

Package: methylclock (via r-universe)

October 14, 2026

Type Package

Title Chronological, Gestational and Biological DNAm Age Estimation with Methylation-Based Clocks

Version 1.99.2

Date 2026-09-14

Description Estimates chronological, gestational and biological DNA methylation (DNAm) age, trait scores and epigenetic age acceleration from methylation beta values. Implements 44 published clocks, selectable by name, family or array platform: chronological (Horvath, Hannum, BNN, skin+blood, PedBE, Wu, AltumAge and others, including the cross-platform Garma clock native to EPICv2), gestational (Knight, Bohlin, Mayne, Lee, NEOage), phenotypic and pace-of-aging (Levine, DunedinPACE), telomere length, mitotic counters (epiTOC1/epiTOC2, HypoClock, stemTOC, RepliTali), causal (CausAge, DamAge, AdaptAge) and trait EpiScores. Accepts matrices, data frames, Bioconductor containers or HDF5 files, computing block-wise out-of-core through BigDataStatMeth when the data exceed memory. Provides configurable missing-value imputation (mean, reference or KNN), cell-type deconvolution, intrinsic and extrinsic epigenetic age acceleration (IEAA, EEAA), quality-control reports and plotting functions.

biocViews DNAMethylation, BiologicalQuestion, Preprocessing, StatisticalMethod, Normalization

License MIT + file LICENSE

SystemRequirements GNU make, C++17

Depends R (>= 4.3.0)

Imports utils, stats, preprocessCore, Rcpp, BigDataStatMeth, ExperimentHub, ggplot2, rlang, quadprog

LinkingTo Rcpp, RcppEigen, Rhdf5lib, BigDataStatMeth, BH

Suggests testthat (>= 3.0.0), digest, impute, knitr, rmarkdown, BiocStyle, kableExtra, minfi, Biobase, SummarizedExperiment

Encoding UTF-8

URL <https://github.com/isglobal-brge/methylclock>

BugReports <https://github.com/isglobal-brge/methylclock/issues>

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs make libicu-dev libpng-dev libssl-dev zlib1g-dev

Repository <https://bioc.r-universe.dev>

Date/Publication 2026-09-14 11:29:34 UTC

RemoteUrl <https://github.com/bioc/methylclock>

RemoteRef HEAD

RemoteSha 30be0284066fda7d2f81ec4bf427da56f0f89d1b

Contents

ageAcceleration	3
ageAccelerationChen	4
anti_trafo	6
as.data.frame.methylclock	7
cellCounts	7
clock_catalog	8
clock_info	9
clock_list	10
clock_predictors	11
clock_register	12
clock_registry	14
clock_reset	14
clock_statuses	15
clock_targets	15
clockAccuracy	16
clockCoverage	17
clockTrajectories	17
compute_clocks	18
DNAmAge	20
DNAmGA	21
EEAA	22
IEAA	23
imputeKNN	25
listCellReferences	26
mc_to_hdf5	26
mcd_backends	28
mcd_cache_clear	29
mcd_manifest	29
mcd_resource	30
mcd_resource_file	30

methyleclock	31
methyleclock_betas	32
methyleclock_cells	33
methyleclock_demo	34
methyleclock_longitudinal	35
methyleclock_references	35
methyleclock_smoking	36
methyleclock_validation	37
new_methyleclock	38
normalize_coef	39
persist	40
plotAccelerationByGroup	41
plotAgeAcceleration	41
plotBlandAltman	42
plotClockCorrelation	43
plotClockDensities	44
plotClockDistributions	45
plotDNAmAge	46
plotGroupDifference	47
plotReferenceRange	49
plotSampleDiscordance	50
plotSamplePCA	52
plotTrajectories	53
qcReport	54
sampleQC	55
theme_methyleclock	56
trajectoryRates	56
Index	58

ageAcceleration	<i>Epigenetic age acceleration, intrinsic and extrinsic</i>
-----------------	---

Description

For each age clock, how much older (or younger) a sample looks than its chronological age. Three measures per clock:

ageAcc The raw difference, predicted minus chronological age.

residual The classic age-acceleration residual: predicted regressed on chronological age. (The published EEAA of Chen and colleagues is a different, cell-weighted measure; [EEAA](#) returns it under its usual name.)

residualCells The residual further adjusted for blood cell composition (whatever panel you supply), so it reflects ageing independent of shifts in cell proportions. Only returned when `cell_counts` is supplied. This is the same adjustment idea as the canonical IEAA of the literature, but that measure is defined with a specific set of seven immune covariates — [IEAA](#) computes it as defined.

Estimate the cell proportions with [cellCounts](#).

Usage

```
ageAcceleration(x, age, cell_counts = NULL, clocks = NULL)
```

Arguments

<code>x</code>	A methylclock result, or a data frame with an <code>id</code> column and one numeric column per clock.
<code>age</code>	Numeric vector of chronological ages, one per sample in <code>x</code> 's order (or named by sample id).
<code>cell_counts</code>	Optional samples-by-cell-types matrix (from cellCounts); adds the cell-adjusted residual <code>residualCells</code> .
<code>clocks</code>	Optional clock names. Default: the age clocks in <code>x</code> .

Value

A data frame with columns `id`, `clock`, `ageAcc`, `residual` and (with `cell_counts`) `residualCells`.

See Also

[cellCounts](#), [plotAgeAcceleration](#)

Examples

```
data(methylclock_demo)
head(ageAcceleration(methylclock_demo, age = methylclock_demo$age))
```

ageAccelerationChen	<i>Chen-style extrinsic age acceleration with data-driven weights</i>
---------------------	---

Description

An immune-aware measure of epigenetic age acceleration in the spirit of Chen et al. (2016): a Kleméra–Doubal weighted average of the Hannum clock with immune cell types that track age (by default naive CD8+ T cells, exhausted CD8+ T cells and plasmablasts) is formed, then regressed on chronological age, and the returned `chenAcc` is that residual.

Usage

```
ageAccelerationChen(
  hannum,
  age,
  cells,
  immune = c("CD8nv", "CD8exhausted", "PlasmaBlast")
)
```

Arguments

hannum	Numeric vector of Hannum-clock estimated ages, one per sample. May be named by sample id.
age	Numeric vector of chronological ages, one per sample.
cells	Samples-by-cell-types matrix with the immune cell measures; it must contain every column named in <code>immune</code> .
immune	Column names in <code>cells</code> for the immune subsets that enter the weighted average. All of them must be present: the blend is only the intended measure with its full set of inputs, so a missing column is an error rather than a silent change of definition.

Details

The defining trait of this function is that the Kleméra–Doubal weights are re-estimated from YOUR data, and the immune inputs are whatever cell measures you supply. The published EEAA instead fixes both: its weights were fixed once in the Women’s Health Initiative cohorts and its cell inputs are three specific CpG-based estimators (the parameters of that fixed recipe are published in European patent EP 3 494 210 B1, and [EEAA](#) implements it). The patent documents the re-estimated flavour as well (its "dynamic" variant) and reports the two to be highly correlated; still, the two are numerically different measures, which is why this function’s output column is named `chenAcc` and not `EEAA`.

Value

A data frame with columns `id`, `age` and `chenAcc`.

Where the immune proportions can come from

The subsets this measure was defined on — naive CD8+ T cells, exhausted CD8+ T cells (CD28-CD45RA-) and plasmablasts — are not resolved by the reference panels bundled here (see [listCellReferences](#)), which report total CD8+ T cells. Two external routes are commonly used, and both produce values you can pass straight to `cells`:

- the *advanced analysis* of the online epigenetic-age calculator of Horvath (2013), which reports these subsets directly;
- an extended immune deconvolution such as the EPIC IDOL-Ext library of Salas et al. (2022), distributed as the **FlowSorted.BloodExtended.EPIC** package under its own licence from the authors’ institution — with the caveat that it resolves naive and memory CD8+ T cells and B cells but NOT exhausted CD8+ T cells or plasmablasts, so proportions from it are stand-ins: the result is a measure in the spirit of Chen’s, not the one defined on the original subsets.

Users who obtain proportions this way should cite the source they used. Note that the rarer subsets (plasmablasts, exhausted CD8+ T cells) are reported by some algorithms as scores linearly related to abundance rather than as cell-type proportions, which is worth keeping in mind when comparing values across sources.

References

Chen BH, Marioni RE, Colicino E, et al. (2016). DNA methylation-based measures of biological age. *Aging* 8(9):1844–1865. Klemnera P, Doubal S (2006). A new approach to the concept and computation of biological age. *Mech Ageing Dev* 127(3):240–248. European patent EP 3 494 210 B1 (2024). DNA methylation based biological age prediction.

See Also

[EEAA](#) for the canonical fixed-weight measure, [IEAA](#), [ageAcceleration](#), [cellCounts](#)

Examples

```
## Not run:
# `subsets` holding naive CD8+, exhausted CD8+ and plasmablast measures
# per sample, e.g. from the Horvath calculator's advanced analysis:
chen <- ageAccelerationChen(res$Hannum, age = pheno$age, cells = subsets)

## End(Not run)
```

anti_trafo	<i>Horvath age transformation (inverse)</i>
------------	---

Description

Several clocks are trained on an age variable that is compressed on a log scale below a reference adult age. This maps a raw prediction back to years.

Usage

```
anti_trafo(x, adult_age = 20)
```

Arguments

- x Numeric vector of raw predictions.
- adult_age Reference adult age used by the transformation. Default 20.

Value

Numeric vector of ages in years.

Examples

```
anti_trafo(c(-0.5, 0, 0.5, 1.5))
```

as.data.frame.methylclock

Coerce a methylclock result to a data frame

Description

Returns the estimates as a data frame, materialising them from disk first if they are not already held in memory. The backend is invisible to the caller.

Usage

```
## S3 method for class 'methylclock'
as.data.frame(x, ...)
```

Arguments

x	A "methylclock" object.
...	Unused.

Value

A data frame of estimates.

cellCounts

Estimate blood cell-type proportions from methylation

Description

Reference-based deconvolution: given a methylation matrix, it estimates the proportion of each blood cell type per sample, by fitting the sample's profile as a non-negative mix of purified-cell reference profiles (Houseman et al. 2012). These proportions are what the intrinsic age acceleration adjusts for.

Usage

```
cellCounts(
  betas,
  reference = "blood gse35069 complete",
  normalize = TRUE,
  array = c("EPIC", "450k")
)
```

Arguments

betas	A methylation matrix (or data frame, HDF5-backed matrix, or an accepted Bio-conductor container) with CpG identifiers as row names.
reference	Reference panel name (see listCellReferences). Default "blood gse35069 complete" (whole-blood, 7 cell types). The high-resolution FlowSorted panels are supported only from the user's own installation of the corresponding package (see Details).
normalize	Quantile-normalise the CpGs to a shipped reference before fitting (ignored for the FlowSorted panels). Default TRUE.
array	"EPIC" or "450k"; selects the marker set for the extended FlowSorted panel. Default "EPIC".

Details

The panels named in [listCellReferences](#) are shipped via the data layer. Two higher-resolution panels are also supported but not shipped, because of their licence: "FlowSorted.Blood.EPIC" (6 cells, GPL-3) and "FlowSorted.BloodExtended.EPIC" (12 immune cells, Salas et al. 2022, a restrictive non-commercial academic licence). Install the package yourself and it is used at run time; nothing licensed is redistributed by methylclock.

Value

A matrix of samples (rows) by cell types (columns), each row summing to approximately one.

References

Houseman EA et al. (2012) *BMC Bioinformatics* 13:86. Salas LA et al. (2022) *Nature Communications* 13:761.

Examples

```
## Not run:
cellCounts(betas)
cellCounts(betas, reference = "FlowSorted.BloodExtended.EPIC")

## End(Not run)
```

clock_catalog

Catalogue of the clocks in the package

Description

A one-row-per-clock overview meant to be read first: what each clock estimates, which arrays it was built for, how many CpGs it uses, and a short reference to the paper. It is a frozen snapshot of the registry (regenerated when clocks are added), so it can be displayed without resolving any coefficients. For the live registry — to filter or query programmatically — use [clock_list](#).

Usage

```
data(clock_catalog)
```

Format

A data frame with one row per clock and columns:

clock Clock name (pass to methylclock(clocks =)).

estimates What the clock reports, with units (e.g. "Age").

tissue The tissue the clock was trained on (whole blood, cord blood, placenta, buccal, skin, multi-tissue...): a clock reads best the tissue it came from.

arrays Illumina platforms the clock supports.

n_cpgs Number of CpGs the clock uses.

generation Clock generation, where defined.

target The registry target family the clock belongs to.

citation Citation key for the clock's paper.

reference Short human-readable reference (author and year).

Examples

```
data(clock_catalog)
clock_catalog[clock_catalog$target == "chronological",
              c("clock", "estimates", "arrays", "n_cpgs", "reference")]
```

clock_info

Look up one clock's registry entry

Description

Look up one clock's registry entry

Usage

```
clock_info(name)
```

Arguments

name Clock name.

Value

The registry entry (class clock_entry); errors if the clock is not registered.

See Also

[clock_list](#)

Examples

```
clock_info("Horvath")
```

clock_list	<i>List and filter registered clocks</i>
------------	--

Description

List and filter registered clocks

Usage

```
clock_list(  
  target = NULL,  
  generation = NULL,  
  predictor = NULL,  
  platform = NULL,  
  status = NULL  
)
```

Arguments

target	Optional filter by biological target.
generation	Optional filter by generation.
predictor	Optional filter by prediction engine.
platform	Optional filter by native platform: keeps clocks whose native platform includes any of the given arrays (e.g. "450K").
status	Optional filter by availability status.

Value

A data frame with one row per matching clock.

See Also

```
clock\_info
```

Examples

```
clock_list(target = "gestational")  
clock_list(platform = "450K")
```

clock_predictors	<i>Supported prediction engines</i>
------------------	-------------------------------------

Description

The predictor families a registered clock can use. A clock's predictor must be one of these values.

Usage

```
clock_predictors
```

Format

An object of class character of length 7.

Details

linear intercept plus a weighted sum of features (most clocks).

pc project features onto a PCA rotation, then a linear model on the principal components.

surrogate two-stage model (methylation surrogates then a combining model); typically requires chronological age and sex.

counter a mean or quantile over a defined set of CpGs.

iterative a counter estimated per sample by numerical optimisation.

nn a neural network.

classifier a model with categorical output.

Value

A character vector of the supported engine names.

Examples

```
clock_predictors
```

clock_register	<i>Register a clock</i>
----------------	-------------------------

Description

Adds (or replaces) one clock in the package registry. A registration is a single declarative description of a clock; the estimation engine reads it to compute the clock's value.

Usage

```
clock_register(
  name,
  target,
  predictor = "linear",
  generation = NA_integer_,
  platform = character(0),
  compatible = character(0),
  tissue = NA_character_,
  resource = NULL,
  aux = NULL,
  coef_col = "CoefficientTraining",
  intercept = TRUE,
  standardize = FALSE,
  normalize = NA_character_,
  agg = "mean",
  transform = identity,
  inputs = character(0),
  feature_type = "cpg",
  output_type = "numeric",
  license = NA_character_,
  source = NA_character_,
  citation = NA_character_,
  status = "ok",
  notes = NA_character_
)
```

Arguments

name	Clock name (unique key), e.g. "Horvath".
target	Biological target; one of clock_targets .
predictor	Prediction engine; one of clock_predictors . Defaults to "linear".
generation	Clock generation (e.g. 1, 2, 3) or NA.
platform	Character vector of the clock's native arrays — the platform(s) it was built and validated on, e.g. c("450K", "EPIC").
compatible	Character vector of additional arrays the clock can be used with through probe conversion or adaptation, i.e. not its native platform.

tissue	The tissue the clock was trained on (e.g. "whole blood", "buccal epithelium (children)", "placenta", "multi-tissue"): a clock reads best the tissue it came from. Empty by default. Selection and estimation treat these as non-native.
resource	Resource identifier (resolved by mcd_resource) holding the coefficients, e.g. "coefHorvath".
aux	Character vector of auxiliary resource identifiers (e.g. normalization means, a PCA rotation, a cell-type reference), or NULL.
coef_col	Name of the coefficient column in the resource. Default "CoefficientTraining".
intercept	Logical: is the first coefficient row the intercept?
standardize	Logical: z-score each sample across the CpGs of the input matrix before applying the coefficients (as some elastic-net clocks require). Default FALSE.
normalize	Optional input normalization applied before the coefficients. "quantile_gs" quantile-normalizes each sample to a gold-standard reference distribution taken from the clock's aux resource. Default NA (none).
agg	For "counter" clocks, how the CpG set is aggregated per sample: "mean" (the default) or list(fun = "quantile", p = <0..1>) for an upper-quantile summary. Ignored by other engines.
transform	Function applied to the raw prediction (e.g. an anti-transform). Default identity.
inputs	Character vector of inputs required beyond the beta matrix (e.g. c("age", "sex")). Default none.
feature_type	Feature space of the predictor. Default "cpg"; kept configurable so the same engine can operate on non-CpG features.
output_type	One of "numeric", "probability", "category". Default "numeric".
license	Short license/provenance tag for the weights. Default NA.
source	Where the weights come from (URL, package, or resource id).
citation	Short citation key.
status	Availability status; one of clock_statuses . Default "ok".
notes	Free-text notes.

Value

Invisibly, the registry entry (a list of class clock_entry).

See Also

[clock_list](#), [clock_info](#), [mcd_resource](#)

Examples

```
clock_register("MyClock", target = "chronological", resource = "coefMyClock")
clock_info("MyClock")
```

clock_registry	<i>The clock registry</i>
----------------	---------------------------

Description

The clock registry

Usage

```
clock_registry()
```

Value

A named list of registry entries (objects of class `clock_entry`), ordered by name.

See Also

[clock_register](#), [clock_list](#)

Examples

```
reg <- clock_registry()
length(reg)
names(reg)[seq_len(5)]
reg[["Horvath"]]
```

clock_reset	<i>Clear the clock registry</i>
-------------	---------------------------------

Description

Removes all registered clocks. Mainly useful in tests, or when building a registry from scratch: the built-in clocks are registered when the package is loaded, so save them first if you want them back.

Usage

```
clock_reset()
```

Value

Invisibly NULL.

See Also

[clock_register](#), [clock_registry](#)

Examples

```
saved <- clock_registry()
clock_reset()
length(clock_registry())
# put the built-in clocks back
for (entry in saved) do.call(clock_register, unclass(entry))
length(clock_registry())
```

clock_statuses	<i>Availability status labels</i>
----------------	-----------------------------------

Description

A coarse status for a registered clock. "ok" is fully available; "license-risk" is usable but carries licensing constraints on its weights; "out-of-scope" does not apply to the package's target tissue; "not-implementable" has no usable published model; "todo" is planned.

Usage

```
clock_statuses
```

Format

An object of class character of length 5.

Value

A character vector of the supported status labels.

Examples

```
clock_statuses
```

clock_targets	<i>Supported biological targets</i>
---------------	-------------------------------------

Description

The quantity a clock estimates. A clock's target must be one of these.

Usage

```
clock_targets
```

Format

An object of class character of length 10.

Value

A character vector of the supported target names.

Examples

```
clock_targets
```

```
clockAccuracy
```

How well each clock tracks chronological age

Description

A quick quality-control summary for a dataset with known ages: for every age clock, how closely its estimate follows chronological age. It reports the number of samples, the Pearson correlation, the R-squared and the median absolute error (in years). Pass `by` to compute the same summary within groups — sex, case/control, cohort, ethnicity — which is the usual way to check that a clock behaves across subgroups rather than only on average.

Usage

```
clockAccuracy(x, age, by = NULL, clocks = NULL)
```

Arguments

<code>x</code>	A methylclock result, or a data frame with an <code>id</code> column and one numeric column per clock.
<code>age</code>	Numeric vector of chronological ages, one per sample in <code>x</code> 's order (or named by sample id).
<code>by</code>	Optional categorical vector (one value per sample, or named by id): the summary is computed separately within each group.
<code>clocks</code>	Optional clock names to restrict to. Default: the clocks that estimate an age in years (a comparison with chronological age is only meaningful for those).

Details

A high correlation with a large error means a clock *follows* age but is *miscalibrated* for your data (a constant offset), which the two columns surface separately.

Value

A data frame with one row per clock (per group), and columns `clock`, optionally `group`, `n`, `r`, `R2` and `MAE`.

Examples

```
data(methylclock_demo)
clockAccuracy(methylclock_demo, age = methylclock_demo$age,
              by = methylclock_demo$sex)
```

clockCoverage	<i>CpG coverage of each clock in a dataset</i>
---------------	--

Description

For every clock, how many of the CpGs it needs are present in your data. A clock built from only a fraction of its CpGs should be read with caution (and below the coverage threshold it is not computed at all). This needs the methylation matrix, not the estimates, so it can look up which CpGs you have.

Usage

```
clockCoverage(betas, clocks = NULL)
```

Arguments

betas	A methylation matrix (or data frame, HDF5-backed matrix, or one of the accepted Bioconductor containers): only its CpG identifiers (row names) are used.
clocks	Optional clock names to restrict to. Default: all clocks.

Value

A data frame with one row per clock and columns clock, n_cpgs (CpGs the clock uses), present (how many are in your data) and pct_present.

Examples

```
data(methylclock_betas)
clockCoverage(methylclock_betas, clocks = c("Horvath", "Hannum", "Levine"))
```

clockTrajectories	<i>Arrange repeated samples into per-subject clock trajectories</i>
-------------------	---

Description

When the same subjects were sampled at several ages, this pairs their samples into trajectories: one row per subject, visit and clock, ordered in time. It is the table the longitudinal plots and rates are built from, and the one to hand to a mixed-model package for formal analysis — for example `lme4::lmer(estimate ~ age + (age | subject), data = tr)` fits a population slope with per-subject deviations. This package shapes and displays longitudinal data; the inference belongs to those tools.

Usage

```
clockTrajectories(x, subject, age, clocks = NULL)
```

Arguments

x	A methylclock result, or a data frame with an id column and one numeric column per clock.
subject	Subject identifier (character or factor), one per sample in x's order, or named by sample id.
age	Numeric chronological age AT EACH SAMPLE, one per sample in x's order (or named by sample id). This is the time axis.
clocks	Optional character vector of clock names. Default: the age clocks present in x.

Details

Samples with a missing subject or age are dropped; repeated measurements of a subject at the same recorded age are averaged (whether they are technical replicates or distinct samples whose ages rounded to the same value is for the caller to know — pass more precise ages to keep them apart); and any subject-and-clock combination seen at fewer than two distinct ages is excluded — one point has no trajectory. Each exclusion is reported in a message.

Value

A data frame with columns subject, visit (1, 2, ... in age order), age, clock and estimate.

Examples

```
data(methylclock_longitudinal)
d <- methylclock_longitudinal
tr <- clockTrajectories(d, subject = d$subject, age = d$age,
                       clocks = "Wu")
head(tr)
```

compute_clocks	<i>Estimate clocks from a methylation matrix</i>
----------------	--

Description

Computes the selected clocks over a methylation beta matrix and returns a [new_methylclock](#) result. This is the engine shared by [methylclock](#), [DNAmAge](#) and [DNAmGA](#).

Usage

```
compute_clocks(
  x,
  clocks = "all",
  target = NULL,
  generation = NULL,
  platform = NULL,
  min.perc = 0.8,
```

```

    impute = c("mean", "reference", "none", "knn"),
    min.perc.sample = 0,
    k = 10,
    maxp = 1500,
    storage = c("auto", "hdf5", "memory"),
    file = NULL,
    compression = NULL,
    ...
)

```

Arguments

x	A methylation matrix with CpG identifiers as row names and samples as columns (a data frame with a CpG-identifier column is also accepted). A large matrix may be supplied on disk as an HDF5-backed matrix, in which case each clock reads only the CpGs it needs rather than the whole array.
clocks	Either "all" or a character vector of clock names.
target	Optional biological target to restrict to (e.g. "gestational"); "!gestational" selects everything else.
generation	Optional generation to restrict to (e.g. 1).
platform	Optional native platform to restrict to (e.g. "450K"); keeps clocks whose native platform includes any of the given arrays.
min.perc	Minimum fraction of a clock's CpGs that must be present. A clock is only computed when its coverage is strictly greater than this; otherwise it is returned as NA. Default 0.8.
impute	How missing values among the present CpGs are filled before the weighted sum. "mean" (default) fills each with that CpG's mean across samples; "reference" fills it with the CpG's reference level from the clock's own coefficients (mean fallback), staying closer to how the clock was trained; "none" fills nothing, so a sample missing any used CpG is returned NA instead of an imputed value; "knn" fills each from the CpG's nearest co-varying CpGs (see imputeKNN).
min.perc.sample	Per-sample coverage floor. For each clock, a sample with fewer than this fraction of the clock's present CpGs observed is returned NA, so a heavily imputed sample is not reported as if it were a real estimate. Default 0 (off).
k	Neighbours averaged when impute = "knn". Default 10.
maxp	Largest block imputed by exact KNN when impute = "knn"; larger row sets are split by 2-means first. Default 1500.
storage	How the estimates are held: "auto" (default) backs them on disk (HDF5, through BigDataStatMeth) and keeps an in-memory cache when they fit; "hdf5" backs them on disk without an in-memory cache; "memory" keeps them in memory only (they will not persist). When a disk write is not possible the estimates fall back to memory, with a warning.
file	HDF5 file to write the estimates to. When NULL (default) a session temporary file is used (or the directory in options(methylclock.hdf5_dir=) if set); pass a path to keep the result, or promote a session result later with persist .

compression	HDF5 compression level (0-9); NULL (default) picks it by size (none for small results, moderate for large ones).
...	Reserved for future inputs (e.g. age, sex, cell composition).

Value

A "methylclock" object.

See Also

[methylclock](#), [persist](#), [clock_list](#)

Examples

```
data(methylclock_betas)
res <- compute_clocks(methylclock_betas, clocks = c("Horvath", "Levine"),
                      storage = "memory")
head(as.data.frame(res))
```

DNAMAge

Estimate chronological and biological DNAm age

Description

Convenience wrapper around [methylclock](#) restricted to non-gestational clocks.

Usage

```
DNAMAge(
  x,
  clocks = "all",
  generation = NULL,
  platform = NULL,
  min.perc = 0.8,
  storage = "auto",
  ...
)
```

Arguments

x	A methylation matrix with CpG identifiers as row names and samples as columns (a data frame with a CpG-identifier column is also accepted), or a large matrix held on disk as an HDF5-backed matrix.
clocks	Either "all" (default) or a character vector of clock names; see clock_list .
generation	Optional generation to restrict to (e.g. 1).
platform	Optional native platform to restrict to (e.g. "450K"), handy when the data come from a known array and only the clocks built for it should run.

<code>min.perc</code>	Minimum fraction of a clock's CpGs that must be present for it to be computed (otherwise NA). Default 0.8.
<code>storage</code>	How the estimates are held: "auto" (default) backs them on disk (HDF5) with an in-memory cache when they fit, "hdf5" on disk only, "memory" in memory only (not persisted). See compute_clocks for the file and compression arguments, and persist to keep a session result.
<code>...</code>	Passed to compute_clocks (e.g. file, compression).

Value

A "methclock" object.

See Also

[methclock](#), [DNAmGA](#)

Examples

```
data(methclock_betas)
head(as.data.frame(DNAmAge(methclock_betas, clocks = "Horvath")))
```

DNAmGA	<i>Estimate gestational DNAm age</i>
--------	--------------------------------------

Description

Convenience wrapper around [methclock](#) restricted to gestational clocks.

Usage

```
DNAmGA(
  x,
  clocks = "all",
  generation = NULL,
  platform = NULL,
  min.perc = 0.8,
  storage = "auto",
  ...
)
```

Arguments

<code>x</code>	A methylation matrix with CpG identifiers as row names and samples as columns (a data frame with a CpG-identifier column is also accepted), or a large matrix held on disk as an HDF5-backed matrix.
<code>clocks</code>	Either "all" (default) or a character vector of clock names; see clock_list .
<code>generation</code>	Optional generation to restrict to (e.g. 1).

platform	Optional native platform to restrict to (e.g. "450K"), handy when the data come from a known array and only the clocks built for it should run.
min.perc	Minimum fraction of a clock's CpGs that must be present for it to be computed (otherwise NA). Default 0.8.
storage	How the estimates are held: "auto" (default) backs them on disk (HDF5) with an in-memory cache when they fit, "hdf5" on disk only, "memory" in memory only (not persisted). See compute_clocks for the file and compression arguments, and persist to keep a session result.
...	Passed to compute_clocks (e.g. file, compression).

Value

A "methclock" object.

See Also

[methclock](#), [DNAmAge](#)

 EEAA

Extrinsic epigenetic age acceleration (canonical EEAA)

Description

The EEAA of the literature (Chen et al. 2016): the Hannum clock blended with three immune cell estimators — plasmablasts, exhausted CD8+ T cells (CD28-CD45RA-) and naive CD8+ T cells — with weights fixed once in the Women's Health Initiative, then regressed on chronological age; EEAA is the residual. By construction it rises with the immune shifts of ageing, which is what makes it "extrinsic", and it is the specific measure researchers mean by "EEAA" (its intrinsic sibling is [IEAA](#)).

Usage

```
EEAA(x, age, betas)
```

Arguments

x	A methclock result, or a data frame with an id column and a numeric Hannum column.
age	Numeric vector of chronological ages, one per sample in x's order (or named by sample id).
betas	The methylation data the estimate came from: a matrix (or data frame, HDF5-backed matrix, or accepted Bioconductor container) with CpG identifiers as row names and one column per sample. Only the rows of the estimator CpGs are read, so an on-disk matrix is not loaded whole.

Details

The three cell estimators are computed here from the methylation data itself, as linear combinations of published CpG coefficients — the same route the original software takes — so no external deconvolution is needed; betas should be blood 450K/EPIC data. All numeric parameters (the estimators' CpG coefficients and the fixed centers, scales and weights of the blend) are the ones published in European patent EP 3 494 210 B1. The estimators' additive intercepts are not published; they shift every sample equally and cancel in the residual, which is why this function returns EEAA and not the intermediate biological-age value. If some of the estimator CpGs are absent from your array a warning reports it: the scores then deviate from the published estimators and comparability with other cohorts' EEAA weakens.

Value

A data frame with columns `id` and `EEAA`.

References

Chen BH, Marioni RE, Colicino E, et al. (2016). DNA methylation-based measures of biological age. *Aging* 8(9):1844–1865. European patent EP 3 494 210 B1 (2024). DNA methylation based biological age prediction.

See Also

[IEAA](#), [ageAcceleration](#), [ageAccelerationChen](#)

Examples

```
## Not run:
res <- DNAmAge(betas, clocks = "Hannum")
head(EEAA(res, age = pheno$age, betas = betas))

## End(Not run)
```

IEAA

Intrinsic epigenetic age acceleration (canonical IEAA)

Description

The IEAA of the literature (Chen et al. 2016): the Horvath clock's estimate regressed on chronological age and seven blood immune cell covariates — naive CD8+ T cells, exhausted CD8+ T cells (CD28-CD45RA-), plasmablasts, CD4+ T cells, natural killer cells, monocytes and granulocytes — whose residual reflects cell-intrinsic ageing independent of shifts in blood cell composition. This is the specific measure researchers mean by "IEAA" (its extrinsic sibling is [EEAA](#)). The same adjustment idea for any clock and any cell panel is available as the `residualCells` column of [ageAcceleration](#), under that plainer name.

Usage

```
IEAA(x, age, betas, cell_counts = NULL)
```

Arguments

<code>x</code>	A methylclock result, or a data frame with an <code>id</code> column and a numeric Horvath column.
<code>age</code>	Numeric vector of chronological ages, one per sample in <code>x</code> 's order (or named by sample id).
<code>betas</code>	The methylation data the estimate came from: a matrix (or data frame, HDF5-backed matrix, or accepted Bioconductor container) with CpG identifiers as row names and one column per sample; blood 450K/EPIC data. The rare-subset estimators read only their CpG rows.
<code>cell_counts</code>	Optional samples-by-cell-types matrix (for example from cellCounts) supplying the common cell types: CD4T, NK, Mono and Gran (or Neu, plus Eos when available). When omitted, <code>cellCounts(betas)</code> is run internally, which loads the full matrix into memory.

Details

The three rare subsets are computed from the methylation data itself, as the published linear combinations of CpGs the original software uses (parameters from European patent EP 3 494 210 B1; their additive intercepts are not published, but constants are absorbed by the regression, so the residual is unaffected). The four common cell types are taken from `cell_counts` if you provide them, or estimated with [cellCounts](#) otherwise; granulocytes are used directly when a `Gran` column exists, and otherwise formed as neutrophils plus eosinophils. Reference-based deconvolution estimates are not numerically identical to the original calculator's own estimates of these four, so the measure is canonical in structure and covariates while small numerical differences with that software remain possible. If some of the estimator CpGs are absent from your array a warning reports it.

Value

A data frame with columns `id` and `IEAA`.

References

Chen BH, Marioni RE, Colicino E, et al. (2016). DNA methylation-based measures of biological age. *Aging* 8(9):1844–1865. European patent EP 3 494 210 B1 (2024). DNA methylation based biological age prediction.

See Also

[EEAA](#), [ageAcceleration](#), [cellCounts](#)

Examples

```
## Not run:
res <- DNAMAge(betas, clocks = "Horvath")
head(IEAA(res, age = pheno$age, betas = betas))

## End(Not run)
```

imputeKNN

KNN-impute a methylation matrix

Description

Fills the missing values of a methylation matrix from each CpG's nearest CpGs – those that co-vary with it across the observed samples – rather than from its marginal mean, so co-methylation is respected. A row set within maxp is imputed exactly; a larger one is first split into blocks of similar CpGs (through a BigDataStatMeth SVD embedding) and each block is imputed exactly. This is the same imputation [compute_clocks](#) applies with impute = "knn"; use it to impute a matrix once and reuse it across analyses.

Usage

```
imputeKNN(x, k = 10, maxp = 1500, rowmax = 0.5, colmax = 0.8)
```

Arguments

x	A methylation matrix with CpG identifiers as row names and samples as columns (a data frame with a CpG-identifier column is also accepted).
k	Number of neighbouring CpGs averaged for each imputed value.
maxp	Largest block of CpGs imputed exactly; larger row sets are split into similar blocks first.
rowmax	If a CpG has a greater fraction of samples missing than this, it is filled with the per-sample mean instead of by KNN.
colmax	If a sample has a greater fraction of CpGs missing than this, a warning is raised (the sample is very sparse); imputation still proceeds.

Value

The matrix with missing values filled, same shape and dimnames.

See Also

[compute_clocks](#)

Examples

```
data(methylclock_betas)
m <- methylclock_betas[seq_len(200), ]
m[5, 2] <- NA
m[40, c(1, 3)] <- NA
filled <- imputeKNN(m)
filled[c(5, 40), seq_len(3)]
```

listCellReferences	<i>Cell-type reference panels available for deconvolution</i>
--------------------	---

Description

The reference panels shipped with the package (via the data layer), each a name accepted by [cellCounts](#)'s reference argument. `cellCounts()` also accepts the higher-resolution "FlowSorted.Blood.EPIC" and "FlowSorted.BloodExtended.EPIC" panels, which are not shipped and are used only from your own installation of those packages (see [cellCounts](#)).

Usage

```
listCellReferences()
```

Value

A character vector of the shipped reference-panel names.

Examples

```
listCellReferences()
```

mc_to_hdf5	<i>Store a methylation matrix on disk as HDF5</i>
------------	---

Description

Writes a methylation matrix to an HDF5 file and returns it as an on-disk matrix that [methylclock](#) can estimate block by block, without holding the whole array in memory. Useful for arrays too large to work with comfortably in RAM: convert once, then pass the result (and let the in-memory copy go).

Usage

```
mc_to_hdf5(  
  x,  
  file = NULL,  
  overwrite = FALSE,  
  sep = NULL,  
  header = TRUE,  
  row.names = TRUE,  
  compression = NULL  
)
```

Arguments

x	A methylation matrix or data frame (CpG identifiers as row names / a CpG-identifier column, samples as columns), or the path to a delimited text file laid out the same way. A text file is imported without being loaded into memory first.
file	Destination HDF5 file. Pass a real path to keep it; a session temporary file is used when NULL (default).
overwrite	Overwrite file if it exists. Default FALSE.
sep	For a text file, the field separator (default: auto by extension, comma for .csv else tab).
header, row.names	For a text file, whether it has a header row / a leading column of CpG identifiers. Both default to TRUE.
compression	HDF5 compression level (0-9); NULL (default) picks it by size.

Value

An on-disk (HDF5-backed) matrix, open for reading; pass it to [methylock](#) and `close()` it when done.

See Also

[methylock](#)

Examples

```
data(methylock_betas)  
f <- tempfile(fileext = ".h5")  
hm <- mc_to_hdf5(methylock_betas, f)  
res <- methylock(hm, clocks = "Horvath")  
head(as.data.frame(res))  
close(hm)  
unlink(f)
```

mcd_backends

Configure the resource backend chain

Description

Sets or returns the ordered list of sources consulted to resolve a resource. The chain is stored in `options(methylclockData.backend=)`.

Usage

```
mcd_backends(backends)
```

Arguments

`backends` Optional character vector to set, e.g. `c("local")`. With no argument, returns the current chain.

Details

Order matters: the first backend that resolves an identifier wins. Supported backends:

"local" a directory of .rda files with an accompanying MANIFEST.csv (set its location with `options(methylclockData.local_root=)` or the METHYLCLOCKDATA_MIRROR environment variable).

"eh" ExperimentHub; resolves the resource by the manifest's `eh_id` column. Requires the **ExperimentHub** package.

"zenodo" downloads and caches the resource for the record in the manifest's `zenodo_doi` column.

A backend returns nothing for a resource whose manifest row has no id for that source, so the chain falls through to the next backend.

Value

The current backend chain; returned invisibly when setting.

See Also

[mcd_resource](#)

Examples

```
mcd_backends()           # the current chain
old <- mcd_backends()
mcd_backends(c("local", "eh")) # set a chain
mcd_backends(old)         # and restore it
```

mcd_cache_clear	<i>Clear the resource cache</i>
-----------------	---------------------------------

Description

Empties the private cache of resolved resources. Resources are re-resolved through the backend chain the next time they are needed.

Usage

```
mcd_cache_clear()
```

Value

Invisibly NULL.

See Also

[mcd_resource](#), [mcd_backends](#)

Examples

```
mcd_cache_clear()
```

mcd_manifest	<i>The resource manifest</i>
--------------	------------------------------

Description

Reads and caches the manifest that describes the available resources (identifier, stored object name, class, dimensions, columns, size, checksum and source). A copy ships with the package; a configured local data root (see [mcd_backends](#)) takes precedence when present.

Usage

```
mcd_manifest()
```

Value

A data frame, one row per resource identifier.

See Also

[mcd_resource](#)

Examples

```
man <- mcd_manifest()
head(man$id)
```

mcd_resource	<i>Resolve a clock resource by identifier</i>
--------------	---

Description

The single access point for coefficient objects. Resolves `id` through the backend chain (see [mcd_backends](#)), caches the result in a private environment and returns it.

Usage

```
mcd_resource(id, verify = FALSE, cache = TRUE)
```

Arguments

<code>id</code>	Resource identifier, e.g. "coefHorvath" (a row of mcd_manifest).
<code>verify</code>	Logical; verify the object's checksum when possible. Default FALSE.
<code>cache</code>	Logical; use and populate the private cache. Default TRUE.

Value

The resource object in canonical form (see [normalize_coef](#)).

See Also

[mcd_manifest](#), [mcd_backends](#), [clock_register](#)

Examples

```
coef <- mcd_resource("coefHorvath")
head(coef)
```

mcd_resource_file	<i>Resolve a file resource to a local path</i>
-------------------	--

Description

Some resources are files read directly by the C++ layer (e.g. an HDF5 bundle of neural-network weights) rather than objects loaded into R. This resolves `id` through the backend chain and returns a local path: the file on disk for the local backend, or a downloaded-and-cached copy for a remote backend.

Usage

```
mcd_resource_file(id)
```

Arguments

`id` Resource identifier (a row of `mcd_manifest` whose class marks it as a file resource, e.g. `"coefAltumAge"`).

Value

A path to the resource file.

See Also

[mcd_resource](#)

methylock	<i>Estimate DNA methylation clocks</i>
-----------	--

Description

Unified entry point: estimates any registered clocks from a methylation matrix. [DNAmAge](#) and [DNAmGA](#) are convenience wrappers that restrict the selection to non-gestational and gestational clocks respectively.

Usage

```
methylock(
  x,
  clocks = "all",
  generation = NULL,
  platform = NULL,
  min.perc = 0.8,
  impute = c("mean", "reference", "none", "knn"),
  min.perc.sample = 0,
  storage = "auto",
  ...
)
```

Arguments

`x` A methylation matrix with CpG identifiers as row names and samples as columns (a data frame with a CpG-identifier column is also accepted), or a large matrix held on disk as an HDF5-backed matrix.

`clocks` Either `"all"` (default) or a character vector of clock names; see [clock_list](#).

`generation` Optional generation to restrict to (e.g. 1).

`platform` Optional native platform to restrict to (e.g. `"450K"`), handy when the data come from a known array and only the clocks built for it should run.

`min.perc` Minimum fraction of a clock's CpGs that must be present for it to be computed (otherwise NA). Default 0.8.

<code>impute</code>	How missing values among the present CpGs are filled: "mean" (default, each CpG's across-sample mean), "reference" (the CpG's reference level from the clock's own coefficients, mean fallback), "none" (a sample missing any used CpG is returned NA), or "knn" (each from the CpG's nearest co-varying CpGs; see imputeKNN). See compute_clocks .
<code>min.perc.sample</code>	Per-sample coverage floor: a sample with fewer than this fraction of a clock's present CpGs observed is returned NA for that clock. Default 0 (off).
<code>storage</code>	How the estimates are held: "auto" (default) backs them on disk (HDF5) with an in-memory cache when they fit, "hdf5" on disk only, "memory" in memory only (not persisted). See compute_clocks for the file and compression arguments, and persist to keep a session result.
<code>...</code>	Passed to compute_clocks (e.g. file, compression).

Value

A "methylclock" object.

See Also

[DNAmAge](#), [DNAmGA](#), [persist](#), [clock_list](#)

Examples

```
data(methylclock_betas)
res <- methylclock(methylclock_betas, clocks = c("Horvath", "Levine"))
head(as.data.frame(res))
```

<code>methylclock_betas</code>	<i>Example methylation beta matrix</i>
--------------------------------	--

Description

A small, complete methylation beta matrix (real data) provided so the vignette can demonstrate the `impute=` options on an actual matrix: compute the clocks on the full data, blank out a fraction of the values, and compare mean and K-nearest-neighbour imputation against the truth. Unlike the other bundled objects, this one is the model **input** (CpGs and samples), not the estimates. It is a subset of the example matrix shipped with earlier versions of the package, restricted to the CpGs of the clocks it fully covers (Horvath, Levine, Wu) to keep it small and let those clocks run without missing CpGs.

Usage

```
data(methylclock_betas)
```

Format

A numeric matrix with 919 rows (CpGs, as row names) and 16 columns (samples). Values are methylation betas in $[0, 1]$; there are no missing values.

Source

Subset of MethylationDataExample55 from earlier releases of this package; a public human methylation series.

Examples

```
data(methylock_betas)
methylock_betas[1:3, 1:3]
# complete-data estimates, then the effect of imputing blanked-out values
truth <- methylock(methylock_betas,
  clocks = c("Horvath", "Levine", "Wu"))
```

methylock_cells	<i>Clock estimates with blood cell proportions</i>
-----------------	--

Description

Clock estimates together with estimated blood cell proportions for an adult whole-blood cohort (795 samples, EPIC), so the intrinsic (cell-adjusted) age acceleration can be computed live. The cell proportions were estimated with [cellCounts](#) from the methylation matrix (not shipped).

Usage

```
data(methylock_cells)
```

Format

A data frame with 795 rows (samples) and columns:

id Sample identifier.

age Chronological age in years.

sex Reported sex (F/M).

Horvath, Hannum, Levine, skinHorvath, EN Estimated age in years.

Bcell, CD4T, CD8T, Eos, Mono, Neu, NK Estimated proportion of each blood cell type.

Source

Grady Trauma Project, GEO accession <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE132203>.

Examples

```
data(methylock_cells)
cells <- c("Bcell", "CD4T", "CD8T", "Eos", "Mono", "Neu", "NK")
head(ageAcceleration(methylock_cells, age = methylock_cells$age,
  cell_counts = as.matrix(methylock_cells[, cells])))
```

methylock_demo

Example clock estimates for a public blood cohort

Description

Per-sample epigenetic clock estimates for a public whole-blood methylation series with known chronological age (GEO accession GSE40279, 656 adult blood samples aged 19–101). The raw methylation matrix is not shipped; this table holds only the computed estimates, chronological age and sex, which is what the plotting functions need. It lets the vignette and the plot examples run on real data without downloading anything.

Usage

```
data(methylock_demo)
```

Format

A data frame with 656 rows (samples) and columns:

id Sample identifier.

age Chronological age in years.

sex Reported sex, a factor with levels F and M.

ethnicity Reported ethnicity, a factor with levels Caucasian (426, "Caucasian - European" in the source) and Hispanic (230, "Hispanic - Mexican"). It lets the cohort be used as two population-specific references in [plotReferenceRange](#).

Horvath, Hannum, Levine, skinHorvath, BLUP, EN, BNN, AltumAge, Lin, VidalBralo, PedBE, Wu Estimated age in years from each clock.

TL Estimated telomere length.

Source

Hannum et al. (2013), GEO accession <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE40279>.

Examples

```
data(methylock_demo)
plotDNAMAge(methylock_demo, age = methylock_demo$age)
```

`methylock_longitudinal`*A longitudinal example: children sampled repeatedly from birth to age 3*

Description

Per-sample clock estimates for a public cohort in which the same children were sampled at several ages — the shape the longitudinal tools ([clockTrajectories](#), [trajectoryRates](#), [plotTrajectories](#)) work on. 67 children from Goma, DR Congo, followed from birth to about age 3 (venous blood, EPIC; GEO accession GSE224573), 164 observations in all, technical replicates flagged rather than removed. The cohort carries substantial peripartum trauma exposure, so it illustrates the mechanics of longitudinal analysis; it does not stand for children in general.

Usage

```
data(methylock_longitudinal)
```

Format

A data frame with 164 rows (observations) and columns `id`, `subject` (the child), `age` (decimal age at the visit), `sex`, `replicate`, and one numeric column per computed clock.

Source

GEO accession <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE224573>.

Examples

```
data(methylock_longitudinal)
with(methylock_longitudinal, table(table(subject)))
```

`methylock_references`*Clock estimates on the shipped reference cohorts*

Description

Per-sample clock estimates, chronological age and sex for public blood cohorts shipped as selectable references for [plotReferenceRange](#) (the raw methylation is not shipped). Each is a single, specific cohort and its bands describe it alone: Swedish (GSE87571), 729 adults aged 14–94 from a population-based cohort recorded as disease state: normal (3 source samples ship no age and are excluded); GENOA (GSE210255), 1394 African American adults recruited through hypertensive sibships (enriched for hypertension, with related samples); Han Chinese controls (GSE116379), the 79 non-schizophrenia controls, all aged 45–51, of a schizophrenia and prenatal-famine study in Changchun; and DRC mothers (GSE224363), 97 mothers aged 14–42 sampled around delivery in Goma, DR Congo, in a cohort with substantial trauma exposure.

Usage

```
data(methylock_references)
```

Format

A data frame with 2299 rows (samples) and columns cohort (factor), id, age, sex, and one numeric column per computed clock.

Source

GEO accessions <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE87571>, <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE210255>, <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE116379> and <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE224363>.

Examples

```
data(methylock_references)
table(methylock_references$cohort)
```

methylock_smoking	<i>Smoking predictor scores with smoking status</i>
-------------------	---

Description

Estimates for a public whole-blood cohort with recorded smoking status (464 adults, 450K), used to show a non-age (trait) clock at work: the methylation smoking predictor separates never, former and current smokers. As with the other bundled data, only the estimates and phenotype are shipped, not the methylation matrix.

Usage

```
data(methylock_smoking)
```

Format

A data frame with 464 rows (samples) and columns:

id Sample identifier.

smoking Smoking status, a factor with levels never, former and current.

age Chronological age in years.

sex Reported sex (F/M).

McCartney.Smoking The methylation smoking-predictor score.

Horvath, Hannum Estimated age in years, for context.

Source

Tsaprouni et al. (2014), GEO accession <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE50660>.

Examples

```
data(methylock_smoking)
tapply(methylock_smoking$McCartney.Smoking,
       methylock_smoking$smoking, median)
```

```
methylock_validation
```

Clock estimates on two independent validation cohorts

Description

Per-sample clock estimates for two public blood cohorts that were not used to train these clocks, so they read the clocks' accuracy out of sample: an adult cohort and a paediatric one. It is what the vignette's validation section runs on. As with [methylock_demo](#), only the estimates and age/sex are shipped, not the methylation matrices.

Usage

```
data(methylock_validation)
```

Format

A data frame with 929 rows (samples) and columns:

cohort Which cohort the sample belongs to, a factor: "adult (GSE132203)" (795 adults, whole blood, EPIC) or "pediatric (GSE36054)" (134 healthy children aged 1–17, blood, 450K).

id Sample identifier.

age Chronological age in years.

sex Reported sex, a factor with levels F and M (may be NA for the paediatric cohort).

Horvath, Hannum, Levine, skinHorvath, BLUP, EN, AltumAge, Lin, PedBE, Wu Estimated age in years from each clock.

Source

Grady Trauma Project, GEO accession <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE132203>; and Alisch et al. (2012), GEO accession <https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE36054>.

Examples

```
data(methylock_validation)
adult <- subset(methylock_validation, cohort == "adult (GSE132203)")
plotDNAMage(adult, age = adult$age, clocks = c("Horvath", "BLUP"))
```

new_methylclock	<i>Construct a methylclock result</i>
-----------------	---------------------------------------

Description

Low-level constructor for the object returned by the clock estimation functions. Most users obtain one from [methylclock](#), [DNAmAge](#) or [DNAmGA](#) rather than calling this directly.

Usage

```
new_methylclock(
  values,
  clocks,
  storage = "memory",
  handle = NULL,
  persistent = FALSE,
  meta = list()
)
```

Arguments

values	A data frame of estimates (one row per sample, one column per clock) when it is held in memory; NULL when the estimates live only on disk and are reached through handle.
clocks	Character vector of the clocks that were computed.
storage	Where the authoritative estimates live: "memory" or "hdf5".
handle	For storage = "hdf5", a descriptor of the on-disk dataset (file, group, dataset); NULL otherwise.
persistent	TRUE if the on-disk file is meant to outlive the session (a user-supplied path); FALSE for a session temporary file.
meta	Named list of metadata (e.g. sample count, call, warnings).

Value

An object of class "methylclock".

See Also

[methylclock](#)

Examples

```
est <- data.frame(id = c("s1", "s2"), Horvath = c(41.2, 55.8))
res <- new_methylclock(est, clocks = "Horvath")
as.data.frame(res)
```

normalize_coef	<i>Normalize a coefficient resource to canonical form</i>
----------------	---

Description

Coerces a coefficient object to the single canonical layout used by the estimation engine (a data frame with a character CpGmarker column and numeric coefficient columns), so that downstream code does not handle the shape differences between clocks. Called by [mcd_resource](#) when a resource is loaded. Objects that are not coefficient tables (e.g. cell-type references) are returned unchanged.

Usage

```
normalize_coef(obj, id = NULL, row = NULL)
```

Arguments

obj	The raw resource object as loaded by a backend.
id	The resource identifier (unused; kept for dispatch).
row	The manifest row for id (unused; kept for dispatch).

Value

The object in canonical form.

See Also

[mcd_resource](#)

Examples

```
# a table whose CpG column is not called CpGmarker
raw <- data.frame(probeID = c("(Intercept)", "cg00075967"),
                  weight = c(0.5, -0.02))
normalize_coef(raw)

# a bare CpG vector becomes a one-column table
normalize_coef(c("cg00075967", "cg00374717"))
```

persist

Persist a result to a file

Description

Save an object to a file so it outlives the current session.

Usage

```
persist(x, file, ...)  
  
## S3 method for class 'methylock'  
persist(x, file, overwrite = FALSE, ...)
```

Arguments

x	The object to persist.
file	Destination file path.
...	Further arguments passed to methods.
overwrite	Overwrite file if it already exists. Default FALSE.

Details

For a "methylock" result the estimates already live in an HDF5 file; when that file is a session temporary, persisting copies it to file (no recomputation) and verifies the copy. A result held only in memory is written out to file. The returned object references the persistent file.

Value

The object, updated to reference the persistent file (invisibly).

Examples

```
data(methylock_betas)  
res <- methylock(methylock_betas, clocks = "Horvath")  
f <- tempfile(fileext = ".h5")  
res <- persist(res, f)  
file.exists(f)  
unlink(f)
```

plotAccelerationByGroup

Plot epigenetic age acceleration by a sample group

Description

Splits the per-clock age acceleration (the residual of predicted age on chronological age) by a categorical sample variable, such as sex or a case/control label. A group whose boxes sit consistently above zero looks epigenetically older than its chronological age across clocks, which is the association these plots are meant to surface.

Usage

```
plotAccelerationByGroup(x, age, group, clocks = NULL)
```

Arguments

x	A methylclock result, or a data frame with an id column and one numeric column per clock.
age	Numeric vector of chronological ages, one per sample in x's order (or named by sample id).
group	Categorical vector (factor or character), one value per sample in x's order, or named by sample id. Samples with a missing group are dropped.
clocks	Optional character vector of clock names. Default: the age clocks present in x.

Value

A ggplot object.

Examples

```
data(methylclock_demo)
d <- methylclock_demo
plotAccelerationByGroup(d, age = d$age, group = d$sex,
  clocks = c("Horvath", "Hannum", "Levine"))
```

plotAgeAcceleration

Plot epigenetic age acceleration

Description

Age acceleration is the residual of the linear fit of predicted age on chronological age: a positive value is a sample that looks older than its years for it. Two views: "box" shows the spread per clock; "heatmap" shows every sample and clock, so a sample accelerated across all clocks (a consistent outlier) stands out.

Usage

```
plotAgeAcceleration(
  x,
  age,
  clocks = NULL,
  type = c("box", "heatmap"),
  color_by = c("auto", "target", "none")
)
```

Arguments

x	A methylclock result, or a data frame with an id column and one numeric column per clock.
age	Numeric vector of chronological ages, one per sample in x's order (or named by sample id).
clocks	Optional character vector of clock names. Default: the age clocks present in x.
type	"box" (per-clock distribution) or "heatmap" (samples x clocks). Default "box".
color_by	For type = "box", colour the boxes by clock family (its registry target): "auto" (default) only when more than one family is shown, "target" always, "none" never.

Value

A ggplot object.

Examples

```
data(methylclock_demo)
d <- methylclock_demo
plotAgeAcceleration(d, age = d$age,
  clocks = c("Horvath", "Hannum", "Levine"))
```

plotBlandAltman	<i>Bland-Altman plot of predicted vs. chronological age</i>
-----------------	---

Description

A method-agreement view that complements the scatter in [plotDNAmAge](#): for each age clock, the difference (predicted minus real age) against the mean of the two. The solid line is the mean difference (the clock's bias) and the dashed lines are the 95% limits of agreement (bias +/- 1.96 SD). A bias away from zero is a miscalibrated clock; wide limits mean noisy predictions.

Usage

```
plotBlandAltman(x, age, clocks = NULL, ncol = 3)
```

Arguments

x	A methylclock result, or a data frame with an id column and one numeric column per clock.
age	Numeric vector of chronological ages, one per sample in x's order (or named by sample id).
clocks	Optional clock names. Default: the age clocks in x.
ncol	Number of facet columns. Default 3.

Value

A ggplot object.

Examples

```
data(methylclock_demo)
d <- methylclock_demo
plotBlandAltman(d, age = d$age, clocks = c("Horvath", "Hannum", "Levine"))
```

plotClockCorrelation *Heatmap of the correlation between clocks*

Description

Correlation of the per-sample estimates across every pair of clocks, as a heatmap. Clocks that measure similar things correlate highly; this is the clearest view of how much the clocks agree (and where they are redundant).

Usage

```
plotClockCorrelation(x, clocks = NULL, method = "pearson", cluster = TRUE)
```

Arguments

x	A methylclock result, or a data frame with an id column and one numeric column per clock.
clocks	Optional character vector of clock names to include. Default: all numeric clocks in x.
method	Correlation method passed to <code>cor</code> ("pearson", "spearman" or "kendall"). Default Pearson.
cluster	Reorder clocks by hierarchical clustering so correlated ones sit together. Default TRUE.

Value

A ggplot object.

Examples

```
data(methylclock_demo)
plotClockCorrelation(methylclock_demo,
                     clocks = c("Horvath", "Hannum", "Levine", "TL"))
```

plotClockDensities	<i>Plot the density of each clock's estimates, optionally split by a group</i>
--------------------	--

Description

A smoothed density curve of the per-sample values of every clock. With group, the curves of the groups are drawn together in each panel, so a shift, a widening or an extra mode between groups shows up as a visible change of shape.

Usage

```
plotClockDensities(
  x,
  group = NULL,
  clocks = NULL,
  what = c("estimate", "acceleration"),
  style = c("overlay", "mirror", "ridge"),
  age = NULL,
  ncol = 4,
  min_n = 5L
)
```

Arguments

x	A methylclock result, or a data frame with an id column and one numeric column per clock.
group	Optional categorical vector (factor or character), one value per sample in x's order, or named by sample id. Samples with a missing group are dropped.
clocks	Optional character vector of clock names. Default: all numeric clocks in x (for what = "acceleration", the age clocks).
what	Plot the raw "estimate" (the default) or the "acceleration", which requires age.
style	"overlay" (the default), "mirror" (two groups, one drawn downwards) or "ridge" (all clocks in one column).
age	Numeric vector of chronological ages, one per sample in x's order (or named by sample id). Only used — and required — when what = "acceleration".
ncol	Facet columns for "overlay" and "mirror". Default 4.
min_n	Minimum number of values a clock/group needs for a curve; below it, the values are drawn as ticks. Default 5.

Details

Three layouts are available through `style`. `"overlay"` (the default) superimposes the group curves in one panel per clock. `"mirror"` needs exactly two groups and draws one upwards and the other downwards from a shared baseline, so any asymmetry between the two shapes is the figure itself. `"ridge"` stacks all the clocks in a single column, one ridge per clock, giving a one-look overview across clocks; because that puts every clock on the same axis, the estimates are z-scored per clock there (accelerations are already in shared units).

With `what = "acceleration"` the curves are drawn over the age acceleration (the residual of predicted age regressed on chronological age) instead of the raw estimates. The distinction matters when the groups differ in chronological age: the raw estimates then differ with it, so their densities separate for that reason alone, while the acceleration removes that trend and leaves the scales on which the group shapes are comparable.

A density smoothed from a handful of points takes its shape mostly from the smoothing, not from the data, so any clock/group with fewer than `min_n` values is drawn as ticks along its baseline rather than as a curve.

Value

A ggplot object.

Examples

```
data(methylclock_demo)
d <- methylclock_demo
cl <- c("Horvath", "Hannum", "Levine")
plotClockDensities(d, group = d$sex, clocks = cl)
plotClockDensities(d, group = d$sex, clocks = cl, what = "acceleration",
  age = d$age, style = "mirror")
plotClockDensities(d, group = d$sex, clocks = cl, style = "ridge")
```

plotClockDistributions

Plot the distribution of each clock's estimates

Description

A violin plus box of every clock's per-sample values, to see the spread and any skew or outliers at a glance. Because clocks live on different scales (years, counts, scores), `scale = TRUE` z-scores each clock to one axis; `scale = FALSE` facets them on their own scales.

Usage

```
plotClockDistributions(
  x,
  clocks = NULL,
  scale = FALSE,
  ncol = 4,
```

```
    color_by = c("auto", "target", "none")
  )
```

Arguments

x	A methylclock result, or a data frame with an id column and one numeric column per clock.
clocks	Optional character vector of clock names. Default: all numeric clocks in x.
scale	z-score each clock onto one axis (TRUE) or facet each on its own scale (FALSE, the default).
ncol	Facet columns when scale = FALSE. Default 4.
color_by	Colour the violins by clock family (its registry target): "auto" (the default) colours only when more than one family is shown, "target" always, "none" never.

Value

A ggplot object.

Examples

```
data(methylclock_demo)
plotClockDistributions(methylclock_demo,
  clocks = c("Horvath", "Hannum", "Levine"))
```

plotDNAMAge

Plot predicted epigenetic age against chronological age

Description

For each age clock, a scatter of the predicted age against the samples' chronological age, with the identity line (predicted equal to chronological, drawn dashed), a linear fit and its confidence band, and the fit's R^2 in the title. Clocks that do not output an age (mitotic counts, trait scores) are dropped unless named in clocks.

Usage

```
plotDNAMAge(x, age, clocks = NULL, group = NULL, ncol = 3, outlier_sd = 3)
```

Arguments

x	A methylclock result, or a data frame with an id column and one numeric column per clock.
age	Numeric vector of chronological ages, one per sample in the same order as x's rows (or named by sample id).

clocks	Optional character vector of clock names to plot. Default: the age clocks present in x.
group	Optional categorical vector (factor or character), one value per sample in x's order, or named by sample id. Samples with a missing group are dropped.
ncol	Number of facet columns. Default 3.
outlier_sd	Points whose residual from the clock's linear fit exceeds this many standard deviations are highlighted, to flag predictions that fall well off the trend. NULL or Inf turns the highlight off, and it is not drawn when group is given. Default 3.

Details

With group, the points and the fitted lines are drawn per group: a group whose line runs above the other's is predicted older at the same chronological age, and lines with different slopes mean that gap changes along the age range. The outlier highlight is turned off in that case, so its colour cannot be mistaken for a group.

Value

A ggplot object.

Examples

```
data(methylclock_demo)
d <- methylclock_demo
plotDNAMAge(d, age = d$age, clocks = c("Horvath", "Hannum", "Levine"))
plotDNAMAge(d, age = d$age, group = d$sex, clocks = "Horvath")
```

plotGroupDifference *Plot the per-clock difference between two groups as a forest plot*

Description

One row per clock, in one of two layouts. The default forest draws the difference between the two groups' means with its confidence interval, all on one shared axis — a one-look answer to which clocks separate the groups and by how much. The dashed line at zero means "no difference": an interval that reaches it is compatible with none, and the points of intervals that leave zero out are drawn filled. `style = "dumbbell"` draws instead the two group means joined by a segment, so both levels are visible rather than only their difference — one group being high tells a different story than the other being low, even when the gap is the same.

Usage

```
plotGroupDifference(
  x,
  group,
  clocks = NULL,
```

```

    what = c("acceleration", "estimate"),
    style = c("forest", "dumbbell"),
    age = NULL,
    conf = 0.95
  )

```

Arguments

x	A methylclock result, or a data frame with an id column and one numeric column per clock.
group	Categorical vector (factor or character) with exactly two levels, one value per sample in x's order, or named by sample id. Samples with a missing group are dropped. The difference is second level minus first, as the axis label states.
clocks	Optional character vector of clock names. Default: the age clocks present in x.
what	Compare the "acceleration" (the default, requires age) or the raw "estimate".
style	"forest" (the default: the difference with its confidence interval) or "dumbbell" (the two group means joined by a segment).
age	Numeric vector of chronological ages, one per sample in x's order (or named by sample id). Only used — and required — when what = "acceleration".
conf	Confidence level of the intervals (Welch t). Default 0.95.

Details

By default the comparison is on the age acceleration (the residual of predicted age regressed on chronological age), which requires age. When the groups differ in chronological age, the raw estimates differ with it, so the acceleration is the scale on which a group difference means something beyond age. When no ages are known, what = "estimate" compares the raw estimates instead; any difference in chronological age between the groups is then part of the number.

Value

A ggplot object.

Examples

```

data(methylclock_demo)
d <- methylclock_demo
cl <- c("Horvath", "Hannum", "Levine")
plotGroupDifference(d, group = d$sex, age = d$age, clocks = cl)
plotGroupDifference(d, group = d$sex, what = "estimate", clocks = cl,
  style = "dumbbell")

```

plotReferenceRange *Plot samples against the percentile bands of a reference cohort*

Description

The growth-chart idea applied to epigenetic age: for each clock, the percentile bands of a reference cohort along chronological age (5th–95th light, 25th–75th darker, median line), with the user's samples drawn on top. A sample inside the central band is where most of the reference sits at that age; one outside the light band is where fewer than one in ten of the reference samples fall. Where the reference has too few samples near an age, no band is drawn — there is no rule to measure against there.

Usage

```
plotReferenceRange(
  x,
  age,
  reference = "auto",
  ref_label = NULL,
  clocks = NULL,
  window = 5,
  min_n = 20L,
  ncol = 2
)
```

Arguments

x	A methylclock result, or a data frame with an id column and one numeric column per clock.
age	Numeric vector of chronological ages, one per sample in x's order (or named by sample id). Required: age is the axis.
reference	"auto" (the default: a shipped cohort picked by the samples' age range), the name of a shipped cohort (see Details), or a data frame with an age column and one column per clock — ideally your own control samples.
ref_label	Label for the reference shown in the subtitle. Defaults to the shipped cohort's name, or "user reference" for a data frame.
clocks	Optional character vector of clock names. Default: the age clocks present in both x and the reference.
window	Half-width, in years, of the sliding age window the percentiles are computed in. Default 5.
min_n	Minimum reference samples a window needs for its band to be drawn. Default 20.
ncol	Facet columns. Default 2.

Details

The bands describe *the named reference cohort*, nothing more: a reference measures well what resembles it (population, tissue, array, processing), so the most meaningful reference is your own control samples, passed as a data frame. When none is given, the function falls back to a cohort shipped with the package, says so in a message, and the subtitle always states which reference was used and its size. The shipped options (see Details) are offered so that something can be seen with a stated caveat rather than nothing without data; they are single, specific cohorts and none of them stands for people in general.

- "GSE40279": 656 adults aged 19–101 (Hannum 2013, blood, 450K). This is the training cohort of the Hannum, BLUP and EN clocks, so its bands for those clocks read tighter than a general cohort's would. Its reported ethnicity splits it into two further references, "GSE40279-Caucasian" (426) and "GSE40279-Hispanic" (230): a reference measures best the population it comes from.
- "GSE132203": 795 adults aged 18–77 (Grady Trauma Project, blood, EPIC), a cohort selected for trauma exposure.
- "GSE36054": 134 children aged 1–17 (blood, 450K).
- "GSE87571": 729 Swedish adults aged 14–94 (blood, 450K), a population-based cohort recorded as disease state: normal.
- "GSE210255": 1394 African American adults (GENOA, blood, EPIC), recruited through hypertensive sibships — enriched for hypertension, with related samples, and spanning roughly ages 25–90 with most samples between 40 and 75.
- "GSE116379": 79 Han Chinese adults from Changchun (blood, 450K) — the non-schizophrenia controls of a schizophrenia and prenatal-famine study, all aged 45–51, so its bands exist only in that narrow window.
- "GSE224363": 97 mothers from Goma, DR Congo (venous blood, EPIC), sampled around delivery in a cohort with substantial trauma exposure, aged 14–42.

Value

A ggplot object.

Examples

```
data(methylclock_demo)
pred <- methylclock_demo[seq_len(10), ]
plotReferenceRange(pred, age = pred$age)
```

plotSampleDiscordance *Plot how much the clocks agree on each sample*

Description

One point per sample, placed by two summaries of its clock estimates once each clock is z-scored across samples (so clocks on different scales become comparable): the consensus — the median z-score, where the clocks jointly place the sample relative to the rest — and the discordance — the standard deviation of those z-scores, how much the clocks contradict each other on that sample. Most samples form a low band; a sample far above it is one the clocks genuinely disagree about, which can be a technical problem with that sample or a biology worth a second look. The label_n most discordant samples are highlighted and labelled by id.

Usage

```
plotSampleDiscordance(
  x,
  clocks = NULL,
  what = c("estimate", "acceleration"),
  age = NULL,
  label_n = 5L
)
```

Arguments

x	A methylclock result, or a data frame with an id column and one numeric column per clock.
clocks	Optional character vector of clock names. Default: the clocks in x known to the registry (for what = "acceleration", the age clocks). At least three are needed for a disagreement to mean anything.
what	Summarise the raw "estimate" (the default) or the "acceleration", which requires age.
age	Numeric vector of chronological ages, one per sample in x's order (or named by sample id). Only used — and required — when what = "acceleration".
label_n	How many of the most discordant samples to highlight and label. Default 5; 0 turns the highlight off.

Details

This complements [sampleQC](#), which scores how jointly *extreme* a sample is: a sample can be extreme with every clock in agreement, or unremarkable on average while the clocks disagree.

With what = "acceleration" (requires age) the same two summaries are computed on the age acceleration instead, so the consensus reads as how jointly accelerated the sample is rather than how old.

Value

A ggplot object.

Examples

```
data(methylclock_demo)
plotSampleDiscordance(methylclock_demo)
```

plotSamplePCA	<i>PCA of samples in clock space</i>
---------------	--------------------------------------

Description

Runs a principal component analysis on the per-sample clock estimates (a samples-by-clocks matrix) and plots the first two components, optionally coloured by a per-sample variable such as age or sex. It shows how samples arrange themselves across the whole panel of clocks: for a healthy age series the leading component typically tracks chronological age.

Usage

```
plotSamplePCA(x, color = NULL, color_label = "group", clocks = NULL)
```

Arguments

<code>x</code>	A methylclock result, or a data frame with an id column and one numeric column per clock.
<code>color</code>	Optional per-sample variable to colour points by (numeric for a continuous scale, factor/character for a discrete one), in x's order or named by sample id.
<code>color_label</code>	Legend title for color. Default "group".
<code>clocks</code>	Optional character vector of clock names to include in the PCA. Default: all numeric clock columns.

Value

A ggplot object.

Examples

```
data(methylclock_demo)
d <- methylclock_demo
plotSamplePCA(d, color = d$age, color_label = "age",
              clocks = c("Horvath", "Hannum", "Levine"))
```

plotTrajectories	<i>Plot each subject's clock trajectory over age</i>
------------------	--

Description

The longitudinal view: for each clock, one thin line per subject joining its visits (chronological age on the x axis, the clock's estimate on the y axis), with the identity line dashed — a subject moving parallel to it ages one epigenetic year per calendar year, and a steeper path runs faster. With group, lines are coloured by group and a thicker line follows each group's median trajectory across the age range it covers.

Usage

```
plotTrajectories(x, subject, age, group = NULL, clocks = NULL, ncol = 3)
```

Arguments

x	A methylclock result, or a data frame with an id column and one numeric column per clock.
subject	Subject identifier (character or factor), one per sample in x's order, or named by sample id.
age	Numeric chronological age AT EACH SAMPLE, one per sample in x's order (or named by sample id). This is the time axis.
group	Optional categorical vector (factor or character), one value per sample in x's order, or named by sample id. Samples with a missing group are dropped.
clocks	Optional character vector of clock names. Default: the age clocks present in x.
ncol	Facet columns. Default 3.

Details

The lines join the observed visits without smoothing: the wobble of a trajectory is a faithful display of how noisy repeated clock estimates are, which is part of what this plot is for. Per-subject rates with their caveats are computed by [trajectoryRates](#).

Value

A ggplot object.

Examples

```
data(methylclock_longitudinal)
d <- methylclock_longitudinal
plotTrajectories(d, subject = d$subject, age = d$age,
  clocks = c("Wu", "PedBE"))
```

qcReport

*Quality-control report for clock estimates***Description**

Bundles the checks worth running before trusting clock estimates into one call: sample descriptives, how many clocks were computable, how well each age clock tracks chronological age (overall and within groups), and — if you pass the methylation matrix — how many of each clock's CpGs your data carry. The chronological ages are optional: without them the accuracy and acceleration cannot be computed, and the report falls back to coverage, agreement and distributions.

Usage

```
qcReport(
  x,
  age = NULL,
  by = NULL,
  clocks = NULL,
  betas = NULL,
  impute = NULL,
  output = c("object", "html"),
  file = NULL
)
```

Arguments

x	A methylclock result, or a data frame with an id column and one numeric column per clock.
age	Optional numeric vector of chronological ages (one per sample in x's order, or named by sample id). Needed only for accuracy (R^2 , MAE) and age acceleration.
by	Optional categorical vector (one value per sample, or named by id): the accuracy is broken down within each group.
clocks	Optional clock names to restrict to. Default: all clocks in x.
betas	Optional methylation matrix (or accepted container): if given, the report adds each clock's CpG coverage (see clockCoverage) and how many of each clock's CpG values had to be imputed.
impute	Optional label of the imputation method used to compute x ("mean", "knn", "reference" or "none"), recorded in the report so the reader knows how missing values were filled.
output	"object" (the default) returns the report object; "html" also renders a small self-contained HTML report to file and returns its path invisibly.
file	Where to write the HTML report when output = "html". Default: a temporary file.

Value

An object of class `methylock_qc`: a list with `samples`, `groups`, `computed`, `accuracy` (or `NULL` without `age`), `coverage` and `missing_by_clock` (or `NULL` without `betas`), `impute` (the recorded method, if any), `notes` (a written reading of each section, in plain language) and `commentary` (the one-line-per-section summary). With `output = "html"` the path to the rendered report is returned invisibly.

See Also

[clockAccuracy](#), [clockCoverage](#)

Examples

```
data(methylock_demo)
qcReport(methylock_demo, age = methylock_demo$age,
         by = methylock_demo$sex)
```

sampleQC	<i>Per-sample quality control</i>
----------	-----------------------------------

Description

Flags samples worth a second look: those whose clock estimates are jointly extreme in clock space (a robust sign of a technical outlier), and — if you pass the methylation matrix — those missing a large fraction of the clock CpGs (values that had to be imputed). A sample high on either count carries estimates that are closer to an approximation than a measurement.

Usage

```
sampleQC(x, betas = NULL, clocks = NULL, outlier_sd = 3)
```

Arguments

<code>x</code>	A <code>methylock</code> result, or a data frame with an <code>id</code> column and one numeric column per clock.
<code>betas</code>	Optional methylation matrix (or accepted container): adds the per-sample percentage of clock CpGs that were missing (and so imputed).
<code>clocks</code>	Optional clock names to restrict to.
<code>outlier_sd</code>	A sample is flagged an outlier when its mean absolute z-score across clocks exceeds this. Default 3.

Value

A data frame with one row per sample: `id`, `outlier_score` (mean absolute z-score across clocks), `is_outlier`, `discordance` (standard deviation of the z-scores across clocks — how much the clocks contradict each other on that sample; drawn by [plotSampleDiscordance](#)), and — with `betas` — `pct_missing`.

Examples

```
data(methylclock_demo)
head(sampleQC(methylclock_demo))
```

theme_methylclock	<i>A clean theme for methylclock plots</i>
-------------------	--

Description

The theme the package's plotting functions use by default: a light background, restrained gridlines and clear facet labels. It returns a standard ggplot2 theme, so you can add it to your own plots or override any part of it — every plotting function returns a ggplot object, so the defaults are a starting point, not a constraint.

Usage

```
theme_methylclock(base_size = 12, base_family = "")
```

Arguments

base_size	Base font size in points. Default 12.
base_family	Base font family. Default "" (the device default).

Value

A ggplot2 theme object.

Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg)) + geom_point() + theme_methylclock()
```

trajectoryRates	<i>Each subject's rate of epigenetic aging, clock by clock</i>
-----------------	--

Description

For every subject and clock, the within-subject slope of the estimate: how many epigenetic years it advances per chronological year across that subject's visits. A slope of 1 tracks the calendar. This is a property of the estimate's trajectory, not a validated measure of biological ageing pace — of the shipped clocks, only DunedinPACE was designed and validated to estimate such a pace, from a single sample. With two visits the slope is the difference quotient and carries no per-subject uncertainty (se is NA: read it as orientative); with three or more it is the slope of a least-squares line with its standard error.

Usage

```
trajectoryRates(x, subject, age, clocks = NULL)
```

Arguments

x	A methylclock result, or a data frame with an id column and one numeric column per clock.
subject	Subject identifier (character or factor), one per sample in x's order, or named by sample id.
age	Numeric chronological age AT EACH SAMPLE, one per sample in x's order (or named by sample id). This is the time axis.
clocks	Optional character vector of clock names. Default: the age clocks present in x.

Details

Read individual slopes with care: the technical replication error of many clocks spans several years (see Higgins-Chen et al. 2022, Nature Aging, on the reliability of clock estimates), and the difference of two measurements carries the noise of both, so over a short follow-up a subject's slope can be mostly measurement. The steadier signal is in the aggregate — the distribution of slopes in a group, or their group difference ([plotGroupDifference](#) accepts these rates: one row per subject).

Value

A data frame with one row per subject and clock: subject, clock, n_visits, span_years (age range covered), slope and se (NA with two visits).

Examples

```
data(methylclock_longitudinal)
d <- methylclock_longitudinal
rates <- trajectoryRates(d, subject = d$subject, age = d$age,
                        clocks = "Wu")
head(rates)
```

Index

* datasets

- clock_catalog, [8](#)
- clock_predictors, [11](#)
- clock_statuses, [15](#)
- clock_targets, [15](#)
- methylock_betas, [32](#)
- methylock_cells, [33](#)
- methylock_demo, [34](#)
- methylock_longitudinal, [35](#)
- methylock_references, [35](#)
- methylock_smoking, [36](#)
- methylock_validation, [37](#)

ageAcceleration, [3](#), [6](#), [23](#), [24](#)

ageAccelerationChen, [4](#), [23](#)

anti_trafo, [6](#)

as.data.frame.methylock, [7](#)

cellCounts, [3](#), [4](#), [6](#), [7](#), [24](#), [26](#), [33](#)

clock_catalog, [8](#)

clock_info, [9](#), [10](#), [13](#)

clock_list, [8](#), [9](#), [10](#), [13](#), [14](#), [20](#), [21](#), [31](#), [32](#)

clock_predictors, [11](#), [12](#)

clock_register, [12](#), [14](#), [30](#)

clock_registry, [14](#), [14](#)

clock_reset, [14](#)

clock_statuses, [13](#), [15](#)

clock_targets, [12](#), [15](#)

clockAccuracy, [16](#), [55](#)

clockCoverage, [17](#), [54](#), [55](#)

clockTrajectories, [17](#), [35](#)

compute_clocks, [18](#), [21](#), [22](#), [25](#), [32](#)

cor, [43](#)

DNAmAge, [18](#), [20](#), [22](#), [31](#), [32](#), [38](#)

DNAmGA, [18](#), [21](#), [21](#), [31](#), [32](#), [38](#)

EEAA, [3](#), [5](#), [6](#), [22](#), [23](#), [24](#)

IEAA, [3](#), [6](#), [22](#), [23](#), [23](#)

imputeKNN, [19](#), [25](#), [32](#)

listCellReferences, [5](#), [8](#), [26](#)

mc_to_hdf5, [26](#)

mcd_backends, [28](#), [29](#), [30](#)

mcd_cache_clear, [29](#)

mcd_manifest, [29](#), [30](#), [31](#)

mcd_resource, [13](#), [28](#), [29](#), [30](#), [31](#), [39](#)

mcd_resource_file, [30](#)

methylock, [18](#), [20–22](#), [26](#), [27](#), [31](#), [38](#)

methylock_betas, [32](#)

methylock_cells, [33](#)

methylock_demo, [34](#), [37](#)

methylock_longitudinal, [35](#)

methylock_references, [35](#)

methylock_smoking, [36](#)

methylock_validation, [37](#)

new_methylock, [18](#), [38](#)

normalize_coef, [30](#), [39](#)

persist, [19–22](#), [32](#), [40](#)

plotAccelerationByGroup, [41](#)

plotAgeAcceleration, [4](#), [41](#)

plotBlandAltman, [42](#)

plotClockCorrelation, [43](#)

plotClockDensities, [44](#)

plotClockDistributions, [45](#)

plotDNAmAge, [42](#), [46](#)

plotGroupDifference, [47](#), [57](#)

plotReferenceRange, [34](#), [35](#), [49](#)

plotSampleDiscordance, [50](#), [55](#)

plotSamplePCA, [52](#)

plotTrajectories, [35](#), [53](#)

qcReport, [54](#)

sampleQC, [51](#), [55](#)

theme_methylock, [56](#)

trajectoryRates, [35](#), [53](#), [56](#)