data to binary values, cut at threshold.
ntrials	Number of replications per level

Details

Simulation of data structures is a very useful tool in psychometric research and teaching. By knowing "truth" it is possible to see how well various algorithms can capture it. For a much longer discussion of the use of simulation in psychometrics, see the accompany vignettes.

There are a number of simulation functions included in psych. sim.omega is just one of them.

The default values for sim.structure is to generate a 4 factor, 12 variable data set with a simplex structure between the factors. This, and the simplex of items (sim.simplex) can also be converted in a STARS model with an autoregressive component (alpha) and a stable trait component (lambda).

Two data structures that are particular challenges to exploratory factor analysis are the simplex structure and the presence of minor factors. Simplex structures sim.simplex will typically occur in developmental or learning contexts and have a correlation structure of r between adjacent variables and r^n for variables n apart. Although just one latent variable (r) needs to be estimated, the structure will have nvar-1 factors.

An alternative version of the simplex is the State-Trait-Auto Regressive Structure (STARS) which has both a simplex state structure, with autoregressive path alpha and a trait structure with path lambda. This simulated in sim.simplex by specifying a non-zero lambda value.

Many simulations of factor structures assume that except for the major factors, all residuals are normally distributed around 0. An alternative, and perhaps more realistic situation, is that the there are a few major (big) factors and many minor (small) factors. The challenge is thus to identify the major factors. sim.minor generates such structures. The structures generated can be thought of as havinga a major factor structure with some small correlated residuals. To make these simulations complete, the possibility of a general factor is considered. For simplicity, sim.minor allows one to specify a set of loadings to be sampled from for g, fmajor and fminor. Alternatively, it is possible to specify the complete factor matrix.

Another structure worth considering is direct modeling of a general factor with several group factors. This is done using sim.general.

Although coefficient ω is a very useful indicator of the general factor saturation of a unifactorial test (one with perhaps several sub factors), it has problems with the case of multiple, independent factors. In this situation, one of the factors is labelled as "general" and the omega estimate is too large. This situation may be explored using the sim.omega function with general left as NULL. If there is a general factor, then results from sim.omega suggests that omega estimated either from EFA or from SEM does a pretty good job of identifying it but that the EFA approach using Schmid-Leiman transformation is somewhat more robust than the SEM approach.

The four irt simulations, sim.rasch, sim.irt, sim.npl and sim.npn, simulate dichotomous items following the Item Response model. sim.irt just calls either sim.npl (for logistic models) or sim.npn (for normal models) depending upon the specification of the model.

The logistic model is

$$P(i,j) = \gamma + \frac{\zeta - \gamma}{1 + e^{\alpha(\delta - \theta)}}$$

where γ is the lower asymptote or guesssing parameter, ζ is the upper asymptote (normally 1), α is item discrimination and δ is item difficulty. For the 1 Paramater Logistic (Rasch) model, gamma=0, zeta=1, alpha=1 and item difficulty is the only free parameter to specify.

For the 2PL and 2PN models, a = α and d = δ are specified.
For the 3PL or 3PN models, items also differ in their guessing parameter c =γ.
For the 4PL and 4PN models, the upper asymptote, z= ζ is also specified.
(Graphics of these may be seen in the demonstrations for the `logistic` function.)

The normal model (irt.npn calculates the probability using pnorm instead of the logistic function used in irt.npl, but the meaning of the parameters are otherwise the same. With the a = α parameter = 1.702 in the logistic model the two models are practically identical.

In parallel to the dichotomous IRT simulations are the poly versions which simulate polytomous item models. They have the additional parameter of how many categories to simulate. In addition, the `sim.poly.ideal` functions will simulate an ideal point or unfolding model in which the response probability varies by the distance from each subject's ideal point. Some have claimed that this is a more appropriate model of the responses to personality questionnaires. It will lead to simplex like structures which may be fit by a two factor model. The middle items form one factor, the extreme a bipolar factor.

By default, the theta parameter is created in each function as normally distributed with mean mu=0 and sd=1. In the case where you want to specify the theta to be equivalent from another simulation or fixed for a particular experimental condition, either take the theta object from the output of a previous simulation, or create it using whatever properties are desired.

The previous functions all assume one latent trait. Alternatively, we can simulate dichotomous or polytomous items with a particular structure using the sim.poly.mat function. This takes as input the population correlation matrix, the population marginals, and the sample size. It returns categorical items with the specified structure.

Other simulation functions in psych are:

`sim.structure` A function to combine a measurement and structural model into one data matrix. Useful for understanding structural equation models. Combined with `structure.diagram` to see the proposed structure.

`sim.congeneric` A function to create congeneric items/tests for demonstrating classical test theory. This is just a special case of sim.structure.

`sim.hierarchical` A function to create data with a hierarchical (bifactor) structure.

`sim.item` A function to create items that either have a simple structure or a circumplex structure.

`sim.circ` Create data with a circumplex structure.

`sim.dichot` Create dichotomous item data with a simple or circumplex structure.

`sim.minor` Create a factor structure for nvar variables defined by nfact major factors and nvar/2 "minor" factors for n observations.

Although the standard factor model assumes that K major factors (K « nvar) will account for the correlations among the variables

$$R = FF' + U^2$$

where R is of rank P and F is a P x K matrix of factor coefficients and U is a diagonal matrix of uniquenesses. However, in many cases, particularly when working with items, there are many small factors (sometimes referred to as correlated residuals) that need to be considered as well. This leads to a data structure such that

$$R = FF' + MM' + U^2$$

where R is a P x P matrix of correlations, F is a P x K factor loading matrix, M is a P x P/2 matrix of minor factor loadings, and U is a diagonal matrix (P x P) of uniquenesses.

Such a correlation matrix will have a poor χ^2 value in terms of goodness of fit if just the K factors are extracted, even though for all intents and purposes, it is well fit.

sim.minor will generate such data sets with big factors with loadings of .6 to .8 and small factors with loadings of -.2 to .2. These may both be adjusted.

`sim.parallel` Create a number of simulated data sets using sim.minor to show how parallel analysis works. The general observation is that with the presence of minor factors, parallel analysis is probably best done with component eigen values rather than factor eigen values, even when using the factor model.

`sim.anova` Simulate a 3 way balanced ANOVA or linear model, with or without repeated measures. Useful for teaching research methods and generating teaching examples.

`sim.multilevel` To understand some of the basic concepts of multilevel modeling, it is useful to create multilevel structures. The correlations of aggregated data is sometimes called an 'ecological correlation'. That group level and individual level correlations are independent makes such inferences problematic. This simulation allows for demonstrations that correlations within groups do not imply, nor are implied by, correlations between group means.

Value

`link{sim.parallel}` returns the results of multiple simulations of fa.parallel for various combinations of the number of variables, numbers of factors, and sample size. `link{sim.general}` returns either a correlation matrix (if n is not specified) or a data matrix with a general factor.

Note

A few of the various sim functions are included here.

Author(s)

https://personality-project.org/revelle.html
Maintainer: William Revelle < `revelle@northwestern.edu` >

References

https://personality-project.org/r/r.omega.html

Revelle, William. (in prep) An introduction to psychometric theory with applications in R. Springer. Working draft available at https://personality-project.org/r/book/

Revelle, W. (1979). Hierarchical cluster analysis and the internal structure of tests. Multivariate Behavioral Research, 14, 57-74. (https://personality-project.org/revelle/publications/iclust.pdf)

Revelle, W. and Zinbarg, R. E. (2009) Coefficients alpha, beta, omega and the glb: comments on Sijtsma. Psychometrika, 74, 1, 145-154. (https://personality-project.org/revelle/publications/rz09.pdf

Waller, N. G. (2017) Direct Schmid-Leiman Transformations and Rank-Deficient Loadings Matrices. Psychometrika. DOI: 10.1007/s11336-017-9599-0

Zinbarg, R.E., Revelle, W., Yovel, I., & Li. W. (2005). Cronbach's Alpha, Revelle's Beta, McDonald's Omega: Their relations with each and two alternative conceptualizations of reliability. Psychometrika. 70, 123-133. https://personality-project.org/revelle/publications/zinbarg.revelle.pmet.05.pdf

Zinbarg, R., Yovel, I. & Revelle, W. (2007). Estimating omega for structures containing two group factors: Perils and prospects. Applied Psychological Measurement. 31 (2), 135-157.

Zinbarg, R., Yovel, I., Revelle, W. & McDonald, R. (2006). Estimating generalizability to a universe of indicators that all have one attribute in common: A comparison of estimators for omega. Applied Psychological Measurement, 30, 121-144. DOI: 10.1177/0146621605278814

See Also

omega

Examples

```
#test <- sim.omega()
```

`sim.structure` *Create correlation matrices or data matrices with a particular measurement and structural model*

Description

Structural Equation Models decompose correlation or correlation matrices into a measurement (factor) model and a structural (regression) model. sim.structural creates data sets with known measurement and structural properties. Population or sample correlation matrices with known properties are generated. Optionally raw data are produced.

It is also possible to specify a measurement model for a set of x variables separately from a set of y variables. They are then combined into one model with the correlation structure between the two sets.

Finally, the general case is given a population correlation matrix, generate data that reproduce (with sampling variability) that correlation matrix. simCor sim.correlation.

Usage

```
sim.structure(fx=NULL,Phi=NULL, fy=NULL, f=NULL, n=0, uniq=NULL, raw=TRUE,
  items = FALSE, low=-2,high=2,d=NULL,cat=5, mu=0)
sim.structural(fx=NULL, Phi=NULL, fy=NULL, f=NULL, n=0, uniq=NULL, raw=TRUE,
      items = FALSE, low=-2,high=2,d=NULL,cat=5, mu=0)  #deprecated
```

```
simCor(R,n=1000,data=FALSE,scale=TRUE, skew=c("none","log","lognormal",
   "sqrt","abs"),vars=NULL,latent=FALSE,quant=NULL)
sim.correlation(R,n=1000,data=FALSE,scale=TRUE, skew=c("none","log","lognormal",
    "sqrt","abs"),vars=NULL,latent=FALSE,quant=NULL)
```

Arguments

fx	The measurement model for x
Phi	The structure matrix of the latent variables
fy	The measurement model for y
f	The measurement model
n	Number of cases to simulate. If n=0, the population matrix is returned.
uniq	The uniquenesses if creating a covariance matrix
raw	if raw=TRUE, raw data are returned as well for n > 0.
items	TRUE if simulating items, FALSE if simulating scales
low	Restrict the item difficulties to range from low to high
high	Restrict the item difficulties to range from low to high
d	A vector of item difficulties, if NULL will range uniformly from low to high
cat	Number of categories when creating binary (2) or polytomous items
mu	A vector of means, defaults to 0
R	The correlation matrix to reproduce
data	if TRUE, return the raw data, otherwise return the sample correlation matrix.
scale	standardize the simulated data?
skew	Defaults to none (the multivariate normal case. Alternatives take the log, the squareroot, or the absolute value of latent or observed data)
vars	Apply the skewing or cuts to just these variables. If NULL, to all the variables/
latent	Should the skewing transforms be applied to the latent variables, or the observed variables?
quant	Either a single number or a vector length nvar. The data will be dichotomized at quant.

Details

Given the measurement model, fx and the structure model Phi, the model is f %*% Phi %*% t(f). Reliability is f %*% t(f). $f\phi f'$ and the reliability for each test is the items communality or just the diag of the model.

If creating a correlation matrix, (uniq=NULL) then the diagonal is set to 1, otherwise the diagonal is diag(model) + uniq and the resulting structure is a covariance matrix.

A special case of a structural model are one factor models such as parallel tests, tau equivalent tests, and congeneric tests. These may be created by letting the structure matrix = 1 and then defining a vector of factor loadings. Alternatively, `sim.congeneric` will do the same.

The general case is to use simCor aka sim.correlation which will create data sampled from a specified correlation matrix for a particular sample size. This follows a procedure described by Kaiser and Dickman (1962). If desired, it will just return the sample correlation matrix. With data=TRUE, it will return the sample data as well. It uses an eigen value decomposition of the original matrix times a matrix of random normal deviates (code adapted from the mvnorm function of Brian Ripley's MASS package). These resulting scores may be transformed using a number of transforms (see the skew option) or made into dichotomous variables (see quant option) for all or a select set (vars option) of the variables.

Value

model	The implied population correlation or covariance matrix
reliability	The population reliability values
r	The sample correlation or covariance matrix
observed	If raw=TRUE, a sample data matrix

Author(s)

William Revelle

References

Kaiser, H.F. and Dickman, W. (1962) Sample and population score matrices and sample correlation matrices from an arbitrary population correlation matrix. Psychometrika, 27, 179-182.

Revelle, W. (in preparation) An Introduction to Psychometric Theory with applications in R. Springer. at https://personality-project.org/r/book/

See Also

make.hierarchical for another structural model and make.congeneric for the one factor case. structure.list and structure.list for making symbolic structures.

Examples

```
#First, create a sem like model with a factor model of x and ys with correlation Phi
fx <-matrix(c( .9,.8,.6,rep(0,4),.6,.8,-.7),ncol=2)
fy <- matrix(c(.6,.5,.4),ncol=1)
rownames(fx) <- c("V","Q","A","nach","Anx")
rownames(fy)<- c("gpa","Pre","MA")
Phi <-matrix( c(1,0,.7,.0,1,.7,.7,.7,1),ncol=3)
#now create this structure
gre.gpa <- sim.structural(fx,Phi,fy)
print(gre.gpa,2)
#correct for attenuation to see structure
#the raw correlations are below the diagonal, the adjusted above
round(correct.cor(gre.gpa$model,gre.gpa$reliability),2)

#These are the population values,
# we can also create a correlation matrix sampled from this population
GRE.GPA  <- sim.structural(fx,Phi,fy,n=250,raw=FALSE)
```

```
lowerMat(GRE.GPA$r)

#or we can show data sampled from such a population
GRE.GPA  <- sim.structural(fx,Phi,fy,n=250,raw=TRUE)
lowerCor(GRE.GPA$observed)



congeneric <- sim.structure(f=c(.9,.8,.7,.6)) # a congeneric model
congeneric

#now take this correlation matrix as a population value and create samples from it
example.congeneric <- sim.correlation(congeneric$model,n=200) #create a sample matrix
lowerMat(example.congeneric ) #show the correlation matrix
#or create another sample and show the data
example.congeneric.data <- simCor(congeneric$model,n=200,data=TRUE)
describe(example.congeneric.data)
lowerCor(example.congeneric.data )
example.skewed <- simCor(congeneric$model,n=200,vars=c(1,2),data=TRUE,skew="log")
describe(example.skewed)
```

`sim.VSS` *create VSS like data*

Description

Simulation is one of most useful techniques in statistics and psychometrics. Here we simulate a correlation matrix with a simple structure composed of a specified number of factors. Each item is assumed to have complexity one. See `circ.sim` and `item.sim` for alternative simulations.

Usage

```
sim.VSS(ncases=1000, nvariables=16, nfactors=4, meanloading=.5,dichot=FALSE,cut=0)
```

Arguments

`ncases`	number of simulated subjects
`nvariables`	Number of variables
`nfactors`	Number of factors to generate
`meanloading`	with a mean loading
`dichot`	dichot=FALSE give continuous variables, dichot=TRUE gives dichotomous variables
`cut`	if dichotomous = TRUE, then items with values > cut are assigned 1, otherwise 0.

Value

a ncases x nvariables matrix

Author(s)

William Revelle

See Also

VSS, ICLUST

Examples

```
## Not run:
simulated <- sim.VSS(1000,20,4,.6)
vss <- VSS(simulated,rotate="varimax")
VSS.plot(vss)

## End(Not run)
```

---

simulation.circ *Simulations of circumplex and simple structure*

Description

Rotations of factor analysis and principal components analysis solutions typically try to represent correlation matrices as simple structured. An alternative structure, appealing to some, is a circumplex structure where the variables are uniformly spaced on the perimeter of a circle in a two dimensional space. Generating these data is straightforward, and is useful for exploring alternative solutions to affect and personality structure.

Usage

```
simulation.circ(samplesize=c(100,200,400,800), numberofvariables=c(16,32,48,72))
circ.sim.plot(x.df)
```

Arguments

samplesize	a vector of sample sizes to simulate
numberofvariables	vector of the number of variables to simulate
x.df	A data frame resulting from simulation.circ

Details

"A common model for representing psychological data is simple structure (Thurstone, 1947). According to one common interpretation, data are simple structured when items or scales have nonzero factor loadings on one and only one factor (Revelle & Rocklin, 1979). Despite the commonplace application of simple structure, some psychological models are defined by a lack of simple

structure. Circumplexes (Guttman, 1954) are one kind of model in which simple structure is lacking.

"A number of elementary requirements can be teased out of the idea of circumplex structure. First, circumplex structure implies minimally that variables are interrelated; random noise does not a circumplex make. Second, circumplex structure implies that the domain in question is optimally represented by two and only two dimensions. Third, circumplex structure implies that variables do not group or clump along the two axes, as in simple structure, but rather that there are always interstitial variables between any orthogonal pair of axes (Saucier, 1992). In the ideal case, this quality will be reflected in equal spacing of variables along the circumference of the circle (Gurtman, 1994; Wiggins, Steiger, & Gaelick, 1981). Fourth, circumplex structure implies that variables have a constant radius from the center of the circle, which implies that all variables have equal communality on the two circumplex dimensions (Fisher, 1997; Gurtman, 1994). Fifth, circumplex structure implies that all rotations are equally good representations of the domain (Conte & Plutchik, 1981; Larsen & Diener, 1992)." (Acton and Revelle, 2004)

Acton and Revelle reviewed the effectiveness of 10 tests of circumplex structure and found that four did a particularly good job of discriminating circumplex structure from simple structure, or circumplexes from ellipsoidal structures. Unfortunately, their work was done in Pascal and is not easily available. Here we release R code to do the four most useful tests:

The Gap test of equal spacing

Fisher's test of equality of axes

A test of indifference to Rotation

A test of equal Variance of squared factor loadings across arbitrary rotations.

Included in this set of functions are simple procedure to generate circumplex structured or simple structured data, the four test statistics, and a simple simulation showing the effectiveness of the four procedures.

`circ.sim.plot` compares the four tests for circumplex, ellipsoid and simple structure data as function of the number of variables and the sample size. What one can see from this plot is that although no one test is sufficient to discriminate these alternative structures, the set of four tests does a very good job of doing so. When testing a particular data set for structure, comparing the results of all four tests to the simulated data will give a good indication of the structural properties of the data.

Value

A data.frame with simulation results for circumplex, ellipsoid, and simple structure data sets for each of the four tests.

Note

The simulations default values are for sample sizes of 100, 200, 400, and 800 cases, with 16, 32, 48 and 72 items.

Author(s)

William Revelle

References

Acton, G. S. and Revelle, W. (2004) Evaluation of Ten Psychometric Criteria for Circumplex Structure. Methods of Psychological Research Online, Vol. 9, No. 1 (formerly at https://www.dgps.de/fachgruppen/methoden/mpr-online/issue22/mpr110_10.pdf and now at https://personality-project.org/revelle/publications/acton.revelle.mpr110_10.pdf.

See Also

See also circ.tests, sim.circ, sim.structural, sim.hierarchical, simCor, sim

Examples

```
#not run
demo <- simulation.circ()
boxplot(demo[3:14])
title("4 tests of Circumplex Structure",sub="Circumplex, Ellipsoid, Simple Structure")
circ.sim.plot(demo[3:14])  #compare these results to real data
```

---

small.msq *A small example data set taken from a larger data set*

Description

A small subset of 200 cases and 14 variables from the msqR data set in the psychTools package. Saved here to allow vignettes to work without using psychTools

Usage

```
data("small.msq")
```

Format

A data frame with 200 observations on the following 14 variables.

`active` a numeric vector

`alert` a numeric vector

`aroused` a numeric vector

`sleepy` a numeric vector

`tired` a numeric vector

`drowsy` a numeric vector

`anxious` a numeric vector

`jittery` a numeric vector

`nervous` a numeric vector

`calm` a numeric vector

`relaxed` a numeric vector

`at.ease` a numeric vector

`gender` a numeric vector

`drug` a numeric vector

Details

See the detailed discussion of the msqR in the psychTools package

References

Revelle, W. and Anderson, K.J. (1998) Personality, motivation and cognitive performance: Final report to the Army Research Institute on contract MDA 903-93-K-0008. (https://www.personality-project.org/revelle/publications/ra.ari.98.pdf).

Examples

```
data(small.msq)
```

`smc` *Find the Squared Multiple Correlation (SMC) of each variable with the remaining variables in a matrix*

Description

The squared multiple correlation of a variable with the remaining variables in a matrix is sometimes used as initial estimates of the communality of a variable.

SMCs are also used when estimating reliability using Guttman's lambda 6 `guttman` coefficient.

The SMC is just 1 - 1/diag(R.inv) where R.inv is the inverse of R.

Usage

```
smc(R,covar=FALSE)
```

Arguments

`R`	A correlation matrix or a dataframe. In the latter case, correlations are found.
`covar`	if covar = TRUE and R is either a covariance matrix or data frame, then return the smc * variance for each item

Value

a vector of squared multiple correlations. Or, if covar=TRUE, a vector of squared multiple correlations * the item variances

If the matrix is not invertible, then a vector of 1s is returned. Note, that I now take the left pseudo inverse so this is less likely to happen (if at all).

In the case of correlation or covariance matrices with some NAs, those variables with NAs are dropped and the SMC for the remaining variables are found. The missing SMCs are then estimated by finding the maximum correlation for that column (with a warning).

Author(s)

William Revelle

See Also

mat.regress, fa

Examples

```
R <- make.hierarchical()
round(smc(R),2)
```

---

spider *Make "radar" or "spider" plots.*

Description

Radar plots and spider plots are just two of the many ways to show multivariate data. radar plots correlations as vectors ranging in length from 0 (corresponding to r=-1) to 1 (corresponding to an r=1). The vectors are arranged radially around a circle. Spider plots connect the end points of each vector. The plots are most appropriate if the variables are organized in some meaningful manner.

Usage

```
spider(y,x,data,labels=NULL,rescale=FALSE,center=FALSE,connect=TRUE,overlay=FALSE,
    scale=1,ncolors=31,fill=FALSE,main=NULL,...)

radar(x,labels=NULL,keys=NULL,center=FALSE,connect=FALSE,scale=1,ncolors=31,fill=FALSE,
    add=FALSE,linetyp="solid", main="Radar Plot",angle=0,absolute=FALSE,
    show=TRUE,digits=2,cut=.2,circles=TRUE, shape=FALSE, clockwise=FALSE,
     delta = NULL,label.pos=NULL,position=NULL,
     xlim=c(-1,1),ylim=c(-1, 1),...)
```

Arguments

`y`	The y variables to plot. Each y is plotted against all the x variables
`x`	The x variables defining each line. Each y is plotted against all the x variables
`data`	A correlation matrix from which the x and y variables are selected
`labels`	Labels (assumed to be colnames of the data matrix) for each x variable
`rescale`	If TRUE, then rescale the data to have mean 0 and sd = 1. This is used if plotting raw data rather than correlations.
`center`	if TRUE, then lines originate at the center of the plot, otherwise they start at the mid point.
`connect`	if TRUE, a spider plot is drawn, if FALSE, just a radar plot
`scale`	can be used to magnify the plot, to make small values appear larger.
`ncolors`	if ncolors > 2, then positive correlations are plotted with shades of blue and negative correlations shades of red. This is particularly useful if fill is TRUE. ncolors should be an odd number, so that neutral values are coded as white.
`fill`	if TRUE, fill the polygons with colors scaled to size of correlation
`overlay`	If TRUE, plot multiple spiders on one plot, otherwise plot them as separate plots
`add`	If TRUE, add a new spider diagram to the previous one.
`linetyp`	see lty in the par options
`main`	A label or set of labels for the plots
`keys`	If a keys list is provided, then variables are grouped by the keys, with labels drawn for the key names
`angle`	Rotate the entire figure angle/nvar to the left. Useful for drawing circumplex structures
`absolute`	If TRUE, then just use color to show correlation size
`show`	If TRUE, show the values at the end of the radar lines if they are > cut
`digits`	round the values to digits
`cut`	Just show values > cut
`circles`	Draw circles at .25, .5 and .75
`shape`	If TRUE, do not draw circles, but rather polygons with nvar sides
`clockwise`	If TRUE, organize the variables clockwise
`delta`	How far from the ends of the lines should the values be placed (defaults to 1.05 * length of line). May be vector.
`label.pos`	How far out should the labels be placed? (defaults to 1.05 which is just outside of the outer circle.)
`position`	A way of passing the pos parameter that includes NULL as a value. (See pos in graphics help)
`xlim`	default values may be changed for more space for labels
`ylim`	default values by be changed for more space for labelssap
`...`	Additional parameters can be passed to the underlying graphics call

Details

Displaying multivariate profiles may be done by a series of lines (see, e.g., matplot), by colors (see, e.g., corPlot, or by radar or spider plots. Spiders are particularly suitable for showing data thought to have circumplex structure.

To show just one variable as a function of several others, use radar. To make multiple plots, use spider. An additional option when comparing just a few y values is to do overlay plots. Alternatively, set the plotting options to do several on one page.

Value

Either a spider or radar plot

Author(s)

William Revelle

See Also

corPlot

Examples

```
op <- par(mfrow=c(3,2))
spider(y=1,x=2:9,data=Thurstone,connect=FALSE) #a radar plot
spider(y=1,x=2:9,data=Thurstone) #same plot as a spider plot
 spider(y=1:3,x=4:9,data=Thurstone,overlay=TRUE)
 #make a somewhat oversized plot
spider(y=26:28,x=1:25,data=cor(bfi,use="pairwise"),fill=TRUE,scale=2)
par(op)

#another example taken from  Lippa (2001, page 193)
lippa.df <-
structure(list(labels = c("Assured - Dominant", "Gregarious\nExtraverted",
"Warm\nAgreeable", "Unassuming\nIngeneous", "Unassured - Submissive",
"Aloof\nIntroverted", "Cold\nHearted", "Arrogant\nCalculating"
), pos = c(0.8, 0.85, 0.83, 0.8, 0.75, 0.83, 0.85, 0.85), values = c(0.41,
-0.29, -0.53, -0.61, -0.38, 0.14, 0.59, 0.6), delta = c(1.1,
1.2, 1.2, 1.1, 1.1, 1.5, 1.2, 1.1)), row.names = c(NA, -8L), class = "data.frame")

radar(lippa.df$values,abs=TRUE,labels=lippa.df$labels,angle=90,clockwise=TRUE,lwd=3,
 label.pos=lippa.df$pos,main="Data from Lippa (2001)",scale=.9,circles=FALSE,
  cut=0,delta=lippa.df$delta)
 segments(-1,0,1,0,lwd=.2)  # Add hairline axes
 segments(0,-1,0,1,lwd=.2)
text(0,1.05,expression(italic("Masculine Instrumentality")))
text(1.05,0,expression(italic("Feminine Communion")),srt=270)

#show how to draw a hexagon
RIASEC.df <- structure(list(labels = c("Realistic", "Investigative", "Artistic",
"Social", "Enterprising", "Conventional"), Su = c(0.84, 0.26,
-0.35, -0.68, 0.04, -0.33), Morris = c(1.14, 0.32, -0.19, -0.38,
```

```
0.22, 0.23)), row.names = c(NA, -6L), class = "data.frame")

 radar(RIASEC.df$Morris,RIASEC.df$labels,clockwise=TRUE,angle=0,absolute=TRUE,circl=FALSE,scale=.7,
 position=c(1,0,0,0,0,0), lwd=4,label.pos=rep(.80,6),main="",cut=0, shape=TRUE,
  delta =c(1.1,1.25,1.25, 1.25, 1.45,1.45) )
 text(-1.04,0,expression(italic("People")),srt=90)
 text(1.04,0,expression(italic("Things")),srt=270)
 text(0,.91,expression(italic("Data")))
 text(0,-.91 ,expression(italic("Ideas")))
 segments(-1,0,1,0,lwd=.2)     #add hairline axes
 segments(0,-.86,0,.86,lwd=.2)
 text(0,1.2, "Data from Su")
```

---

splitHalf *Alternative estimates of test reliabiity*

Description

Eight alternative estimates of test reliability include the six discussed by Guttman (1945), four discussed by ten Berge and Zergers (1978) ($\mu_0 \ldots \mu_3$) as well as β (the worst split half, Revelle, 1979), the glb (greatest lowest bound) discussed by Bentler and Woodward (1980), and ω_h and ω_t (McDonald, 1999; Zinbarg et al., 2005). Greatest and lowest split-half values are found by brute force or sampling.

Usage

```
splitHalf(r,raw=FALSE,brute=FALSE,n.sample=15000,covar=FALSE,check.keys=TRUE,
           key=NULL,ci=.05,use="pairwise")
guttman(r,key=NULL)
tenberge(r)
glb(r,key=NULL)
glb.fa(r,key=NULL)
```

Arguments

r	A correlation or covariance matrix or raw data matrix.
raw	return a vector of split half reliabilities
brute	Use brute force to try all combinations of n take n/2. Be careful. For e.g., n=24, this is 1,352,078 possible splits!
n.sample	If brute is false, how many samples of split halves should be tried? (16 items takes 12,780)
covar	Should the covariances or correlations be used for reliability calculations
check.keys	If TRUE, any item with a negative loading on the first factor will be flipped in sign

key	a vector of -1, 0, 1 to select or reverse key items. See scoreItems for an example of keying. If the key vector is less than the number of variables, then item numbers to be reverse can be specified.
use	Should we find the correlations using "pairwise" or "complete" (see ?cor)
ci	The alpha level to use for the confidence intervals of the split half estimates

Details

Surprisingly, more than a century after Spearman (1904) introduced the concept of reliability to psychologists, there are still multiple approaches for measuring it. Although very popular, Cronbach's α (1951) underestimates the reliability of a test and over estimates the first factor saturation. Using splitHalf for tests with 16 or fewer items, all possible splits may be found fairly easily. For tests with 17 or more items, n.sample splits are randomly found. Thus, for 16 or fewer items, the upper and lower bounds are precise. For 17 or more items, they are close but will probably slightly underestimate the highest and overestimate the lowest reliabilities.

The guttman function includes the six estimates discussed by Guttman (1945), four of ten Berge and Zergers (1978), as well as Revelle's β (1979) using splitHalf. The companion function, omega calculates omega hierarchical (ω_h) and omega total (ω_t).

Guttman's first estimate λ_1 assumes that all the variance of an item is error:

$$\lambda_1 = 1 - \frac{tr(\vec{V_x})}{V_x} = \frac{V_x - tr(\vec{V_x})}{V_x}$$

This is a clear underestimate.

The second bound, λ_2, replaces the diagonal with a function of the square root of the sums of squares of the off diagonal elements. Let $C_2 = \vec{1}(\vec{V} - diag(\vec{V})^2\vec{1}'$, then

$$\lambda_2 = \lambda_1 + \frac{\sqrt{\frac{n}{n-1}C_2}}{V_x} = \frac{V_x - tr(\vec{V_x}) + \sqrt{\frac{n}{n-1}C_2}}{V_x}$$

Effectively, this is replacing the diagonal with n * the square root of the average squared off diagonal element.

Guttman's 3rd lower bound, λ_3, also modifies λ_1 and estimates the true variance of each item as the average covariance between items and is, of course, the same as Cronbach's α.

$$\lambda_3 = \lambda_1 + \frac{\frac{V_X - tr(\vec{V_X})}{n(n-1)}}{V_X} = \frac{n\lambda_1}{n-1} = \frac{n}{n-1}\Big(1 - \frac{tr(\vec{V})_x}{V_x}\Big) = \frac{n}{n-1}\frac{V_x - tr(\vec{V_x})}{V_x} = \alpha$$

This is just replacing the diagonal elements with the average off diagonal elements. $\lambda_2 \geq \lambda_3$ with $\lambda_2 > \lambda_3$ if the covariances are not identical.

λ_3 and λ_2 are both corrections to λ_1 and this correction may be generalized as an infinite set of successive improvements. (Ten Berge and Zegers, 1978)

$$\mu_r = \frac{1}{V_x}\big(p_o + (p_1 + (p_2 + \ldots (p_{r-1} + (p_r)^{1/2})^{1/2} \ldots)^{1/2})^{1/2}\big), r = 0, 1, 2, \ldots$$

where

$$p_h = \sum_{i \neq j} \sigma_{ij}^{2h}, h = 0, 1, 2, \ldots r-1$$

and

$$p_h = \frac{n}{n-1}\sigma_{ij}^{2h}, h = r$$

tenberge and Zegers (1978). Clearly $\mu_0 = \lambda_3 = \alpha$ and $\mu_1 = \lambda_2$. $\mu_r \geq \mu_{r-1} \geq \ldots \mu_1 \geq \mu_0$, although the series does not improve much after the first two steps.

Guttman's fourth lower bound, λ_4 was originally proposed as any spit half reliability but has been interpreted as the greatest split half reliability. If $\vec{X}$ is split into two parts, $\vec{X}_a$ and $\vec{X}_b$, with correlation r_{ab} and covariance C_{ab} then

$$\lambda_4 = 2\Big(1 - \frac{V_{X_a} + V_{X_b}}{V_X}\Big) = \frac{4C_{ab}}{V_x} = \frac{4C_{ab}}{V_{X_a} + V_{X_b} + 2C_{ab}}$$

which is just the normal split half reliability, but in this case, of the most similar splits. For 16 or fewer items, this is found by trying all possible splits. For 17 or more items, this is estimated by taking n.sample random splits.

λ_5, Guttman's fifth lower bound, replaces the diagonal values with twice the square root of the maximum (across items) of the sums of squared interitem covariances

$$\lambda_5 = \lambda_1 + \frac{2\sqrt{\bar{C}_2}}{V_X}.$$

Although superior to λ_1, λ_5 underestimates the correction to the diagonal. A better estimate would be analogous to the correction used in λ_3:

$$\lambda_{5+} = \lambda_1 + \frac{n}{n-1}\frac{2\sqrt{\bar{C}_2}}{V_X}.$$

λ_6,Guttman's final bound considers the amount of variance in each item that can be accounted for the linear regression of all of the other items (the squared multiple correlation or smc), or more precisely, the variance of the errors, e_j^2, and is

$$\lambda_6 = 1 - \frac{\sum e_j^2}{V_x} = 1 - \frac{\sum(1 - r_{smc}^2)}{V_x}$$

.

The smc is found from all the items. A modification to Guttman λ_6, λ_6* reported by the `scoreItems` function is to find the smc from the entire pool of items given, not just the items on the selected scale.

Guttman's λ_4 is the greatest split half reliability. Although originally found here by combining the output from three different approaches,this has now been replaced by using `splitHalf` to find the maximum value by brute force (for 16 or fewer items) or by taking a substantial number of random splits.

The algorithms that had been tried before included:

a) Do an ICLUST of the reversed correlation matrix. ICLUST normally forms the most distinct clusters. By reversing the correlations, it will tend to find the most related clusters. Truly a weird approach but tends to work.

b) Alternatively, a kmeans clustering of the correlations (with the diagonal replaced with 0 to make pseudo distances) can produce 2 similar clusters.

c) Clusters identified by assigning items to two clusters based upon their order on the first principal factor. (Highest to cluster 1, next 2 to cluster 2, etc.)

These three procedures will produce keys vectors for assigning items to the two splits. The maximum split half reliability is found by taking the maximum of these three approaches. This is not elegant but is fast.

The brute force and the sampling procedures seem to provide more stable and larger estimates.

Yet another procedure, implemented in `splitHalf` is actually to form all possible (for n items <= 16) or sample 15,000 (or more) split halfs corrected for test length. This function returns the best and worst splits as item keys that can be used for scoring purposes, if desired. Can do up to 24 items in reasonable time, but gets much slower for more than about 24 items. To do all possible splits of 24 items considers 1,352,078 splits. This will give an exact value, but this will not differ that much from random samples. For a 24 item problem with exactly 2 factors (by simulation), the worst split half is much lower than just random sampling would indicate.

The number of possible splits grows exponentially. See `choose` for various split sizes.

When consider split halfs, it is important to remember these are of roughly equal size. So a correlation matrix of 3 unrelated factors (sharing no general factor) will have a "worst" split which is not 0. See the examples.

Timings on a MacPro for a 24 item problem with a 2.4 GHz 8 core are .24 secs for the default 10,000 samples, .678 for 30,000 samples and 22.58 sec for all possible. The values of the maximum split for these sample sizes were .799,/.804 and .800 for three replications of the default sample size of 10000, .805 and .806 for two sets of 30,000 and .812 for an exhaustive search.

There are three greatest lower bound functions. One, glb finds the greatest split half reliability, λ_4. This considers the test as set of items and examines how best to partition the items into splits. The other two, `glb.fa` and `glb.algebraic`, are alternative ways of weighting the diagonal of the matrix.

`glb.fa` estimates the communalities of the variables from a factor model where the number of factors is the number with positive eigen values. Then reliability is found by

$$glb = 1 - \frac{\sum e_j^2}{V_x} = 1 - \frac{\sum(1-h^2)}{V_x}$$

This estimate will differ slightly from that found by `glb.algebraic`, written by Andreas Moeltner which uses calls to csdp in the Rcsdp package. His algorithm, which more closely matches the description of the glb by Jackson and Woodhouse, seems to have a positive bias (i.e., will over estimate the reliability of some items; they are said to be = 1) for small sample sizes. More exploration of these two algorithms is underway.

Compared to `glb.algebraic`, `glb.fa` seems to have less (positive) bias for smallish sample sizes (n < 500) but larger for large (> 1000) sample sizes. This interacts with the number of variables so that equal bias sample size differs as a function of the number of variables. The differences are, however small. As samples sizes grow, `glb.algebraic` seems to converge on the population value while glb.fa has a positive bias.

Value

`beta` The worst split half reliability. This is an estimate of the general factor saturation.

maxrb	The maximimum split half reliability. This is Guttman's lambda 4
alpha	Also known as Guttman's Lambda 3
ci	The 2.5%, 50%, and 97.5% values of the raw or sampled split half. Note that it necessary to specify raw=TRUE to get these.
tenberge$mu1	tenBerge mu 1 is functionally alpha
tenberge$mu2	one of the sequence of estimates mu1 ... mu3
glb	glb found from factor analysis

Author(s)

William Revelle

References

Cronbach, L.J. (1951) Coefficient alpha and the internal strucuture of tests. Psychometrika, 16, 297-334.

Guttman, L. (1945). A basis for analyzing test-retest reliability. Psychometrika, 10 (4), 255-282.

Revelle, W. (1979). Hierarchical cluster-analysis and the internal structure of tests. Multivariate Behavioral Research, 14 (1), 57-74.

Revelle, W. and Condon, D.M. (2019) Reliability from alpha to omega: A tutorial. Psychological Assessment, 31, 12, 1395-1411. DOI: 10.1037/pas0000754. https://osf.io/preprints/psyarxiv/2y3w9 Preprint available from PsyArxiv

Revelle, W. and Zinbarg, R. E. (2009) Coefficients alpha, beta, omega and the glb: comments on Sijtsma. Psychometrika, 2009.

Ten Berge, J. M. F., & Zegers, F. E. (1978). A series of lower bounds to the reliability of a test. Psychometrika, 43 (4), 575-579.

Zinbarg, R. E., Revelle, W., Yovel, I., & Li, W. (2005). Cronbach's α , Revelle's β , and McDonald's ω_h): Their relations with each other and two alternative conceptualizations of reliability. doi:10.1007/s1133600309747 Psychometrika, 70 (1), 123-133.

See Also

reliability, alpha, omega, ICLUST, unidim, glb.algebraic

Examples

```
data(attitude)
splitHalf(attitude)
splitHalf(attitude,covar=TRUE) #do it on the covariances
temp <- splitHalf(attitude,raw=TRUE)
temp$ci #to show the confidence intervals, you need to specify that raw=TRUE

glb(attitude)
glb.fa(attitude)
if(require(Rcsdp)) {glb.algebraic(cor(attitude)) }
guttman(attitude)
```

```
#to show the histogram of all possible splits for the ability test
#sp <- splitHalf(psychTools::ability,raw=TRUE)  #this saves the results
#hist(sp$raw,breaks=101,ylab="SplitHalf reliability",main="SplitHalf
#    reliabilities of a test with 16 ability items")
sp <- splitHalf(bfi[1:10],key=c(1,9,10))


#An example of how split half does not estimate the amount of general factor variance
#When there are more than 2 factors in the data

F <- matrix(c(rep(.8,3),rep(0,9),rep(.8,3),rep(0,9),rep(.8,3)),ncol=3)
R <- F %*% t(F)
diag(R) <- 1
corPlot(R)
sp <- splitHalf(R)   #shows a worst split half of .25
 #    which is clearly not the amount of general variance
sp
sp$minAB      #these equal size splits

#but  ICLUST gets it right
iclust(R,plot=FALSE)$reliability
```

---

statsBy *Find statistics (including correlations) within and between groups for basic multilevel analyses*

Description

When examining data at two levels (e.g., the individual and by some set of grouping variables), it is useful to find basic descriptive statistics (means, sds, ns per group, within group correlations) as well as between group statistics (over all descriptive statistics, and overall between group correlations). Of particular use is the ability to decompose a matrix of correlations at the individual level into correlations within group and correlations between groups.

Usage

```
statsBy(data, group, cors = FALSE, cor="cor", method="pearson", use="pairwise",
 poly=FALSE, na.rm=TRUE,alpha=.05,minlength=5,
 weights=NULL,mean.weights=NULL, min.n=1)
statsBy.boot(data,group,ntrials=10,cors=FALSE,replace=TRUE,method="pearson")
statsBy.boot.summary(res.list,var="ICC2")
faBy(stats, nfactors = 1, rotate = "oblimin", fm = "minres", free = TRUE, all=FALSE,
   min.n = 12,quant=.1, ...)
```

Arguments

data	A matrix or dataframe with rows for subjects, columns for variables. One of these columns should be the values of a grouping variable.

group	The names or numbers of the variable in data to use as the grouping variables.
cors	Should the results include the correlation matrix within each group? Default is FALSE.
cor	Type of correlation/covariance to find within groups and between groups. The default is Pearson correlation. To find within and between covariances, set cor="cov". Although polychoric, tetrachoric, and mixed correlations can be found within groups, this does not make sense for the between groups or the pooled within groups. In this case, correlations for each group will be as specified, but the between groups and pooled within will be Pearson. See the discussion below.
method	What kind of correlations should be found (default is Pearson product moment)
use	How to treat missing data. use="pairwise" is the default
poly	Find polychoric.tetrachoric correlations within groups if requested.
na.rm	Should missing values be deleted (na.rm=TRUE) or should we assume the data are clean?
alpha	The alpha level for the confidence intervals for the ICC1 and ICC2, and rwg, rbg
minlength	The minimum length to use when abbreviating the labels for confidence intervals
weights	If specified, weight the groups by weights when finding the pooled within group correlation. Otherwise weight by sample size.
mean.weights	If not NULL, what weights should be use we wait the withingroup variables by some weights vector?
ntrials	The number of trials to run when bootstrapping statistics
replace	Should the bootstrap be done by permuting the data (replace=FALSE) or sampling with replacement (replace=TRUE)
res.list	The results from statsBy.boot may be summarized using boot.stats
var	Name of the variable to be summarized from statsBy.boot
stats	The output of statsBy
nfactors	The number of factors to extract in each subgroup
rotate	The factor rotation/transformation
fm	The factor method (see fa for details)
free	Allow the factor solution to be freely estimated for each individual (see note).
all	Report individual factor analyses for each group as well as the summary table
min.n	The minimum number of within subject cases before we factor analyze it or find group statistics. Only works for one grouping variable.
quant	Show the upper and lower quant quantile of the factor loadings in faBy
...	Other parameters to pass to the fa function

Details

Multilevel data are endemic in psychological research. In multilevel data, observations are taken on subjects who are nested within some higher level grouping variable. The data might be experimental (participants are nested within experimental conditions) or observational (students are nested within classrooms, students are nested within college majors.) To analyze this type of data, one uses random effects models or mixed effect models, or more generally, multilevel models. There are at least two very powerful packages (nlme and multilevel) which allow for complex analysis of hierarchical (multilevel) data structures. `statsBy` is a much simpler function to give some of the basic descriptive statistics for two level models. It is meant to supplement true multilevel modeling.

For a group variable (group) for a data.frame or matrix (data), basic descriptive statistics (mean, sd, n) as well as within group correlations (cors=TRUE) are found for each group.

The amount of variance associated with the grouping variable compared to the total variance is the type 1 IntraClass Correlation (ICC1): $ICC1 = (MSb - MSw)/(MSb + MSw * (npr - 1))$ where npr is the average number of cases within each group.

The reliability of the group differences may be found by the ICC2 which reflects how different the means are with respect to the within group variability. $ICC2 = (MSb - MSw)/MSb$. Because the mean square between is sensitive to sample size, this estimate will also reflect sample size.

Perhaps the most useful part of `statsBy` is that it decomposes the observed correlations between variables into two parts: the within group and the between group correlation. This follows the decomposition of an observed correlation into the pooled correlation within groups (rwg) and the weighted correlation of the means between groups discussed by Pedazur (1997) and by Bliese in the multilevel package.

$r_{xy} = eta_{x_{wg}} * eta_{y_{wg}} * r_{xy_{wg}} + eta_{x_{bg}} * eta_{y_{bg}} * r_{xy_{bg}}$

where r_{xy} is the normal correlation which may be decomposed into a within group and between group correlations $r_{xy_{wg}}$ and $r_{xy_{bg}}$ and eta is the correlation of the data with the within group values, or the group means.

It is important to realize that the within group and between group correlations are independent of each other. That is to say, inferring from the 'ecological correlation' (between groups) to the lower level (within group) correlation is inappropriate. However, these between group correlations are still very meaningful, if inferences are made at the higher level.

There are actually two ways of finding the within group correlations pooled across groups. We can find the correlations within every group, weight these by the sample size and then report this pooled value (pooled). This is found if the cors option is set to TRUE. It is logically equivalent to doing a sample size weighted meta-analytic correlation. The other way, rwg, considers the covariances, variances, and thus correlations when each subject's scores are given as deviation score from the group mean.

If finding tetrachoric, polychoric, or mixed correlations, these two estimates will differ, for the pooled value is the weighted polychoric correlation, but the rwg is the Pearson correlation.

If the weights parameter is specified, the pooled correlations are found by weighting the groups by the specified weight, rather than sample size.

Confidence values and significance of $r_{xy_{wg}}$, pwg, reflect the pooled number of cases within groups, while $r_{xy_{bg}}$, pbg, the number of groups. These are not corrected for multiple comparisons.

`withinBetween` is an example data set of the mixture of within and between group correlations. `sim.multilevel` will generate simulated data with a multilevel structure.

The statsBy.boot function will randomize the grouping variable ntrials times and find the statsBy output. This can take a long time and will produce a great deal of output. This output can then be summarized for relevant variables using the statsBy.boot.summary function specifying the variable of interest. These two functions are useful in order to find if the mere act of grouping leads to large between group correlations.

Consider the case of the relationship between various tests of ability when the data are grouped by level of education (statsBy(sat.act,"education")) or when affect data are analyzed within and between an affect manipulation (statsBy(flat,group="Film")). Note in this latter example, that because subjects were randomly assigned to Film condition for the pretest, that the pretest ICC1s cluster around 0.

faBy uses the output of statsBy to perform a factor analysis on the correlation matrix within each group. If the free parameter is FALSE, then each solution is rotated towards the group solution (as much as possible). The output is a list of each factor solution, as well as a summary matrix of loadings and interfactor correlations for all groups.

Value

means	The means for each group for each variable.
sd	The standard deviations for each group for each variable.
n	The number of cases for each group and for each variable.
ICC1	The intraclass correlation reflects the amount of total variance associated with the grouping variable.
ICC2	The intraclass correlation (2) reflecting how much the groups means differ.
ci1	The confidence intervals for the ICC1
ci2	The confidence intervals for the ICC2
F	The F from a one-way anova of group means.
rwg	The pooled within group correlations.
ci.wg	The confidence intervals of the pooled within group correlations.
rbg	The sample size weighted between group correlations.
c.bg	The confidence intervals of the rbg values
etawg	The correlation of the data with the within group values.
etabg	The correlation of the data with the group means.
pbg	The probability of the between group correlation
pwg	The probability of the within group correlation
r	In the case that we want the correlations in each group, r is a list of the within group correlations for every group. Set cors=TRUE
within	is just another way of displaying these correlations. within is a matrix which reports the lower off diagonal correlations as one row for each group.
pooled	The sample size weighted correlations. This is just within weighted by the sample sizes. The cors option must be set to TRUE to get this. See the note.
slope	The within group slopes – if using cor="cov" these will be the b weights

Note

If finding polychoric correlations, the two estimates of the pooled within group correlations will differ, for the pooled value is the weighted polychoric correlation, but the rwg is the Pearson correlation. As of March, 2020 the average is based upon the weighted FisherZ correlation (back transformed) rather than the average correlation.

The value of rbg (the between group correlation) is the group size weighted correlation. This is not the same as just finding the correlation of the group means (i.e. cor(means)).

The statsBy.boot function will sometimes fail if sampling with replacement because if the group sizes differ drastically, some groups will be empty. In this case, sample without replacement.

The statsBy.boot function can take a long time. (As I am writing this, I am running 1000 replications of a problem with 64,000 cases and 84 groups. It is taking about 3 seconds per replication on a MacBook Pro.)

The faBy function takes the output of statsBy (with the cors=TRUE option) and then factors each individual subject. By default, the solutions are organized so that the factors "match" the group solution in terms of their order. It is also possible to attempt to force the solutions to match by order and also by using the TargetQ rotation function. (free=FALSE)

Author(s)

William Revelle

References

Pedhazur, E.J. (1997) Multiple regression in behavioral research: explanation and prediction. Harcourt Brace.

See Also

describeBy and the functions within the multilevel package.

Examples

```
#Taken from Pedhazur, 1997
pedhazur <- structure(list(Group = c(1L, 1L, 1L, 1L, 1L, 2L, 2L, 2L, 2L,
2L), X = c(5L, 2L, 4L, 6L, 3L, 8L, 5L, 7L, 9L, 6L), Y = 1:10),
class = "data.frame", row.names = c(NA, -10L))

pedhazur
ped.stats <- statsBy(pedhazur,"Group")
ped.stats


#Now do this for the sat.act data set
sat.stats <- statsBy(sat.act,c("education","gender"),cors=TRUE)  #group by two grouping variables
print(sat.stats,short=FALSE)
lowerMat(sat.stats$pbg)  #get the probability values

#show means by groups
round(sat.stats$mean)
```

```
#Do separate factor analyses for each group
sb.bfi <- statsBy(bfi[1:10], group=bfi$gender,cors=TRUE)
faBy(sb.bfi,1)  #one factor per group
faBy(sb.bfi,2) #two factors per group
```

---

structure.diagram *Draw a structural equation model specified by two measurement models and a structural model*

Description

Graphic presentations of structural equation models are a very useful way to conceptualize sem and confirmatory factor models. Given a measurement model on x (xmodel) and on y (ymodel) as well as a path model connecting x and y (phi), draw the graph. If the ymodel is not specified, just draw the measurement model (xmodel + phi). If the Rx or Ry matrices are specified, show the correlations between the x variables, or y variables.

Perhaps even more usefully, the function returns a model appropriate for running directly in the *sem package* written by John Fox or the *lavaan* package by Yves Rosseel. For this option to work directly, it is necessary to specfy that errrors=TRUE.

Input can be specified as matrices or the output from fa, factanal, or a rotation package such as *GPArotation*.

For symbolic graphs, the input matrices can be character strings or mixtures of character strings and numeric vectors.

As an option, for those without Rgraphviz installed, structure.sem will just create the sem model and skip the graph. (This functionality is now included in structure.diagram.)

structure.diagram will draw the diagram without using Rgraphviz and is probably the preferred option. structure.graph will be removed eventually.

lavaan.diagram will draw either cfa or sem results from the lavaan package. It has been tested for cfa, sem and mimic type output. It takes the output object from *lavaan* and then calls structure.diagram.

Usage

```
structure.diagram(fx, Phi=NULL,fy=NULL,labels=NULL,cut=.3,errors=FALSE,simple=TRUE,
   regression=FALSE,lr=TRUE,Rx=NULL,Ry=NULL,digits=1,e.size=.1,
    main="Structural model", ...)
structure.graph(fx,  Phi = NULL,fy = NULL, out.file = NULL, labels = NULL, cut = 0.3,
   errors=TRUE, simple=TRUE,regression=FALSE, size = c(8, 6),
    node.font = c("Helvetica", 14), edge.font = c("Helvetica", 10),
    rank.direction = c("RL", "TB", "LR", "BT"), digits = 1,
     title = "Structural model", ...)
structure.sem(fx,  Phi = NULL, fy = NULL,out.file = NULL, labels = NULL,
     cut = 0.3, errors=TRUE, simple=TRUE,regression=FALSE)
```

```
lavaan.diagram(fit,main,e.size=.1,...)
sem.diagram(fit,main="A SEM from the sem package",...)
sem.graph(fit,out.file=NULL,main= "A SEM from the sem package",...)
```

Arguments

fx	a factor model on the x variables.
Phi	A matrix of directed relationships. Lower diagonal values are drawn. If the upper diagonal values match the lower diagonal, two headed arrows are drawn. For a single, directed path, just the value may be specified.
fy	a factor model on the y variables (can be empty)
Rx	The correlation matrix among the x variables
Ry	The correlation matrix among the y variables
out.file	name a file to send dot language instructions.
labels	variable labels if not specified as colnames for the matrices
cut	Draw paths for values > cut
fit	The output from a lavaan cfa or sem
errors	draw an error term for observerd variables
simple	Just draw one path per x or y variable
regression	Draw a regression diagram (observed variables cause Y)
lr	Direction of diagram is from left to right (lr=TRUE, default) or from bottom to top (lr=FALSE)
e.size	size of the ellipses in structure.diagram
main	main title of diagram
size	page size of graphic
node.font	font type for graph
edge.font	font type for graph
rank.direction	Which direction should the graph be oriented
digits	Number of digits to draw
title	Title of graphic
...	other options to pass to Rgraphviz

Details

The recommended function is structure.diagram which does not use Rgraphviz but which does not produce dot code either.

All three structure function return a matrix of commands suitable for using in the sem or lavaan packages. (Specify errors=TRUE to get code that will run directly in the sem package.)

The structure.graph output can be directed to an output file for post processing using the dot graphic language but requires that Rgraphviz is installed.

lavaan.diagram will create sem, cfa, or mimic diagrams depending upon the lavaan input.

sem.diagram and sem.graph convert the output from a simple CFA done with the sem package and draw them using structure.diagram or structure.graph. lavaan.diagram converts the output (fit) from a simple CFA done with the lavaan package and draws them using structure.diagram. The figure is organized to show the appropriate paths between:

The correlations between the X variables (if Rx is specified)
The X variables and their latent factors (if fx is specified)
The latent X and the latent Y (if Phi is specified)
The latent Y and the observed Y (if fy is specified)
The correlations between the Y variables (if Ry is specified)

A confirmatory factor model would specify just fx and Phi, a structural model would include fx, Phi, and fy. The raw correlations could be shown by just including Rx and Ry.

lavaan.diagram may be called from the diagram function which also will call fa.diagram, omega.diagram or iclust.diagram, depending upon the class of the fit.

Other diagram functions include fa.diagram, omega.diagram. All of these functions use the various dia functions such as dia.rect, dia.ellipse, dia.arrow, dia.curve, dia.curved.arrow, and dia.shape.

Value

sem	(invisible) a model matrix (partially) ready for input to John Fox's sem package. It is of class "mod" for prettier output.
lavaan	(invisible) A model specification for the lavaan package.
dotfile	If out.file is specified, a dot language file suitable for using in a dot graphics program such as graphviz or Omnigraffle.

A graphic structural diagram in the graphics window

Author(s)

William Revelle

See Also

fa.graph, omega.graph, sim.structural to create artificial data sets with particular structural properties.

Examples

```
#A set of measurement and structural models
#First set up the various matrices
fx <-  matrix(c(.9,.8,.7,rep(0,9), .6,.7,-.8,rep(0,9),.5,.6,.4),ncol=3)
fy <- matrix(c(.9,.8,.6,rep(0,4),.6,.8,-.7),ncol=2)
Phi <- matrix(c(1,.35,0,0,0,
                .35,1,.5,0,0,
                0,.5, 1,0,0,
                .7,-.6, 0, 1,0,
                .0, 0, .4,0,1 ),ncol=5,byrow=TRUE)
#now draw a number of models
```

```
f1 <- structure.diagram(fx,main = "A measurement model for x")
f2 <- structure.diagram(fx,Phi, main = "A measurement model for x")
f3 <- structure.diagram(fy=fy, main = "A measurement model for y")
f4 <- structure.diagram(fx,Phi,fy,main="A structural path diagram")
f5 <- structure.diagram(fx,Phi,fy,main="A structural path diagram",errors=TRUE)

#a mimic model
fy <- matrix(c(.9,.8,.6,rep(0,4),.6,.8,-.7),ncol=2)
fx <- matrix(c(.6,.5,0,.4),ncol=2)
mimic <- structure.diagram(fx,fy=fy,simple=FALSE,errors=TRUE, main="A mimic diagram")

fy <- matrix(c(rep(.9,8),rep(0,16),rep(.8,8)),ncol=2)
structure.diagram(fx,fy=fy, e.size=.05)

#symbolic input
X2 <- matrix(c("a",0,0,"b","e1",0,0,"e2"),ncol=4)
colnames(X2) <- c("X1","X2","E1","E2")
phi2 <- diag(1,4,4)
phi2[2,1] <- phi2[1,2] <- "r"
f2 <- structure.diagram(X2,Phi=phi2,errors=FALSE,main="A symbolic model")

#symbolic input with error
X2 <- matrix(c("a",0,0,"b"),ncol=2)
colnames(X2) <- c("X1","X2")
phi2 <- diag(1,2,2)
phi2[2,1] <- phi2[1,2] <- "r"
f3 <- structure.diagram(X2,Phi=phi2,main="an alternative representation",e.size=.4)

#and yet another one
X6 <- matrix(c("a","b","c",rep(0,6),"d","e","f"),nrow=6)
colnames(X6) <- c("L1","L2")
rownames(X6) <- c("x1","x2","x3","x4","x5","x6")
Y3 <- matrix(c("u","w","z"),ncol=1)
colnames(Y3) <- "Y"
rownames(Y3) <- c("y1","y2","y3")
phi21 <- matrix(c(1,0,"r1",0,1,"r2",0,0,1),ncol=3)
colnames(phi21) <- rownames(phi21) <-  c("L1","L2","Y")
f4 <- structure.diagram(X6,phi21,Y3)

###the following example is not run but is included to show how to work with lavaan

library(lavaan)
mod.1 <- 'A =~ A1 + A2 + A3 + A4 + A5
         C =~ C1 + C2 + C3 + C4 + C5
         E =~ E1 +E2 + E3 + E4 +E5'
fit.1 <- sem(mod.1,bfi[complete.cases(bfi),],std.lv=TRUE)
lavaan.diagram(fit.1)   #a normal cfa

#compare with
f3 <- fa(bfi[complete.cases(bfi),1:15],3)
fa.diagram(f3)
```

```
#a sem model
mod.2 <- 'A =~ A1 + A2 + A3 + A4 + A5
         C =~ C1 + C2 + C3 + C4 + C5
         E =~ E1 +E2 + E3 + E4 +E5
         E ~ A + C '

fit.2 <- sem(mod.2,bfi[complete.cases(bfi),],std.lv=TRUE)
lavaan.diagram(fit.2, cut=0,simple=FALSE,main="sem  model")  # A SEM Model


#a mimic model
mod.3 <- 'A =~ A1 + A2 + A3 + A4 + A5
         C =~ C1 + C2 + C3 + C4 + C5
         E =~ E1 +E2 + E3 + E4 +E5
         A ~ age + gender
         C ~ age + gender
         E ~ age + gender'

fit.3 <- sem(mod.3,bfi[complete.cases(bfi),],std.lv=TRUE)
lavaan.diagram(fit.3, cut=0,simple=FALSE,main="mimic model", e.size=.03)





# and finally, a regression model
X7 <- matrix(c("a","b","c","d","e","f"),nrow=6)
f5 <- structure.diagram(X7,regression=TRUE,main = "Regression model")

#and a really messy regession model
x8 <- c("b1","b2","b3")
r8 <- matrix(c(1,"r12","r13","r12",1,"r23","r13","r23",1),ncol=3)
f6<- structure.diagram(x8,Phi=r8,regression=TRUE,main="Regression model")
```

`structure.list` *Create factor model matrices from an input list*

Description

When creating a structural diagram or a structural model, it is convenient to not have to specify all of the zero loadings in a structural matrix. structure.list converts list input into a design matrix. phi.list does the same for a correlation matrix. Factors with NULL values are filled with 0s.

Usage

```
structure.list(nvars, f.list,f=NULL, f.labels = NULL, item.labels = NULL)
phi.list(nf,f.list, f.labels = NULL)
```

Arguments

nvars	Number of variables in the design matrix
f.list	A list of items included in each factor (for structure.list, or the factors that correlate with the specified factor for phi.list
f	prefix for parameters – needed in case of creating an X set and a Y set
f.labels	Names for the factors
item.labels	Item labels
nf	Number of factors in the phi matrix

Details

This is almost self explanatory. See the examples.

Value

factor.matrix	a matrix of factor loadings to model

See Also

structure.graph for drawing it, or sim.structure for creating this data structure.

Examples

```
fx <- structure.list(9,list(F1=c(1,2,3),F2=c(4,5,6),F3=c(7,8,9)))
fy <- structure.list(3,list(Y=c(1,2,3)),"Y")
phi <- phi.list(4,list(F1=c(4),F2=c(1,4),F3=c(2),F4=c(1,2,3)))
fx
phi
fy
```

---

superMatrix *Form a super matrix from two sub matrices.*

Description

Given the matrices nXm, and jYk, form the super matrix of dimensions (n+j) and (m+k) with with elements x and y along the super diagonal. Useful when considering structural equations. The measurement models x and y can be combined into a larger measurement model of all of the variables. If either x or y is a list of matrices, then recursively form a super matrix of all of those elements. superCor will form a matrix from two matrices and the intercorrelation of the elements of the two.

Usage

```
superMatrix(x,y)
superCor(x,y=NULL, xy=NULL)
super.matrix(x, y)  #Deprecated
```

Arguments

x	A n x m matrix or a list of such matrices, or the output object from `link{scoreOverlap}`
y	A j x k matrix or a list of such matrices
xy	A n x k matrix

Details

Several functions, e.g., `sim.structural`,`structure.graph`, `make.keys` use matrices that can be thought of as formed from a set of submatrices. In particular, when using `make.keys` in order to score a set of items (`scoreItems` or `scoreOverlap`) or to form specified clusters (`cluster.cor`), it is convenient to define different sets of scoring keys for different sets of items and to combine these scoring keys into one super key.

When developing scales and examining items using `bestScales` it is sometimes helpful to combine the matrix output from `scoreOverlap` with the original correlation matrix. Thus, let x = correlation of the scales from `scoreOverlap`, y = the correlations of the items used to form the scales, and xy = the correlation of the scales with the items.

Value

A (n+j) x (m +k) matrix with appropriate row and column names

Author(s)

William Revelle

See Also

`sim.structural`,`structure.graph`, `make.keys`

Examples

```
mx <- matrix(c(.9,.8,.7,rep(0,4),.8,.7,.6),ncol=2)
my <- matrix(c(.6,.5,.4))

colnames(mx) <- paste("X",1:dim(mx)[2],sep="")
rownames(mx) <- paste("Xv",1:dim(mx)[1],sep="")
colnames(my) <- "Y"
rownames(my) <- paste("Yv",1:3,sep="")
mxy <- superMatrix(mx,my)
#show the use of a list to do this as well
key1 <- make.keys(6,list(first=c(1,-2,3),second=4:6,all=1:6))  #make a scoring key
key2 <- make.keys(4,list(EA=c(1,2),TA=c(3,4)))
superMatrix(list(key1,key2))
```

```
r <- cor(bfi[1:15],use="pairwise")
bfi.scores <- scoreOverlap(bfi.keys[1:2], r,select=FALSE) #note the select = FALSE
R <- superCor(bfi.scores,r)
lowerMat(R)
#or to just get the scale correlations with the items
R <- superCor(bfi.scores)
round(R,2)
```

`table2matrix` *Convert a table with counts to a matrix or data.frame representing those counts.*

Description

Some historical sets are reported as summary tables of counts in a limited number of bins. Transforming these tables to data.frames representing the original values is useful for pedagogical purposes. (E.g., transforming the original Galton table of height x cubits in order to demonstrate regression.) The column and row names must be able to be converted to numeric values.

Usage

```
table2matrix(x, labs = NULL)
table2df(x, count=NULL,labs = NULL)
```

Arguments

`x`	A two dimensional table of counts with row and column names that can be converted to numeric values.
`count`	if present, then duplicate each row count times
`labs`	Labels for the rows and columns. These will be used for the names of the two columns of the resulting matrix

Details

The original Galton (1888) of heights by cubits (arm length) is in tabular form. To show this as a correlation or as a scatter plot, it is useful to convert the table to a matrix or data frame of two columns.

This function may also be used to convert an item response pattern table into a data table. e.g., the Bock data set `bock`.

Value

A matrix (or data.frame) of sum(x) rows and two columns.

Author(s)

William Revelle

See Also

cubits and bock data sets

Examples

```
if(require(psychTools)) {
data(cubits)
cubit <- table2matrix(psychTools::cubits,labs=c("height","cubit"))
describe(cubit)
ellipses(cubit,n=1)
data(bock)
responses <- table2df(bock.table[,2:6],count=bock.table[,7],labs= paste("lsat6.",1:5,sep=""))
describe(responses)
}
```

Tal_Or *Data set testing causal direction in presumed media influence*

Description

Nurit Tal-Or, Jonanathan Cohen, Yariv Tasfati, and Albert Gunther (2010) examined the presumed effect of media on other people and change in attitudes. This data set is from Study 2, and examined the effect of presumed influence of the media upon subsequent actions. It is used as an example of mediation by Hayes (2013) and for the mediate function.

Usage

```
data("Tal.Or")
```

Format

A data frame with 123 observations on the following 6 variables.

`cond` Experimental Condition: 0 low media importance, 1 high media importance

`pmi` Presumed media influence (based upon the mean of two items

`import` Importance of the issue

`reaction` Subjects rated agreement about possible reactions to the story (mean of 4 items).

`gender` 1 = male, 2 = female

`age` a numeric vector

Details

Tal-Or et al. (2010) examined the presumed effect of the media in two experimental studies. These data are from study 2. '... perceptions regarding the influence of a news story about an expected shortage in sugar were manipulated indirectly, by manipulating the perceived exposure to the news story, and behavioral intentions resulting from the story were consequently measured." (p 801).

Source

The data were downloaded from the webpages of Andrew Hayes (https://www.afhayes.com/public/hayes2018data.zip) supporting the first and second edition of his book. The name of the original data set was pmi. (Gender was recoded to reflect the number of X chromosomes).

The original data are from Nurit Tal-Or, Jonathan Cohen, Yariv Tsfati, and Albert C. Gunther and are used with their kind permission.

References

Nurit Tal-Or, Jonathan Cohen, Yariv Tsfati and Albert C. Gunther (2010), Testing Causal Direction in the Influence of Presumed Media Influence, Communication Research, 37, 801-824.

Hayes, Andrew F. (2013) Introduction to mediation, moderation, and conditional process analysis: A regression-based approach. Guilford Press.

Examples

```
data(Tal.Or)
mediate(reaction ~ cond + (pmi), data =Tal.Or,n.iter=50)
```

---

test.irt *A simple demonstration (and test) of various IRT scoring algorthims.*

Description

Item Response Theory provides a number of alternative ways of estimating latent scores. Here we compare 6 different ways to estimate the latent variable associated with a pattern of responses. Originally developed as a test for scoreIrt, but perhaps useful for demonstration purposes. Items are simulated using `sim.irt` and then scored using factor scores from `factor.scores` using statistics found using `irt.fa`, simple weighted models for 1 and 2 PL and 2 PN. Results show almost perfect agreement with estimates from MIRT and ltm for the dichotomous case and with MIRT for the polytomous case. (Results from ltm are unstable for the polytomous case, sometimes agreeing with `scoreIrt` and MIRT, sometimes being much worse.)

Usage

```
test.irt(nvar = 9, n.obs = 1000, mod = "logistic",type="tetra", low = -3, high = 3,
 seed = NULL)
```

Arguments

`nvar`	Number of variables to create (simulate) and score
`n.obs`	Number of simulated subjects
`mod`	"logistic" or "normal" theory data are generated
`type`	"tetra" for dichotomous, "poly" for polytomous

low	items range from low to high
high	items range from low to high
seed	Set the random number seed using some non-nul value. Otherwise, use the existing sequence of random numbers

Details

n.obs observations (0/1) on nvar variables are simulated using either a logistic or normal theory model. Then, a number of different scoring algorithms are applied and shown graphically. Requires the ltm package to be installed to compare ltm scores.

Value

A dataframe of scores as well as the generating theta true score. A graphic display of the correlations is also shown.

Author(s)

William Revelle

See Also

scoreIrt,irt.fa

Examples

```
#not run
#test.irt(9,1000)
```

---

test.psych *Testing of functions in the psych package*

Description

Test to make sure the psych functions run on basic test data sets

Usage

```
test.psych(first=1,last=5,short=TRUE,all=FALSE,fapc=FALSE)
```

Arguments

first	first=1: start with dataset first
last	last=5: test for datasets until last
short	short=TRUE - don't return any analyses
all	To get around a failure on certain Solaris 32 bit systems, all=FALSE is the default
fapc	if fapc=TRUE, then do a whole series of tests of factor and principal component extraction and rotations.

Details

When modifying the psych package, it is useful to make sure that adding some code does not break something else. The test.psych function tests the major functions on various standard data sets. It also shows off a number of the capabilities of the psych package.

Uses 5 standard data sets:
USArrests Violent Crime Rates by US State (4 variables)
attitude The Chatterjee-Price Attitude Data
Harman23.cor$cov Harman Example 2.3 8 physical measurements
Harman74.cor$cov Harman Example 7.4 24 mental measurements
ability.cov$cov 8 Ability and Intelligence Tests

It also uses the bfi and ability data sets from psych

Value

`out` if short=FALSE, then list of the output from all functions tested

Warning

Warning messages will be thrown by fa.parallel and sometimes by fa for random datasets.

Note

Although test.psych may be used as a quick demo of the various functions in the psych packge, in general, it is better to try the specific functions themselves. The main purpose of test.psych is to make sure functions throw error messages or correct for weird conditions.

The datasets tested are part of the standard R data sets and represent some of the basic problems encountered.

When version 1.1.10 was released, it caused errors when compiling and testing on some Solaris 32 bit systems. The all option was added to avoid this problem (since I can't replicate the problem on Macs or PCs). all=TRUE adds one more test, for a non-positive definite matrix.

Author(s)

William Revelle

Examples

```
#test <- test.psych()
#not run
#test.psych(all=TRUE)
#    f3 <- fa(bfi[1:15],3,n.iter=5)
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="Varimax")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="varimax")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="bifactor")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="varimin")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="bentlerT")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="geominT")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="equamax")
```

```
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="Promax")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="cluster")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="biquartimin")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="equamax")
#    f3 <- fa(bfi[1:15],3,n.iter=5,rotate="Promax")
#
#     fpoly <- fa(bfi[1:10],2,n.iter=5,cor="poly")
#     f1 <- fa(psychTools::ability,n.iter=4)
#     f1p <- fa(psychTools::ability,n.iter=4,cor="tet")
```

`testRetest` *Find various test-retest statistics, including test, person and item reliability*

Description

Given two presentations of a test, it is straightforward to find the test-retest reliablity, as well as the item reliability and person stability across items. Using the multi-level structure of the data, it is also possible to do a variance deomposition to find variance components for people, items, time, people x time, people x items, and items x time as well as the residual variance. This leads to various generalizability cofficients.

Usage

```
testRetest(t1,t2=NULL,keys=NULL,id="id", time=  "time", select=NULL,
check.keys=TRUE, warnings=TRUE,lmer=TRUE,sort=TRUE)
```

Arguments

`t1`	a data.frame or matrix for the first time of measurement.
`t2`	a data.frame or matrix for the second time of measurement. May be NULL if time is specifed in t1
`keys`	item names (or locations) to analyze, preface by "-" to reverse score.
`id`	subject identification codes to match across time
`time`	The name of the time variable identifying time 1 or 2 if just one data set is supplied.
`select`	A subset of items to analyze
`check.keys`	If TRUE will automatically reverse items based upon their correlation with the first principal component. Will throw a warning when doing so, but some people seem to miss this kind of message.
`warnings`	If TRUE, then warn when items are reverse scored
`lmer`	If TRUE, include the lmer variance decomposition. By default, this is true, but this can lead to long times for large data sets.
`sort`	If TRUE, the data are sorted by id and time. This allows for random ordering of data, but will fail if ids are duplicated in different studies. In that case, we need to add a constant to the ids for each study. See the last example.

Details

There are many ways of measuring reliability. Test - Retest is one way. If the time interval is very short (or immediate), this is known as a dependability correlation, if the time interval is longer, a stability coefficient. In all cases, this is a correlation between two measures at different time points. Given the multi-level nature of these data, it is possible to find variance components associated with individuals, time, item, and time by item, etc. This leads to several different estimates of reliability (see `multilevel.reliability` for a discussion and references).

It is also possible to find the subject reliability across time (this is the correlation across the items at time 1 with time 2 for each subject). This is a sign of subject reliability (Wood et al, 2017). Items can show differing amounts of test-retest reliability over time. Unfortunately, the within person correlation has problems if people do not differ very much across items. If all items are in the same keyed direction, and measuring the same construct, then the response profile for an individual is essentially flat. This implies that the even with almost perfect reproducibility, that the correlation can actually be negative. The within person distance (d2) across items is just the mean of the squared differences for each item. Although highly negatively correlated with the rqq score, this does distinguish between random responders (high dqq and low rqq) from consistent responders with lower variance (low dqq and low rqq).

Several individual statistics are reported in the scores object. These can be displayed by using `pairs.panels` for a graphic display of the relationship and ranges of the various measures.

Although meant to decompose the variance for tests with items nested within tests, if just given two tests, the variance components for people and for time will also be shown. The resulting variance ratio of people to total variance is the intraclass correlation between the two tests. See also `ICC` for the more general case.

Value

`r12`	The time 1 time 2 correlation of scaled scores across time
`alpha`	Guttman's lambda 3 (aka alpha) and lambda 6* (item reliabilities based upon smcs) are found for the scales at times 1 and 2.
`rqq`	The within subject test retest reliability of response patterns over items
`item.stats`	Item reliabilities, item loadings at time 1 and 2, item means at time 1 and time 2
`scores`	A data frame of principal component scores at time 1 and time 2, raw scores from time 1 and time 2, the within person standard deviation for time 1 and time 2, and the rqq and dqq scores for each subject.
`xy.df`	If given separate t1 and t2 data.frames, this is combination suitable for using `multilevel.reliability`
`key`	A key vector showing which items have been reversed
`ml`	The multilevel output

Note

lmer=TRUE is the default and will do the variance decomposition using lmer. This will take some time. For 3032 cases with 10 items from the msqR and sai data set, this takes 92 seconds, but just .63 seconds if lmer = FALSE. For the 1895 subjects with repeated measures on the `sai`, it takes 85 seconds with lmer and .38 without out lmer.

In the case of just two tests (no items specified), the item based statistics (alpha, rqq, item.stats, scores, xy.df) are not reported.

Several examples are given. The first takes 200 cases from the sai data set. Subjects were given the link[psychTools]{sai} twice with an intervening mood manipulation (four types of short film clips, with or without placebo/caffeine). The test retest stability of the sai are based upon the 20 sai items. The second example compares the scores of the 10 sai items that overlap with 10 items from the msqR data set from the same study. link[psychTools]{sai} and msqR were given immediately after each other and although the format differs slightly, can be seen as measures of dependability.

The third example considers the case of the Impulsivity scale from the Eysenck Personality Inventory. This is a nice example of how different estimates of reliability will differ. The alpha reliability is .51 but the test-retest correlation across several weeks is .71! That is to say, the items don't correlate very much with each other (alpha) but do with themselves across time (test-retest).

The fourth example takes the same data and jumble the subjects (who are identified by their ids). This result should be the same as the prior result because the data are automatically sorted by id.

Although typically, there are as many subjects at time 1 as time 2, testRetest will handle the case of a different number of subjects. The data are first sorted by time and id, and then those cases from time 1 that are matched at time 2 are analyzed. It is important to note that the subject id numbers must be unique.

Author(s)

William Revelle

References

Cattell, R. B. (1964). Validity and reliability: A proposed more basic set of concepts. Journal of Educational Psychology, 55(1), 1 - 22. doi: 10.1037/h0046462

Cranford, J. A., Shrout, P. E., Iida, M., Rafaeli, E., Yip, T., & Bolger, N. (2006). A procedure for evaluating sensitivity to within-person change: Can mood measures in diary studies detect change reliably? Personality and Social Psychology Bulletin, 32(7), 917-929.

DeSimone, J. A. (2015). New techniques for evaluating temporal consistency. Organizational Research Methods, 18(1), 133-152. doi: 10.1177/1094428114553061

Revelle, W. and Condon, D. Reliability from alpha to omega: A tutorial. Psychological Assessment, 31 (12) 1395-1411.

Revelle, W. (in preparation) An introduction to psychometric theory with applications in R. Springer. (Available online at https://personality-project.org/r/book/).

Shrout, P. E., & Lane, S. P. (2012). Psychometrics. In Handbook of research methods for studying daily life. Guilford Press.

Wood, D., Harms, P. D., Lowman, G. H., & DeSimone, J. A. (2017). Response speed and response consistency as mutually validating indicators of data quality in online samples. Social Psychological and Personality Science, 8(4), 454-464. doi: 10.1177/1948550617703168

See Also

alpha, omega scoreItems, cor2

Examples

```
  #for faster compiling, dont test
#lmer set to FALSE for speed.
#set lmer to TRUE to get variance components
if(require(psychTools)){
sai.xray <- subset(psychTools::sai,psychTools::sai$study=="XRAY")
#The case where the two measures are identified by time
#automatically reverses items but throws a warning
stability <- testRetest(sai.xray[-c(1,3)],lmer=FALSE)
stability  #show the results
#get a second data set
sai.xray1 <- subset(sai.xray,sai.xray$time==1)
msq.xray <- subset(psychTools::msqR,
 (psychTools::msqR$study=="XRAY") & (psychTools::msqR$time==1))
select <- colnames(sai.xray1)[is.element(colnames(sai.xray1 ),colnames(psychTools::msqR))]

select <-select[-c(1:3)]  #get rid of the id information
#The case where the two times are in the form x, y

dependability <-  testRetest(sai.xray1,msq.xray,keys=select,lmer=FALSE)
dependability  #show the results

#now examine the Impulsivity subscale of the EPI
#use the epiR data set which includes epi.keys

#Imp <- selectFromKeys(epi.keys$Imp)   #fixed temporarily with
Imp <- c("V1", "V3", "V8", "V10","V13" ,"V22", "V39" , "V5" , "V41")
imp.analysis <- testRetest(psychTools::epiR,select=Imp) #test-retest = .7, alpha=.51,.51
imp.analysis

#demonstrate random ordering  -- the results should be the same
n.obs <- NROW(psychTools::epiR)
set.seed(42)
ss <- sample(n.obs,n.obs)
temp.epi <- psychTools::epiR
temp.epi <-char2numeric(temp.epi)  #make the study numeric
temp.epi$id <- temp.epi$id + 300*temp.epi$study
random.epi <- temp.epi[ss,]
random.imp.analysis <- testRetest(random.epi,select=Imp)
}
```

`tetrachoric` *Tetrachoric, polychoric, biserial and polyserial correlations from various types of input*

Description

The tetrachoric correlation is the inferred Pearson Correlation from a two x two table with the assumption of bivariate normality. The polychoric correlation generalizes this to the n x m table.

Particularly important when doing Item Response Theory or converting comorbidity statistics using normal theory to correlations. Input may be a 2 x 2 table of cell frequencies, a vector of cell frequencies, or a data.frame or matrix of dichotomous data (for tetrachoric) or of numeric data (for polychoric). The biserial correlation is between a continuous x variable and a dichotmous y variable, which is assumed to have resulted from a dichotomized normal variable. Biserial is a special case of the polyserial correlation, which is the inferred latent correlation between a continuous variable (X) and a ordered categorical variable (e.g., an item response). Input for these later two are data frames or matrices. Requires the mnormt package.

Usage

```
tetrachoric(x,y=NULL,correct=.5,smooth=TRUE,global=TRUE,weight=NULL,na.rm=TRUE,
     delete=TRUE)
polychoric(x,y=NULL,smooth=TRUE,global=TRUE,polycor=FALSE,ML=FALSE, std.err=FALSE,
     weight=NULL,correct=.5,progress=TRUE,na.rm=TRUE,  delete=TRUE,max.cat=8)
biserial(x,y, weight=NULL)
polyserial(x,y, weight=NULL)
polydi(p,d,taup,taud,global=TRUE,ML = FALSE, std.err = FALSE,
     weight=NULL,progress=TRUE,na.rm=TRUE,delete=TRUE,correct=.5)
#deprecated  use polychoric instead
poly.mat(x, short = TRUE, std.err = FALSE, ML = FALSE)
```

Arguments

`x` The input may be in one of four forms:

a) a data frame or matrix of dichotmous data (e.g., the lsat6 from the bock data set) or discrete numerical (i.e., not too many levels, e.g., the big 5 data set, bfi) for polychoric, or continuous for the case of biserial and polyserial.

b) a 2 x 2 table of cell counts or cell frequencies (for tetrachoric) or an n x m table of cell counts (for both tetrachoric and polychoric).

c) a vector with elements corresponding to the four cell frequencies (for tetrachoric)

d) a vector with elements of the two marginal frequencies (row and column) and the comorbidity (for tetrachoric)

`y` A (matrix or dataframe) of discrete scores. In the case of tetrachoric, these should be dichotomous, for polychoric not too many levels, for biserial they should be discrete (e.g., item responses) with not too many (<10?) categories.

`correct` Correction value to use to correct for continuity in the case of zero entry cell for tetrachoric, polychoric, polybi, and mixed.cor. See the examples for the effect of correcting versus not correcting for continuity.

`smooth` if TRUE and if the tetrachoric/polychoric matrix is not positive definite, then apply a simple smoothing algorithm using cor.smooth

`global` When finding pairwise correlations, should we use the global values of the tau parameter (which is somewhat faster), or the local values (global=FALSE)? The local option is equivalent to the polycor solution, or to doing one correlation at a time. global=TRUE borrows information for one item pair from the other pairs using those item's frequencies. This will make a difference in the presence

	of lots of missing data. With very small sample sizes with global=FALSE and correct=TRUE, the function will fail (for as yet underdetermined reasons.
`polycor`	A no longer used option, kept to stop other packages from breaking.
`weight`	A vector of length of the number of observations that specifies the weights to apply to each case. The NULL case is equivalent of weights of 1 for all cases.
`short`	short=TRUE, just show the correlations, short=FALSE give the full hetcor output from John Fox's hetcor function if installed and if doing polychoric Deprecated
`std.err`	std.err=FALSE does not report the standard errors (faster) deprecated
`progress`	Show the progress bar (if not doing multicores)
`ML`	ML=FALSE do a quick two step procedure, ML=TRUE, do longer maximum likelihood — very slow! Deprecated
`na.rm`	Should missing data be deleted
`delete`	Cases with no variance are deleted with a warning before proceeding.
`max.cat`	The maximum number of categories to bother with for polychoric.
`p`	The polytomous input to polydi
`d`	The dichotomous input to polydi
`taup`	The tau values for the polytomous variables – if global=TRUE
`taud`	The tau values for the dichotomous variables – if globabl = TRUE

Details

Tetrachoric correlations infer a latent Pearson correlation from a two x two table of frequencies with the assumption of bivariate normality. The estimation procedure is two stage ML. Cell frequencies for each pair of items are found. In the case of tetrachorics, cells with zero counts are replaced with .5 as a correction for continuity (correct=TRUE).

The data typically will be a raw data matrix of responses to a questionnaire scored either true/false (tetrachoric) or with a limited number of responses (polychoric). In both cases, the marginal frequencies are converted to normal theory thresholds and the resulting table for each item pair is converted to the (inferred) latent Pearson correlation that would produce the observed cell frequencies with the observed marginals. (See `draw.tetra` and `draw.cor` for illustrations.)

This is a computationally intensive function which can be speeded up considerably by using multiple cores and using the parallel package. The number of cores to use when doing polychoric or tetrachoric may be specified using the options command. The greatest step up in speed is going from 1 cores to 2. This is about a 50% savings. Going to 4 cores seems to have about at 66% savings, and 8 a 75% savings. The number of parallel processes defaults to 2 but can be modified by using the `options` command: options("mc.cores"=4) will set the number of cores to 4.

`tetrachoric` and `polychoric` can find non-symmetric correlation matrices of set of x variables (coluumns) and y variable (rows). This is useful if extending a solution from a base set of items to a different set. See `fa.extension` for an application of this.

The tetrachoric correlation is used in a variety of contexts, one important one being in Item Response Theory (IRT) analyses of test scores, a second in the conversion of comorbity statistics to correlation coefficients. It is in this second context that examples of the sensitivity of the coefficient to the cell frequencies becomes apparent:

Consider the test data set from Kirk (1973) who reports the effectiveness of a ML algorithm for the tetrachoric correlation (see examples).

Examples include the lsat6 and lsat7 data sets in the bock data.

The polychoric function forms matrices of polychoric correlations by an local function (polyc) and will also report the tau values for each alternative. Earlier versions used John Fox's polychor function which has now been replaced by the polyc function.

For finding one polychoric correlation from a table, see the Olsson example (below).

polychoric replaces poly.mat and is recommended. poly.mat was an alternative wrapper to the polycor function.

biserial and polyserial correlations are the inferred latent correlations equivalent to the observed point-biserial and point-polyserial correlations (which are themselves just Pearson correlations).

The polyserial function is meant to work with matrix or dataframe input and treats missing data by finding the pairwise Pearson r corrected by the overall (all observed cases) probability of response frequency. This is particularly useful for SAPA procedures (https://www.sapa-project.org/) (Revelle et al. 2010, 2016, 2020) with large amounts of missing data and no complete cases. See also the International Cognitive Ability Resource (https://www.icar-project.org/) for similar data.

Ability tests and personality test matrices will typically have a cleaner structure when using tetrachoric or polychoric correlations than when using the normal Pearson correlation. However, if either alpha or omega is used to find the reliability, this will be an overestimate of the squared correlation of a latent variable with the observed variable.

A biserial correlation (not to be confused with the point-biserial correlation which is just a Pearson correlation) is the latent correlation between x and y where y is continuous and x is dichotomous but assumed to represent an (unobserved) continuous normal variable. Let p = probability of x level 1, and q = 1 - p. Let zp = the normal ordinate of the z score associated with p. Then, $rbi = rs * \sqrt{(pq)}/zp$.

As nicely discussed by MacCallum et al, 2002, if artificially dichotomizing at the mean, the point biserial will be .8 of the biserial (actually .798). Similarly, the phi coefficient (a Pearson on dichotmous data will be 2[arcsin(rho)/pi) of the real value (rho).

The 'ad hoc' polyserial correlation, rps is just $r = r * sqrt(n-1)/n)\sigma y/\sum(zpi)$ where zpi are the ordinates of the normal curve at the normal equivalent of the cut point boundaries between the item responses. (Olsson, 1982)

All of these were inspired by (and adapted from) John Fox's polycor package which should be used for precise ML estimates of the correlations. See, in particular, the hetcor function in the polychor package. The results from polychoric match the polychor answers to at least 5 decimals when using correct=FALSE, and global = FALSE.

Particularly for tetrachoric correlations from sets of data with missing data, the matrix will sometimes not be positive definite. Various smoothing alternatives are possible, the one done here is to do an eigen value decomposition of the correlation matrix, set all negative eigen values to 10 * .Machine$double.eps, normalize the positive eigen values to sum to the number of variables, and then reconstitute the correlation matrix. A warning is issued when this is done.

For very small data sets, the correction for continuity for the polychoric correlations can lead to difficulties, particularly if using the global=FALSE option, or if doing just one correlation at a time. Setting a smaller correction value (i.e., correct =.1) seems to help.

John Uebersax (2015) makes the interesting point that both polychoric and tetrachoric correlations should be called latent correlations or latent continuous correlations because of the way they are found and not tetrachoric or polychoric which is the way they were found in the past. That is, what is the correlation between two latent variables that when artificially broken into two (tetrachoric) or more (polychoric) values produces the n x n table of observed frequencies.

For combinations of continous, categorical, and dichotomous variables, see `mixed.cor`.

If using data with a variable number of response alternatives, it is necessary to use the global=FALSE option in polychoric. (This is set automatically if this condition is detected).

For relatively small samples with dichotomous data if some cells are empty, or if the resampled matrices are not positive semi-definite, warnings are issued. this leads to serious problems if using multi.cores (the default if using a Mac). The solution seems to be to not use multi.cores (e.g., options(mc.cores =1)

`mixedCor` which calls `polychoric` for the N = 4,000 with 145 items of the spi data set, on a Mac book Pro with a 2.4 GHz 8-core Intel i9, 1 core took 130 seconds, 2 cores, 68 seconds, 4 37, 8 22 and 16 22.8. Just finding the polychoric correlations (spi[11:145]) for 4 cores took 34 and for 8 cores, 22 seconds.

(Update in 2022: with an M1 max chip) mixedCor for all 145 variables: 1 core = 54, 2 cores 28, 4 core = 15.8 seconds, 8 cores= 8.9 seconds. For just the polychorics 4 cores = 13.7, 8 cores = 7.4 .

As the number of categories increases, the computation time does as well. For more than about 8 categories, the difference between a normal Pearson correlation and the polychoric is not very large*. However, the number of categories (num.cat) may be set to large values if desired. (Added for version 2.0.6).

*A recent article by Foldnes and Gronneberg suggests using polychoric is appropriate for categorical data even if the number of categories is large. This particularly the case for very non-normal distributions of the category frequencies.

Value

`rho`	The (matrix) of tetrachoric/polychoric/biserial correlations
`tau`	The normal equivalent of the cutpoints
`fixed`	If any correlations were adjusted for continuity, the total number of adjustments will be reported.

Note

For tetrachoric, in the degenerate case of a cell entry with zero observations, a correction for continuity is applied and .5 is added to the cell entry. A warning is issued. If correct=FALSE the correction is not applied. This correction is, by default, on. It can be adjusted by specifying a smaller value. See the examples.

For correct=FALSE, the results agree perfectly with John Fox's polycor function.

Switched to using sadmvn from the mnormt package to speed up by 50%.

Some small data sets with differing number of alternatives and missing values can lead to errors. This seems to be solved by setting the correct=0 option.

Author(s)

William Revelle

References

A. Gunther and M. Hofler. Different results on tetrachorical correlations in mplus and stata-stata announces modified procedure. Int J Methods Psychiatr Res, 15(3):157-66, 2006.

David Kirk (1973) On the numerical approximation of the bivariate normal (tetrachoric) correlation coefficient. Psychometrika, 38, 259-268.

Foldnes, N., & Gronneberg, S. (2021, April 1). The Sensitivity of Structural Equation Modeling With Ordinal Data to Underlying Non-Normality and Observed Distributional Forms. Psychological Methods. Advance online publication. http://dx.doi.org/10.1037/met0000385

MacCallum, Robert C. and Zhang, Shaobo and Preacher, Kristopher J. and Rucker, Derek D. (2002) On the practice of dichotomization of quantitative variables., Psychological Methods, 7, (1) 19-40.

Olsson, U. Maximum Likelihood Estimation of the Polychoric Correlation Coefficient, Psychometrika, 44:443-460.

U.Olsson, F.Drasgow, and N.Dorans (1982). The polyserial correlation coefficient. Psychometrika, 47:337-347.

Revelle, W., Wilt, J., and Rosenthal, A. (2010) Individual Differences in Cognition: New Methods for examining the Personality-Cognition Link In Gruszka, A. and Matthews, G. and Szymura, B. (Eds.) Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control, Springer.

Revelle, W, Condon, D.M., Wilt, J., French, J.A., Brown, A., and Elleman, L.G. (2016) Web and phone based data collection using planned missing designs. In Fielding, N.G., Lee, R.M. and Blank, G. (Eds). SAGE Handbook of Online Research Methods (2nd Ed), Sage Publcations

W. Revelle, E.M. Dworak and D.M. Condon (2020) Exploring the persome: The power of the item in understanding personality structure. Personality and Individual Differences, doi:10.1016/j.paid.2020.109905

See Also

`mixed.cor` to find the correlations between mixtures of continuous, polytomous, and dichtomous variables. See the `bigCor` function for very large correlation matrices (uses Pearson correlations). See also the polychor function in the polycor package. `irt.fa` uses the tetrachoric function to do item analysis with the `fa` factor analysis function. `draw.tetra` shows the logic behind a tetrachoric correlation (for teaching purpuses.)

Examples

```
#if(require(mnormt)) {
data(bock)
tetrachoric(lsat6)
polychoric(lsat6)  #values should be the same
tetrachoric(matrix(c(44268,193,14,0),2,2))  #MPLUS reports.24

#Do not apply continuity correction -- compare with previous analysis!
tetrachoric(matrix(c(44268,193,14,0),2,2),correct=0)
```

```
#the default is to add correct=.5 to 0 cells
tetrachoric(matrix(c(61661,1610,85,20),2,2)) #Mplus reports .35
tetrachoric(matrix(c(62503,105,768,0),2,2)) #Mplus reports -.10
tetrachoric(matrix(c(24875,265,47,0),2,2)) #Mplus reports  0

polychoric(matrix(c(61661,1610,85,20),2,2)) #Mplus reports .35
polychoric(matrix(c(62503,105,768,0),2,2)) #Mplus reports -.10
polychoric(matrix(c(24875,265,47,0),2,2)) #Mplus reports  0

#Do not apply continuity correction- compare with previous analysis
tetrachoric(matrix(c(24875,265,47,0),2,2), correct=0)
polychoric(matrix(c(24875,265,47,0),2,2), correct=0)  #the same result


#examples from Kirk 1973
#note that Kirk's tables have joint probability followed by marginals, but
#tetrachoric needs marginals followed by joint probability

tetrachoric(c(.5,.5,.333333))   #should be .5
tetrachoric(c(.5,.5,.1150267))  #should be -.75
tetrachoric(c(.5,.5,.397584))   #should e .8
tetrachoric(c(.158655254,.158655254,.145003)) #should be .99


#the example from Olsson, 1979
 x <- as.table(matrix(c(13,69,41,6,113,132,0,22,104),3,3))
 polychoric(x,correct=FALSE)
#Olsson reports rho = .49, tau row = -1.77, -.14 and tau col =  -.69, .67

#give a vector of two marginals and the comorbidity
tetrachoric(c(.2, .15, .1))
tetrachoric(c(.2, .1001, .1))
 #} else {
 #       message("Sorry, you must have mnormt installed")}

# 4 plots comparing biserial to point biserial and latent Pearson correlation
set.seed(42)
x.4 <- sim.congeneric(loads =c(.9,.6,.3,0),N=1000,short=FALSE)
y  <- x.4$latent[,1]
for(i in 1:4) {
x <- x.4$observed[,i]
r <- round(cor(x,y),1)
ylow <- y[x<= 0]
yhigh <- y[x > 0]
yc <- c(ylow,yhigh)
rpb <- round(cor((x>=0),y),2)
rbis <- round(biserial(y,(x>=0)),2)
ellipses(x,y,ylim=c(-3,3),xlim=c(-4,3),pch=21 - (x>0),
       main =paste("r = ",r,"rpb = ",rpb,"rbis =",rbis))

dlow <- density(ylow)
dhigh <- density(yhigh)
```

```
points(dlow$y*5-4,dlow$x,typ="l",lty="dashed")
lines(dhigh$y*5-4,dhigh$x,typ="l")
}

#show non-symmeteric results
if(require(psychTools)) {
test1 <- tetrachoric(psychTools::ability[,1:4],psychTools::ability[,5:10])
test2 <-  polychoric(psychTools::ability[,1:4],psychTools::ability[,5:10])
all <- tetrachoric(psychTools::ability[,1:10])

}
```

---

thurstone *Thurstone Case V scaling*

Description

Thurstone Case V scaling allows for a scaling of objects compared to other objects. As one of the cases considered by Thurstone, Case V makes the assumption of equal variances and uncorrelated distributions.

Usage

```
thurstone(x, ranks = FALSE, digits = 2)
```

Arguments

x	A square matrix or data frame of preferences, or a rectangular data frame or matrix rank order choices.
ranks	TRUE if rank orders are presented
digits	number of digits in the goodness of fit

Details

Louis L. Thurstone was a pioneer in psychometric theory and measurement of attitudes, interests, and abilities. Among his many contributions was a systematic analysis of the process of comparative judgment (thurstone, 1927). He considered the case of asking subjects to successively compare pairs of objects. If the same subject does this repeatedly, or if subjects act as random replicates of each other, their judgments can be thought of as sampled from a normal distribution of underlying (latent) scale scores for each object, Thurstone proposed that the comparison between the value of two objects could be represented as representing the differences of the average value for each object compared to the standard deviation of the differences between objects. The basic model is that each item has a normal distribution of response strength and that choice represents the stronger of the two response strengths. A justification for the normality assumption is that each decision represents the sum of many independent inputs and thus, through the central limit theorem, is normally distributed.

Thurstone considered five different sets of assumptions about the equality and independence of the variances for each item (Thurston, 1927). Torgerson expanded this analysis slightly by considering three classes of data collection (with individuals, between individuals and mixes of within

and between) crossed with three sets of assumptions (equal covariance of decision process, equal correlations and small differences in variance, equal variances).

The data may be either a square matrix of dataframe of preferences (as proportions with the probability of the column variable being chosen over the row variable) or a matrix or dataframe of rank orders (1 being prefered to 2, etc.)

The second example creates 100 random permutations of ranks 1-5. These data are then converted to a matrix of choices and then scaled. The goodness of fit is practically perfect, even though the data are meaningless.

This suggests a better goodness of fit test should be applied.

Value

GF	Goodness of fit 1 = 1 - sum(squared residuals/squared original) for lower off diagonal.
	Goodness of fit 2 = 1 - sum(squared residuals/squared original) for full matrix.
residual	square matrix of residuals (of class dist)
choice	The original choice data
...	

Author(s)

William Revelle

References

Thurstone, L. L. (1927) A law of comparative judgments. Psychological Review, 34, 273-286.

Revelle, W. An introduction to psychometric theory with applications in R. (in preparation), Springer. https://personality-project.org/r/book/

Examples

```
if(require(psychTools)) {
data(psychTools::vegetables)
thurstone(psychTools::veg)
#But consider the case of 100 random orders
set.seed((42))
ranks <- matrix(NA,nrow=100,ncol=5)
for(i in 1:100) ranks[i,] <- sample(5,5)
 t <- thurstone(ranks,TRUE)
 t   #show the fits
 t$hoice #show the choice matrix

}
```

tr — *Find the trace of a square matrix*

Description

Hardly worth coding, if it didn't appear in so many formulae in psychometrics, the trace of a (square) matrix is just the sum of the diagonal elements.

Usage

```
tr(m)
```

Arguments

m	A square matrix

Details

The tr function is used in various matrix operations and is the sum of the diagonal elements of a matrix.

Value

The sum of the diagonal elements of a square matrix.
i.e. tr(m) <- sum(diag(m)).

Examples

```
m <- matrix(1:16,ncol=4)
m
tr(m)
```

Tucker — *9 Cognitive variables discussed by Tucker and Lewis (1973)*

Description

Tucker and Lewis (1973) introduced a reliability coefficient for ML factor analysis. Their example data set was previously reported by Tucker (1958) and taken from Thurstone and Thurstone (1941). The correlation matrix is a 9 x 9 for 710 subjects and has two correlated factors of ability: Word Fluency and Verbal.

Usage

```
data(Tucker)
```

Format

A data frame with 9 observations on the following 9 variables.

- `t42` Prefixes
- `t54` Suffixes
- `t45` Chicago Reading Test: Vocabulary
- `t46` Chicago Reading Test: Sentences
- `t23` First and last letters
- `t24` First letters
- `t27` Four letter words
- `t10` Completion
- `t51` Same or Opposite

Details

The correlation matrix from Tucker (1958) was used in Tucker and Lewis (1973) for the Tucker-Lewis Index of factoring reliability.

Source

Tucker, Ledyard (1958) An inter-battery method of factor analysis, Psychometrika, 23, 111-136.

References

L.~Tucker and C.~Lewis. (1973) A reliability coefficient for maximum likelihood factor analysis. Psychometrika, 38(1):1–10.

F.~J. Floyd and K.~F. Widaman. (1995) Factor analysis in the development and refinement of clinical assessment instruments., Psychological Assessment, 7(3):286 – 299.

Examples

```
data(Tucker)
fa(Tucker,2,n.obs=710)
omega(Tucker,2)
```

`unidim` *Several indices of the unidimensionality of a set of variables.*

Description

There are a variety of ways of assessing whether a set of items measures one latent trait. `unidim` is just one more way. If a one factor model holds in the data, then the factor analytic decomposition F implies that FF' should reproduce the correlations with communalities along the diagonal. In this case, the fit FF' should be identical to the correlation matrix minus the uniquenesses. unidim is just the ratio of these two estimates. The higher it is, the more the evidence for unidimensionality. A number of alternative statistics are estimated.

Usage

```
unidim(keys=NULL,x=NULL,cor="cor",use="pairwise", fm="minres", correct=.5,
       check.keys=TRUE,n.obs=NA,nfactors=3)
```

Arguments

x	An input matrix or data frame. If x is not a correlation matrix, then the correlations are found.
keys	If specified, then a number of scales can be tested at once. (See scoreItems for a similar procedure.)
cor	By default, find the Pearson correlation, other options are "spearman","kendall","tet"(for tetrachoric), "poly" (for polychoric), or "mixed"
use	pairwise complete cases is the default
fm	factor extraction method defaults to "minres" but could be "mle" or "minrank"
correct	If using "tetrachoric" or "polychoric" correlations, should we correct empty cells for continuity, and if so, by how much. (See tetrachoric for a discussion of this correction)
check.keys	If TRUE, then items will be keyed based upon their loadings on the first factor. Automatically done if key.list is NULL.
n.obs	Number of observations. If given a correlation matrix as input, n.obs is required for some of the goodness of fit estimates.
nfactors	The number of factors used (if possible) to estimate omega_t

Details

This is set of new indices to test the unidimensionality of scale. A number of test cases suggest that u provides high values when the data are in fact unidimensional, low values when they are not.

The logic is deceptively simple: Unidimensionality implies that a one factor model of the data fits the covariances of the data. If this is the case, then factor model implies R = FF' + U2 will have residuals of 0. Similarly, this also implies that the observed correlations will equal the model. Thus, the sum of the observed correlations (with the diagonal replaced by the communalities) should match the factor model. Compare these two models: R - U2 versus FF'. This is the rho_c estimate. It is basically a test of whether a congeneric model fits. (That is, all the items have loadings on just one factor.)

This works well, but when some of the loadings are very small, even though 1 factor is correct, it is probably not a good idea to think of the items as forming a unidimen- sional scale. Thus, an alternative model (the Tau statistic) considers the residuals found by subtracting the average correlation from the observed correlations. This will achieve a maximum if the item covariances are all identical (a tau equivalent model).

The product of rho_c and Tau is the measure of unidimensionality, u That is, congeneric fit x tau equivalent fit as a measure of unidimensionality.

The main unidim estimates of the results are reported in the uni object. Conventional factor goodness of fit are in the fa.stats object.

In a recent article (Revelle and Condon, 2025) we compare the u statistic to other estimates of unidimensionality.

From the abstract:

"How to evaluate how well a psychological scale measures just one construct is a recurring problem in assessment. We introduce an index, u, of the unidimensionality and homogeneity of a scale. u is just the product of two other indices: τ (a measure of τ equivalence) and ρ_c (a measure of congeneric fit). By combining these two indices into one, we provide a simple index of the unidimensionality and homogeneity of a scale. We evaluate u through simulations and with real data sets. Simulations of u across one-factor scales ranging from three to 24 items with various levels of factor homogeneity show that τ and, therefore, u are sensitive to the degree of factor homogeneity. Additional tests with multifactorial scales representing 9, 18, 27, and 36 items with a hierarchical factor structure varying in a general factor loading show that ρ_c and, therefore, u are sensitive to the general factor saturation of a test. We also demonstrate the performance of u on 45 different publicly available personality and ability measures. Comparisons with traditional measures (i.e., $\omega_h, \alpha, \omega_t$, comparative fit index (CFI), and explained common variance (ECV)) show that u has greater sensitivity to unidimensional structure and less sensitivity to the number of items in a scale. u is easily calculated with open source statistical packages and is relatively robust to sample sizes ranging from 100 to 5,000. "

Value

`u`	The estimate of unidimensionality which is just the product of
`Tau`	The fit of the average r to the correlation matrix
`rho_c`	The off diagonal fit from `fa`
`alpha`	Standardized alpha of the keyed items (after appropriate reversals)
`av.r`	The average interitem correlation of the keyed items.
`median.r`	The median value of the iteritem correlations of the keyed items.
`Unidim.A`	The unidimensional criterion when items are keyed in positive direction.
`Unidim`	The raw value of the unidimensional criterion
`raw.model`	The ratio of the FF' model to the sum(R)
`adj.model`	The ratio of the FF' model to the sum(R) when items are flipped.
`Total`	The ratio of the sum(R - uniqueness)/sum(R)
`Total.A`	Same ratio with flipped items
`CFI`	Comparative Fit Index
`ECV`	Explained Common Variance

Note

A new statistic of unidimensionality. It is (perhaps) useful to compare the unidim statistics with those generated by `omega`. A quick way to do this is to use the `reliability` function which will find alpha, omega_h and omega_t as well as split half reliablities and the unidim measures. When this done, it can be seen that u is not sensitive to the number of items and seems robust across various sample sizes.

Author(s)

William Revelle

References

Revelle, W. and Condon, D.M (2025) Using undim rather than omega in estimating undimensionality. Psychological Methods (Advanced online publication) doi:10.1037/met0000729 (Open access)

See Also

fa for factor analysis, omega and reliability for reliability.

Examples

```
#test the unidimensionality of the five factors of the bfi data set.


unidim(bfi.keys,bfi)

if(require(psychTools)) {

unidim(psychTools::ability.keys,psychTools::ability)
}

#Try a known 3 factor structure
x <- sim.minor(nfact=3,bipolar=FALSE)   #this makes all the items positive
unidim(x$model)
keys.list <- list(first =paste0("V",1:4) ,second = paste0("V",5:8),
   third=paste0("V",9:12),all= paste0("V",1:12))
unidim(keys.list, x$model)


x <- sim.minor(nfact=3)
unidim(keys.list,x$model)   #we flip the negative items

#what about a hierarchical model?
H <- sim.hierarchical()  # by default, a nice hierarchical model
H.keys <- list(First = paste0("V",1:3),Second=paste0("V",4:6),Third=paste0("V",7:9),
  All = paste0("V",1:9))
unidim(H.keys,H)
```

VSS *Apply the Very Simple Structure, MAP, and other criteria to determine the appropriate number of factors.*

Description

There are multiple ways to determine the appropriate number of factors in exploratory factor analysis. Routines for the Very Simple Structure (VSS) criterion allow one to compare solutions of varying complexity and for different number of factors. Graphic output indicates the "optimal"

number of factors for different levels of complexity. The Velicer MAP criterion is another good choice. nfactors finds and plots several of these alternative estimates.

Usage

```
vss(x, n = 8, rotate = "varimax", diagonal = FALSE, fm = "minres",
 n.obs=NULL,plot=TRUE,title="Very Simple Structure",use="pairwise",cor="cor",...)
VSS(x, n = 8, rotate = "varimax", diagonal = FALSE, fm = "minres",
 n.obs=NULL,plot=TRUE,title="Very Simple Structure",use="pairwise",cor="cor",...)
 nfactors(x,n=20,rotate="varimax",diagonal=FALSE,fm="minres",n.obs=NULL,
             title="Number of Factors",pch=16,use="pairwise", cor="cor",...)
vssSelect(keys,x,cor="cor", fm="minres",plot=FALSE)
eigenCi(x,n.iter=1000, use="pairwise", alpha=.05,plot=FALSE,root=FALSE)
```

Arguments

x	a correlation matrix or a data matrix
n	Number of factors to extract – should be more than hypothesized!
rotate	what rotation to use c("none", "varimax", "oblimin","promax")
diagonal	Should we fit the diagonal as well
fm	factoring method – fm="pa" Principal Axis Factor Analysis, fm = "minres" minimum residual (OLS) factoring fm="mle" Maximum Likelihood FA, fm="pc" Principal Components"
n.obs	Number of observations if doing a factor analysis of correlation matrix. This value is ignored by VSS but is necessary for the ML factor analysis package.
plot	plot=TRUE Automatically call VSS.plot with the VSS output, otherwise don't plot
keys	A keys list specifying the subsets of items to analyze
title	a title to be passed on to VSS.plot
pch	the plot character for the nfactors plots
use	If doing covariances or Pearson R, should we use "pairwise" or "complete cases"
cor	What kind of correlation to find, defaults to Pearson but see fa for the choices
n.iter	How many iterations of the bootstrap for eigenCi
alpha	Width of confidence intervals = 1- alpha
root	By default show the eigen values, if root=TRUE plot the squareroots of the eigen values.
...	parameters to pass to the factor analysis program The most important of these is if using a correlation matrix is covmat= xx

Details

Determining the most interpretable number of factors from a factor analysis is perhaps one of the greatest challenges in factor analysis. There are many solutions to this problem, none of which is uniformly the best. "Solving the number of factors problem is easy, I do it everyday before

breakfast." But knowing the right solution is harder. (Horn and Engstrom, 1979) (Henry Kaiser in personal communication with J.L. Horn, as cited by Horn and Engstrom, 1979, MBR p 283).

Techniques most commonly used include

1) Extracting factors until the chi square of the residual matrix is not significant.

2) Extracting factors until the change in chi square from factor n to factor n+1 is not significant.

3) Extracting factors until the eigen values of the real data are less than the corresponding eigen values of a random data set of the same size (parallel analysis) `fa.parallel`.

4) Plotting the magnitude of the successive eigen values and applying the scree test (a sudden drop in eigen values analogous to the change in slope seen when scrambling up the talus slope of a mountain and approaching the rock face.

5) Extracting principal components until the eigen value < 1.

6) Extracting factors as long as they are interpetable.

7) Using the Very Simple Structure Criterion (VSS).

8) Using Wayne Velicer's Minimum Average Partial (MAP) criterion.

Each of the procedures has its advantages and disadvantages. Using either the chi square test or the change in square test is, of course, sensitive to the number of subjects and leads to the nonsensical condition that if one wants to find many factors, one simply runs more subjects. Parallel analysis is partially sensitive to sample size in that for large samples the eigen values of random factors will be very small. The scree test is quite appealling but can lead to differences of interpretation as to when the scree "breaks". The eigen value of 1 rule, although the default for many programs, seems to be a rough way of dividing the number of variables by 3. Extracting interpretable factors means that the number of factors reflects the investigators creativity more than the data. VSS, while very simple to understand, will not work very well if the data are very factorially complex. (Simulations suggests it will work fine if the complexities of some of the items are no more than 2).

Most users of factor analysis tend to interpret factor output by focusing their attention on the largest loadings for every variable and ignoring the smaller ones. Very Simple Structure operationalizes this tendency by comparing the original correlation matrix to that reproduced by a simplified version (S) of the original factor matrix (F). R = SS' + U2. S is composed of just the c greatest (in absolute value) loadings for each variable. C (or complexity) is a parameter of the model and may vary from 1 to the number of factors.

The VSS criterion compares the fit of the simplified model to the original correlations: VSS = 1 -sumsquares(r*)/sumsquares(r) where R* is the residual matrix R* = R - SS' and r* and r are the elements of R* and R respectively.

VSS for a given complexity will tend to peak at the optimal (most interpretable) number of factors (Revelle and Rocklin, 1979).

Although originally written in Fortran for main frame computers, VSS has been adapted to micro computers (e.g., Macintosh OS 6-9) using Pascal. We now release R code for calculating VSS.

Note that if using a correlation matrix (e.g., my.matrix) and doing a factor analysis, the parameters n.obs should be specified for the factor analysis: e.g., the call is VSS(my.matrix,n.obs=500). Otherwise it defaults to 1000.

Wayne Velicer's MAP criterion has been added as an additional test for the optimal number of components to extract. Note that VSS and MAP will not always agree as to the optimal number.

The nfactors function will do a VSS, find MAP, and report a number of other criteria (e.g., BIC, complexity, chi square, ...)

A variety of rotation options are available. These include varimax, promax, and oblimin. Others can be added. Suggestions are welcome.

vssSelect will perform a VSS analysis on subsets of the data where the subsets are specified by a keys list.

Value

A data.frame with entries: map: Velicer's MAP values (lower values are better)
dof: degrees of freedom (if using FA)
chisq: chi square (from the factor analysis output (if using FA)
prob: probability of residual matrix > 0 (if using FA)
sqresid: squared residual correlations
RMSEA: the RMSEA for each number of factors
BIC: the BIC for each number of factors
eChiSq: the empirically found chi square
eRMS: Empirically found mean residual
eCRMS: Empirically found mean residual corrected for df
eBIC: The empirically found BIC based upon the eChiSq
fit: factor fit of the complete model
cfit.1: VSS fit of complexity 1
cfit.2: VSS fit of complexity 2
...
cfit.8: VSS fit of complexity 8
cresidiual.1: sum squared residual correlations for complexity 1
...: sum squared residual correlations for complexity 2 ..8

Author(s)

William Revelle

References

https://personality-project.org/r/vss.html, Revelle, W. An introduction to psychometric theory with applications in R (in prep) Springer. Draft chapters available at https://personality-project.org/r/book/

Revelle, W. and Rocklin, T. 1979, Very Simple Structure: an Alternative Procedure for Estimating the Optimal Number of Interpretable Factors, Multivariate Behavioral Research, 14, 403-414. https://personality-project.org/revelle/publications/vss.pdf

Velicer, W. (1976) Determining the number of components from the matrix of partial correlations. Psychometrika, 41, 321-327.

See Also

VSS.plot, ICLUST, omega, fa.parallel

Examples

```
#test.data <- Harman74.cor$cov
#my.vss <- VSS(test.data,title="VSS of 24 mental tests")
#print(my.vss[,1:12],digits =2)
#VSS.plot(my.vss, title="VSS of 24 mental tests")

#now, some simulated data with two factors
#VSS(sim.circ(nvar=24),fm="minres" ,title="VSS of 24 circumplex variables")
VSS(sim.item(nvar=24)$item,fm="minres" ,title="VSS of 24 simple structure variables")

#vssSelect(bfi.keys, bfi)
```

VSS.parallel *Compare real and random VSS solutions*

Description

Another useful test for the number of factors is when the eigen values of a random matrix are greater than the eigen values of a a real matrix. Here we show VSS solutions to random data. A better test is probably `fa.parallel`.

Usage

```
VSS.parallel(ncases, nvariables,scree=FALSE,rotate="none")
```

Arguments

`ncases`	Number of simulated cases
`nvariables`	number of simulated variables
`scree`	Show a scree plot for random data – see `omega`
`rotate`	rotate="none" or rotate="varimax"

Value

VSS like output to be plotted by VSS.plot

Author(s)

William Revelle

References

Very Simple Structure (VSS)

See Also

`fa.parallel`, `VSS.plot`, `ICLUST`, `omega`

Examples

```
#VSS.plot(VSS.parallel(200,24))
```

VSS.plot *Plot VSS fits*

Description

The Very Simple Structure criterion (VSS) for estimating the optimal number of factors is plotted as a function of the increasing complexity and increasing number of factors.

Usage

```
VSS.plot(x, title = "Very Simple Structure", line = FALSE)
```

Arguments

x	output from VSS
title	any title
line	connect different complexities

Details

Item-factor models differ in their "complexity". Complexity 1 means that all except the greatest (absolute) loading for an item are ignored. Basically a cluster model (e.g., ICLUST). Complexity 2 implies all except the greatest two, etc.

Different complexities can suggest different number of optimal number of factors to extract. For personality items, complexity 1 and 2 are probably the most meaningful.

The Very Simple Structure criterion will tend to peak at the number of factors that are most interpretable for a given level of complexity. Note that some problems, the most interpretable number of factors will differ as a function of complexity. For instance, when doing the Harman 24 psychological variable problems, an unrotated solution of complexity one suggests one factor (g), while a complexity two solution suggests that a four factor solution is most appropriate. This latter probably reflects a bi-factor structure.

For examples of VSS.plot output, see https://personality-project.org/r/r.vss.html

Value

A plot window showing the VSS criterion varying as the number of factors and the complexity of the items.

Author(s)

Maintainer: William Revelle <revelle@northwestern.edu>

References

https://personality-project.org/r/r.vss.html

See Also

VSS, ICLUST, omega

Examples

```
test.data <- Harman74.cor$cov
my.vss <- VSS(test.data)       #suggests that 4 factor complexity two solution is optimal
VSS.plot(my.vss,title="VSS of Holzinger-Harmon problem")           #see the graphics window
```

VSS.scree *Plot the successive eigen values for a scree test*

Description

Cattell's scree test is one of most simple ways of testing the number of components or factors in a correlation matrix. Here we plot the eigen values of a correlation matrix as well as the eigen values of a factor analysis.

Usage

```
scree(rx,factors=TRUE,pc=TRUE,main="Scree plot",hline=NULL,add=FALSE,sqrt=FALSE)
VSS.scree(rx, main = "scree plot",sqrt=FALSE)
```

Arguments

rx	a correlation matrix or a data matrix. If data, then correlations are found using pairwise deletions.
factors	If true, draw the scree for factors
pc	If true, draw the scree for components
hline	if null, draw a horizontal line at 1, otherwise draw it at hline (make negative to not draw it)
main	Title
add	Should multiple plots be drawn?
sqrt	If TRUE, take the sqrt of the eigen value before plotting

Details

Among the many ways to choose the optimal number of factors is the scree test. A better function to show the scree as well as compare it to randomly parallel solutions is found found in fa.parallel

Following a suggestion from Marco Del Giudice, I added the sqrt option for version 2.2.12. -

Author(s)

William Revelle

References

https://personality-project.org/r/vss.html

See Also

fa.parallel VSS.plot, ICLUST, omega

Examples

```
scree(attitude)
#VSS.scree(cor(attitude)
```

---

winsor *Find the Winsorized scores, means, sds or variances for a vector, matrix, or data.frame*

Description

Among the robust estimates of central tendency are trimmed means and Winsorized means. This function finds the Winsorized scores. The top and bottom trim values are given values of the trimmed and 1- trimmed quantiles. Then means, sds, and variances are found.

Usage

```
winsor(x, trim = 0.2, na.rm = TRUE)
winsor.mean(x, trim = 0.2, na.rm = TRUE)
winsor.means(x, trim = 0.2, na.rm = TRUE)
winsor.sd(x, trim = 0.2, na.rm = TRUE)
winsor.var(x, trim = 0.2, na.rm = TRUE)
```

Arguments

x	A data vector, matrix or data frame
trim	Percentage of data to move from the top and bottom of the distributions
na.rm	Missing data are removed

Details

Among the many robust estimates of central tendency, some recommend the Winsorized mean. Rather than just dropping the top and bottom trim percent, these extreme values are replaced with values at the trim and 1- trim quantiles.

Value

A scalar or vector of winsorized scores or winsorized means, sds, or variances (depending upon the call).

Author(s)

William Revelle with modifications suggested by Joe Paxton and a further correction added (January, 2009) to preserve the original order for the winsor case.

References

Wilcox, Rand R. (2005) Introduction to robust estimation and hypothesis testing. Elsevier/Academic Press. Amsterdam ; Boston.

See Also

interp.median

Examples

```
data(sat.act)
winsor.means(sat.act) #compare with the means of the winsorized scores
y <- winsor(sat.act)
describe(y)
xy <- data.frame(sat.act,y)
#pairs.panels(xy) #to see the effect of winsorizing
x <- matrix(1:100,ncol=5)
winsor(x)
winsor.means(x)
y <- 1:11
winsor(y,trim=.5)
```

---

withinBetween — *An example of the distinction between within group and between group correlations*

Description

A demonstration that a correlation may be decomposed to a within group correlation and a between group correlations and these two correlations are independent. Between group correlations are sometimes called ecological correlations, the decomposition into within and between group correlations is a basic concept in multilevel modeling. This data set shows the composite correlations between 9 variables, representing 16 cases with four groups.

Usage

```
data(withinBetween)
```

Format

A data frame with 16 observations on the following 10 variables.

`Group` An example grouping factor.

`V1` A column of 16 observations

`V2` A column of 16 observations

`V3` A column of 16 observations

`V4` A column of 16 observations

`V5` A column of 16 observations

`V6` A column of 16 observations

`V7` A column of 16 observations

`V8` A column of 16 observations

`V9` A column of 16 observations

Details

Correlations between individuals who belong to different natural groups (based upon e.g., ethnicity, age, gender, college major,or country) reflect an unknown mixture of the pooled correlation within each group as well as the correlation of the means of these groups. These two correlations are independent and do not allow inferences from one level (the group) to the other level (the individual). This data set shows this independence. The within group correlations between 9 variables are set to be 1, 0, and -1 while those between groups are also set to be 1, 0, -1. These two sets of correlations are crossed such that V1, V4, and V7 have within group correlations of 1, as do V2, V5 and V8, and V3, V6 and V9. V1 has a within group correlation of 0 with V2, V5, and V8, and a -1 within group correlation with V3, V6 and V9. V1, V2, and V3 share a between group correlation of 1, as do V4, V5 and V6, and V7, V8 and V9. The first group has a 0 between group correlation with the second and a -1 with the third group.

`statsBy` can decompose the observed correlation in the between and within correlations. `sim.multilevel` can produce similar data.

Source

The data were created for this example

References

P. D. Bliese. Multilevel modeling in R (2.3) a brief introduction to R, the multilevel package and the nlme package, 2009.

Pedhazur, E.J. (1997) Multiple regression in behavioral research: explanation and prediction. Harcourt Brace.

Revelle, W. An introduction to psychometric theory with applications in R (in prep) Springer. Draft chapters available at https://personality-project.org/r/book/

See Also

`statsBy`, `describeBy`, and `sim.multilevel`

Examples

```
data(withinBetween)
pairs.panels(withinBetween,bg=c("red","blue","white","black")[withinBetween[,1]],
     pch=21,ellipses=FALSE,lm=TRUE)
stats <- statsBy(withinBetween,'Group')
print(stats,short=FALSE)
```

Yule *From a two by two table, find the Yule coefficients of association, convert to phi, or tetrachoric, recreate table the table to create the Yule coefficient.*

Description

One of the many measures of association is the Yule coefficient. Given a two x two table of counts

a	b	R1
c	d	R2
C1	C2	n

Yule Q is (ad - bc)/(ad+bc).
Conceptually, this is the number of pairs in agreement (ad) - the number in disagreement (bc) over the total number of paired observations. Warren (2008) has shown that Yule's Q is one of the "coefficients that have zero value under statistical independence, maximum value unity, and minimum value minus unity independent of the marginal distributions" (p 787).
ad/bc is the odds ratio and Q = (OR-1)/(OR+1)
Yule's coefficient of colligation is Y = (sqrt(OR) - 1)/(sqrt(OR)+1) Yule.inv finds the cell entries for a particular Q and the marginals (a+b,c+d,a+c, b+d). This is useful for converting old tables of correlations into more conventional `phi` or tetrachoric correlations `tetrachoric`
Yule2phi and Yule2tetra convert the Yule Q with set marginals to the correponding phi or tetrachoric correlation.

Bonett and Price show that the Q and Y coefficients are both part of a general family of coefficients raising the OR to a power (c). If c=1, then this is Yule's Q. If .5, then Yule's Y, if c = .75, then this is Digby's H. They propose that c = .5 - (.5 * min(cell probabilty)^2 is a more general coefficient. YuleBonett implements this for the 2 x 2 case, YuleCor for the data matrix case.

Usage

```
YuleBonett(x,c=1,bonett=FALSE,alpha=.05) #find the generalized Yule cofficients
YuleCor(x,c=1,bonett=FALSE,alpha=.05) #do this for a matrix
Yule(x,Y=FALSE)  #find Yule given a two by two table of frequencies
 #find the frequencies that produce a Yule Q given the Q and marginals
Yule.inv(Q,m,n=NULL)
#find the phi coefficient that matches the Yule Q given the marginals
```

```
Yule2phi(Q,m,n=NULL)
Yule2tetra(Q,m,n=NULL,correct=TRUE)


   #Find the tetrachoric correlation given the Yule Q and the marginals
#(deprecated) Find the tetrachoric correlation given the Yule Q and the marginals
Yule2poly(Q,m,n=NULL,correct=TRUE)
```

Arguments

x	A vector of four elements or a two by two matrix, or, in the case of YuleBonett or YuleCor, this can also be a data matrix
c	1 returns Yule Q, .5, Yule's Y, .75 Digby's H
bonett	If FALSE, then find Q, Y, or H, if TRUE, then find the generalized Bonett coefficient
alpha	The two tailed probability for confidence intervals
Y	Y=TRUE return Yule's Y coefficient of colligation
Q	Either a single Yule coefficient or a matrix of Yule coefficients
m	The vector c(R1,C2) or a two x two matrix of marginals or a four element vector of marginals. The preferred form is c(R1,C1)
n	The number of subjects (if the marginals are given as frequencies
correct	When finding a tetrachoric correlation, should small cell sizes be corrected for continuity. See `{link{tetrachoric}` for a discussion.

Details

Yule developed two measures of association for two by two tables. Both are functions of the odds ratio

Value

Q	The Yule Q coefficient
R	A two by two matrix of counts
result	If given matrix input, then a matrix of phis or tetrachorics
rho	From YuleBonett and YuleCor
ci	The upper and lower confidence intervals in matrix form (From YuleBonett and YuleCor).

Note

Yule.inv is currently done by using the optimize function, but presumably could be redone by solving a quadratic equation.

Author(s)

William Revelle

References

Yule, G. Uday (1912) On the methods of measuring association between two attributes. Journal of the Royal Statistical Society, LXXV, 579-652

Bonett, D.G. and Price, R.M, (2007) Statistical Inference for Generalized Yule Coefficients in 2 x 2 Contingency Tables. Sociological Methods and Research, 35, 429-446.

Warrens, Matthijs (2008), On Association Coefficients for 2x2 Tables and Properties That Do Not Depend on the Marginal Distributions. Psychometrika, 73, 777-789.

See Also

See Also as phi, tetrachoric, Yule2poly.matrix, Yule2phi.matrix

Examples

```
Nach <- matrix(c(40,10,20,50),ncol=2,byrow=TRUE)
Yule(Nach)
Yule.inv(.81818,c(50,60),n=120)
Yule2phi(.81818,c(50,60),n=120)
Yule2tetra(.81818,c(50,60),n=120)
phi(Nach)  #much less
#or express as percents and do not specify n
Nach <- matrix(c(40,10,20,50),ncol=2,byrow=TRUE)
Nach/120
Yule(Nach)
Yule.inv(.81818,c(.41667,.5))
Yule2phi(.81818,c(.41667,.5))
Yule2tetra(.81818,c(.41667,.5))
phi(Nach)  #much less
if(require(psychTools)) {
YuleCor(psychTools::ability[,1:4],,TRUE)
}
YuleBonett(Nach,1)  #Yule Q
YuleBonett(Nach,.5)  #Yule Y
YuleBonett(Nach,.75)  #Digby H
YuleBonett(Nach,,TRUE)  #Yule* is a generalized Yule
```

Index

∗ **cluster**
- 00.psych, 6
- cluster.fit, 59
- cluster.loadings, 60
- cluster.plot, 62
- iclust, 225
- ICLUST.cluster, 234
- iclust.diagram, 235
- ICLUST.graph, 237
- ICLUST.rgraph, 240

∗ **datagen**
- sim, 396
- sim.congeneric, 404
- sim.hierarchical, 406
- sim.irt, 409
- sim.item, 414
- sim.omega, 420
- sim.structure, 424
- sim.VSS, 427
- simulation.circ, 428

∗ **datasets**
- Bechtoldt, 30
- bfi, 38
- bock, 49
- cattell, 50
- Dwyer, 120
- Garcia, 209
- Gleser, 215
- Gorsuch, 216
- Harman, 217
- sat.act, 364
- Schmid, 371
- small.msq, 430
- Tal_Or, 453
- Tucker, 469
- withinBetween, 481

∗ **hplot**
- bi.bars, 41
- biPlot, 45
- cluster.plot, 62
- corPlot, 85
- densityBy, 107
- diagram, 114
- draw.tetra, 118
- ellipses, 122
- error.bars, 124
- error.bars.by, 127
- error.crosses, 131
- error.dots, 133
- errorCircles, 136
- fa.diagram, 157
- iclust.diagram, 235
- ICLUST.graph, 237
- ICLUST.rgraph, 240
- manhattan, 270
- multi.hist, 286
- pairs.panels, 312
- scatterHist, 366
- spider, 432
- structure.diagram, 445
- VSS.scree, 479

∗ **models**
- 00.psych, 6
- alpha, 16
- anova.psych, 22
- bassAckward, 26
- bestScales, 33
- bigCor, 42
- CFA, 51
- circ.tests, 57
- cohen.d, 64
- congruence, 75
- cor.smooth, 77
- cor2dist, 81
- corCi, 81
- corFiml, 84
- correct.cor, 89
- corTest, 94

cta, 103
describe, 109
describeBy, 113
dummy.code, 119
eigen.loadings, 121
esem, 138
fa, 143
fa.lookup, 165
fa.multi, 168
fa.poly, 177
fa.random, 178
faCor, 185
factor.congruence, 187
factor.fit, 190
factor.model, 191
factor.residuals, 192
factor.rotate, 193
factor.scores, 195
factor.stats, 197
factor2cluster, 200
faRotations, 201
fisherz, 204
fsi, 207
ICLUST.sort, 242
irt.1p, 245
irt.fa, 247
irt.item.diff.rasch, 251
irt.responses, 253
kaiser, 254
KMO, 255
lmCor, 257
make.keys, 268
mardia, 272
mat.sort, 274
mediate, 276
mixedCor, 281
mssd, 285
multilevel.reliability, 288
omega, 293
omegaStats, 305
outlier, 308
p.rep, 309
paired.r, 311
pairwiseCount, 315
phi, 321
phi.demo, 323
phi2tetra, 324
polychor.matrix, 331
predict.psych, 332
predicted.validity, 334
principal, 335
Promax, 341
r.test, 349
rangeCorrection, 352
reliability, 354
rescale, 358
residuals.psych, 359
RMSEA, 361
RV, 362
scaling.fits, 365
schmid, 369
score.alpha, 373
score.multiple.choice, 374
scoreIrt, 376
scoreItems, 381
scoreOverlap, 388
scoreWtd, 392
SD, 395
sim.anova, 402
sim.hierarchical, 406
sim.multilevel, 417
sim.VSS, 427
statsBy, 440
structure.list, 449
table2matrix, 452
test.irt, 454
testRetest, 457
thurstone, 467
unidim, 470
VSS, 473
VSS.parallel, 477
VSS.plot, 478
Yule, 483

∗ **multivariate**

00.psych, 6
alpha, 16
anova.psych, 22
AUC, 23
bassAckward, 26
bestScales, 33
bigCor, 42
biPlot, 45
block.random, 48
CFA, 51
circ.tests, 57
cluster.fit, 59

cluster.loadings, 60
cluster.plot, 62
cluster2keys, 63
cohen.d, 64
cohen.kappa, 70
comorbidity, 74
congruence, 75
cor.smooth, 77
cor.wt, 79
cor2dist, 81
corCi, 81
corFiml, 84
corPlot, 85
correct.cor, 89
corTest, 94
cortest, 91
cortest.bartlett, 97
cosinor, 99
densityBy, 107
describe, 109
diagram, 114
draw.tetra, 118
dummy.code, 119
eigen.loadings, 121
ellipses, 122
error.bars, 124
error.bars.by, 127
error.crosses, 131
error.dots, 133
errorCircles, 136
esem, 138
fa, 143
fa.diagram, 157
fa.extension, 161
fa.lookup, 165
fa.multi, 168
fa.parallel, 172
fa.poly, 177
fa.random, 178
fa.sort, 184
faCor, 185
factor.congruence, 187
factor.model, 191
factor.residuals, 192
factor.rotate, 193
factor.scores, 195
factor.stats, 197
factor2cluster, 200
faRotations, 201
fisherz, 204
fsi, 207
geometric.mean, 211
glb.algebraic, 212
harmonic.mean, 219
headTail, 220
ICC, 222
iclust, 225
ICLUST.cluster, 234
iclust.diagram, 235
ICLUST.graph, 237
ICLUST.rgraph, 240
ICLUST.sort, 242
irt.1p, 245
irt.fa, 247
irt.item.diff.rasch, 251
irt.responses, 253
kaiser, 254
KMO, 255
lmCor, 257
logistic, 265
lowerUpper, 266
make.keys, 268
manhattan, 270
mardia, 272
mat.sort, 274
matrix.addition, 275
mediate, 276
mixedCor, 281
mssd, 285
multi.hist, 286
multilevel.reliability, 288
omega, 293
omega.graph, 302
omegaStats, 305
outlier, 308
paired.r, 311
pairs.panels, 312
pairwiseCount, 315
parcels, 318
partial.r, 319
phi, 321
phi.demo, 323
Pinv, 325
plot.psych, 327
polar, 329
polychor.matrix, 331

predict.psych, 332
predicted.validity, 334
principal, 335
print.psych, 340
Promax, 341
psych.misc, 345
r.test, 349
rangeCorrection, 352
reliability, 354
rescale, 358
residuals.psych, 359
reverse.code, 360
RMSEA, 361
RV, 362
scatterHist, 366
schmid, 369
score.alpha, 373
score.multiple.choice, 374
scoreIrt, 376
scoreItems, 381
scoreOverlap, 388
scoreWtd, 392
scrub, 393
sim, 396
sim.anova, 402
sim.congeneric, 404
sim.hierarchical, 406
sim.irt, 409
sim.item, 414
sim.multilevel, 417
sim.omega, 420
sim.structure, 424
sim.VSS, 427
simulation.circ, 428
smc, 431
spider, 432
splitHalf, 435
statsBy, 440
structure.diagram, 445
structure.list, 449
superMatrix, 450
test.irt, 454
test.psych, 455
testRetest, 457
tetrachoric, 460
tr, 469
unidim, 470
VSS, 473
VSS.plot, 478
VSS.scree, 479
Yule, 483

∗ **package**
00.psych, 6

∗ **tree**
bestScales, 33

∗ **univar**
describe, 109
describeBy, 113
interp.median, 244
p.rep, 309
rescale, 358
winsor, 480

∗ **utilities**
fparse, 206

%+% *(matrix.addition)*, 275
00.psych, 6

ability, *10*, *317*
acs *(psych.misc)*, 345
alpha, *7*, *9*, *12*, 16, *222*, *260*, *354*, *373*, *374*, *387*, *391*, *439*, *459*
alpha.ci, *19*
alpha2r *(alpha)*, 16
anova.psych, 22, *142*, *155*, *261*
AUC, 23, *23*, *75*, *322*
auc *(AUC)*, 23
autoR, *285*, *418*
autoR *(mssd)*, 285

bassAckward, 26, *115*, *116*, *156*, *186*
bassAckward.diagram, *28*, *160*
Bechtoldt, 30, *218*, *344*
bestItems, *35*, *36*, *166*, *167*
bestItems *(bestScales)*, 33
bestScales, *7*, *12*, 33, *33–35*, *37*, *133*, *135*, *165*, *167*, *261*, *269–271*, *332*, *333*, *386*, *392*, *393*, *451*
bfi, *10*, *15*, 38, *53*, *174*, *180*, *230*, *386*
bfi.dictionary, *35*, *166*
bi.bars, *40*, 41, *109*, *288*
bifactor, *30*, *159*, *217*
bifactor *(Promax)*, 341
bigCor, *11*, 42, *316*, *465*
biPlot, 45
biplot.psych *(biPlot)*, 45
biquartimin, *30*, *159*
biquartimin *(Promax)*, 341

BISCUIT, *7*, *33, 34*
BISCUIT *(*bestScales*)*, 33
biscuit *(*bestScales*)*, 33
BISCWIT, *33, 34*
BISCWIT *(*bestScales*)*, 33
biscwit *(*bestScales*)*, 33
biserial, *282*
biserial *(*tetrachoric*)*, 460
blant, *146*
block.random, 48
bock, 49, *452, 453*, *463*
burt, *78*, *154*, *218, 219*

cancorDiagram, *262*
cancorDiagram *(*esem*)*, 138
cattell, 50
cd.validity *(*cohen.d*)*, 64
CFA, *6*, *11*, 51, *51*, *53*, *155*, *206–209*, *305*, *307*
CFA.bifactor, *6*, *11*, *53*, *155*, *206*
char2numeric, *347, 348*
char2numeric *(*psych.misc*)*, 345
Chen *(*Schmid*)*, 371
chi2r *(*fisherz*)*, 204
choose, *438*
circ.sim, *427*
circ.sim *(*sim.item*)*, 414
circ.sim.plot, *429*
circ.sim.plot *(*simulation.circ*)*, 428
circ.simulation, *58*
circ.simulation *(*simulation.circ*)*, 428
circ.tests, *10*, 57, *330*, *416*, *430*
circadian.cor, *8*, *14*, *100, 101*
circadian.cor *(*cosinor*)*, 99
circadian.F, *101*
circadian.F *(*cosinor*)*, 99
circadian.linear.cor, *8*, *14*, *101*
circadian.linear.cor *(*cosinor*)*, 99
circadian.mean, *8*, *14*, *101*
circadian.mean *(*cosinor*)*, 99
circadian.phase *(*cosinor*)*, 99
circadian.reliability, *101*
circadian.reliability *(*cosinor*)*, 99
circadian.sd *(*cosinor*)*, 99
circadian.stats, *101*
circadian.stats *(*cosinor*)*, 99
circular.cor, *101*
circular.cor *(*cosinor*)*, 99
circular.mean, *101*
circular.mean *(*cosinor*)*, 99
cities, *10*, *15*
cluster.cor, *7*, *9*, *12*, *18*, *21*, *35*, *60, 61*, *63, 64*, *82, 83*, *90*, *96*, *201*, *260*, *262*, *268, 269*, *341*, *381*, *387*, *390*, *451*
cluster.cor *(*scoreOverlap*)*, 388
cluster.fit, 59, *192*, *232*, *234*, *243*
cluster.loadings, *12*, 60, *90*, *341*, *374*, *387*
cluster.plot, *9*, 62, *329, 330*
cluster2keys, 63
cohen.d, *25*, 64, *66, 67*, *113, 114*, *133*, *135*, *204*
cohen.d.by, *66*, *114*
cohen.d.ci, *66*, *114*
cohen.kappa, 70
cohen.profile, *76*, *188*
cohen.profile *(*congruence*)*, 75
cohenPooled *(*cohen.d*)*, 64
comorbidity, *14*, *26*, 74, *322*
con2cat *(*sim.item*)*, 414
congeneric.sim *(*sim.congeneric*)*, 404
congruence, 75, *76*, *188*
cor, *95*, *282, 283*
cor.ci, *86*, *88*, *96*, *314*, *386*
cor.ci *(*corCi*)*, 81
cor.plot, *14*, *83*, *313, 314*
cor.plot *(*corPlot*)*, 85
cor.plot.upperLowerCi, *82, 83*, *314*
cor.smooth, 77, *154*
cor.smoother *(*cor.smooth*)*, 77
cor.test, *96*, *312*
cor.wt, 79, *113*, *314*
cor2, *345*, *347*, *459*
cor2 *(*psych.misc*)*, 345
cor2cov *(*fisherz*)*, 204
cor2dist, 81
corCi, 81, *133*
corFiml, 84
corInfo, *96*
corInfo *(*corTest*)*, 94
corPlot, *11*, 85, *88*, *232*, *267*, *275*, *316*, *347*, *434*
corPlotUpperLowerCi, *88*
corPlotUpperLowerCi *(*corPlot*)*, 85
corr.p, *96*, *320*
corr.p *(*corTest*)*, 94
corr.test, *9*, *11*, *83*, *93*, *133*, *312*, *314*, *322*, *348*
corr.test *(*corTest*)*, 94

correct.cor, *12*, 89, *374*, *387*
corTest, *88*, 94, *351*
cortest, 91, *92*
cortest.bartlett, *14*, *92*, *93*, 97, *142*, *149*, *155*, *256*
cortest.jennrich, *92*, *98*
cortest.mat, *14*, *92*, *96*, *98*, *351*
cortest.normal, *92*, *98*
cosinor, *8*, *14*, 99
count.pairwise, *12*, *85*, *247*
count.pairwise (pairwiseCount), 315
cov.wt, *80*
crossValidation, *7*, *36*, *261*
crossValidation (lmCor), 257
crossValidationBoot, *261*
crossValidationBoot (lmCor), 257
cs, *8*, *347*, *348*
cs (psych.misc), 345
cta, *8*, 103, *104*, *105*
cta.15, *104*, *105*
cubits, *10*, *15*, *453*

d.ci (cohen.d), 64
d.robust, *66*
d.robust (cohen.d), 64
d2CL, *25*, *67*
d2CL (cohen.d), 64
d2OVL, *67*
d2OVL (cohen.d), 64
d2OVL2, *67*
d2OVL2 (cohen.d), 64
d2r (cohen.d), 64
d2t (cohen.d), 64
d2U3, *67*
d2U3 (cohen.d), 64
densityBy, *41*, 107, *112*, *114*, *288*
describe, *6*, *8*, *11*, *41*, *68*, 109, *109*, *111*, *113*, *114*, *124*, *125*, *132*, *133*, *135*, *274*, *286*, *296*, *341*
describe.by, *11*, *274*, *395*
describe.by (describeBy), 113
describeBy, *6*, *8*, *41*, *68*, *109*–*112*, 113, *128*, *132*, *133*, *135*, *137*, *286*, *444*, *482*
describeData, *110*, *111*
describeData (describe), 109
describeFast, *111*
describeFast (describe), 109
df2latex, *14*, *166*
dfOrder, *8*
dia.arrow, *236*, *447*
dia.arrow (diagram), 114
dia.cone (diagram), 114
dia.curve, *236*, *447*
dia.curve (diagram), 114
dia.curved.arrow, *447*
dia.curved.arrow (diagram), 114
dia.ellipse, *236*, *447*
dia.ellipse (diagram), 114
dia.ellipse1 (diagram), 114
dia.rect, *236*, *447*
dia.rect (diagram), 114
dia.self (diagram), 114
dia.shape, *447*
dia.shape (diagram), 114
dia.triangle (diagram), 114
diagram, *14*, *28*, 114, *157*, *159*, *160*, *230*–*232*, *447*
diff, *285*
directSl, *298*, *370*, *371*
directSl (omega), 293
distance, *76*, *188*
distance (congruence), 75
draw.cor, *462*
draw.cor (draw.tetra), 118
draw.tetra, 118, *462*, *465*
dummy.code, *14*, 119, *120*
Dwyer, 120, *164*

eigen.loadings, *12*, 121
eigenCi (VSS), 473
ellipses, 122
epi.bfi, *15*
equamax (Promax), 341
error.bars, *6*, *8*, *11*, *41*, *109*, *112*, *114*, 124, *125*, *129*, *130*, *132*, *135*
error.bars.by, *11*, *41*, *109*, *112*, *114*, *125*, *126*, 127, *132*, *135*, *137*, *206*
error.bars.tab, *125*
error.crosses, *11*, *110*, *112*, *126*, *130*, 131, *137*
error.dots, *6*, *8*, *9*, *36*, *37*, *66*, *68*, *126*, *130*, 133, *271*, *356*
errorCircles, *9*, *132*, 136
esem, *51*, 138, *141*, *362*
esemDiagram, *141*
esemDiagram (esem), 138
extension.diagram, *159*
extension.diagram (fa.diagram), 157

fa, *6*, *9*, *11*, *12*, *22*, *28*, *29*, *37*, *45–47*, *51*, *54*, *62*, *78*, *85*, *86*, *88*, *92*, *98*, *115*, *135*, *139*, *140*, 143, *148*, *149*, *153*, *159*, *161–164*, *167*, *171–173*, *176–178*, *180*, *183–185*, *187*, *189*, *193*, *195–198*, *200–203*, *207–209*, *218*, *231*, *247–249*, *252*, *254–256*, *260*, *275*, *295*, *306*, *307*, *316*, *327–330*, *333*, *336*, *337*, *339*, *343*, *344*, *359*, *361*, *362*, *369*, *390*, *406*, *432*, *441*, *445*, *465*, *472*, *473*
fa.congruence *(factor.congruence)*, 187
fa.diagram, *9*, *11*, *14*, *116*, *117*, 157, *157*, *159*, *185*, *303*, *447*
fa.extend, *159*, *163*, *187*, *249*
fa.extend *(fa.extension)*, 161
fa.extension, *11*, *13*, *139*, *142*, *148*, *155*, 161, *162*, *163*, *187*, *359*, *462*
fa.graph, *11*, *13*, *16*, *63*, *157*, *371*, *447*
fa.graph *(fa.diagram)*, 157
fa.lookup, *28*, *142*, *155*, *160*, 165, *165–167*
fa.multi, *142*, *155*, 168, *301*
fa.organize, *142*, *155*
fa.organize *(fa.sort)*, 184
fa.parallel, *7*, *9*, *11*, 172, *174*, *301*, *339*, *475–477*, *479*, *480*
fa.parallel.poly, *11*, *174*
fa.plot, *46*, *47*, *329*
fa.plot *(cluster.plot)*, 62
fa.poly, *45–47*, *119*, *149*, 177
fa.random, *155*, *156*, 178
fa.rgraph, *157*
fa.rgraph *(fa.diagram)*, 157
fa.sapa, *150*
fa.sort, *11*, *142*, *155*, *167*, 184, *243*
fa.stats, *54*, *306*
fa.stats *(factor.stats)*, 197
fa2irt, *163*, *249*
fa2irt *(irt.fa)*, 247
faBy, *10*, *443*, *444*
faBy *(statsBy)*, 440
fac *(fa)*, 143
faCor, *28*, 185, *189*, *326*
factanal, *146–148*, *151*, *181*, *199*, *338*
factor.congruence, *13*, *76*, *77*, *163*, *186*, 187, *339*
factor.fit, *13*, *59*, *60*, 190, *191*, *192*
factor.minres, *9*, *11*, *200*, *343*
factor.minres *(fa.poly)*, 177
factor.model, *13*, 191
factor.pa, *9*, *11*, *13*, *200*, *327*, *343*
factor.pa *(fa.poly)*, 177
factor.plot *(cluster.plot)*, 62
factor.residuals, *13*, 192
factor.rotate, *13*, *156*, 193, *195*, *344*
factor.scores, *11*, *46*, *144*, *148*, *152*, *153*, *155*, *179*, *182*, 195, *196*, *454*
factor.stats, 197, *197*
factor.wls, *11*
factor.wls *(fa.poly)*, 177
factor2cluster, *7*, *9*, *13*, *60*, *61*, *63*, *64*, *142*, *155*, 200, *201*, *243*, *260*, *262*, *339*, *389–391*
faReg, *162*
faReg *(fa.extension)*, 161
faRegression, *142*, *163*
faRegression *(fa.extension)*, 161
faRotate *(Promax)*, 341
faRotations, 201, *202*, *343*
fisherz, *14*, *82*, 204
fisherz2r, *14*
fisherz2r *(fisherz)*, 204
fparse, 206
fromTo, *348*
fromTo *(psych.misc)*, 345
fsi, *145*, *152*, *153*, 207, *207*, *208*

g2r *(fisherz)*, 204
galton, *10*, *15*
Garcia, 209, *279*
geometric.mean, *11*, 211
glb *(splitHalf)*, 435
glb.algebraic, *12*, *16*, 212, *438*, *439*
glb.fa, *214*, *438*
Gleser, 215
Gorsuch, 216
GSBE, *262*, *279*
GSBE *(Garcia)*, 209
guttman, *7*, *9*, *11–13*, *21*, *212*, *214*, *296*, *384*, *387*, *431*
guttman *(splitHalf)*, 435

Harman, *31*, 217
Harman.5, *218*, *219*
Harman.5 *(Harman)*, 217
Harman.8, *218*
Harman.8 *(Harman)*, 217

Harman.Burt (Harman), 217
Harman.Holzinger (Harman), 217
Harman.political, *218, 219*, *256*
Harman.political (Harman), 217
Harman74.cor, *218*
Harman_5, *219*
Harman_5 (Harman), 217
harmonic.mean, *11*, *212*, 219
head, *220, 221*
headTail, 220, *221*
headtail, *11*
headtail (headTail), 220
heights, *10*, *15*
het.diagram, *116*, *157*, *159*
het.diagram (fa.diagram), 157
histBy, *109*, *130*, *368*
histBy (multi.hist), 286
histo.density (multi.hist), 286
Holzinger, *217*, *344*
Holzinger (Bechtoldt), 30
Holzinger.9, *217*
holzinger.swineford, *217*

ICC, *7*, *12*, *14*, *70*, 222, *458*
ICLUST, *7*, *9*, *18*, *21*, *59–63*, *141*, *150*, *155*, *159*, *181*, *190–193*, *198*, *201*, *234–238*, *241, 242*, *260*, *262*, *296*, *301*, *327–330*, *341*, *370, 371*, *383*, *390*, *408*, *428*, *439*, *476–480*
ICLUST (iclust), 225
iclust, *11*, *22*, *29*, *37*, *88*, *115*, *167*, *184, 185*, *187*, 225, *226*, *229*, *298*, *319*, *356, 357*, *391*
ICLUST.cluster, *192*, *232*, 233, *243*
ICLUST.diagram, *11*, *14*, *116, 117*, *230*, *232*, *301*
ICLUST.diagram (iclust.diagram), 235
iclust.diagram, *230*, 235, *447*
ICLUST.graph, *9*, *11*, *13*, *63*, *160*, *192*, *230*, *232*, *234, 235*, 237, *241–243*
iclust.graph (ICLUST.graph), 237
ICLUST.rgraph, *11*, *16*, *230*, *232*, *235–237*, 240, *304*
ICLUST.sort, *61*, 242
iclust.sort, *232*
iclust.sort (ICLUST.sort), 242
interbattery, *31*, *51*
interbattery (esem), 138
interp.boxplot (interp.median), 244
interp.median, *11*, *112*, 244, *481*
interp.q (interp.median), 244
interp.qplot.by (interp.median), 244
interp.quantiles (interp.median), 244
interp.quart (interp.median), 244
interp.quartiles (interp.median), 244
interp.values (interp.median), 244
invMatSqrt, *326*
invMatSqrt (Pinv), 325
iqitems, *15*
irt.0p (irt.1p), 245
irt.1p, 245
irt.2p (irt.1p), 245
irt.CFA, *6, 7*, *53, 54*, *386*
irt.CFA (CFA), 51
irt.discrim, *246*
irt.discrim (irt.item.diff.rasch), 251
irt.fa, *7*, *9*, *11*, *39, 40*, *49*, *53*, *78*, *119*, *141*, *148, 149*, *155*, *195*, *245, 246*, 247, *247–249*, *252*, *283*, *299*, *327–329*, *359*, *376*, *378, 379*, *386, 387*, *406*, *454, 455*, *465*
irt.item.diff.rasch, *15*, *247*, 251
irt.person.rasch, *15*, *252*
irt.person.rasch (irt.1p), 245
irt.responses, *249*, 253, *379*, *387*
irt.se, *378*
irt.se (scoreIrt), 376
irt.select, *248*
irt.select (irt.fa), 247
irt.stats.like, *328*, *378*
irt.stats.like (scoreIrt), 376
irt.tau, *378*
irt.tau (scoreIrt), 376
isCorrelation (psych.misc), 345
isCovariance (psych.misc), 345
item.dichot (sim.item), 414
item.lookup, *167*
item.lookup (fa.lookup), 165
item.sim, *323*, *406*, *427*
item.sim (sim.item), 414
item.validity, *334*, *357*
item.validity (predicted.validity), 334
itemSort, *167*
itemSort (fa.lookup), 165

kaiser, *11*, *152*, 254, *254*
keys.lookup (fa.lookup), 165
keys2list, *201*, *268*

keys2list (make.keys), 268
keysort (parcels), 318
KMO, *98*, *142*, *149*, *155*, 255
kurtosi, *11*, *112*, *395*
kurtosi (mardia), 272

lavaan.diagram, *116*, *445*, *447*
lavaan.diagram (structure.diagram), 445
lavParse, *206*
lavParse (fparse), 206
layout, *118*
levels2numeric, *347*
levels2numeric (psych.misc), 345
lm, *259*, *392*, *393*
lmCor, *7*, *12*, *13*, *141*, *206*, 257, *260*, *261*, *279*, *320*, *362*, *363*, *390*, *392*, *393*
lmCor.diagram, *279*
lmCorLookup, *261*
lmCorLookup (fa.lookup), 165
lmDiagram (lmCor), 257
logistic, *247*, 265, *399*, *411*, *422*
logit (logistic), 265
lookup, *35*, *166*, *167*
lookup (fa.lookup), 165
lookupFromKeys (fa.lookup), 165
lookupItems, *167*
lookupItems (fa.lookup), 165
lowerCor, *83*, *96*, *322*, *347*
lowerCor (psych.misc), 345
lowerMat, *83*, *96*, *320*, *347*
lowerMat (psych.misc), 345
lowerUpper, *88*, *96*, 266, *348*
lsat6 (bock), 49
lsat7 (bock), 49

m2d (cohen.d), 64
m2t, *66*
m2t (cohen.d), 64
mahalanobis, *308*
make.congeneric, *426*
make.congeneric (sim.congeneric), 404
make.hierarchical, *301*, *304*, *426*
make.hierarchical (sim.hierarchical), 406
make.irt.stats (scoreIrt), 376
make.keys, *9*, *12*, *39*, *64*, *83*, *201*, 268, *268*, *378*, *381*–*383*, *387*, *389*, *451*
makePositiveKeys (make.keys), 268
manhattan, 270
MAP, *7*, *9*, *12*, *174*
MAP (VSS), 473
mardia, 272
mat.regress, *9*, *13*, *391*, *432*
mat.regress (lmCor), 257
mat.sort, *88*, *232*, 274
matMult (psych.misc), 345
matPlot, *261*
matPlot (lmCor), 257
matReg, *261*
matReg (lmCor), 257
matrix.addition, 275
matSort (mat.sort), 274
matSqrt, *326*
matSqrt (Pinv), 325
mean, *212*
median, *245*
mediate, *7*, *22*, *206*, *210*, *258*, *261*, *262*, 276
minkowski (ellipses), 122
misc (psych.misc), 345
mixed.cor, *12*, *88*, *148*, *283*, *464*, *465*
mixed.cor (mixedCor), 281
mixedCor, *13*, *52*, *145*, 281, *283*, *464*
mlArrange, *10*, *11*, *291*
mlArrange (multilevel.reliability), 288
mlPlot, *10*, *291*
mlPlot (multilevel.reliability), 288
mlr, *10*, *354*, *357*
mlr (multilevel.reliability), 288
moderate.diagram (mediate), 276
msq, *15*, *391*
msqR, *459*
mssd, 285
multi.arrow, *116*
multi.arrow (diagram), 114
multi.curved.arrow, *116*
multi.curved.arrow (diagram), 114
multi.hist, *11*, 286, *368*
multi.rect, *116*
multi.rect (diagram), 114
multi.self, *116*
multi.self (diagram), 114
multilevel.reliability, *7*, *10*, 288, *458*

nchar2numeric, *348*
nchar2numeric (psych.misc), 345
nfactors, *9*, *141*, *155*, *174*, *176*, *301*, *474*
nfactors (VSS), 473

omega, 7, 9, 10, 12, 16, 18, 21, 22, 28–30, 51, 86, 115, 142, 155, 159, 161, 162, 167, 171, 187, 192, 217, 222, 226, 228, 232, 234, 248, 293, 296–299, 302, 304, 305, 307, 316, 327–329, 354, 356, 357, 359, 362, 371, 374, 375, 383, 384, 387, 407, 408, 416, 424, 436, 439, 459, 472, 473, 476, 477, 479, 480
omega.diagram, 7, 14, 116, 117, 447
omega.diagram (omega.graph), 302
omega.graph, 12, 13, 160, 294, 301, 302, 371, 447
omegaDirect, 298
omegaDirect (omega), 293
omegaFromSem, 297, 299, 300
omegaFromSem (omega), 293
omegah, 356
omegah (omega), 293
omegaSem, 7, 12, 30, 297–300
omegaSem (omega), 293
omegaStats, 54, 155, 305, 305
optim, 147
options, 283, 462
outlier, 308, 314

p.adjust, 94, 96
p.rep, 8, 14, 309
p.rep.r, 312
paired.r, 14, 311, 351
pairs, 314
pairs.panels, 6, 8, 11, 47, 96, 110, 112, 122, 123, 312, 313, 368, 458
pairwiseCount, 248, 315, 316
pairwiseCountBig, 45, 316
pairwiseCountBig (pairwiseCount), 315
pairwiseDescribe (pairwiseCount), 315
pairwiseImpute, 317
pairwiseImpute (pairwiseCount), 315
pairwisePlot (pairwiseCount), 315
pairwiseReport, 316
pairwiseReport (pairwiseCount), 315
pairwiseSample, 317
pairwiseSample (pairwiseCount), 315
pairwiseZero, 317
pairwiseZero (pairwiseCount), 315
panel.cor (pairs.panels), 312
panel.ellipse (pairs.panels), 312
panel.hist (pairs.panels), 312
panel.lm (pairs.panels), 312
panel.smoother (pairs.panels), 312
parcels, 318
partial.r, 7, 12, 96, 256, 262, 319, 320
paSelect (fa.parallel), 172
pca, 6, 9, 28, 29, 167, 185, 187, 202, 203, 333
pca (principal), 335
peas, 10, 15
phi, 10, 14, 25, 26, 75, 321, 483, 485
phi.demo, 13, 14, 323, 331
phi.list (structure.list), 449
phi2poly, 14, 331
phi2poly (phi2tetra), 324
phi2poly.matrix, 14, 325
phi2poly.matrix (polychor.matrix), 331
phi2tetra, 26, 75, 322, 324
Pinv, 325, 326
plot, 68
plot.irt, 247, 248
plot.irt (plot.psych), 327
plot.poly, 247, 248
plot.poly (plot.psych), 327
plot.poly.parallel (fa.parallel), 172
plot.psych, 14, 63, 249, 327
plot.reliability, 328, 329, 354, 356
plot.reliability (reliability), 354
plot.residuals, 328
plot.residuals (plot.psych), 327
pmi (Tal_Or), 453
polar, 10, 14, 329
poly.mat, 12, 463
poly.mat (tetrachoric), 460
polychor.matrix, 331
polychoric, 7, 9, 12, 13, 26, 39, 42, 46, 52, 75, 77, 78, 148, 149, 232, 247–249, 252, 282–284, 299, 322, 347, 406, 462–464
polychoric (tetrachoric), 460
polydi, 283
polydi (tetrachoric), 460
polyserial, 12, 13, 282, 283
polyserial (tetrachoric), 460
predict, 12, 262
predict.psych, 35, 142, 152, 155, 182, 332, 339
predicted.validity, 334, 334, 357
principal, 9, 11, 37, 45–47, 86, 98, 115, 121, 141, 146, 150, 155, 164, 167, 181,

184, 189, 193, 198, 200, 201, 260, 262, 295, 327–329, 333, 335, 343, 344, 359, 390
princomp, 337
print.psych, 150, 181, 185, 231, 340
Procrustes (Promax), 341
progressBar, 283, 347
progressBar (psych.misc), 345
Promax, 7, 254, 255, 299, 341
promax, 299, 344
protest (Garcia), 209
psych, 10, 110, 290
psych (00.psych), 6
psych-package (00.psych), 6
psych.misc, 345
psychTools, 6

quickView (headTail), 220

r.con, 8, 14, 351
r.con (fisherz), 204
r.test, 8, 10, 14, 96, 312, 348, 349
r2alpha (alpha), 16
r2c (fisherz), 204
r2chi (fisherz), 204
r2d (cohen.d), 64
r2p (fisherz), 204
r2t (fisherz), 204
radar, 432, 434
radar (spider), 432
rangeCorrection, 352, 352
rd2html, 14
read.clipboard, 6, 8, 10, 110, 112, 388
read.clipboard.csv, 10
read.clipboard.lower, 10, 267
read.clipboard.upper, 10
read.file, 6, 8, 10
reflect, 184
reflect (psych.misc), 345
Reise (Bechtoldt), 30
reliability, 7, 9–12, 226, 230, 231, 287, 297, 301, 334, 335, 354, 354, 356, 357, 439, 472, 473
removeMissing (scoreItems), 381
rescale, 11, 358, 358, 394
resid.psych (residuals.psych), 359
residuals, 151, 182
residuals.psych, 359
response.frequencies, 381
response.frequencies (scoreItems), 381
responseFrequency (scoreItems), 381
reverse.code, 360, 394
RMSEA, 361
rmssd, 286
rmssd (mssd), 285
RV, 362

sai, 458, 459
SAPAfy (psych.misc), 345
sat.act, 10, 15, 364
scale, 359
scaling.fits, 14, 365
scatter.hist, 119, 314
scatter.hist (scatterHist), 366
scatterHist, 14, 67, 68, 109, 130, 288, 366
Schmid, 371
schmid, 6, 12, 16, 294, 295, 301, 326, 369, 408
schmid.leiman (Schmid), 371
Schutz, 146
score.alpha, 373
score.irt, 246, 249, 252–254
score.irt (scoreIrt), 376
score.items, 9, 12, 18, 39, 63, 64, 90, 283, 318, 341, 375
score.items (scoreItems), 381
score.multiple.choice, 7, 9, 12, 253, 254, 374, 387
scoreBy, 389, 390
scoreBy (scoreOverlap), 388
scoreFast, 335, 381, 385, 386
scoreFast (scoreItems), 381
scoreIrt, 7, 10, 284, 376, 376, 378, 386, 387, 454, 455
scoreIrt.1pl, 7, 10, 268, 269, 378, 386, 387
scoreIrt.2pl, 7, 10, 268, 269, 376, 378
scoreItems, 7, 9, 12, 19–21, 35, 52, 82, 83, 88, 142, 148, 155, 231, 268–270, 284, 296, 319, 335, 337, 354–357, 373, 374, 378, 379, 381, 385, 386, 389–392, 436, 437, 451, 459, 471
scoreOverlap, 7, 10, 12, 52, 83, 88, 268, 269, 284, 316, 354, 386, 387, 388, 390, 451
scoreVeryFast, 381, 385, 386
scoreVeryFast (scoreItems), 381
scoreWtd, 35, 387, 392, 392
scree (VSS.scree), 479
scrub, 14, 393

SD, 395
selectFromKeys, *268*
selectFromKeys (make.keys), 268
sem.diagram (structure.diagram), 445
sem.graph (structure.diagram), 445
Sensitivity (AUC), 23
set.cor, *312*
set.cor (lmCor), 257
setCor, *13*, *22*, *210*
setCor (lmCor), 257
shannon, *347*
shannon (psych.misc), 345
sim, *13*, *206*, 396, *396*, *398*, *403*, *409*, *420*, *430*
sim.anova, *10*, *13*, *397*, *400*, 402, *409*, *413*, *420*, *423*
sim.bonds, *407*
sim.bonds (sim.hierarchical), 406
sim.circ, *8*, *10*, *13*, *58*, *396*, *400*, *409*, *412*, *420*, *422*, *430*
sim.circ (sim.item), 414
sim.congeneric, *8*, *13*, *396*, *400*, 404, *409*, *412*, *420*, *422*, *425*
sim.correlation, *424*, *426*
sim.correlation (sim.structure), 424
sim.dichot, *10*, *396*, *400*, *409*, *412*, *420*, *422*
sim.dichot (sim.item), 414
sim.general, *399*, *411*, *421*
sim.general (sim.omega), 420
sim.hierarchical, *8*, *13*, *28*, *396*, *400*, 406, *407*, *409*, *412*, *416*, *420*, *422*, *430*
sim.irt, *13*, *247*, *249*, *396*, *399*, 409, *409*, *420*, *454*
sim.item, *8*, *10*, *13*, *396*, *400*, *409*, *412*, 414, *420*, *422*
sim.minor, *13*, *147*, *174*, *176*, *396*, *398*, *400*, *409*, *411*, *412*, *420–422*
sim.multi, *292*, *418*
sim.multi (sim.multilevel), 417
sim.multilevel, *292*, *400*, *413*, 417, *418*, *423*, *442*, *482*
sim.npl, *396*, *399*, *409*, *420*
sim.npl (sim.irt), 409
sim.npn, *396*, *399*, *409*, *420*
sim.npn (sim.irt), 409
sim.omega, *396*, *399*, *409*, *411*, 420, *420*, *421*
sim.parallel, *396*, *400*, *409*, *413*, *420*, *423*
sim.parallel (sim.omega), 420
sim.poly, *396*, *409*, *420*
sim.poly (sim.irt), 409
sim.poly.ideal, *397*, *399*, *409*, *412*, *420*, *422*
sim.poly.ideal.npl, *397*, *409*, *420*
sim.poly.ideal.npn, *397*, *409*, *420*
sim.poly.npl, *396*, *409*, *420*
sim.poly.npn, *396*, *409*, *420*
sim.rasch, *247*, *396*, *399*, *409*, *420*
sim.rasch (sim.irt), 409
sim.simplex, *396*, *398*, *409*, *411*, *420*, *421*
sim.spherical, *13*, *416*
sim.spherical (sim.item), 414
sim.structural, *8*, *10*, *13*, *396*, *398*, *409*, *416*, *420*, *430*, *447*, *451*
sim.structural (sim.structure), 424
sim.structure, *206*, *398*, *400*, *408*, *411*, *412*, *421*, *422*, 424, *450*
sim.VSS, *13*, *397*, *409*, *420*, 427
simCor, *396*, *416*, *424*, *426*, *430*
simCor (sim.structure), 424
simulation.circ, *416*, 428, *428*
skew, *11*, *112*, *395*
skew (mardia), 272
small.msq, 430
smc, *12*, *174*, 431
Specificity (AUC), 23
spider, *14*, 432, *434*
splitHalf, *10*, *12*, *226*, *228*, *354–357*, 435, *436–438*
stack, *291*
statsBy, *7*, *8*, *10*, *11*, *41*, *80*, *109*, *113*, *114*, *132*, *133*, *135–137*, *220*, *286*, *292*, *314*, *389*, *390*, *418*, *419*, 440, *442*, *443*, *482*
statsBy.boot, *443*
statsBy.boot.summary, *443*
structure.diagram, *14*, *116*, *117*, *159*, *160*, *303*, *400*, *412*, *422*, 445, *445*
structure.graph, *13*, *16*, *450*, *451*
structure.graph (structure.diagram), 445
structure.list, *426*, 449
structure.sem, *445*
structure.sem (structure.diagram), 445
summary, *110*
summary.psych (print.psych), 340
super.matrix (superMatrix), 450
superCor (superMatrix), 450
superMatrix, *269*, 450

t2d (cohen.d), 64

t2r (fisherz), 204
table2df, *11*, *49*
table2df (table2matrix), 452
table2matrix, 452
tableF, *347*
tableF (psych.misc), 345
tail, *220, 221*
Tal.Or, *320*
Tal.Or (Tal_Or), 453
Tal_Or, 453
target.rot, *7*, *148, 149*, *294*, *337*
target.rot (Promax), 341
TargetQ (Promax), 341
TargetT (Promax), 341
tctg (Tal_Or), 453
tenberge (splitHalf), 435
test.all, *347*
test.all (psych.misc), 345
test.irt, 454
test.psych, *15*, 455
testReliability (testRetest), 457
testRetest, *223*, *354*, *357*, 457
tetrachor, *14*
tetrachor (tetrachoric), 460
tetrachoric, *7*, *9*, *12, 13*, *24–26*, *46*, *49*, *52*, *75*, *77, 78*, *118, 119*, *148, 149*, *173*, *232*, *247–249*, *252*, *282–284*, *322*, *324, 325*, *345*, *347*, 460, *462*, *471*, *483*, *485*
Thurstone, *15*, *300*
Thurstone (Bechtoldt), 30
thurstone, *14*, *366*, 467
topBottom, *221*
topBottom (headTail), 220
tr, *14*, *362*, 469
Tucker, *15*, 469

unidim, *7*, *9*, *12*, *297*, *354–357*, *439*, 470, *470*
usaf, *218*

validityItem (predicted.validity), 334
varimin (Promax), 341
vech (psych.misc), 345
vegetables, *10*, *15*, *366*
vgQ.bimin (Promax), 341
vgQ.targetQ (Promax), 341
vgQ.varimin (Promax), 341
View, *220, 221*
violin, *288*
violin (densityBy), 107
violinBy, *41*, *112*, *114*, *288*
violinBy (densityBy), 107
VSS, *7*, *9*, *12*, *59, 60*, *141*, *150*, *155*, *173, 174*, *176*, *181*, *190–193*, *198*, *225*, *230*, *232*, *234*, *243*, *301*, *327*, *329, 330*, *339*, *371*, *408*, *428*, 473, *478, 479*
vss (VSS), 473
VSS.parallel, *9*, *12*, *176*, 477
VSS.plot, *12*, *176*, *238*, *242*, *329*, *476, 477*, 478, *480*
VSS.scree, *9*, *12*, *339*, 479
VSS.sim (sim.VSS), 427
VSS.simulate, *323*
VSS.simulate (sim.VSS), 427
vssSelect (VSS), 473

West (Schmid), 371
winsor, 480
withinBetween, *419*, *442*, 481
wkappa, *14*
wkappa (cohen.kappa), 70

Yule, *10*, *14*, *26*, *75*, *322*, 483
Yule.inv, *15*, *26*, *75*, *322*
Yule2phi, *15*, *26*, *75*, *322*, *331*
Yule2phi (Yule), 483
Yule2phi.matrix, *325*, *485*
Yule2phi.matrix (polychor.matrix), 331
Yule2poly, *331*
Yule2poly (Yule), 483
Yule2poly.matrix, *485*
Yule2poly.matrix (polychor.matrix), 331
Yule2tetra, *15*, *331*
Yule2tetra (Yule), 483
YuleBonett (Yule), 483
YuleCor (Yule), 483