

Package: exuber (via r-universe)

October 7, 2026

Type Package

Title Econometric Analysis of Explosive Time Series

Version 2.0.0

Description Testing for and dating periods of explosive dynamics (exuberance) in time series using the univariate and panel recursive unit root tests proposed by Phillips et al. (2015) <[doi:10.1111/iere.12132](https://doi.org/10.1111/iere.12132)> and Pavlidis et al. (2016) <[doi:10.1007/s11146-015-9531-2](https://doi.org/10.1007/s11146-015-9531-2)>. The recursive least-squares algorithm uses the matrix inversion lemma, so no matrix has to be inverted at each step, which makes the tests much faster to compute. The package also simulates a variety of periodically collapsing bubble processes. Details can be found in Vasilopoulos et al. (2022) <[doi:10.18637/jss.v103.i10](https://doi.org/10.18637/jss.v103.i10)>.

License GPL-3

URL <https://kvasilopoulos.github.io/exuber/>,
<https://github.com/kvasilopoulos/exuber>

BugReports <https://github.com/kvasilopoulos/exuber/issues>

Depends R (>= 4.0)

Imports cli (>= 1.1.0), doFuture (>= 1.0.0), dplyr (>= 1.1.1), foreach (>= 1.4.4), future (>= 1.33.0), generics (>= 0.0.2), ggplot2 (>= 3.4.0), glue (>= 1.3.1), lifecycle, lubridate (>= 1.7.4), parallel, progressr (>= 0.14.0), purrr (>= 0.3.2), quantreg (>= 5.9), Rcpp (>= 0.12.17), rlang (>= 0.3.4), tibble (>= 3.0.2), tidyr (>= 0.8.3)

Suggests curl, magrittr (>= 1.5), pkgload, forcats (>= 0.5.0), gridExtra (>= 2.3), knitr (>= 1.22), rmarkdown (>= 1.12), spelling (>= 2.1), stringr (>= 1.4.0), testthat (>= 2.1.1), withr (>= 2.1.2)

LinkingTo Rcpp (>= 1.0.1), RcppArmadillo (>= 0.9.400.2.0)

VignetteBuilder knitr

Config/exuber/exubercore-version v0.3.1

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE, roclets = c("`rd", `` namespace",
`` collate"))

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libicu-dev

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

Date/Publication 2026-10-06 13:48:59 UTC

RemoteUrl <https://github.com/kvasilopoulos/exuber>

RemoteRef HEAD

RemoteSha d92157b390b735a825202745480320b8e6c083b5

Contents

augment.radf_cv	4
augment.radf_obj	5
augment_join	5
augment_join.radf_obj	6
autoplot.cobubble_test_obj	7
autoplot.contagion_reg_obj	7
autoplot.cusum_test_obj	8
autoplot.dating_hls_obj	9
autoplot.dating_hlw_obj	9
autoplot.dating_knp_obj	10
autoplot.dating_pdc_obj	10
autoplot.ds_radf	11
autoplot.lbi_test_obj	12
autoplot.monitor_cusum_obj	12
autoplot.monitor_lbi_obj	13
autoplot.monitor_obj	14
autoplot.monitor_quantile_obj	14
autoplot.quantile_test_obj	15
autoplot.radf_distr	16
autoplot.radf_obj	16
autoplot.radf_recovery_obj	19
autoplot.radf_sbz_union_obj	20
autoplot.rootstamp_episodes	20
autoplot.rootstamp_est	21
autoplot.ssu_test_obj	22
cobubble_test	22
contagion_reg	24
cusum_test	26
datestamp	28

dating_hls	30
dating_hlw	31
dating_knp	33
dating_pdc	35
diagnostics	38
exuber_functions	39
index-rd	40
lbi_test	41
monitor	42
monitor_cusum	45
monitor_lbi	48
monitor_quantile	50
ps_tb	53
psy_minw	54
quantile_test	55
radf	57
radf_common	59
radf_common_cv	60
radf_kp	62
radf_mc_cv	63
radf_recovery	65
radf_recovery_cv	67
radf_sb_cv	68
radf_sbz	70
radf_sbz_cv	72
radf_sbz_union	74
radf_sign	76
radf_sign_cv	78
radf_sign_dm	79
radf_sign_dm_cv	81
radf_tt	82
radf_tt_cv	84
radf_wb_cv	85
radf_wb_ps_cv	88
rootstamp	90
scale_exuber_manual	92
series_names	94
sim_blan	95
sim_coexplosive	97
sim_common	98
sim_data	100
sim_div	101
sim_evans	102
sim_falsebubble	104
sim_fi	106
sim_innov	107
sim_mar	108
sim_msbubble	110

sim_psl	111
sim_psy1	113
sim_psy2	116
sim_tree	118
sim_vol_break	119
sim_vol_cir	120
sim_vol_garch	122
sim_vol_sv	123
ssu_test	124
summary.radf_obj	126
tidy.ds_radf	127
tidy.radf_cv	128
tidy.radf_distr	129
tidy.radf_obj	129
tidy_join	130
tidy_join.radf_obj	131
Index	132

augment.radf_cv	<i>Augment a radf_cv object</i>
-----------------	---------------------------------

Description

Returns the full critical-value sequences (badf_cv, bsadf_cv) of a radf_cv, one row per observation; see [tidy.radf_cv](#) for the scalar critical values.

Usage

```
## S3 method for class 'radf_cv'
augment(x, format = c("wide", "long"), trunc = TRUE, ...)
```

Arguments

- x An object of class radf_cv.
- format Long or wide format (default = "wide").
- trunc Whether to remove the period of the minimum window from the plot (default = TRUE).
- ... Further arguments passed to methods. Not used.

Value

A `tibble::tibble()`

augment.radf_obj	<i>Augment a radf_obj object</i>
------------------	----------------------------------

Description

Returns the full test-statistic sequences (badf, bsadf) of a radf_obj, one row per observation; see [tidy.radf_obj](#) for the scalar statistics.

Usage

```
## S3 method for class 'radf_obj'
augment(x, format = c("wide", "long"), panel = FALSE, trunc = TRUE, ...)
```

Arguments

x	An object of class radf_obj.
format	Long or wide format (default = "wide").
panel	If TRUE then returns the panel statistics
trunc	Whether to remove the period of the minimum window from the plot (default = TRUE).
...	Further arguments passed to methods. Not used.

Value

A `tibble::tibble()`

augment_join	<i>Augment into a joint model</i>
--------------	-----------------------------------

Description

Augment and then join objects.

Usage

```
augment_join(x, y, ...)
```

Arguments

x	An object of class obj.
y	An object of class cv.
...	Further arguments passed to methods.

Value

A tibble joining the augmented statistics of x with the augmented critical values of y; see the methods for the exact columns.

augment_join.radf_obj *Augment into a joint model*

Description

Augment and then join the full statistic sequences of a `radf_obj` with the critical-value sequences of a `radf_cv`, with one row for each observation. This is the table `autoplot.radf_obj` is built on.

Usage

```
## S3 method for class 'radf_obj'
augment_join(x, y = NULL, trunc = TRUE, ...)
```

Arguments

<code>x</code>	An object of class <code>radf_obj</code> .
<code>y</code>	An object of class <code>radf_cv</code> . The output will depend on the type of critical value.
<code>trunc</code>	Whether to remove the period of the minimum window from the plot (default = <code>TRUE</code>).
<code>...</code>	Further arguments passed to methods. Not used.

Value

A `tibble::tibble()`

Examples

```
rsim_data <- radf(sim_data, minw = 20)
cv <- radf_wb_cv(sim_data, minw = 20)

# Join the statistic path and the critical-value path. This is the table autoplot() is built on
aj <- augment_join(rsim_data, cv)
aj

# Reproduce (a simplified version of) autoplot()'s own bsadf-vs-crit line plot
library(ggplot2)
aj %>%
  dplyr::filter(sig == 95, stat == "bsadf") %>%
  tidyr::pivot_longer(c(tstat, crit), names_to = "series") %>%
  ggplot(aes(index, value, col = series)) +
  geom_line() +
  facet_wrap(~id, scales = "free")
```

autoplot.cobubble_test_obj

Plot method for cobubble_test() output

Description

Plots the two input series over the sample, with the test statistic and the lag in the title.

Usage

```
## S3 method for class 'cobubble_test_obj'
autoplot(object, ...)
```

Arguments

object	An object of class <code>cobubble_test</code> , the output of <code>cobubble_test</code> .
...	Further arguments passed to methods. Not used.

Value

A `ggplot`

See Also

`cobubble_test`

autoplot.contagion_reg_obj

Plot method for contagion_reg() output

Description

Plots the time-varying contagion coefficient $\delta_2(r)$ against the sample fraction r , with a zero reference line.

Usage

```
## S3 method for class 'contagion_reg_obj'
autoplot(object, ...)
```

Arguments

object	An object of class <code>contagion_reg_obj</code> , the output of <code>contagion_reg</code> .
...	Further arguments passed to methods. Not used.

Value

A [ggplot](#)

See Also

[contagion_reg](#)

autoplot.cusum_test_obj

Plot method for cusum_test() output

Description

Plots the CUSUM-type statistic path against its critical values, with one panel for each series. For the two-sided CUSUM-of-squares tests, the sup path is shown against the upper critical value and the inf path against the lower one.

Usage

```
## S3 method for class 'cusum_test_obj'  
autoplot(object, ...)
```

Arguments

object	An object of class <code>cusum_test_obj</code> , the output of cusum_test .
...	Further arguments passed to methods. Not used.

Value

A [ggplot](#)

See Also

[cusum_test](#)

`autoplot.dating_hls_obj`*Plot method for dating_hls() output*

Description

Plots each series with vertical markers at the estimated origination, collapse and recovery dates.

Usage

```
## S3 method for class 'dating_hls_obj'  
autoplot(object, ...)
```

Arguments

<code>object</code>	An object of class <code>dating_hls_obj</code> , the output of dating_hls .
<code>...</code>	Further arguments passed to methods. Not used.

Value

A [ggplot](#)

See Also

[dating_hls](#)

`autoplot.dating_hlw_obj`*Plot method for dating_hlw() output*

Description

Plots each series with vertical markers at every estimated origination, collapse and recovery date.

Usage

```
## S3 method for class 'dating_hlw_obj'  
autoplot(object, ...)
```

Arguments

<code>object</code>	An object of class <code>dating_hlw_obj</code> , the output of dating_hlw .
<code>...</code>	Further arguments passed to methods. Not used.

Value

A [ggplot](#)

See Also

[dating_hlw](#)

autoplot.dating_knp_obj

Plot method for dating_knp() output

Description

Plots each series with vertical markers at the estimated origination and collapse dates.

Usage

```
## S3 method for class 'dating_knp_obj'  
autoplot(object, ...)
```

Arguments

object	An object of class <code>dating_knp_obj</code> , the output of dating_knp .
...	Further arguments passed to methods. Not used.

Value

A [ggplot](#)

See Also

[dating_knp](#)

autoplot.dating_pdc_obj

Plot method for dating_pdc() output

Description

Plots each series with vertical markers at the estimated origination, collapse and, where estimated, recovery dates.

Usage

```
## S3 method for class 'dating_pdc_obj'  
autoplot(object, ...)
```

Arguments

object An object of class `dating_pdc_obj`, the output of `dating_pdc`.
 ... Further arguments passed to methods. Not used.

Value

A `ggplot`

See Also

`dating_pdc`

<code>autoplot.ds_radf</code>	<i>Plotting a ds_radf object</i>
-------------------------------	----------------------------------

Description

Takes a `ds_radf` object and returns a `ggplot2` object with a `geom_segment` layer.

Usage

```
## S3 method for class 'ds_radf'
autoplot(object, trunc = TRUE, ...)
```

Arguments

object An object of class `ds_radf`. The output of `datestamp`
 trunc Whether to remove the period of the minimum window from the plot (default = TRUE).
 ... Further arguments passed to methods. Not used.

Value

`ggplot`

Examples

```
sim_data_wdate %>%
  radf() %>%
  datestamp() %>%
  autoplot()

# Change the colour manually
sim_data_wdate %>%
  radf() %>%
  datestamp() %>%
```

```
autoplot() +
  ggplot2::scale_colour_manual(values = rep("black", 4))
```

`autoplot.lbi_test_obj` *Plot method for lbi_test() output*

Description

Bar chart of the LBI statistic for each series against its critical value. Series that exceed it are flagged as detected.

Usage

```
## S3 method for class 'lbi_test_obj'
autoplot(object, ...)
```

Arguments

<code>object</code>	An object of class <code>lbi_test_obj</code> , the output of <code>lbi_test</code> .
<code>...</code>	Further arguments passed to methods. Not used.

Value

A `ggplot`

See Also

`lbi_test`

`autoplot.monitor_cusum_obj`
Plot method for monitor_cusum() output

Description

Plots the CUSUM detector path against its boundary, with one panel for each series and vertical markers at the end of the training sample and at the alarm date.

Usage

```
## S3 method for class 'monitor_cusum_obj'
autoplot(object, ...)
```

Arguments

object An object of class `monitor_cusum_obj`, the output of `monitor_cusum`.
... Further arguments passed to methods. Not used.

Value

A `ggplot`

See Also

`monitor_cusum`

`autoplot.monitor_lbi_obj`

Plot method for `monitor_lbi()` output

Description

Plots the LBI CUSUM detector path against its boundary, with one panel for each series and a vertical marker at the alarm date.

Usage

```
## S3 method for class 'monitor_lbi_obj'  
autoplot(object, ...)
```

Arguments

object An object of class `monitor_lbi_obj`, the output of `monitor_lbi`.
... Further arguments passed to methods. Not used.

Value

A `ggplot`

See Also

`monitor_lbi`

`autoplot.monitor_obj` *Plot method for monitor() output*

Description

Plots the monitoring statistic against its boundary, with one panel for each series and vertical markers at the end of the training sample and at the alarm date.

Usage

```
## S3 method for class 'monitor_obj'
autoplot(object, ...)
```

Arguments

<code>object</code>	An object of class <code>monitor_obj</code> , the output of <code>monitor</code> .
<code>...</code>	Further arguments passed to methods. Not used.

Value

A `ggplot`

See Also

`monitor`

`autoplot.monitor_quantile_obj`
Plot method for monitor_quantile() output

Description

Plots the quantile monitoring statistic against its boundary, with one panel for each series and a vertical marker at the alarm date.

Usage

```
## S3 method for class 'monitor_quantile_obj'
autoplot(object, ...)
```

Arguments

<code>object</code>	An object of class <code>monitor_quantile_obj</code> , the output of <code>monitor_quantile</code> .
<code>...</code>	Further arguments passed to methods. Not used.

Value

A [ggplot](#)

See Also

[monitor_quantile](#)

autoplot.quantile_test_obj

Plot method for quantile_test() output

Description

Bar chart of the quantile-DF statistic for each series against its critical value. Series that exceed it are flagged as detected.

Usage

```
## S3 method for class 'quantile_test_obj'  
autoplot(object, ...)
```

Arguments

object	An object of class quantile_test_obj, the output of quantile_test .
...	Further arguments passed to methods. Not used.

Value

A [ggplot](#)

See Also

[quantile_test](#)

`autoplot.radf_distr` *Plotting a radf_distr object*

Description

Takes a `radf_distr` object and returns a `ggplot2` object.

Usage

```
## S3 method for class 'radf_distr'
autoplot(object, ...)
```

Arguments

<code>object</code>	An object of class <code>radf_distr</code> .
<code>...</code>	Further arguments passed to methods, used only in <code>wb_distr</code> facet options.

Value

A `ggplot2::ggplot()`

Examples

```
# Monte Carlo distribution
mdist <- radf_mc_distr(n = 100, nrep = 1000)
autoplot(mdist)

# Wild bootstrap distribution (one facet per series)
wdist <- radf_wb_distr(sim_data)
autoplot(wdist)

# Panel sieve bootstrap distribution
sdist <- radf_sb_distr(sim_data, nboot = 500)
autoplot(sdist)
```

`autoplot.radf_obj` *Plotting radf models*

Description

`autoplot.radf_obj` takes a `radf_obj` and a `radf_cv` and returns a faceted `ggplot` object. `shade` is used as an input to `shade_opt`. It modifies the `geom_rect` layer that marks the exuberance periods.

Usage

```
## S3 method for class 'radf_obj'
autoplot(
  object,
  cv = NULL,
  sig_lvl = 95,
  option = c("gsadf", "sadf"),
  min_duration = 0L,
  select_series = NULL,
  nonrejected = FALSE,
  shade_opt = shade(),
  trunc = TRUE,
  include_negative = "DEPRECATED",
  ...
)

## S3 method for class 'radf_obj'
autoplot2(
  object,
  cv = NULL,
  sig_lvl = 95,
  option = c("gsadf", "sadf"),
  min_duration = 0L,
  select_series = NULL,
  nonrejected = FALSE,
  trunc = TRUE,
  shade_opt = shade(),
  ...
)

shade(
  fill = "grey55",
  fill_negative = fill,
  fill_ongoing = NULL,
  opacity = 0.3,
  ...
)
```

Arguments

object	An object of class obj.
cv	An object of class cv.
sig_lvl	Significance level. It could be one of 90, 95 or 99.
option	Whether to apply the "gsadf" or the "sadf" methodology (default = "gsadf"). Unlike datestamp , this function does not support "svadf", because that option has no critical-value band to shade.
min_duration	The minimum duration of an explosive period for it to be reported (default = 0).

<code>select_series</code>	A vector of column names or numbers that specifies the series to plot. The order of the series does not change the order used in the plot.
<code>nonrejected</code>	If TRUE, plot all series, whether or not they reject the null at the 5 percent significance level.
<code>shade_opt</code>	Shading options, typically set using <code>shade</code> function.
<code>trunc</code>	Whether to remove the period of the minimum window from the plot (default = TRUE).
<code>include_negative</code>	Argument name is deprecated and substituted with <code>nonrejected</code> .
<code>...</code>	Further arguments passed to <code>ggplot2::facet_wrap</code> and <code>ggplot2::geom_rect</code> for shade.
<code>fill</code>	The shade color for the exuberance periods with a positive signal.
<code>fill_negative</code>	The shade color for the exuberance periods with a negative signal, that is, from series that do not reject the null hypothesis.
<code>fill_ongoing</code>	The shade color for the exuberance periods that are ongoing.
<code>opacity</code>	The opacity of the shade color (alpha).

Value

A [ggplot](#)

Examples

```
rsim_data <- radf(sim_data_wdate)

autoplot(rsim_data)

# Modify facet_wrap options through ellipsis
autoplot(rsim_data, scales = "free_y", dir = "v")

# Modify the shading options
autoplot(rsim_data, shade_opt = shade(fill = "pink", opacity = 0.5))

# Allow for nonrejected series to be plotted
autoplot(rsim_data, nonrejected = TRUE)

# Remove the shading completely (2 ways)
autoplot(rsim_data, shade_opt = NULL)
autoplot(rsim_data, shade_opt = shade(opacity = 0))

# Plot only the series with the shading options
autoplot2(rsim_data)
autoplot2(rsim_data, trunc = FALSE) # keep the minw period

# We will need ggplot2 from here on out
library(ggplot2)

# Change (overwrite) color, size or linetype
```

```

autoplot(rsim_data) +
  scale_color_manual(values = c("black", "black")) +
  scale_linewidth_manual(values = c(0.9, 1)) +
  scale_linetype_manual(values = c("solid", "solid"))

# Change names through labeller (first way)
custom_labels <- c("psy1" = "new_name_for_psy1", "psy2" = "new_name_for_psy2")
autoplot(rsim_data, labeller = labeller(.default = label_value, id = as_labeller(custom_labels)))

# Change names through labeller (second way)
custom_labels2 <- series_names(rsim_data)
names(custom_labels2) <- custom_labels2
custom_labels2[c(3, 5)] <- c("Evans", "Blanchard")
autoplot(rsim_data, labeller = labeller(id = custom_labels2))

# Or change names before plotting
series_names(rsim_data) <- LETTERS[1:5]
autoplot(rsim_data)

# Change Theme options
autoplot(rsim_data) +
  theme(legend.position = "right")

```

```
autoplot.radf_recovery_obj
```

Plot method for radf_recovery() output

Description

Plots each series with vertical markers at the estimated collapse and recovery dates.

Usage

```
## S3 method for class 'radf_recovery_obj'
autoplot(object, ...)
```

Arguments

object	An object of class <code>radf_recovery_obj</code> , the output of radf_recovery .
...	Further arguments passed to methods. Not used.

Value

A [ggplot](#)

See Also

[radf_recovery](#)

```
autoplot.radf_sbz_union_obj
```

Plot method for radf_sbz_union() output

Description

Plots the supDF, supBZ and union statistics of each series against their bootstrap critical values at the chosen significance level.

Usage

```
## S3 method for class 'radf_sbz_union_obj'
autoplot(object, sig_lvl = 95, ...)
```

Arguments

<code>object</code>	A <code>radf_sbz_union</code> object.
<code>sig_lvl</code>	Significance level at which to plot the critical value, one of 90, 95 (default) or 99.
<code>...</code>	Further arguments passed to methods. Not used.

Value

A [ggplot](#)

See Also

[radf_sbz_union](#)

```
autoplot.rootstamp_episodes
```

Plot method for rootstamp() output on datestamped episodes

Description

Plots the estimated root and its confidence interval for every episode, one panel per series.

Usage

```
## S3 method for class 'rootstamp_episodes'
autoplot(object, ...)
```

Arguments

object	An object of class rootstamp_episodes, the output of the radf_obj rootstamp method.
...	Further arguments passed to methods. Not used.

Value

A [ggplot](#)

See Also

[rootstamp](#)

autoplot.rootstamp_est

Plot method for rootstamp() output on a single sub-sample

Description

Plots the sub-sample against the fitted explosive path implied by the estimated root, $y_1 \rho^{t-1}$.

Usage

```
## S3 method for class 'rootstamp_est'
autoplot(object, ...)
```

Arguments

object	An object of class rootstamp_est, the output of the default rootstamp method.
...	Further arguments passed to methods. Not used.

Value

A [ggplot](#)

See Also

[rootstamp](#)

`autoplot.ssu_test_obj` *Plot method for ssu_test() output*

Description

Plots the recursive SSU statistic path against its critical value, with one panel for each series.

Usage

```
## S3 method for class 'ssu_test_obj'
autoplot(object, ...)
```

Arguments

`object` An object of class `ssu_test_obj`, the output of `ssu_test`.
`...` Further arguments passed to methods. Not used.

Value

A `ggplot`

See Also

[ssu_test](#)

`cobubble_test` *Test for Co-explosive Behaviour Between Two Series*

Description

`cobubble_test` tests whether two series that each contain an explosive episode are *co-explosive*. That is, it tests whether the linear combination $y_t - \alpha - \beta * x_{\{t-lag\}}$ is stationary, so that the explosive dynamics in y and x are the same underlying phenomenon, possibly migrating from one series to the other with a lead or lag, and not independent explosive episodes.

Usage

```
cobubble_test(
  y,
  x,
  lag = NULL,
  lag_grid = -6:6,
  nboot = 499L,
  sig_lvl = 95,
  seed = NULL
)
```

Arguments

<code>y, x</code>	Numeric vectors of equal length, or objects that <code>as.numeric()</code> can coerce to one. <code>x</code> is the candidate regressor with the explosive episode, and <code>y</code> is tested for co-explosivity with $x_{\{t-lag\}}$.
<code>lag</code>	The lead or lag i in $x_{\{t-lag\}}$. If NULL (default), it is estimated from <code>lag_grid</code> by minimizing the residual variance ($i_{\hat{}}$ in Section VI).
<code>lag_grid</code>	Candidate lag values searched when <code>lag = NULL</code> . The default <code>-6:6</code> follows the simulation design of the paper.
<code>nboot</code>	Number of wild bootstrap replications.
<code>sig_lvl</code>	Significance level, on the same 0 to 100 scale as the <code>sig_lvl</code> of <code>datestamp</code> (default 95, which gives a 5% upper-tail rejection region).
<code>seed</code>	Optional seed for the bootstrap draws.

Details

`radf` is a right-tailed ADF-family test for the presence of explosiveness. This function is a stationarity (KPSS-type) test instead. The null hypothesis is co-explosivity, that is, that the residuals of `y` regressed on a constant and $x_{\{t-lag\}}$ are $I(0)$. The null limiting distribution of the statistic depends on the pattern of heteroskedasticity in the errors (Evripidou, Harvey, Leybourne & Sollis 2022, Theorem 1). The critical values therefore come from a wild bootstrap that reproduces this pattern of heteroskedasticity in the bootstrap samples (Theorem 2).

Value

An object of class `cobubble_test_obj`: a list with the observed statistic `S`, the (given or estimated) lag, the bootstrap critical value `cv` at `sig_lvl`, the bootstrap p-value `p_value` and `reject`, which is TRUE if `S` exceeds `cv`, that is, if co-explosivity is rejected.

Status

[Experimental]

Note

The critical value comes from a wild bootstrap of the residuals (Theorem 2), which is computed internally in each call. There is no separate, reusable `cv` function for this test.

The function returns its own class and not `radf_obj`, so it does not work with `summary()`, `\link{datestamp}` and `tidy`. It has its own `print()` and `autoplot()` methods instead. `print()` shows the statistic, the critical value and the p-value. See `vignette("naming-and-analysis", package = "exuber")` for which functions fit the shared pipeline and which do not.

References

Evripidou, A. C., Harvey, D. I., Leybourne, S. J., & Sollis, R. (2022). Testing for co-explosive behaviour in financial time series. *Oxford Bulletin of Economics and Statistics*, 84(3), 624-650.

See Also

Other multivariate: `contagion_reg()`, `radf_common()`

Examples

```
# A co-explosive pair (the process of Evripidou et al.), which is not rejected
xy <- sim_coexplosive(n = 100, seed = 123)
res <- cobubble_test(xy$y, xy$x, nboot = 199L, seed = 1)
print(res)

# Force a specific lead or lag instead of estimating it
res_lag0 <- cobubble_test(xy$y, xy$x, lag = 0L, nboot = 199L, seed = 1)
print(res_lag0)

# Two independent bubbles: co-explosivity correctly rejected
cobubble_test(sim_data$psy1, sim_data$psy2, nboot = 199L, seed = 1)

# Plot the two series that are tested for co-explosivity
autoplot(res)
```

contagion_reg

Bubble Contagion Regression (Greenaway-McGrevy & Phillips 2016)

Description

`contagion_reg` estimates the time-varying contagion coefficient of Greenaway-McGrevy & Phillips (2016). It computes a fixed-window rolling AR(1) coefficient sequence for a "core" series and for a "satellite" series y and relates them with a functional (Nadaraya-Watson kernel) regression at a chosen delay d . The coefficient shows how strongly, and how it varies over time, the local persistence of the core series transmits to y , d periods later.

Usage

```
contagion_reg(
  y,
  core,
  S = NULL,
  d = 0L,
  h = NULL,
  r_grid = seq(0, 1, length.out = 100)
)
```

Arguments

<code>y</code>	Satellite (dependent) series, a numeric vector.
<code>core</code>	Core (reference) series, a numeric vector of the same length as y .

<code>S</code>	Fixed rolling-window width for the AR(1) coefficient sequence (default <code>floor(0.33 * length(y))</code> , the choice of the paper).
<code>d</code>	Non-negative integer delay (default 0).
<code>h</code>	Bandwidth for the Nadaraya-Watson regression. The default NULL selects it by leave-one-out cross-validation (eq. 7).
<code>r_grid</code>	Evaluation points for the time-varying coefficient, as fractions of the sample (default <code>seq(0, 1, length.out = 100)</code>).

Details

This is a minimal subset of the procedure in the paper. It contains the fixed-window AR(1) coefficient sequence (their eq. 1), the Nadaraya-Watson regression at a single supplied `d` (eq. 6) and leave-one-out cross-validated bandwidth selection (eq. 7). Their eq. 8, the automatic search over `d`, is not implemented. If you need a search, call `contagion_reg` once for each candidate `d` and compare the fit.

The paper performs no formal inference on the contagion coefficient itself, with no confidence bands and no hypothesis test. As in the paper, this function is a tool for point estimation and visualization and not a test.

Value

An object of class `contagion_reg_obj`: a list with the fixed-window AR(1) coefficient sequences (`beta_core` and `beta_j`), the selected or supplied bandwidth (`h`) and the estimated time-varying contagion coefficient (`delta2`, aligned with `r_grid`).

Status

[Experimental]

Note

The function is not a hypothesis test. It performs no formal inference on the contagion coefficient, with no confidence bands and no significance test, so there is no critical value for it.

The function returns its own class and not `radf_obj`, so it does not work with `summary()`, `\link{datestamp}` and `tidy`. It has its own `print()` and `autoplot()` methods instead. `print()` shows the coefficient path, because the function performs no formal inference. See `vignette("naming-and-analysis", package = "exuber")` for which functions fit the shared pipeline and which do not.

References

Greenaway-McGrevy, R., & Phillips, P. C. B. (2016). Hot property in New Zealand: Empirical evidence of housing bubbles in the metropolitan centres. *New Zealand Economic Papers*, 50(1), 88-113.

See Also

[cobubble_test](#) for a different, symmetric bivariate bubble relationship that uses a hypothesis test.
Other multivariate: [cobubble_test\(\)](#), [radf_common\(\)](#)

Examples

```
# A co-explosive pair: the AR coefficient of y follows that of x almost one for one
xy <- sim_coexplosive(n = 100, seed = 123)
res <- contagion_reg(xy$y, xy$x, d = 0L)
print(res)

# Plot the estimated time-varying contagion coefficient
autoplot(res)

# Compare a one-period lead (d = 1) with the contemporaneous case
res_d1 <- contagion_reg(xy$y, xy$x, d = 1L)
autoplot(res) +
  ggplot2::geom_line(data = data.frame(r = res_d1$r_grid, delta2 = res_d1$delta2),
    ggplot2::aes(r, delta2), color = "red", inherit.aes = FALSE)
```

cusum_test	<i>CUSUM and CUSUM-of-Squares Bubble Tests (Kurozumi & Nishi 2025)</i>
------------	--

Description

`cusum_test` implements the retrospective CUSUM ("cs"), generalized CUSUM ("gcs"), CUSUM-of-squares ("cssq") and generalized CUSUM-of-squares ("gcssq") tests of Kurozumi & Nishi (2025). These are the parameter-constancy statistics of Brown et al. (1975), applied to the first differences. The generalized versions take the supremum over every window start as well as every end point.

Usage

```
cusum_test(data, sig_lvl = 95, type = c("cs", "gcs", "cssq", "gcssq"))
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in <code>badf</code> and <code>bsadf</code> and excluded from the <code>adf</code> , <code>sadf</code> and <code>gsadf</code> of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (<code>bsadf_panel</code> and <code>gsadf_panel</code>) are not available, and the function returns NA for them with a warning.
sig_lvl	Significance level on the 0 to 100 scale used throughout the package, one of 90, 95 or 99.
type	One of "cs", "gcs", "cssq" or "gcssq".

Details

CS and GCS reject when the cumulated increments become too large (right tail). CSSQ and GCSSQ are two-sided. They reject when the cumulated squared increments drift too far above *or* below their full-sample average, with half the level in each tail. The paper finds that the CUSUM-type tests lose almost all their power once the explosive coefficient is stochastic, while the CUSUM-SQ type keeps it. See [ssu_test](#) for the more powerful statistics of the paper.

Value

An object of class `cusum_test_obj`: a list with the statistic path `stat` (one value for each end point, and for the generalized versions the sup over window starts), `stat_inf` (the inf path, for CSSQ and GCSSQ only), the statistic sup (and inf), the critical values `crit` and `detected`.

Status

[Experimental]

Note

All critical values are published asymptotic values (Table I of Kurozumi & Nishi 2025). No minimum window is needed, because the paper finds the statistics insensitive to it.

References

Kurozumi, E., & Nishi, M. (2025). Bubble testing with stochastically varying explosive coefficient. *Journal of Time Series Analysis*, 46(5), 945-965.

Brown, R. L., Durbin, J., & Evans, J. M. (1975). Techniques for testing the constancy of regression relationships over time. *Journal of the Royal Statistical Society B*, 37(2), 149-192.

See Also

[ssu_test](#), and [monitor_cusum](#) for real-time CUSUM monitoring.

Other volatility-robust tests: [radf_kp\(\)](#), [radf_sbz\(\)](#), [radf_sbz_union\(\)](#), [radf_sign\(\)](#), [radf_sign_dm\(\)](#), [radf_tt\(\)](#), [ssu_test\(\)](#)

Examples

```
y <- sim_psy1(n = 150, te = 75, tf = 150, c = 3, alpha = 1, seed = 2001,
  coef_noise = rnorm(149), coef_a = 4)
cusum_test(y, type = "cssq")
autoplot(cusum_test(y, type = "gcssq"))
```

datestamp

*Date-stamping periods of mildly explosive behavior***Description**

Computes the origination, termination and duration of the episodes in which a time series shows explosive dynamics.

Usage

```
datestamp(object, cv = NULL, min_duration = 0L, ...)
```

```
## S3 method for class 'radf_obj'
datestamp(
  object,
  cv = NULL,
  min_duration = 0L,
  sig_lvl = 95,
  option = c("gsadf", "sadf", "svadf"),
  nonrejected = FALSE,
  ...
)
```

Arguments

object	An object of class obj.
cv	An object of class cv.
min_duration	The minimum duration of an explosive period for it to be reported (default = 0).
...	Further arguments passed to methods.
sig_lvl	Significance level, one of 90, 95 or 99. It is ignored when option = "svadf".
option	One of "gsadf" or "sadf", which date episodes (PWY/PSY) against the critical values in cv, or "svadf", the SV-ADF asymmetric-threshold dating of Sarkar & Wells (2026). The "svadf" option compares the badf sequence of radf() with two closed-form thresholds that depend only on the sample size, $\log(t)/10$ for origination and $\log(t)/2$ for collapse, so it needs no cv. See Caveats.
nonrejected	logical. Whether to apply the datestamping technique to the series that do not reject the null hypothesis. It is ignored when option = "svadf".

Details

datestamp also stores a vector that takes the value 1 when there is a period of explosive behavior and 0 otherwise. You can use it as a dummy variable for the occurrence of exuberance.

Value

A table with the following columns:

- Start:
- Peak:
- End:
- Duration:
- Signal:
- Ongoing:

A list with the estimated origination and termination dates of the episodes of explosive behavior and their duration.

Caveats

option = "svadf": **[Experimental]** Sarkar & Wells (2026) is a preprint that has not been peer reviewed, which is a weaker standard of evidence than for every other source this package implements. The function emits the same note as a message when you call it with this option. It detects at most one origination and collapse pair per series, as the procedure of the paper does, and it does not find every recurring episode as "gsadf" and "sadf" do.

References

- Phillips, P. C. B., Shi, S., & Yu, J. (2015). Testing for Multiple Bubbles: Historical Episodes of Exuberance and Collapse in the S&P 500. *International Economic Review*, 56(4), 1043-1078. [doi:10.1111/iere.12132](https://doi.org/10.1111/iere.12132)
- Sarkar, A., & Wells, M. T. (2026). Is there an AI bubble? Robust date-stamping for periods of exuberance. arXiv:2604.12062.

Examples

```
rsim_data <- radf(sim_data)

# SV-ADF asymmetric-threshold dating (no critical values needed)
datestamp(rsim_data, option = "svadf")

# The default `cv` is fetched from the shared critical-value store
# (network on first use); pass `cv = radf_mc_cv(nrow(sim_data))` to stay offline
ds_data <- datestamp(rsim_data)
ds_data

# Choose minimum window
datestamp(rsim_data, min_duration = psy_ds(nrow(sim_data)))

autoplot(ds_data)
```

dating_hls

*SSR/BIC Bubble Dating (Harvey, Leybourne & Sollis 2017)***Description**

dating_hls dates a single bubble episode by fitting four candidate regime-dummy regressions of Δy_t on y_{t-1} (unit-root-to-end, unit-root-bubble-unit-root, unit-root-bubble-collapse and unit-root-bubble-collapse-unit-root). It fits each by minimizing the residual sum of squares over candidate break fractions and selects among them by BIC.

Usage

```
dating_hls(data, trim = 0.05)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
trim	Minimum fraction of the (differenced) sample required in every regime (default 0.05, following the choice of Harvey, Leybourne & Sollis in their empirical application. Their simulations use 0.1).

Details

[datestamp](#) uses threshold crossing on the recursive BSADF statistic, and [dating_pdc](#) uses a fixed structure of three or four regimes with breaks that are estimated sequentially and not jointly. This function searches for the breakpoints jointly within each of four candidate regime structures and lets the BIC pick the structure itself. It can therefore distinguish a bubble that collapses to a new stationary regime (Model 3) from a bubble that fully reverts to a unit root (Model 4) and from a bubble that is still ongoing at the end of the sample (Model 1), which the fixed regime count of dating_pdc cannot. The cost is a joint grid search, in place of the sequential scan of dating_pdc that finds one break at a time.

Value

An object of class dating_hls_obj: a list with the selected model (model, one of 1:4), its breakpoint date or dates (origination, collapse and recovery, with NA for the breakpoints that the selected model does not have), and the BIC value of every candidate model (bic, which shows how close the selection was).

Status**[Experimental]****Note**

This is an SSR/BIC model-selection dating procedure and not a hypothesis test, so it needs no critical values.

The function returns its own class and not `radf_obj`, so it does not work with `summary()`, `\link{datestamp}` and `tidy`. It has its own `print()` and `autoplot()` methods instead. `print()` shows the dating table (model, origination, collapse, recovery). See `vignette("naming-and-analysis", package = "exuber")` for which functions fit the shared pipeline and which do not.

References

Harvey, D. I., Leybourne, S. J., & Sollis, R. (2017). Improving the accuracy of asset price bubble start and end date estimators. *Journal of Empirical Finance*, 40, 121-138.

See Also

[dating_pdc](#) for the cheaper sequential-splitting alternative that this function complements, and [datestamp](#) for the original threshold-crossing rule of PSY.

Other dating: [dating_hlw\(\)](#), [dating_knp\(\)](#), [dating_pdc\(\)](#), [radf_recovery\(\)](#), [rootstamp\(\)](#)

Examples

```
res <- dating_hls(sim_data$psy1, trim = 0.05)
print(res)

# Plot the series with the selected model's breakpoint(s) overlaid
autoplot(res)

# A whole panel at once, with one subplot for each series
autoplot(dating_hls(sim_data, trim = 0.05))
```

dating_hlw

Multi-Bubble SSR/BIC Dating (Harvey, Leybourne & Whitehouse 2020)

Description

`dating_hlw` extends [dating_hls](#) to series with more than one explosive episode. It first runs the existing detection and dating of PSY ([radf](#) and [datestamp](#)) to locate a preliminary start and end for each episode. It then splits the sample into disjoint date windows around them and dates each window again with the SSR/BIC fitting of [dating_hls](#), restricted to Models 2 and 4 for every window except the last.

Usage

```

dating_hlw(
  data,
  cv = NULL,
  minw = NULL,
  trim = 0.1,
  min_duration = NULL,
  nboot = 199L,
  seed = NULL,
  join = 3L
)

```

Arguments

<code>data</code>	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in <code>badf</code> and <code>bsadf</code> and excluded from the <code>adf</code> , <code>sadf</code> and <code>gsadf</code> of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (<code>bsadf_panel</code> and <code>gsadf_panel</code>) are not available, and the function returns NA for them with a warning.
<code>cv</code>	Critical values for the step-1 PSY detection and dating, as accepted by datestamp . The default NULL computes radf_wb_cv internally.
<code>minw</code>	Minimum window size for the step-1 radf call. The default is psy_minw .
<code>trim</code>	Minimum fraction of the (differenced) sample required in every regime (default 0.05, following the choice of Harvey, Leybourne & Sollis in their empirical application. Their simulations use 0.1).
<code>min_duration</code>	Minimum duration (in observations) for a step-1 PSY episode to be counted. The default is psy_ds (the $\ln(T)$ rule of HLW).
<code>nboot, seed</code>	Passed to radf_wb_cv when <code>cv</code> is not supplied.
<code>join</code>	The run-joining rule of HLW for fragmented step-1 detections. Two explosive runs that are separated by at most <code>join</code> non-rejections and are each at least $\ln(T)$ long are treated as one episode. The default is 3, the value in the paper, and 0 disables joining.

Details

When exactly one episode is detected, the function reduces to [dating_hls](#) applied to the whole series, which is a stated property of the paper. The single window then runs over $[1, n]$ and fits all four models.

Value

An object of class `dating_hlw_obj`: a list with one element for each series. Each element is a data frame with one row for each detected episode (model, origination, collapse, recovery). A series with no step-1 detected episode gets a data frame with zero rows.

Status**[Experimental]****Note**

The step-2 SSR/BIC dating within each window needs no critical values, as in [dating_hls](#). The step-1 PSY detection and dating pass does use a wild bootstrap critical value (cv, nboot and seed below), but only to locate the preliminary episode windows and not for the dating step.

The function returns its own class and not `radf_obj`, so it does not work with `summary()`, `\link{datestamp}` and `tidy`. It has its own `print()` and `autoplot()` methods instead. `print()` shows the dating table (model, origination, collapse, recovery). See `vignette("naming-and-analysis", package = "exuber")` for which functions fit the shared pipeline and which do not.

References

Harvey, D. I., Leybourne, S. J., & Whitehouse, E. J. (2020). Date-stamping multiple bubble regimes. *Journal of Empirical Finance*, 58, 226-246.

See Also

[dating_hls](#) for the single-bubble fitting that this function wraps, and [datestamp](#) for the multi-bubble threshold-crossing dating of PSY.

Other dating: [dating_hls\(\)](#), [dating_knp\(\)](#), [dating_pdc\(\)](#), [radf_recovery\(\)](#), [rootstamp\(\)](#)

Examples

```
res <- dating_hlw(sim_data$psy1, trim = 0.1, nboot = 199L, seed = 1)
print(res)

# Plot the breakpoints of every detected episode over the series
autoplot(res)

# A two-bubble series, for which dating_hls() alone would fit only one bubble
res2 <- dating_hlw(sim_psy2(n = 200, seed = 123), trim = 0.1, nboot = 199L, seed = 1)
autoplot(res2)
```

dating_knp

*Bias-Corrected Bubble Dating (Kejriwal, Nguyen & Perron 2025)***Description**

`dating_knp` dates bubble episodes (origination and collapse) by minimizing a sum of squared residuals that is corrected by omitting a residual. The model has alternating regimes: a unit root, an explosive regime, and a unit root that resumes from a shifted level after an instantaneous collapse, and so on. Plain OLS over this model is provably inconsistent, because the estimate of the origination

date converges to the true *collapse* date and not to the origination date. The default `omit = TRUE` fixes this by dropping the squared residual at each candidate collapse date from the objective before minimizing.

Usage

```
dating_knp(data, trim = 0.05, omit = TRUE, breaks = 2L)
```

Arguments

<code>data</code>	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in <code>badf</code> and <code>bsadf</code> and excluded from the <code>adf</code> , <code>sadf</code> and <code>gsadf</code> of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (<code>bsadf_panel</code> and <code>gsadf_panel</code>) are not available, and the function returns NA for them with a warning.
<code>trim</code>	Minimum fraction of the (differenced) sample required in each regime (default 0.05).
<code>omit</code>	Use the consistency-restoring correction of Kejriwal, Nguyen & Perron (default TRUE). FALSE gives the plain OLS estimator, which is provably inconsistent (their Theorem 1). It is kept mainly to demonstrate the effect of the correction and not for practical dating.
<code>breaks</code>	Number of break dates (the paper's m), two for each bubble. An odd number lets the last bubble run to the end of the sample, and its collapse is then NA.

Details

`breaks = 2` (the default) is the single-bubble model. More breaks use the dynamic-programming algorithm of Kejriwal, Nguyen & Perron, which returns the exact global minimizer of the objective in $O(\text{breaks} * n^2)$. As in the paper, the number of breaks is taken as given, for example two for each episode that `datestamp` finds.

Value

An object of class `dating_knp_obj`: a list with `origination` and `collapse` (dates) and `delta` (the fitted explosive AR coefficient). For a single bubble these are named vectors with one value for each series, and for more bubbles they are matrices with one row for each bubble and one column for each series.

Status

[Experimental]

Note

This is a model-selection dating procedure based on the residual sum of squares and not a hypothesis test, so it needs no critical values.

The function returns its own class and not `radf_obj`, so it does not work with `summary()`, `\link{datestamp}` and `tidy`. It has its own `print()` and `autoplot()` methods instead. `print()` shows the dating table (model, origination, collapse, recovery). See `vignette("naming-and-analysis", package = "exuber")` for which functions fit the shared pipeline and which do not.

References

Kejriwal, M., Nguyen, L., & Perron, P. (2025). An improved procedure for retrospectively dating the emergence and collapse of bubbles. *Journal of Time Series Analysis*, 46(5), 867-883.

See Also

`dating_hls` and `dating_pdc` for related SSR-based dating approaches.

Other dating: `dating_hls()`, `dating_hlw()`, `dating_pdc()`, `radf_recovery()`, `rootstamp()`

Examples

```
res <- dating_knp(sim_data$psy1, trim = 0.05)
print(res)
autoplot(res)

# Compare the bias-corrected estimate with the plain (inconsistent) OLS one,
# adding an extra reference line to the output of autoplot()
res_plain <- dating_knp(sim_data$psy1, trim = 0.05, omit = FALSE)
autoplot(res) +
  ggplot2::geom_vline(xintercept = as.numeric(res_plain$origination), linetype = 3)

# Two bubbles
dating_knp(sim_data$psy2, breaks = 4)
```

dating_pdc

Sequential Sample-Splitting Bubble Dating (PDC/KS)

Description

`dating_pdc` dates a single bubble episode with the sequential sample-splitting method of Pang, Du & Chong (2021) and its four-regime extension by Kurozumi & Skrobotov (2023). The regime structure is fixed: a unit root, an explosive regime, a stationary collapse and, optionally, a final unit-root recovery regime. The breakpoints are estimated one at a time. Each is a closed-form residual-sum-of-squares minimization over a no-intercept AR(1) model, computed in $O(T)$ time with cumulative sums.

Usage

```

dating_pdc(
  data,
  regimes = 3L,
  trim = 0.05,
  type = c("ols", "wls"),
  kernel = c("gaussian", "uniform"),
  h = NULL
)

```

Arguments

<code>data</code>	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in <code>badf</code> and <code>bsadf</code> and excluded from the <code>adf</code> , <code>sadf</code> and <code>gsadf</code> of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (<code>bsadf_panel</code> and <code>gsadf_panel</code>) are not available, and the function returns NA for them with a warning.
<code>regimes</code>	Either 3 (PDC: unit root, explosive, stationary collapse) or 4 (KS: adds a final unit-root recovery regime after the collapse).
<code>trim</code>	Minimum fraction of the (differenced) sample required on either side of each breakpoint search (default 0.05, as in the empirical application of KS. PDC use 0.05 to 0.1 in their simulations).
<code>type</code>	"ols" (default) for the plain homoskedastic estimator, or "wls" for the volatility-corrected two-step estimator of Kurozumi & Skrobotov (2023).
<code>kernel</code>	Kernel for the spot-volatility estimator when <code>type = "wls"</code> , "gaussian" (default) or "uniform". It is ignored when <code>type = "ols"</code> .
<code>h</code>	Bandwidth for the spot-volatility estimator when <code>type = "wls"</code> . The default is leave-one-out cross-validation. It is ignored when <code>type = "ols"</code> .

Details

`datestamp` finds where the recursive BSADF statistic crosses a critical value. This function instead fits an explicit regime-switching model directly to the series, and it needs no critical values. PDC prove that the collapse date is identified first, because its effect on the residual sum of squares dominates that of the origination date. This justifies estimating the breaks sequentially and not jointly. The alternative of Harvey, Leybourne & Sollis (2017) selects among models by BIC and fits them jointly, and it is not implemented here (see the package's enhancement notes for the cost and benefit considerations).

`type = "wls"` adds the correction of Kurozumi & Skrobotov (2023) for time-varying volatility. The function first fits the plain ("ols") model and collects its fitted residuals for each regime. It then smooths their squares nonparametrically, with the same Nadaraya-Watson kernel and leave-one-out bandwidth estimator that `exuber` already uses in `radf_sbz_cv` and `radf_kp`. Finally it reruns the same sequential break search with each squared term weighted by the inverse of the estimated spot

variance. This needs no new critical-value theory. Like the OLS version, it is a point estimate and not a threshold-crossing test.

Value

An object of class `dating_pdc_obj`. It is a `data.frame` with one row for each series and the columns `origination`, `collapse` and, if `regimes = 4`, `recovery`, which give the estimated break dates (or observation indices, if no date index is available). It has its own `print()` and `autoplot()` methods.

Status

[Experimental]

Note

The function returns its own class and not `radf_obj`, so it does not work with `summary()`, `\link{datestamp}` and `tidy`. It has its own `print()` and `autoplot()` methods instead. `print()` shows the dating table (model, origination, collapse, recovery). See `vignette("naming-and-analysis", package = "exuber")` for which functions fit the shared pipeline and which do not.

References

- Pang, T., Du, L., & Chong, T. T. L. (2021). Estimating multiple breaks in the bubble regime with SSR minimization. *Journal of Management Science and Engineering*.
- Kurozumi, E., & Skrobotov, A. (2023). Bubble dating: a sequential testing approach.
- Kurozumi, E., & Skrobotov, A. (2023). Improving the accuracy of bubble date estimators under time-varying volatility. *arXiv:2306.02977*.

See Also

`datestamp` for the threshold-crossing alternative of PSY.

Other dating: `dating_hls()`, `dating_hlw()`, `dating_knp()`, `radf_recovery()`, `rootstamp()`

Examples

```
# The unit-root, explosive, collapse and recovery process of sim_ps1() is the
# regime structure that dating_pdc() fits (true breaks at 40, 60 and 70)
y <- sim_ps1(n = 100, seed = 1)
res <- dating_pdc(y, regimes = 3L, trim = 0.05)
print(res)
autoplot(res)

# Four-regime extension, which adds a post-collapse recovery breakpoint
res4 <- dating_pdc(y, regimes = 4L, trim = 0.05)
autoplot(res4)

# Volatility-weighted (WLS) variant, robust to time-varying volatility
y_vol <- sim_ps1(n = 100, seed = 1, e = sim_vol_break(99))
autoplot(dating_pdc(y_vol, type = "wls"))
```

diagnostics

Diagnostics on hypothesis testing

Description

Reports whether the null hypothesis of a unit root is rejected against the alternative of explosive behavior, for each series in a dataset.

Usage

```
diagnostics(object, cv = NULL, ...)

## S3 method for class 'radf_obj'
diagnostics(object, cv = NULL, option = c("gsadf", "sadf"), sig_lvl = 95, ...)
```

Arguments

<code>object</code>	An object of class <code>obj</code> .
<code>cv</code>	An object of class <code>cv</code> .
<code>...</code>	Further arguments passed to methods.
<code>option</code>	Whether to apply the "gsadf" or "sadf" methodology (default = "gsadf").
<code>sig_lvl</code>	Significance level, one of 90, 95 or 99, that decides whether a series counts as "positive" (rejects the null). It does not depend on the test statistic chosen with <code>option</code> .

Details

`diagnostics` also stores a vector that takes the value 1 when there is a period of explosive behavior and 0 otherwise.

Value

A list with the series that reject the null hypothesis (positive) and the series that do not (negative), together with the significance level.

Examples

```
# The default `cv` is fetched from the shared critical-value store
# (network on first use); pass `cv = radf_mc_cv(nrow(sim_data))` to stay offline
rsim_data <- radf(sim_data)
diagnostics(rsim_data)

diagnostics(rsim_data, option = "sadf")
```

```
# Gate on the 90% critical value instead of the 95% default
diagnostics(rsim_data, sig_lvl = 90)
```

exuber_functions	<i>Look Up exuber's Test/Procedure Functions by Family</i>
------------------	--

Description

The naming conventions (`radf_`, `_test`, `dating_` and `monitor_`, see `vignette("naming-and-analysis")`) are a guide and not a contract. They are easy to misremember, and they are occasionally traded off deliberately: `monitor()` uses the ADF family internally but is named after what it does. This function is the queryable source of truth for which of the package's test, dating, monitoring and root-inference functions belong to which family.

Usage

```
exuber_functions(family = NULL)
```

Arguments

family	One of "adf" (built on the recursive ADF core), "test" (a standalone hypothesis test), "dating" (point estimation or model selection, with no formal test), "monitor" (real-time or sequential), "root" (confidence-interval inference on the explosive root), "regression" (point estimation, no test) or NULL (default) for every function. A function can belong to more than one family: <code>monitor()</code> is both "adf" and "monitor".
--------	--

Value

A tibble with the columns name, family and description, and one row for each function.

See Also

`vignette("naming-and-analysis", package = "exuber")` for the full naming scheme and for which functions work with `summary()`, [datestamp](#), `tidy` and `autoplot`.

Examples

```
exuber_functions()
exuber_functions(family = "monitor")
exuber_functions(family = "test")
```

index-rd	<i>Retrieve/Replace the index</i>
----------	-----------------------------------

Description

Retrieve or replace the index of an object.

Usage

```
index(x, ...)  
  
index(x) <- value
```

Arguments

x	An object.
...	Further arguments passed to methods.
value	An ordered vector of the same length as the index attribute of x.

Details

If you do not specify an index for the estimation, the function generates a pseudo-index, a sequential numeric series. After the estimation you can use `index()` to retrieve the index or `index<-( )` to replace it. The index can be numeric or Date.

Value

The index of the object (a Date vector when the input data carried one, otherwise an integer sequence). The replacement form returns the modified object.

Examples

```
# A plain numeric vector gets a pseudo-index (1, 2, 3, ...)  
rsim <- radf(sim_data)  
head(index(rsim))  
  
# A data.frame with a Date column uses it as the index automatically  
rsim_wdate <- radf(sim_data_wdate)  
head(index(rsim_wdate))  
class(index(rsim_wdate))  
  
# autoplot() uses index() internally for the x-axis  
autoplot(rsim_wdate)  
  
# Replace the index, e.g. with a custom Date sequence  
index(rsim) <- seq(as.Date("2000-01-01"), by = "month", length.out = length(index(rsim)))  
head(index(rsim))  
autoplot(rsim)
```

lbi_test

*Locally Best Invariant Test for a Bubble (Breitung & Diegel 2025)***Description**

`lbi_test` implements the static locally best invariant (LBI) test of Breitung & Diegel (2025) for a bubble that is known, or assumed, to span the entire sample: $LB\dot{I} = (y_T - y_1) / (\sigma_{\text{tilde}} * \sqrt{T - 1})$, where σ_{tilde}^2 is the sample variance of the first differences. The test is robust to heteroskedasticity by construction, because the invariance property of the statistic does not depend on the exact form of the innovation variance. The null distribution is standard normal, so the test needs no bootstrap, no simulation and no published table.

Usage

```
lbi_test(data, sig_lvl = 95)
```

Arguments

<code>data</code>	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a <code>data.frame</code> . A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in <code>badf</code> and <code>bsadf</code> and excluded from the <code>adf</code> , <code>sadf</code> and <code>gsadf</code> of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (<code>bsadf_panel</code> and <code>gsadf_panel</code>) are not available, and the function returns NA for them with a warning.
<code>sig_lvl</code>	Significance level of the one-sided, right-tailed test (positive bubbles only), on the 0 to 100 scale used throughout the package (default 95). Any value in $[50, 100]$ is accepted, because the critical value is a closed-form normal quantile.

Details

Only the static test, which uses a single full-sample window, is implemented. The main contribution of Breitung & Diegel is a sequential, exponentially weighted extension for monitoring a series when the start date is unknown. Its exact weighting scheme and boundary constant are not pinned down here, so this function does not implement it.

Value

An object of class `lbi_test_obj`: a list with the test statistic `stat`, the standard-normal critical value `crit` and detected (logical, `stat > crit`).

Status

[Experimental]

Note

The critical value is closed-form: it is the standard normal (qnorm) quantile at `sig_lvl`, so no bootstrap, simulation or table is needed.

The function returns its own class and not `radf_obj`, so it does not work with `summary()`, `\link{datestamp}` and `tidy`. It has its own `print()` and `autoplot()` methods instead. `print()` shows the statistic, the critical value and whether the bubble was detected. See `vignette("naming-and-analysis", package = "exuber")` for which functions fit the shared pipeline and which do not.

References

Breitung, J., & Diegel, M. (2025). A locally best invariant sequential test for explosive behavior in the presence of nonstationary volatility. *Journal of Time Series Analysis*.

See Also

[radf](#) for the recursive ADF-family alternative that this test complements.

Other alternative tests: [quantile_test\(\)](#)

Examples

```
y <- sim_psy1(n = 60, te = 1, tf = 60, seed = 1) # explosive from the start
res <- lbi_test(y)
print(res)

# Compare the statistic to its critical value
autoplot(res)
```

 monitor

Real-Time Monitoring for Explosive Bubbles

Description

`monitor` implements real-time monitoring. You fix a training window $[1, T^*]$ that is assumed free of exuberance and calibrate a critical value on it. The function then compares the running recursive statistic at each subsequent point $T^*+1, \dots, T$ with that fixed boundary and flags the first date at which the boundary is breached.

Usage

```
monitor(
  data,
  r_star = 0.5,
  minw = NULL,
  nboot = 500L,
  sig_lvl = 95,
```

```

lag = 0L,
type = c("fixed", "aic", "bic"),
seed = NULL,
boundary = c("bootstrap", "kurozumi", "fluc"),
s0 = 0
)

```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
r_star	The end of the training window: a fraction in (0, 1) of the sample (default 0.5), or an integer number of observations if ≥ 1 .
minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
nboot	Number of wild bootstrap replications for the training critical value. It is ignored unless boundary = "bootstrap".
sig_lvl	Significance level for the monitoring boundary on the 0 to 100 scale used throughout the package, one of 90, 95 (default) or 99.
lag	A non-negative integer. The lag length of the Augmented Dickey-Fuller regression (default = 0L).
type	Lag selection for the wild bootstrap process, passed to radf_wb_ps_cv . It is ignored unless boundary = "bootstrap".
seed	Optional seed for the bootstrap draws. It is ignored unless boundary = "bootstrap".
boundary	"bootstrap" (default, Phillips & Shi 2020), "kurozumi" (the closed-form SADF/GSADF boundary of Kurozumi 2020) or "fluc" (the FLUC boundary of Homm & Breitung 2012).
s0	The range of window starts of Kurozumi (2020), as a fraction of the training length. It is used only when boundary = "kurozumi". The default 0 is the SADF case, with the window start fixed at 1. 0.4 or 0.8 switches to the GSADF_{s0} case, where the window start ranges over $[1, \text{floor}(T * s0)]$. These are the only two values for which the scaling constants of his boundary function are tabulated.

Details

boundary = "bootstrap" (the default) implements Phillips & Shi (2020). The boundary is a wild-bootstrap quantile of the GSADF-type statistic (see the tb parameter of [radf_wb_ps_cv](#)), and it is compared with the bsadf sequence of [radf\(\)](#). The function calibrates on the training window *only* (data[1:T*]) and not on the full series. The null-model fit inside [radf_wb_ps_cv](#) ([adf_res\(\)](#))

uses all the data it is given and does not truncate them to tb , so passing data after T^* , which may be explosive, directly to it would leak future information into the calibration of the null.

`boundary = "kurozumi"` implements the closed-form alternative of Kurozumi (2020). It needs no bootstrap and compares a published constant (his Table 1) with the `badf` sequence of `radf()`. The default `s0 = 0` gives his $SADF(k)$ detector, where the window start is fixed at 1. Setting `s0` to 0.4 or 0.8 switches to his $GSADF_{\{s0\}}(k)$ generalization. The window start then ranges over $[1, \text{floor}(T * s0)]$ and is not fixed at 1, and the comparison uses his boundary function, which varies with k and is not constant, together with its own published scaling constant. `sig_lvl` must be one of 90, 95 or 99, the levels that his table tabulates.

`boundary = "fluc"` implements the FLUC detector of Hogg & Breitung (2012). Their $DF_{\{t/n\}}$ is also exactly the `badf` sequence of `radf()`, and it is compared with a published constant from their Table 7 (the case without detrending) and not with a simulated one. `sig_lvl` must be one of 90, 95 or 99.

Value

An object of class `monitor_obj`: a list with the full-sample statistic path (`stat`, which is `bsadf` for `boundary = "bootstrap"` and `badf` for `"kurozumi"` and `"fluc"`), the calibrated boundary (one flat value for each series), the length of the training window `T_star`, and `alarm` and `alarm_date` (the first observation or date in the monitoring period at which `stat` breaches the boundary, NA if it never does).

Status

[Experimental]

Note

The function returns its own class and not `radf_obj`, so it does not work with `summary()`, `\link{datestamp}` and `tidy`. It has its own `print()` and `autoplot()` methods instead. `print()` shows the boundary and the alarm. See `vignette("naming-and-analysis", package = "exuber")` for which functions fit the shared pipeline and which do not.

References

- Phillips, P. C., & Shi, S. (2020). Real time monitoring of asset markets: Bubbles and crises. In *Handbook of Statistics* (Vol. 42, pp. 61-80). Elsevier.
- Kurozumi, E. (2020). Asymptotic properties of bubble monitoring tests. *Econometric Reviews*, 39(5), 510-538.
- Hogg, U., & Breitung, J. (2012). Testing for speculative bubbles in stock markets: A comparison of alternative methods. *Journal of Financial Econometrics*, 10(1), 198-231.

See Also

`radf_wb_ps_cv` for the underlying wild bootstrap, and `datestamp` for the existing full-sample dating of origination and collapse, which is not a monitoring procedure.

Other monitoring: `monitor_cusum()`, `monitor_lbi()`, `monitor_quantile()`

Examples

```
# A bubble-free training window (first half), explosive from t = 150 on
y <- sim_psy1(n = 200, te = 150, tf = 200, seed = 7)
# Default: Phillips & Shi (2020) wild bootstrap boundary
mon <- monitor(y, r_star = 0.5, nboot = 200)
print(mon)
autoplot(mon)

# Closed-form boundary of Kurozumi (2020), which needs no bootstrap
mon_kz <- monitor(y, r_star = 0.5, boundary = "kurozumi")
autoplot(mon_kz)

# Homm & Breitung (2012) FLUC boundary
autoplot(monitor(y, r_star = 0.5, boundary = "fluc"))
```

monitor_cusum

CUSUM Real-Time Monitoring for Explosive Bubbles

Description

monitor_cusum implements the CUSUM real-time monitoring procedure of Homm & Breitung (2012). You fix a training window $[1, T^*]$ that is assumed free of exuberance. The function then compares the standardized cumulative sum of the post-training first differences, $S_t = (y_t - y_{T^*}) / \sigma_{\hat{t}}$, with the closed-form boundary $c_t * \sqrt{t}$, where $c_t = \sqrt{b_{\alpha} + \log(t / T^*)}$, and flags the first date at which the boundary is breached.

Usage

```
monitor_cusum(
  data,
  r_star = 0.5,
  b_alpha = 4.6,
  boundary = c("asymptotic", "finite"),
  sig_lvl = 95,
  type = c("standard", "kernel"),
  h = 20,
  kernel = c("gaussian", "uniform")
)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the
------	---

	middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
r_star	The end of the training window: a fraction in $(0, 1)$ of the sample (default 0.5), or an integer number of observations if ≥ 1 .
b_alpha	The boundary constant (eq. 29 of HB). The default 4.6 is the one-sided asymptotic calibration of HB for a 5% Section 3). It is an asymptotic upper bound on the false-alarm probability (Chu, Stinchcombe & White 1996) and not an exact size, so it is typically conservative in finite samples. It is ignored when boundary = "finite".
boundary	"asymptotic" (default) uses b_alpha directly. "finite" instead looks up the finite-sample boundary constant of HB (their Table 8) from sig_lvl and the realized ratio of training length to monitoring horizon. sig_lvl must then be one of 90, 95 or 99.
sig_lvl	Significance level on the 0 to 100 scale used throughout the package when boundary = "finite" (default 95). It is ignored when boundary = "asymptotic".
type	"standard" (default) for the original CUSUM statistic of Homm & Breitung (2012), or "kernel" for the volatility-robust "CUSUMV" variant of Astill, Harvey, Leybourne, Taylor & Zu (2023).
h	Bandwidth (window length, N in AHLTZ) of the one-sided kernel spot-variance estimator when type = "kernel". The default 20 is the value that the authors recommend from their experiments (their Section 3: "setting H = 20 delivered a procedure with the best trade-off" between robustness of the false-alarm rate and power). It is ignored when type = "standard".
kernel	Kernel for the spot-variance estimator when type = "kernel", "gaussian" (default) or "uniform". It is ignored when type = "standard".

Value

An object of class `monitor_cusum_obj`: a list with the statistic path in the monitoring region (S) and boundary, the length of the training window `T_star`, and `alarm` and `alarm_date` (the first breach, NA if there is none).

Status

[Experimental]

Note

The boundary is closed-form throughout. It is either a fixed asymptotic constant (boundary = "asymptotic", `b_alpha` = 4.6) or a lookup in the published finite-sample table (boundary = "finite", Table 8 of Homm & Breitung 2012). No simulation and no separate `cv` function is needed.

`monitor` (Family A) uses a recursive ADF-family statistic whose boundary needs a wild bootstrap for calibration. This function uses a structurally different statistic, a standardized running sum and not a recursive regression. Its boundary is an asymptotic closed form (the inequality of Chu, Stinchcombe & White 1996, eq. 28 of HB), so it needs no bootstrap and no simulation and does not depend on the data beyond the running variance estimate itself.

type = "kernel" instead uses the volatility-robust modification ("CUSUMV") of Astill, Harvey, Leybourne, Taylor & Zu (2023). Before cumulating, it standardizes each first difference by its own one-sided kernel spot-variance estimate (their eq. 6-7) and not by a single running variance. Their Corollary 1 establishes that the *same* boundary function gives a controlled asymptotic false-alarm rate even under time-varying volatility. The standard CUSUM statistic does not have this property, because its size-control result requires homoskedasticity.

The function returns its own class and not `radf_obj`, so it does not work with `summary()`, `\link{datestamp}` and `tidy`. It has its own `print()` and `autoplot()` methods instead. `print()` shows the boundary and the alarm. See `vignette("naming-and-analysis", package = "exuber")` for which functions fit the shared pipeline and which do not.

References

- Homm, U., & Breitung, J. (2012). Testing for speculative bubbles in stock markets: A comparison of alternative methods. *Journal of Financial Econometrics*, 10(1), 198-231.
- Chu, C. S. J., Stinchcombe, M., & White, H. (1996). Monitoring structural change. *Econometrica*, 64(5), 1045-1065.
- Astill, S., Harvey, D. I., Leybourne, S. J., Taylor, A. M. R., & Zu, Y. (2023). CUSUM-based monitoring for explosive episodes in financial data in the presence of time-varying volatility. *Journal of Financial Econometrics*, 21(1), 187-227.

See Also

`monitor` for the monitoring alternative based on the recursive ADF (Family A).

Other monitoring: `monitor()`, `monitor_lbi()`, `monitor_quantile()`

Examples

```
# A martingale training window, explosive from t = 150 to the sample end
y <- sim_psy1(n = 200, te = 150, tf = 200, seed = 7)
res <- monitor_cusum(y, r_star = 0.5)
print(res) # alarm should fire soon after t = 150
autoplot(res)

# Volatility-robust "CUSUMV" variant (Astill, Harvey, Leybourne, Taylor & Zu 2023):
# the same bubble, but volatility triples at t = 120, after the training window
y_vol <- sim_psy1(n = 200, te = 150, tf = 200, seed = 7,
  e = sim_vol_break(199, tau = 0.6))
res_kernel <- monitor_cusum(y_vol, r_star = 0.5, type = "kernel")
autoplot(res_kernel)
```

monitor_lbi	<i>Sequential LBI Monitoring for an Unknown Bubble Start Date (Breitung & Diegel 2025)</i>
-------------	--

Description

monitor_lbi implements the sequential, constant-boundary extension of the locally best invariant statistic of [lbi_test](#). It monitors a series in real time when the start date of the bubble is unknown. After a training window $[1, T^*]$ that is assumed free of exuberance, it compares the (optionally exponentially weighted) partial sum of the post-training first differences with a constant boundary and flags the first monitoring date at which the boundary is breached.

Usage

```
monitor_lbi(data, r_star = 0.5, c_bar = 0, sig_lvl = 95)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
r_star	The end of the training window: a fraction in $(0, 1)$ of the sample (default 0.5), or an integer number of observations if ≥ 1 .
c_bar	Exponential up-weighting parameter for later monitoring observations, which are more likely to be bubble-like (their eq. 12), ≥ 0 . The default 0 is the flat-weight "mCUSUM" variant, which is appropriate when a bubble is equally likely to start at any point in the monitoring window. For a moderate gain in power when a bubble that starts partway through the window is more plausible, the paper suggests 2. The critical values (sig_lvl) are the same for every c_bar.
sig_lvl	Significance level on the 0 to 100 scale used throughout the package, one of 90, 95, 97.5, 99 or 99.5. Table 1 of Breitung & Diegel tabulates only these.

Details

Their eq. 15 shows that this partial sum, normalized by the fixed length of the monitoring horizon, converges to a standard Brownian motion on $[0, 1]$ under the null. The normalization is not $\sqrt{t}$, which differs from the Chu-Stinchcombe-White-style boundary of [monitor_cusum](#). A single constant boundary therefore controls size uniformly across the whole monitoring window. The paper shows that this constant-boundary detector ("mCUSUM" at $c_{\text{bar}} = 0$, "wCUSUM" at $c_{\text{bar}} > 0$) is more powerful than the classical CUSUM test with a time-varying boundary, to which it is compared.

Value

An object of class `monitor_lbi_obj`: a list with the statistic path in the monitoring region (`stat`), the constant boundary, the length of the training window `T_star`, and `alarm` and `alarm_date` (the first breach, NA if there is none).

Status

[Experimental]

Note

The critical value is the published constant boundary in Table 1 of Breitung & Diegel (2025), so it is a table lookup and needs no simulation.

The function returns its own class and not `radf_obj`, so it does not work with `summary()`, `\link{datestamp}` and `tidy`. It has its own `print()` and `autoplot()` methods instead. `print()` shows the boundary and the alarm. See `vignette("naming-and-analysis", package = "exuber")` for which functions fit the shared pipeline and which do not.

References

Breitung, J., & Diegel, M. (2025). A locally best invariant sequential test for explosive behavior in the presence of nonstationary volatility. *Journal of Time Series Analysis*.

See Also

`lbi_test` for the static version, which assumes a known bubble window that spans the full sample. `monitor_cusum` and `monitor` are monitoring detectors with a structurally different construction.

Other monitoring: `monitor()`, `monitor_cusum()`, `monitor_quantile()`

Examples

```
# A martingale training window, explosive from t = 150 to the sample end
y <- sim_psy1(n = 200, te = 150, tf = 200, seed = 7)
res <- monitor_lbi(y, r_star = 100)
print(res) # alarm should fire soon after t = 150
autoplot(res)
```

```
# wCUSUM: exponentially up-weight later monitoring observations
autoplot(monitor_lbi(y, r_star = 100, c_bar = 2))
```

monitor_quantile

*QPWY/QPSY Recursive Quantile Monitoring (Wu, Shi & Wu 2025)***Description**

monitor_quantile implements the QPWY and QPSY real-time monitoring strategies of Wu, Shi & Wu (2025). They are quantile-regression (QR) analogues of the recursive ADF t-statistics of PWY and PSY, and they test at a chosen conditional quantile τ , in contrast to the single full-sample test in [quantile_test](#). type = "qpwy" uses the expanding window $[1, r]$, which has the shape of [badf](#) in [radf](#). type = "qpsy" also takes the supremum over every window start, which has the shape of [bsadf](#).

Usage

```
monitor_quantile(
  data,
  tau = 0.5,
  minw = NULL,
  nrep = 500L,
  sig_lvl = 95,
  seed = NULL,
  type = c("qpwy", "qpsy"),
  boundary = c("asymptotic", "bootstrap")
)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf , sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
tau	Quantile to test at, in $(0, 1)$. It is fixed, in contrast to the "optimal" grid search in quantile_test , because eq. 25 of WSW takes τ as a given parameter of the monitoring statistic and does not reselect it at each recursion point.
minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
nrep	Number of replications for the boundary: Monte Carlo draws of the limit with boundary = "asymptotic", bootstrap resamples with boundary = "bootstrap".
sig_lvl	Significance level, one of 90, 95, 99.
seed	Optional seed for the Monte Carlo draws.
type	"qpwy" (expanding window) or "qpsy" (also the supremum over window starts).

boundary "asymptotic" (simulated limit, the default) or "bootstrap" (Algorithm 1 of Wu, Shi & Wu 2025).

Details

The point statistic needs genuine QR fits, because there is no closed-form recursive update as there is for OLS. QPWY needs $O(T)$ fits and QPSY needs $O(T^2)$, so QPSY takes seconds for each series at $n = 200$ and the time grows quadratically.

The boundary is a single **flat** value and not one value for each r . It is the quantile of the supremum of each null path, constructed like the `sadf_cv` of `radf_mc_cv`, which controls the first-crossing false-alarm rate. `boundary` chooses how the null paths are generated.

With `boundary = "asymptotic"` (the default) the function simulates the limiting null distribution $\delta \cdot Q_{\{r_1, r_2\}} + \sqrt{1 - \delta^2} \cdot Z_{\{r_1, r_2\}}$ in each call. Here δ is a correlation coefficient estimated from the data (as in `quantile_test`), Q is the Dickey-Fuller t functional and Z is its counterpart driven by an independent Brownian motion. Both are simulated for every window, so no QR fits are needed and the boundary is cheap.

With `boundary = "bootstrap"` the function applies Algorithm 1 of Wu, Shi & Wu (2025) to the whole path. It resamples the centred first differences of the series with replacement, cumulates them into a null random walk and recomputes the QPWY or QPSY path on it, `nrep` times. The boundary is the quantile of the `nrep` path maxima. It follows the finite-sample distribution of the statistic in the data at hand, so it removes most of the size distortion of the asymptotic boundary described below. Each replicate costs a full statistic path, which is $O(T)$ QR fits for QPWY and $O(T^2)$ for QPSY. The paper also discards the first 100 draws of each resample. We omit this step because the statistic has an intercept, so it does not depend on the level of the series, and the draws are i.i.d., so the retained values already have the same distribution. Set `options(exuber.parallel = TRUE)` to spread the replicates over workers.

Value

An object of class `monitor_quantile_obj`: a list with the statistic path `stat`, the flat boundary, the estimated δ (reported for both boundaries, used only by the asymptotic one), and `alarm` and `alarm_date` (the first breach, NA if there is none).

Caveats

The asymptotic boundary is well sized near the median in finite samples, with a false-alarm rate of 3.5 to 4.0% innovations, $\tau = 0.5$). Away from the median the small early windows make both statistics oversized, and QPSY badly so even with Gaussian innovations. The false-alarm rate of QPSY is 35% innovations it is 21% ($n = 100$). With t_3 innovations the rate for QPWY is 7.5 to 8.5% $\tau = 0.2$ and 0.8 and 12.5% Wu, Shi & Wu advise against extreme quantiles in small samples and use bootstrap critical values for monitoring, which is what `boundary = "bootstrap"` provides. With it the false-alarm rate was between 4.5 and 6.5% the cases above ($n = 100$ for QPWY, $n = 60$ for QPSY), including QPSY at $\tau = 0.8$ and 0.9 . The exception is QPSY with t_3 innovations at $\tau = 0.9$, which stays at 10% less extreme τ help. For `type = "qpsy"` with τ away from 0.5 and the asymptotic boundary, the function emits a short pointer as a message (see `suppressMessages`) and stores it as `attr(x, "caveat")`. The numbers for both boundaries are in `docs/alternative-paradigms.md`.

Status**[Experimental]****Note**

The function simulates the critical value (the boundary) internally in each call, with an unexported helper, `quantile_boundary_sim`. There is currently no reusable exported cv counterpart for this function. This gap is tracked separately and is not addressed here.

The function returns its own class and not `radf_obj`, so it does not work with `summary()`, `\link{datestamp}` and `tidy`. It has its own `print()` and `autoplot()` methods instead. `print()` shows the statistic, the boundary and delta. See `vignette("naming-and-analysis", package = "exuber")` for which functions fit the shared pipeline and which do not.

References

Wu, R., Shi, S., & Wu, J. (2025). Quantile analysis for financial bubble detection and surveillance. *Journal of Time Series Analysis*, 46(5), 908-931.

See Also

`quantile_test` for the static, full-sample version of this test, and `monitor` for the monitoring alternative based on OLS.

Other monitoring: `monitor()`, `monitor_cusum()`, `monitor_lbi()`

Examples

```
# Heavy-tailed (t3) innovations, explosive from t = 150 to the sample end
y <- sim_psy1(n = 200, te = 150, tf = 200, seed = 7,
  e = sim_innov(199, dist = "t", df = 3))
res <- monitor_quantile(y, tau = 0.5, nrep = 100, seed = 1)
print(res)
autoplot(res)

# Monitoring an upper quantile is typically more powerful for right-tailed
# bubbles, but see the Caveats section on extreme quantiles
autoplot(monitor_quantile(y, tau = 0.8, nrep = 100, seed = 1))

# QPSY: supremum over window starts too (O(n^2) QR fits, slower)
monitor_quantile(y[101:200], tau = 0.5, nrep = 100, seed = 1, type = "qpsy")

# Bootstrap boundary (Algorithm 1 of Wu, Shi & Wu): one full statistic path per
# replicate, so keep nrep small in a first run
monitor_quantile(y, tau = 0.8, nrep = 49, seed = 1, boundary = "bootstrap")
```

ps_tb

*Helper function to find tb from Phillips and Shi (2020)***Description**

This function finds the number of observations in the window over which size is to be controlled.

Usage

```
ps_tb(n, freq = c("monthly", "quarterly", "annual", "weekly"), size = 2)
```

Arguments

n	A positive integer. The sample size.
freq	The type of date-interval.
size	The size to be controlled.

Value

A single integer, the length of the training window in observations, suitable for the `r_star` argument of `monitor`.

References

Phillips, P. C., & Shi, S. (2020). Real time monitoring of asset markets: Bubbles and crises. In Handbook of Statistics (Vol. 42, pp. 61-80). Elsevier.

Shi, S., Hurn, S., Phillips, P.C.B., 2018. Causal change detection in possibly integrated systems: Revisiting the money-income relationship.

Examples

```
# Training window that controls size over a 2-year span of monthly data
tb <- ps_tb(100, freq = "monthly", size = 2)
tb
```

```
# Use it directly as the training window of monitor()
monitor(sim_data, r_star = tb, boundary = "kurozumi")
```

 psy_minw

Helper functions in accordance to PSY(2015)

Description

psy_minw and psy_ds use the rules of thumb proposed by Phillips et al. (2015) to compute the minimum window size and the minimum duration of an episode of exuberance, respectively.

Usage

```
psy_minw(n)
```

```
psy_ds(n, rule = 1, delta = 1)
```

Arguments

n	A positive integer. The sample size.
rule	Rule to compute the minimum duration of an episode (default: rule = 1, where T denotes the sample size). Rule 1 corresponds to $\log(T)$, while rule 2 to $\log(T)/T$.
delta	Frequency-dependent parameter (default; delta = 1). See details.

Details

For the minimum duration period, psy_ds allows the user to choose from two rules:

$$\text{rule}_1 = \delta \log(n) \quad \text{and} \quad \text{rule}_2 = \delta \log(n)/n$$

delta depends on the frequency of the data and the minimal duration condition.

Value

psy_minw: a single integer, the minimum window length. psy_ds: a single integer, the minimum episode duration in observations.

References

Phillips, P. C. B., Shi, S., & Yu, J. (2015). Testing for Multiple Bubbles: Historical Episodes of Exuberance and Collapse in the S&P 500. *International Economic Review*, 56(4), 1043-1078. [doi:10.1111/iere.12132](https://doi.org/10.1111/iere.12132)

Examples

```
psy_minw(100)
psy_ds(100)
```

quantile_test

*Quantile Unit Root Test for Bubble Detection (Global Test)***Description**

quantile_test implements the "global test" of Wu, Shi & Wu (2025). It is a quantile-regression (QR) analogue of the Dickey-Fuller t-ratio, and it tests for explosive behavior at a chosen conditional quantile τ of y_t given y_{t-1} , and not at the conditional mean. It is a single static test and not a recursive scan. The comparable statistic in `radf` is the single-shot adf statistic and not the recursive bsadf.

Usage

```
quantile_test(
  data,
  tau = "optimal",
  tau_grid = seq(0.2, 0.8, by = 0.05),
  nrep = 1000L,
  sig_lvl = 95,
  seed = NULL
)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
tau	Quantile to test at, in (0, 1), or "optimal" (default) to select it with the grid search of eq. 33.
tau_grid	Grid searched when tau = "optimal". The default seq(0.2, 0.8, by = 0.05) matches the practical range that the paper recommends, which excludes the extreme quantiles 0.1 and 0.9.
nrep	Number of Monte Carlo replications for the critical value.
sig_lvl	Significance level, one of 90, 95, 99.
seed	Optional seed for the Monte Carlo draws.

Details

tau = "optimal" (the default) selects the quantile that minimizes the asymptotic variance of the QR estimator (their eq. 33) by grid search over tau_grid. The grid excludes the extreme quantiles, which the paper itself recommends avoiding at practical sample sizes.

The function simulates the critical value in each call and does not use a fixed table. There is currently no reusable exported cv counterpart for this function. This gap is tracked separately and is not addressed here. The limiting null distribution of the statistic is $\sqrt{1 - \delta^2} * z + \delta * Q$, where $z \sim N(0, 1)$, δ is a correlation coefficient estimated from the data, and Q is the standard demeaned Dickey-Fuller t-statistic distribution. The function simulates Q with the same random-walk-plus-OLS t-statistic construction used elsewhere in this package (see [radf_mc_cv](#)).

Value

An object of class `quantile_test_obj`: a list with the test statistic `tstat`, the selected tau, the estimated correlation `delta`, the simulated crit value and detected (logical, `tstat > crit`).

Status

[Experimental]

Note

The function returns its own class and not `radf_obj`, so it does not work with `summary()`, `\link{datestamp}` and `tidy`. It has its own `print()` and `autoplot()` methods instead. `print()` shows the statistic, the critical value and `delta`. See `vignette("naming-and-analysis", package = "exuber")` for which functions fit the shared pipeline and which do not.

References

Wu, R., Shi, S., & Wu, J. (2025). Quantile analysis for financial bubble detection and surveillance. *Journal of Time Series Analysis*, 46(5), 908-931.

See Also

[radf](#) for the family of tests based on mean regression (ADF, SADF and GSADF) that this test complements.

Other alternative tests: [lbi_test\(\)](#)

Examples

```
# Heavy-tailed (t3) innovations, where a quantile test is more useful than a test of the mean
y <- sim_psy1(n = 100, seed = 1, e = sim_innov(99, dist = "t", df = 3))
res <- quantile_test(y, nrep = 100, seed = 1)
print(res)
autoplot(res)

# Test at a fixed upper quantile instead of the optimal one
autoplot(quantile_test(y, tau = 0.9, nrep = 100, seed = 1))
```

radf	<i>Recursive Augmented Dickey-Fuller Test</i>
------	---

Description

radf computes the recursive univariate and panel augmented Dickey-Fuller test statistics (ADF, SADF, GSADF and their backward sequences).

Usage

```
radf(data, minw = NULL, lag = 0L)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
lag	A non-negative integer. The lag length of the Augmented Dickey-Fuller regression (default = 0L).

Details

radf() is vectorized: it handles several series at once. The panel statistics are computed as a by-product of the univariate estimations, at almost no additional cost.

Value

A list that contains the unit root test statistics (sequence):

adf	Augmented Dickey-Fuller
badf	Backward Augmented Dickey-Fuller
sadf	Supremum Augmented Dickey-Fuller
bsadf	Backward Supremum Augmented Dickey-Fuller
gsadf	Generalized Supremum Augmented Dickey-Fuller
bsadf_panel	Panel Backward Supremum Augmented Dickey-Fuller
gsadf_panel	Panel Generalized Supremum Augmented Dickey-Fuller

And attributes:

mat	The matrix used in the estimation.
index	The index parsed from the dataset.
lag	The lag used in the estimation.
n	The number of rows.
minw	The minimum window used in the estimation.
series_names	The series names.

References

Phillips, P. C. B., Wu, Y., & Yu, J. (2011). Explosive Behavior in The 1990s Nasdaq: When Did Exuberance Escalate Asset Values? *International Economic Review*, 52(1), 201-226.

Phillips, P. C. B., Shi, S., & Yu, J. (2015). Testing for Multiple Bubbles: Historical Episodes of Exuberance and Collapse in the S&P 500. *International Economic Review*, 56(4), 1043-1078. doi:[10.1111/iere.12132](https://doi.org/10.1111/iere.12132)

Pavlidis, E., Yusupova, A., Paya, I., Peel, D., Martínez-García, E., Mack, A., & Grossman, V. (2016). Episodes of exuberance in housing markets: in search of the smoking gun. *The Journal of Real Estate Finance and Economics*, 53(4), 419-449. doi:[10.1007/s1114601595312](https://doi.org/10.1007/s1114601595312)

Examples

```
# We will use simulated data that are stored as data
sim_data

rsim <- radf(sim_data)

str(rsim)

# We would also use data that contain a Date column
sim_data_wdate

rsim_wdate <- radf(sim_data_wdate)

tidy(rsim_wdate)

augment(rsim_wdate)

tidy(rsim_wdate, panel = TRUE)

head(index(rsim_wdate))

# For lag = 1 and minimum window = 20
rsim_20 <- radf(sim_data, minw = 20, lag = 1)
```

radf_common

*Common-Bubble Detection via PCA + PSY***Description**

radf_common tests for a bubble that is common to a panel of series (Chen, Phillips & Shi, 2023). It extracts the first principal component of the panel and runs the ordinary [radf](#) test on it. The output is an ordinary radf_obj, so every downstream method (tidy(), autoplot(), datestamp() and so on) works on it without further effort.

Usage

```
radf_common(data, minw = NULL, r = 1)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
r	Number of principal components to extract (default 1, which the paper recommends as "sufficient... for the purpose of bubble identification"). Only the first is used for detection. The others are returned for inspection in the "prcomp" attribute.

Details

Theorem 4.3 of the paper claims that the null limiting distribution of the resulting statistic is asymptotically identical to the standard PSY/GSADF distribution, which would let [radf_mc_cv](#) apply directly. An independent validation found that this identity does **not** hold at practical panel widths N. At N = 100 the true critical value is more than double that of [radf_mc_cv](#), and the gap grows as N increases. PCA on a panel of independent (non-cointegrated) I(1) series does not behave like a single random walk once there are more series from which transient co-movement can arise. Use [radf_common_cv](#) for the critical values and **not** [radf_mc_cv](#). The critical values of [radf_mc_cv](#) do not depend on the panel width, and they are badly undersized here once N grows past a handful of series.

Value

A radf_obj (see [radf](#)) computed on the first principal component of the panel, with the fitted prcomp object attached as an attribute (attr(x, "prcomp")).

Status**[Experimental]****Note**

The function returns the output of `radf()` computed on the extracted factor, and `radf_common_cv()` computes the full time-varying boundary alongside the scalar critical values. The `full_summary()`, `datestamp`, `tidy` and `autoplot` pipeline therefore works (see `vignette("naming-and-analysis", package = "exuber")`).

References

Chen, Y., Phillips, P. C. B., & Shi, S. (2023). Common Bubble Detection in Large Dimensional Financial Systems. *Journal of Financial Econometrics*, 21(4), 989-1063.

See Also

`radf` for the underlying test, which is unmodified, and `radf_common_cv` for its critical values, which are specific to the panel width.

Other multivariate: `cobubble_test()`, `contagion_reg()`

Examples

```
# A panel of 5 series driven by one shared latent bubble factor
x <- sim_common(n_series = 5, n = 100, seed = 123)
res <- radf_common(x, minw = 20)
print(res)

# radf_common_cv() is needed here and radf_mc_cv() does not apply (see Details)
cv <- radf_common_cv(n = 100, N = ncol(x), minw = 20)
summary(res, cv = cv)

# The result is an ordinary radf_obj, so autoplot() and datestamp() work directly
autoplot(res, cv = cv)
datestamp(res, cv = cv)
```

radf_common_cv

Critical Values for the Common-Bubble (PCA + PSY) Test

Description

`radf_common_cv` simulates critical values for `radf_common` under its own null of no common explosive factor. The null is a panel of N *independent* random walks, from which one principal component is extracted and tested exactly as in `radf_common`. Independent validation showed that `radf_mc_cv`, whose critical values do not depend on the panel width, is badly undersized as a stand-in for this null once N grows past a handful of series. The null distribution here does depend on N , so N must match the panel on which `radf_common` was run.

Usage

```
radf_common_cv(n, N, minw = NULL, nrep = 1000L, seed = NULL)
```

Arguments

n	A positive integer. The sample size (number of time periods).
N	A positive integer, at least 2. The panel width (number of series) on which radf_common will be run. The critical value depends on it, unlike that of radf_mc_cv .
minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
nrep	A positive integer. The number of Monte Carlo simulations.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Value

A list with `adf_cv`, `sadf_cv`, `gsadf_cv`, `badf_cv` and `bsadf_cv`. It has the same shape as the return value of [radf_mc_cv](#), so you can use it as the `cv` argument of [datestamp](#), [tidy](#) and [autoplot](#) on a [radf_common](#) result.

Status

[Experimental]

References

Chen, Y., Phillips, P. C. B., & Shi, S. (2023). Common Bubble Detection in Large Dimensional Financial Systems. *Journal of Financial Econometrics*, 21(4), 989-1063.

See Also

[radf_common](#), [radf_mc_cv](#)

Other critical values: [radf_mc_cv\(\)](#), [radf_recovery_cv\(\)](#), [radf_sb_cv\(\)](#), [radf_sbz_cv\(\)](#), [radf_sign_cv\(\)](#), [radf_sign_dm_cv\(\)](#), [radf_tt_cv\(\)](#), [radf_wb_cv\(\)](#), [radf_wb_ps_cv\(\)](#)

Examples

```
cv <- radf_common_cv(n = 100, N = 5, minw = 20, nrep = 200)
tidy(cv)
```

radf_kp

*Kernel-Purged Heteroskedasticity-Robust PSY Test***Description**

radf_kp implements the heteroskedasticity-robust PSY test of Harvey, Leybourne, Taylor & Zu (2024), which needs no bootstrap. It "purges" unconditional heteroskedasticity by dividing each first difference of the series by a kernel spot-volatility estimate (eq. 4-5) and cumulating the result. It then runs the ordinary [radf](#), with an intercept, on the purged series.

Usage

```
radf_kp(data, minw = NULL, kernel = c("gaussian", "uniform"), h = NULL)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
kernel	Kernel for the spot-volatility estimator, "gaussian" (default, as in the paper) or "uniform".
h	Bandwidth for the spot-volatility estimator. The default is $0.1 * T^{(-0.25)}$, the setting of the paper (Table I, Section 6).

Details

The paper proves (Theorem 1 and Remark 3.2) that the null limiting distribution of the purged statistic is identical to the standard homoskedastic GSADF null. [radf_mc_cv](#), the existing and already fast Monte Carlo critical values of exuber, therefore apply directly to the result. Unlike [radf_wb_cv](#) and [radf_sbz_cv](#), no new bootstrap or simulation code is needed.

Only the with-intercept variant (PSY_{σ} in the paper) is implemented. The paper also proposes a variant without an intercept and a union-of-rejections test that combines both. They are not implemented here (see the package's enhancement notes for the cost and benefit considerations).

Value

A radf_obj with the same structure as the output of [radf](#), computed on the volatility-purged series, so [radf_mc_cv](#), tidy() and the other methods apply directly.

Status**[Experimental]****Note**

The function returns the unmodified output of `radf`, so the full `summary()`, `datestamp`, `tidy` and `autoplot` pipeline works exactly as it does for plain `radf()` (see `vignette("naming-and-analysis", package = "exuber")`).

References

Harvey, D. I., Leybourne, S. J., Taylor, A. M. R., & Zu, Y. (2024). A new heteroskedasticity-robust test for explosive bubbles. *Journal of Time Series Analysis*. doi:10.1111/jtsa.12784

See Also

`radf_mc_cv` for the critical values of this test, which are unmodified, `radf_wb_cv` for a bootstrap-based alternative, and `radf_tt` for another alternative that needs no bootstrap.

Other volatility-robust tests: `cusum_test()`, `radf_sbz()`, `radf_sbz_union()`, `radf_sign()`, `radf_sign_dm()`, `radf_tt()`, `ssu_test()`

Examples

```
# Volatility triples half-way through the sample. This is the case of
# non-stationary volatility that this test is built for, and plain radf()
# over-rejects here
y <- sim_psy1(n = 200, seed = 1, e = sim_vol_break(199))
res <- radf_kp(y, minw = 20)
print(res)

# radf_mc_cv() applies unmodified (see Details)
cv <- radf_mc_cv(n = attr(res, "n"), minw = 20)
summary(res, cv = cv)
autoplot(res, cv = cv)
```

radf_mc_cv

Monte Carlo Critical Values

Description

`radf_mc_cv` computes Monte Carlo critical values for the recursive unit root tests. `radf_mc_distr` computes the simulated distribution.

Usage

```
radf_mc_cv(n, minw = NULL, nrep = 1000L, seed = NULL, lag = 0)

radf_mc_distr(n, minw = NULL, nrep = 1000L, seed = NULL, lag = 0)
```

Arguments

<code>n</code>	A positive integer. The sample size.
<code>minw</code>	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
<code>nrep</code>	A positive integer. The number of Monte Carlo simulations.
<code>seed</code>	An object specifying if and how the random number generator (rng) should be initialized. It is either <code>NULL</code> or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, <code>NULL</code> , leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.
<code>lag</code>	A non-negative integer. Number of lags in the auxiliary regression, as in radf .

Value

For `radf_mc_cv`, a list with the critical values for the ADF, BADF, BSADF and GSADF test statistics. For `radf_mc_distr`, a list with the ADF, SADF and GSADF distributions.

See Also

[radf_wb_cv](#) for wild bootstrap critical values and [radf_sb_cv](#) for sieve bootstrap critical values
 Other critical values: [radf_common_cv\(\)](#), [radf_recovery_cv\(\)](#), [radf_sb_cv\(\)](#), [radf_sbz_cv\(\)](#), [radf_sign_cv\(\)](#), [radf_sign_dm_cv\(\)](#), [radf_tt_cv\(\)](#), [radf_wb_cv\(\)](#), [radf_wb_ps_cv\(\)](#)

Examples

```
# Default minimum window
mc <- radf_mc_cv(n = 100)

tidy(mc)

# Change the minimum window and the number of simulations
mc2 <- radf_mc_cv(n = 100, nrep = 600, minw = 20)

tidy(mc2)

mdist <- radf_mc_distr(n = 100, nrep = 1000)

autoplot(mdist)

# Apply the critical values to actual data
rsim_data <- radf(sim_data, minw = 20)
autoplot(rsim_data, cv = mc2)
```

radf_recovery	<i>Reverse-Regression Dating of Crisis Origination and Market Recovery</i>
---------------	--

Description

radf_recovery implements the reverse-regression dating of Phillips & Shi (2014). It reverses the series and runs the existing bsadf recursion of radf() on it. It then locates the first up-crossing of a critical-value boundary calibrated for the reversal, which is the market recovery date, and the next down-crossing, which is the crisis (collapse) origination date in the original series. Both are mapped back to the original time index.

Usage

```
radf_recovery(
  data,
  minw = NULL,
  lag = 0,
  nrep = 1000L,
  sig_lvl = 95,
  seed = NULL
)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
lag	A non-negative integer. The lag length of the Augmented Dickey-Fuller regression (default = 0L).
nrep	Number of Monte Carlo replications for the critical value in radf_recovery_cv .
sig_lvl	Significance level, one of 90, 95, 99.
seed	Optional seed for the Monte Carlo draws.

Details

The function returns two dates for each series. `f_c` is the crisis origination (collapse-onset) date, a reverse-regression alternative to the collapse date that [datestamp](#) already dates from the forward test. `f_r` is the market recovery date, and `f_c <= f_r` always holds by construction, because the down-crossing is searched only after the up-crossing. If no up-crossing is found, neither date is identified (NA, `detected = FALSE`). If an up-crossing is found but no later down-crossing occurs before the reverse-time sample ends, `f_c` is NA and `censored = TRUE`, which means that the crisis origination predates the observed sample.

Value

An object of class `radf_recovery_obj`: a list with `f_c` and `f_r` (the estimated dates, NA if not identified), `detected` (logical, whether an up-crossing was found at all) and `censored` (logical, whether `f_c` is left-censored by the start of the reverse-time sample).

Caveats

[Experimental]

Validation status (2026-08-10). `f_r` (the recovery date) validates well against synthetic collapse-then-recovery data. Its bias is in the same range as in the Monte Carlo study of the paper, a few observations early. `f_c` (the crisis origination date) shows a materially larger residual bias in Monte Carlo checks. The empirical false-detection rate under a pure random-walk null ($n = 100$, $\min w = 20$, 95\%) than the comparable numbers for the forward tests elsewhere in this package. We found and fixed one artifact of the synthetic process during validation, where a level jump at a regime boundary produced a spurious spike. The remaining bias in `f_c` and the elevated false-detection rate are not fully explained. They may be genuine finite-sample noise from the literal first-down-crossing rule of the paper. The `inf` in its eq. 9 has no persistence requirement, so a transient dip below the boundary is enough to trigger a premature `f_c`. We have not ruled out a subtler implementation issue. Treat `f_c` and the overall detection rate as exploratory until they are validated further, and see [docs/dating-and-root-inference.md](#) for the full numbers. The function emits the same short pointer as a message when it is called (use [suppressMessages](#) to silence it) and stores it as `attr(x, "caveat")` on the returned object.

Note

The function returns its own class and not `radf_obj`, so it does not work with `summary()`, `\link{datestamp}` and `tidy`. It has its own `print()` and `autoplot()` methods instead. `print()` shows the origination and recovery dates. See `vignette("naming-and-analysis", package = "exuber")` for which functions fit the shared pipeline and which do not.

References

Phillips, P. C. B., & Shi, S. (2014). Financial Bubble Implosion and Reverse Regression. Cowles Foundation Discussion Paper No. 1967, Yale University. Published in *Econometric Theory*.

See Also

[datestamp](#) for the forward, non-reversed dating of origination and collapse that this function complements.

Other dating: [dating_hls\(\)](#), [dating_hlw\(\)](#), [dating_knp\(\)](#), [dating_pdc\(\)](#), [rootstamp\(\)](#)

Examples

```
# The expansion, bubble, collapse and recovery process of sim_ps1()
y <- sim_ps1(n = 100, seed = 1)
res <- radf_recovery(y, minw = 15, nrep = 200, seed = 1)
print(res)

# Plot the series with the estimated collapse (f_c) and recovery (f_r) points
autoplot(res)
```

radf_recovery_cv	<i>Monte Carlo Critical Values for Reverse-Regression Recovery Dating</i>
------------------	---

Description

Computes critical values for the reverse-regression BSADF statistic that [radf_recovery](#) uses. They are calibrated to the null limiting distribution of that statistic (Theorem 1 of Phillips & Shi 2014) and not to the standard forward boundary of [radf_mc_cv](#). The simulated null path is reversed before the recursive computation, because reversal induces an endogeneity that has no analogue in the forward regression (see the Details of [radf_recovery](#)).

Usage

```
radf_recovery_cv(n, minw = NULL, nrep = 1000L, seed = NULL, lag = 0)
```

Arguments

n	A positive integer. The sample size.
minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
nrep	A positive integer. The number of Monte Carlo simulations.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.
lag	A non-negative integer. Number of lags in the auxiliary regression, as in radf .

Value

A list of class `radf_cv` with a single element, `bsadf_cv`. It is a matrix of critical values (columns 90%, 95%, 99%) with one row for each reverse-time position, aligned in the same way as the `bsadf_cv` of [radf_mc_cv](#) is aligned to `radf()`\$bsadf.

Status**[Experimental]****Note**

`print()` and `tidy()` are not yet implemented for the class of this object. `recovery_cv` has no `tidy_radf_cv` method. The objects from `radf_sign_cv()` and `radf_tt_cv()` fall back to the method of `mc_cv`, but that fallback does not apply here, because this object carries only `bsadf_cv` and not the `adf_cv`, `sadf_cv` and `gsadf_cv` fields that the method expects. Inspect `cv$bsadf_cv` directly. Calling `print(cv)` or `tidy(cv)` currently gives an error.

See Also

[radf_recovery](#), [radf_mc_cv](#)

Other critical values: [radf_common_cv\(\)](#), [radf_mc_cv\(\)](#), [radf_sb_cv\(\)](#), [radf_sbz_cv\(\)](#), [radf_sign_cv\(\)](#), [radf_sign_dm_cv\(\)](#), [radf_tt_cv\(\)](#), [radf_wb_cv\(\)](#), [radf_wb_ps_cv\(\)](#)

Examples

```
cv <- radf_recovery_cv(n = 100, minw = 20, nrep = 200)
range(cv$bsadf_cv)
```

radf_sb_cv

Panel Sieve Bootstrap Critical Values

Description

`radf_sb_cv` computes critical values for the panel recursive unit root test with the sieve bootstrap procedure of Pavlidis et al. (2016). `radf_sb_distr` computes the distribution.

Usage

```
radf_sb_cv(
  data,
  minw = NULL,
  lag = 0L,
  nboot = 500L,
  type = c("fixed", "aic", "bic"),
  max_lag = 8L,
  seed = NULL
)

radf_sb_distr(
  data,
  minw = NULL,
```

```

lag = 0L,
nboot = 500L,
type = c("fixed", "aic", "bic"),
max_lag = 8L,
seed = NULL
)

```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
lag	A non-negative integer. The lag length of the Augmented Dickey-Fuller regression (default = 0L).
nboot	A positive integer. Number of bootstraps (default = 500L).
type	Lag-order selection. "fixed" (default) uses lag as given, as in the single-lag behavior of radf . "aic" and "bic" select the lag automatically for each series with lag_select() (internal), and the function takes the maximum across the panel because the rest of it assumes one common lag order. This is the fix of Pedersen & Schütte (2020) for the size distortion that a fixed lag causes under autocorrelated innovations.
max_lag	Maximum lag order to search over when type is "aic" or "bic". It is ignored when type = "fixed".
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to set.seed before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns .Random.seed as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Value

For radf_sb_cv, a list with the critical values for the panel BSADF and panel GSADF test statistics.
 For radf_sb_distr, a numeric vector with the distribution of the panel GSADF statistic.

References

Pavlidis, E., Yusupova, A., Paya, I., Peel, D., Martínez-García, E., Mack, A., & Grossman, V. (2016). Episodes of exuberance in housing markets: In search of the smoking gun. *The Journal of Real Estate Finance and Economics*, 53(4), 419-449. doi:10.1007/s1114601595312

Pedersen, T. Q., & Schütte, E. C. M. (2020). Testing for explosive bubbles in the presence of autocorrelated innovations. *Journal of Empirical Finance*, 58, 207-225.

See Also

[radf_mc_cv](#) for Monte Carlo critical values and [radf_wb_cv](#) for wild Bootstrap critical values

Other critical values: [radf_common_cv\(\)](#), [radf_mc_cv\(\)](#), [radf_recovery_cv\(\)](#), [radf_sbz_cv\(\)](#), [radf_sign_cv\(\)](#), [radf_sign_dm_cv\(\)](#), [radf_tt_cv\(\)](#), [radf_wb_cv\(\)](#), [radf_wb_ps_cv\(\)](#)

Examples

```
rsim_data <- radf(sim_data, lag = 1)

# Critical values should have the same lag length as \code{radf()}
sb <- radf_sb_cv(sim_data, lag = 1)

tidy(sb)

summary(rsim_data, cv = sb)

autoplot(rsim_data, cv = sb)

# Simulate distribution
sdist <- radf_sb_distr(sim_data, lag = 1, nboot = 1000)

autoplot(sdist)

# Automatic BIC lag selection instead of a fixed lag
sb_bic <- radf_sb_cv(sim_data, type = "bic")
```

radf_sbz

WLS/Kernel-Volatility Bubble Statistic (SBZ)

Description

`radf_sbz` computes the WLS (kernel-volatility-weighted) recursive sup-ADF statistic of Harvey, Leybourne & Zu (2019), called supBZ in their notation, with `wls_dfstat_grid()` (internal). It returns the same shape as [radf](#): the scalars `adf`, `sadf` and `gsadf` plus the full recursive paths `badf` and `bsadf`. The result therefore carries the `radf_obj` class, and the full `summary()`, [datestamp](#), `tidy` and `autoplot` pipeline works with it when it is paired with [radf_sbz_cv](#).

Usage

```
radf_sbz(data, minw = NULL, kernel = c("gaussian", "uniform"), h = NULL)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
kernel	Kernel for the spot-volatility estimator (eq. 6 of Harvey, Leybourne & Zu 2019), "gaussian" (default, as in the paper) or "uniform".
h	Bandwidth for the spot-volatility estimator. The default is leave-one-out cross-validation over the search range of the paper.

Details

The bundled [radf_sbz_union](#) combines this statistic with the classic supDF statistic into a bootstrap-calibrated union test. supBZ alone needs a bootstrap only to be tested and not to be *defined*, so it splits into a statistic and a critical-value function, as most of exuber does.

Value

An object of class `radf_sbz_obj/radf_obj`: a list with `adf`, `sadf` and `gsadf` (one value per series) and `badf` and `bsadf` (matrices, one column per series).

Status

[Experimental]

Note

The test needs the critical values from [radf_sbz_cv](#), and neither [radf_wb_cv](#) nor [radf_mc_cv](#) applies. The null distribution of supBZ depends on the WLS weighting, so it needs its own critical-value function, which is data-dependent and uses a wild bootstrap, for the same reason as [radf\(\)](#) and [radf_wb_cv](#).

References

Harvey, D. I., Leybourne, S. J., & Zu, Y. (2019). Testing explosive bubbles with time-varying volatility. *Econometric Reviews*, 38(10), 1131-1151.

See Also

[radf_sbz_cv](#) for critical values, and [radf_sbz_union](#) for the main bootstrap union-of-rejections test of the paper, against the classic supDF statistic.

Other volatility-robust tests: [cusum_test\(\)](#), [radf_kp\(\)](#), [radf_sbz_union\(\)](#), [radf_sign\(\)](#), [radf_sign_dm\(\)](#), [radf_tt\(\)](#), [ssu_test\(\)](#)

Examples

```
# Volatility triples at t = 100, then a strong explosive regime (rho = 1.03)
# runs from t = 120 to the sample end. The kernel-volatility weighting of supBZ
# costs enough power that the milder default bubble of sim_psy1() does not clear it
y <- sim_psy1(n = 200, te = 120, tf = 200, c = 0.03, alpha = 0, seed = 1,
  e = sim_vol_break(199))
res <- radf_sbz(y, minw = 20)
print(res)

cv <- radf_sbz_cv(y, minw = 20, nboot = 200, seed = 1)
summary(res, cv = cv)
tidy(res, cv = cv)
datestamp(res, cv = cv)
autoplot(res, cv = cv)
```

radf_sbz_cv

Wild Bootstrap Critical Values for the SBZ Statistic

Description

radf_sbz_cv performs the HLST (2016) wild bootstrap. It is the same algorithm as [radf_wb_cv](#), applied to the WLS/kernel-volatility statistic of [radf_sbz](#) instead of the classic supDF statistic. It generates critical values that include the time-varying badf_cv and bsadf_cv boundary that [datestamp](#) and [autoplot](#) need, and not only the three scalar critical values that [summary\(\)](#) uses.

Usage

```
radf_sbz_cv(
  data,
  minw = NULL,
  nboot = 499L,
  kernel = c("gaussian", "uniform"),
  h = NULL,
  seed = NULL
)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
------	--

minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
nboot	A positive integer. Number of bootstraps (default = 500L).
kernel	Kernel for the spot-volatility estimator (eq. 6 of Harvey, Leybourne & Zu 2019), "gaussian" (default, as in the paper) or "uniform".
h	Bandwidth for the spot-volatility estimator. The default is leave-one-out cross-validation over the search range of the paper.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Value

An object of class `radf_cv/sbz_cv/wb_cv`: a list with the critical values `adf_cv`, `sadf_cv` and `gsadf_cv` (one row per series) and `badf_cv` and `bsadf_cv` (one array per series, one row per recursion point).

Status

[Experimental]

References

Harvey, D. I., Leybourne, S. J., & Zu, Y. (2019). Testing explosive bubbles with time-varying volatility. *Econometric Reviews*, 38(10), 1131-1151.

See Also

[radf_sbz](#) for the statistic that this function pairs with, and [radf_sbz_union](#) for the bundled union-of-rejections test against the classic supDF statistic. You cannot obtain that test from this function and [radf_wb_cv](#) separately (see the Details of [radf_sbz_union](#) for why).

Other critical values: [radf_common_cv\(\)](#), [radf_mc_cv\(\)](#), [radf_recovery_cv\(\)](#), [radf_sb_cv\(\)](#), [radf_sign_cv\(\)](#), [radf_sign_dm_cv\(\)](#), [radf_tt_cv\(\)](#), [radf_wb_cv\(\)](#), [radf_wb_ps_cv\(\)](#)

Examples

```
y <- sim_psy1(150, seed = 1)
res <- radf_sbz(y, minw = 20)
cv <- radf_sbz_cv(y, minw = 20, nboot = 200, seed = 1)
summary(res, cv = cv)
datestamp(res, cv = cv)
```

radf_sbz_union

*SBZ Weighted Least Squares Bubble Test with Union-of-Rejections***Description**

radf_sbz_union performs the HLST (2016) wild bootstrap, the same algorithm as [radf_wb_cv](#), jointly on the classic sup-ADF statistic (supDF, that is, the sadf of `radf()`) and on the WLS/kernel-volatility statistic supBZ of Harvey, Leybourne & Zu (2019). It combines them into the union-of-rejections statistic U of the paper. supBZ can have substantially higher power than supDF under many patterns of time-varying volatility, and lower power under others, for example upward volatility trends. U is designed to capture whichever of the two is more powerful for a given series.

Usage

```
radf_sbz_union(
  data,
  minw = NULL,
  nboot = 499L,
  kernel = c("gaussian", "uniform"),
  h = NULL,
  seed = NULL
)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
nboot	A positive integer. Number of bootstraps (default = 500L).
kernel	Kernel for the spot-volatility estimator (eq. 6), "gaussian" (default, as in the paper) or "uniform".
h	Bandwidth for the spot-volatility estimator. The default is leave-one-out cross-validation over the search range of the paper.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

The value of U , and not only its significance, is defined with a bootstrap-calibrated scaling ratio between the 95% supDF and supBZ (Section 2.3 of the paper). The union also keeps its size guarantee (Theorem 3 of the paper) only if the joint bootstrap computes supDF and supBZ from the *same* resampled series in each replication. This coupling is why the function stays a single bundled function, and does not split into a statistic and a critical-value function as most of exuber does. supBZ alone has no such coupling, so it does split. See [radf_sbz](#) and [radf_sbz_cv](#) for the route that uses only supBZ, with the usual `summary()`, `datestamp`, `tidy` and `autoplot` pipeline.

Value

A list with bootstrap p-values (`p_supDF`, `p_supBZ`, `p_U`) and critical values (`supDF_cv`, `supBZ_cv`, `U_cv`) for each series.

Status

[Experimental]

Note

This function bundles the statistic and its critical values in a single call. Unlike `radf()` and `radf_wb_cv()`, there is no separate statistic function without critical values and no other critical-value function to pair it with, because the value of U requires the bootstrap by construction (see Details).

The function returns its own class and not `radf_obj`, so it does not work with `summary()`, `\link{datestamp}` and `tidy`. It has its own `print()` and `autoplot()` methods instead. `print()` shows the test statistics together with their critical values, because the object bundles both. The `autoplot()` method compares supDF, supBZ and U with their critical values for each series. See `vignette("naming-and-analysis", package = "exuber")` for which functions fit the shared pipeline and which do not.

References

Harvey, D. I., Leybourne, S. J., & Zu, Y. (2019). Testing explosive bubbles with time-varying volatility. *Econometric Reviews*, 38(10), 1131-1151.

See Also

[radf_wb_cv](#) for the underlying wild bootstrap, which uses supDF only, [radf_sbz](#) and [radf_sbz_cv](#) for the route that uses supBZ only and has full pipeline support, and [radf_tt](#) for a heteroskedasticity-robust alternative that needs no bootstrap.

Other volatility-robust tests: [cusum_test\(\)](#), [radf_kp\(\)](#), [radf_sbz\(\)](#), [radf_sign\(\)](#), [radf_sign_dm\(\)](#), [radf_tt\(\)](#), [ssu_test\(\)](#)

Examples

```
y <- sim_psy1(n = 200, te = 120, tf = 200, c = 0.03, alpha = 0, seed = 1,
  e = sim_vol_break(199))
res <- radf_sbz_union(y, nboot = 200, seed = 1)
print(res)
```

```
autoplot(res)
```

radf_sign

Sign-Based Bubble Test (sPWY / sPSY)

Description

radf_sign computes the sign-based variant of the recursive right-tailed unit root test of Harvey, Leybourne & Zu (2020). Instead of applying the (double-)supremum ADF test to the series itself, it applies it to the cumulated sign of the first differences, $C_t = \text{sum}(\text{sign}(\text{diff}(y)))$. The `sign()` function removes all information about magnitudes, so the recursive DF statistic of C_t is *exactly* invariant to the pattern of volatility in the innovations, even when volatility changes over time. Unlike `radf`, the test needs no wild bootstrap to control its size under heteroskedasticity. The critical values in `radf_sign_cv` are pivotal, so they are computed once and not for each dataset.

Usage

```
radf_sign(data, minw = NULL)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in <code>badf</code> and <code>bsadf</code> and excluded from the <code>adf</code> , <code>sadf</code> and <code>gsadf</code> of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (<code>bsadf_panel</code> and <code>gsadf_panel</code>) are not available, and the function returns NA for them with a warning.
minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).

Details

The price of this invariance is power. The paper finds that the sign-based test outperforms the standard PSY test for many specifications of time-varying volatility and bubbles, but not for all of them, and the standard test can still win in some. The strategy that the paper recommends in practice is a bootstrap-based union of rejections that combines both tests. We have not implemented it (see the package's enhancement notes for the cost and benefit considerations), and this function provides the standalone sign-based test only. `sadf` is the single-supremum sPWY statistic ($r_1 = 0$ fixed), and `gsadf` is the double-supremum sPSY statistic.

Value

An object of class `radf_sign_obj/radf_obj`. It is the same `adf/badf/sadf/bsadf/gsadf` list as for `radf`, computed on the sign-transformed series, and it pairs with `radf_sign_cv`.

Level-shift robustness

Harvey, Leybourne, Tatlow & Zu (2025) show that this test keeps its standard null distribution, the one without level shifts, in the presence of deterministic level shifts, provided that the number of shifts grows strictly more slowly than $\sqrt{T}$. The size of the shifts does not matter. This is a materially weaker requirement than the one the standard PSY test needs for size control, which restricts the number **and** the magnitude of the shifts jointly. In their simulations the standard test is never correctly sized once the number of shifts grows at rate $\sqrt{T}$, while this test stays close to its nominal size.

Status

[Experimental]

Note

The test needs the critical values from `radf_sign_cv`, and neither `radf_wb_cv` nor any other bootstrap applies. The statistic is pivotal (exactly invariant to heteroskedasticity), so its critical values are simulated once and not for each dataset.

The result carries the `radf_obj` class. Since 2026-08-18 the full `summary()`, `datestamp`, `tidy` and `autoplot` pipeline works, because `radf_sign_cv()` now computes the time-varying `badf_cv` and `bsadf_cv` boundary that the last two need, and not only the three scalar critical values that `summary()` uses. See `vignette("naming-and-analysis", package = "exuber")`.

References

Harvey, D. I., Leybourne, S. J., & Zu, Y. (2020). Sign-based unit root tests for explosive financial bubbles in the presence of deterministically time-varying volatility. *Econometric Theory*, 36(1), 122-169.

Harvey, D. I., Leybourne, S. J., Tatlow, D., & Zu, Y. (2025). Unit root tests for explosive financial bubbles in the presence of deterministic level shifts. *Oxford Bulletin of Economics and Statistics*, 87(5), 879-901. doi:10.1111/obes.12668

See Also

`radf_sign_cv` for critical values, `radf_sign_dm` for the recursively demeaned sign-based analogue, which has the same level-shift robustness, and `radf` for the standard test, which is not invariant.

Other volatility-robust tests: `cusum_test()`, `radf_kp()`, `radf_sbz()`, `radf_sbz_union()`, `radf_sign_dm()`, `radf_tt()`, `ssu_test()`

Examples

```
# Volatility triples half-way through the sample. This is the case of
# non-stationary volatility that this test is built for, and plain radf()
# over-rejects here
y <- sim_psy1(n = 200, seed = 1, e = sim_vol_break(199))
res <- radf_sign(y, minw = 20)
print(res)
```

```

cv <- radf_sign_cv(n = 200, minw = 20)
summary(res, cv = cv)
tidy(res, cv = cv)
datestamp(res, cv = cv)
autoplot(res, cv = cv)

```

radf_sign_cv

Monte Carlo Critical Values for the Sign-Based Test

Description

Simulates the asymptotic null distribution of the statistic of [radf_sign](#). By Theorem 2 of Harvey, Leybourne & Zu (2020), this distribution does not depend on the volatility process at all (exact invariance). Like [radf_tt_cv](#), and unlike [radf_wb_cv](#), it therefore does not have to be recomputed for each dataset. A large n with the default $nrep$ approximates the $T \rightarrow \infty$ limit of the paper.

Usage

```
radf_sign_cv(n, minw = NULL, nrep = 2000L, seed = NULL)
```

Arguments

<code>n</code>	A positive integer. The sample size.
<code>minw</code>	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
<code>nrep</code>	A positive integer. The number of Monte Carlo simulations.
<code>seed</code>	An object specifying if and how the random number generator (rng) should be initialized. It is either <code>NULL</code> or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, <code>NULL</code> , leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

You can check `sadf_cv` (single-supremum, $r1 = 0$ fixed) against the asymptotic ($T = \infty$) sPWY values in Table 1 of the paper. For $minw/n = 0.1$, the values at $(10 \backslash gsadf_cv$ (double-supremum) corresponds to the sPSY row: (2.933, 3.180, 3.655).

Value

An object of class `radf_cv/sign_cv/mc_cv` with the same structure as [radf_mc_cv](#).

Status

[Experimental]

References

Harvey, D. I., Leybourne, S. J., & Zu, Y. (2020). Sign-based unit root tests for explosive financial bubbles in the presence of deterministically time-varying volatility. *Econometric Theory*, 36(1), 122-169.

See Also

Other critical values: [radf_common_cv\(\)](#), [radf_mc_cv\(\)](#), [radf_recovery_cv\(\)](#), [radf_sb_cv\(\)](#), [radf_sbz_cv\(\)](#), [radf_sign_dm_cv\(\)](#), [radf_tt_cv\(\)](#), [radf_wb_cv\(\)](#), [radf_wb_ps_cv\(\)](#)

Examples

```
cv <- radf_sign_cv(n = 200, minw = 20)
tidy(cv)
```

radf_sign_dm	<i>Recursively Demeaned Sign-Based Bubble Test ($\bar{s}PWY$ / $\bar{s}PSY$)</i>
--------------	--

Description

`radf_sign_dm` computes the second sign-based analogue of the recursive right-tailed unit root test of Harvey, Leybourne & Zu (2020), which the paper denotes $\bar{s}PWY/\bar{s}PSY$. The construction is the same as in [radf_sign](#), but it is built on a recursively (expanding-window) demeaned cumulated-sign series, $C_{tilde_t} = \sum_{i=2}^t \{ \text{sign}(\text{diff}(y)_i) - \text{mean}(\text{sign}(\text{diff}(y)_{2:i})) \}$, and not on the raw cumulated sign that `radf_sign` uses.

Usage

```
radf_sign_dm(data, minw = NULL)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in <code>badf</code> and <code>bsadf</code> and excluded from the <code>adf</code> , <code>sadf</code> and <code>gsadf</code> of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (<code>bsadf_panel</code> and <code>gsadf_panel</code>) are not available, and the function returns NA for them with a warning.
minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).

Details

Harvey, Leybourne, Tatlow & Zu (2025) show that this statistic shares the asymptotic level-shift robustness of [radf_sign](#) (see the Level-shift robustness section of that function). It does not need Assumption 2 of the underlying HLZ (2020) theory, that the median of the innovations is zero, which is a strictly weaker requirement than the one [radf_sign](#) needs for its own invariance result. Their finite-sample simulations also find that the recursive demeaning tends to reduce the size distortion under level shifts further than [radf_sign](#) does, although both are asymptotically robust to level shifts under the same condition.

Value

An object of class `radf_sign_dm_obj/radf_obj`. It is the same `adf/badf/sadf/bsadf/gsadf` list as for [radf](#), and it pairs with [radf_sign_dm_cv](#).

Status

[Experimental]

Note

The test needs the critical values from [radf_sign_dm_cv](#), and [radf_sign_cv](#) does not apply, because it is calibrated to the non-demeaned [radf_sign](#) statistic. The statistic is pivotal like [radf_sign](#), so no bootstrap is needed for each dataset.

The result carries the `radf_obj` class. Since 2026-08-18 the full `summary()`, `datestamp`, `tidy` and `autoplot` pipeline works, with the same fix as for [radf_sign](#): `radf_sign_dm_cv()` now also computes `badf_cv` and `bsadf_cv`, and not only the three scalar critical values. See `vignette("naming-and-analysis", package = "exuber")`.

References

- Harvey, D. I., Leybourne, S. J., & Zu, Y. (2020). Sign-based unit root tests for explosive financial bubbles in the presence of deterministically time-varying volatility. *Econometric Theory*, 36(1), 122-169.
- Harvey, D. I., Leybourne, S. J., Tatlow, D., & Zu, Y. (2025). Unit root tests for explosive financial bubbles in the presence of deterministic level shifts. *Oxford Bulletin of Economics and Statistics*, 87(5), 879-901. doi:10.1111/obes.12668

See Also

[radf_sign_dm_cv](#) for critical values, and [radf_sign](#) for the non-demeaned sign-based analogue.

Other volatility-robust tests: [cusum_test\(\)](#), [radf_kp\(\)](#), [radf_sbz\(\)](#), [radf_sbz_union\(\)](#), [radf_sign\(\)](#), [radf_tt\(\)](#), [ssu_test\(\)](#)

Examples

```
# Volatility triples half-way through the sample. This is the case of
# non-stationary volatility that this test is built for, and plain radf()
# over-rejects here
y <- sim_psy1(n = 200, seed = 1, e = sim_vol_break(199))
```

```

res <- radf_sign_dm(y, minw = 20)
print(res)

cv <- radf_sign_dm_cv(n = 200, minw = 20)
summary(res, cv = cv)
tidy(res, cv = cv)
datestamp(res, cv = cv)
autoplot(res, cv = cv)

```

radf_sign_dm_cv	<i>Monte Carlo Critical Values for the Recursively Demeaned Sign-Based Test</i>
-----------------	---

Description

Simulates the asymptotic null distribution of the statistic of [radf_sign_dm](#). As for [radf_sign_cv](#), this distribution does not depend on the volatility process (exact invariance, the analogue for this variant of Theorem 2 of HLZ 2020), so it does not have to be recomputed for each dataset.

Usage

```
radf_sign_dm_cv(n, minw = NULL, nrep = 2000L, seed = NULL)
```

Arguments

n	A positive integer. The sample size.
minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
nrep	A positive integer. The number of Monte Carlo simulations.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Value

An object of class `radf_cv/sign_dm_cv/mc_cv` with the same structure as [radf_mc_cv](#).

Status

[Experimental]

References

Harvey, D. I., Leybourne, S. J., & Zu, Y. (2020). Sign-based unit root tests for explosive financial bubbles in the presence of deterministically time-varying volatility. *Econometric Theory*, 36(1), 122-169.

See Also

Other critical values: [radf_common_cv\(\)](#), [radf_mc_cv\(\)](#), [radf_recovery_cv\(\)](#), [radf_sb_cv\(\)](#), [radf_sbz_cv\(\)](#), [radf_sign_cv\(\)](#), [radf_tt_cv\(\)](#), [radf_wb_cv\(\)](#), [radf_wb_ps_cv\(\)](#)

Examples

```
cv <- radf_sign_dm_cv(n = 200, minw = 20)
tidy(cv)
```

radf_tt	<i>Time-Transformed Test for Explosive Bubbles under Non-stationary Volatility</i>
---------	--

Description

radf_tt computes the STADF and GSTADF test statistics of Kurozumi, Skrobotov & Tsarev, a heteroskedasticity-robust alternative to [radf](#) that needs no bootstrap. It time-deforms the series with a nonparametric estimate of its variance profile, after which the usual asymptotic recursive sup-ADF critical values for homoskedastic errors apply.

Usage

```
radf_tt(data, minw = NULL, kernel = c("uniform", "gaussian"), h = NULL)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
kernel	Kernel used in the local variance-profile regression, "uniform" (default, as in the simulations of the paper) or "gaussian".

`h` Bandwidth for the variance-profile kernel regression. The default is $T^{-2/5}$, the midpoint on the log scale of the cross-validation search range $[T^{-0.5}, T^{-0.3}]$ of the paper.

Details

We recommend `radf_tt_cv` for the critical values. They are pivotal (asymptotically free of the volatility process), so they do not have to be recomputed for each dataset, unlike a bootstrap. The wild bootstrap of Harvey, Leybourne, Sollis & Taylor in `radf_wb_cv` is a bootstrap-based alternative. Consider it if non-pivotality or the finite-sample robustness of the bootstrap is a specific concern.

Value

An object of class `radf_tt_obj/radf_obj`: the same `adf/badf/sadf/bsadf/gsadf` list as for `radf`, computed on the time-transformed series. It works with `summary()`, `datestamp()`, `tidy()` and `autoplot()` when paired with `radf_tt_cv`.

Status

[Experimental]

Note

The result carries the `radf_obj` class. Since 2026-08-18 the full `summary()`, `datestamp`, `tidy` and `autoplot` pipeline works, because `radf_tt_cv()` now computes the time-varying `badf_cv` and `bsadf_cv` boundary that the last two need, and not only the three scalar critical values that `summary()` uses. See `vignette("naming-and-analysis", package = "exuber")`.

References

Kurozumi, E., Skrobotov, A., & Tsarev, A. (2024). Time-Transformed Test for Bubbles under Non-stationary Volatility. *Journal of Financial Econometrics*. doi:10.1093/jfinec/nbae026

See Also

`radf_tt_cv` for the pivotal asymptotic critical values that need no bootstrap, and `radf_wb_cv` for the bootstrap-based alternative (Harvey, Leybourne, Sollis & Taylor).

Other volatility-robust tests: `cusum_test()`, `radf_kp()`, `radf_sbz()`, `radf_sbz_union()`, `radf_sign()`, `radf_sign_dm()`, `ssu_test()`

Examples

```
# Volatility triples half-way through the sample. This is the case of
# non-stationary volatility that this test is built for, and plain radf()
# over-rejects here
y <- sim_psy1(n = 200, seed = 1, e = sim_vol_break(199))
res <- radf_tt(y, minw = 20)
print(res)
```

```

cv <- radf_tt_cv(n = 200, minw = 20)
summary(res, cv = cv)
tidy(res, cv = cv)
datestamp(res, cv = cv)
autoplot(res, cv = cv)

```

radf_tt_cv	<i>Monte Carlo critical values for the time-transformed test (STADF/GSTADF)</i>
------------	---

Description

This is the dedicated critical-value function for [radf_tt](#). It simulates the asymptotic null distribution of the GLS-demeaned recursive sup-ADF statistic that [radf_tt](#) uses. By Theorem 1 of Kurozumi, Skrobotov & Tsarev, this distribution is free of the volatility process (pivotal). Unlike [radf_wb_cv](#), it therefore does not have to be recomputed for each dataset. A large n with the default $nrep$ approximates well the $T \rightarrow \infty$ limit used in the paper.

Usage

```
radf_tt_cv(n, minw = NULL, nrep = 2000L, seed = NULL)
```

Arguments

<code>n</code>	A positive integer. The sample size.
<code>minw</code>	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
<code>nrep</code>	A positive integer. The number of Monte Carlo simulations.
<code>seed</code>	An object specifying if and how the random number generator (rng) should be initialized. It is either <code>NULL</code> or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, <code>NULL</code> , leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

You can check the `sadf_cv` column (STADF, that is, $r_1 = 0$ fixed) against the asymptotic values of Whitehouse (2019) quoted in footnote 4 of Kurozumi, Skrobotov & Tsarev. For $minw/n = 0.1$, the values at $(10 \setminus 1) \setminus GSTADF$ (`gsadf_cv`). The paper gives its GSTADF critical values not as numbers in the text but only as "easily computed from" the code of the authors in R.

Value

An object of class `radf_cv/tt_cv/mc_cv` with the same structure as [radf_mc_cv](#): the scalars `adf_cv`, `sadf_cv` and `gsadf_cv` for each level plus the sequences `badf_cv` and `bsadf_cv`. You can use it wherever a `radf_cv` is accepted.

Status**[Experimental]****Note**

Since 2026-08-18 the function also computes `badf_cv` and `bsadf_cv`, a time-varying boundary with one row for each recursion point. `datestamp` and `autoplot` therefore work on `radf_tt` results, and not only `summary()` and `tidy`. The `bsadf_cv` of `radf_mc_cv` is a `cummax()` across replicates, which works around the output shape of the base C++ engine. Here no such shortcut is needed, because the `bsadf` of `gls_dfstat_grid()` (internal) is already the sup over all window starts at each point, and the boundary is the per-time-point quantile across replicates. We validated the function in three ways. The last row of `badf_cv` is bit-identical to `adf_cv`, an exact identity because `adf` is the last point of `badf` in every replicate. The empirical false-alarm rate under H_0 is conservative relative to the nominal level (3.3% power on a synthetic bubble matches the established `radf()` and `radf_mc_cv()` pipeline almost exactly (18% series. `radf_sign_cv()` and `radf_sign_dm_cv()` now compute the same boundary (see `vignette("naming-and-analysis")`)).

References

Kurozumi, E., Skrobotov, A., & Tsarev, A. (2024). Time-Transformed Test for Bubbles under Non-stationary Volatility. *Journal of Financial Econometrics*. doi:10.1093/jfinec/nbae026

See Also

Other critical values: `radf_common_cv()`, `radf_mc_cv()`, `radf_recovery_cv()`, `radf_sb_cv()`, `radf_sbz_cv()`, `radf_sign_cv()`, `radf_sign_dm_cv()`, `radf_wb_cv()`, `radf_wb_ps_cv()`

Examples

```
cv <- radf_tt_cv(n = 200, minw = 20)
tidy(cv)

# Volatility triples half-way through the sample. This is the case of
# non-stationary volatility that this test is built for, and plain radf()
# over-rejects here
y <- sim_psy1(n = 200, seed = 1, e = sim_vol_break(199))
res <- radf_tt(y, minw = 20)
datestamp(res, cv = cv)
autoplot(res, cv = cv)
```

Description

radf_wb_cv generates critical values for the recursive unit root tests with the wild bootstrap of Harvey et al. (2016), which is asymptotically robust to non-stationary volatility. radf_wb_distr computes the distribution.

Usage

```
radf_wb_cv(
  data,
  minw = NULL,
  nboot = 500L,
  dist_rad = FALSE,
  dist_skew = FALSE,
  seed = NULL
)

radf_wb_distr(
  data,
  minw = NULL,
  nboot = 500L,
  dist_rad = FALSE,
  dist_skew = FALSE,
  seed = NULL
)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
minw	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
nboot	A positive integer. Number of bootstraps (default = 500L).
dist_rad	Logical. If TRUE then the Rademacher distribution will be used.
dist_skew	Logical. If TRUE, use the fixed right-skewed multiplier distribution of Hafner (2020) instead of the default standard normal or (dist_rad = TRUE) Rademacher one. It is appropriate when the return distribution of the series is itself clearly right-skewed, as for the cryptocurrency returns in the application of that paper. At most one of dist_rad and dist_skew may be TRUE.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to set.seed before the simulation. If you set it, the value is saved as the "seed" attribute of the

returned value. The default, NULL, leaves the state of the rng unchanged and returns `.Random.seed` as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

The function applies a wild bootstrap re-sampling scheme to construct the bootstrap analogue of the test of Phillips et al. (2015). The bootstrap test is asymptotically robust to non-stationary volatility.

Value

For `radf_wb_cv`, a list with the critical values for the ADF, BADF, BSADF and GSADF tests. For `radf_wb_distr`, a list with the ADF, SADF and GSADF distributions.

References

- Harvey, D. I., Leybourne, S. J., Sollis, R., & Taylor, A. M. R. (2016). Tests for explosive financial bubbles in the presence of non-stationary volatility. *Journal of Empirical Finance*, 38(Part B), 548-574.
- Phillips, P. C. B., Shi, S., & Yu, J. (2015). Testing for Multiple Bubbles: Historical Episodes of Exuberance and Collapse in the S&P 500. *International Economic Review*, 56(4), 1043-1078. [doi:10.1111/iere.12132](https://doi.org/10.1111/iere.12132)
- Hafner, C. M. (2020). Testing for bubbles in cryptocurrencies with time-varying volatility. *Journal of Financial Econometrics*, 18(2), 233-249.

See Also

[radf_mc_cv](#) for Monte Carlo critical values and [radf_sb_cv](#) for sieve bootstrap critical values.

Other critical values: [radf_common_cv\(\)](#), [radf_mc_cv\(\)](#), [radf_recovery_cv\(\)](#), [radf_sb_cv\(\)](#), [radf_sbz_cv\(\)](#), [radf_sign_cv\(\)](#), [radf_sign_dm_cv\(\)](#), [radf_tt_cv\(\)](#), [radf_wb_ps_cv\(\)](#)

Examples

```
# Volatility triples half-way through the sample. This is the case of
# non-stationary volatility that the wild bootstrap is built for, and plain
# radf_mc_cv() over-rejects here
y <- sim_psy1(n = 200, seed = 1, e = sim_vol_break(199))
# Default minimum window
wb <- radf_wb_cv(y)

tidy(wb)

# Change the minimum window and the number of bootstraps
wb2 <- radf_wb_cv(y, nboot = 600, minw = 20)

tidy(wb2)

# Simulate distribution
wdist <- radf_wb_distr(y)
```

```

autoplot(wdist)

# Apply the critical values to actual data
rsim_data <- radf(y, minw = 20)
autoplot(rsim_data, cv = wb2)

```

radf_wb_ps_cv

Wild Bootstrap Critical Values (Phillips & Shi 2020)

Description

radf_wb_ps_cv generates critical values for the recursive unit root tests with the wild bootstrap of Phillips & Shi (2020). The scheme fits a null AR model and resamples its residuals, and it is asymptotically robust to non-stationary volatility. radf_wb_ps_distr computes the distribution. In contrast to the non-parametric multiplier bootstrap of Harvey et al. (2016) in [radf_wb_cv](#), this version supports a training-window boundary (tb), which is what [monitor](#) uses it for.

Usage

```

radf_wb_ps_cv(
  data,
  minw = NULL,
  nboot = 500L,
  adflag = 0,
  type = c("fixed", "aic", "bic"),
  tb = NULL,
  seed = NULL
)

radf_wb_ps_distr(
  data,
  minw = NULL,
  nboot = 500L,
  adflag = 0,
  type = c("fixed", "aic", "bic"),
  tb = NULL,
  seed = NULL
)

```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way,
------	---

	the panel statistics (<code>bsadf_panel</code> and <code>gsadf_panel</code>) are not available, and the function returns NA for them with a warning.
<code>minw</code>	A positive integer. The minimum window size (default = $(0.01 + 1.8/\sqrt{T})T$, where T denotes the sample size).
<code>nboot</code>	A positive integer. Number of bootstraps (default = 500L).
<code>adflag</code>	A positive integer. Number of lags when type is "fixed" or number of max lags when type is either "aic" or "bic".
<code>type</code>	Character. "fixed" for fixed lag, "aic" or "bic" for automatic lag selection according to the criterion.
<code>tb</code>	A positive integer. The simulated sample size.
<code>seed</code>	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Value

For `radf_wb_ps_cv`, a list with the critical values for the ADF, BADF, BSADF and GSADF tests.
 For `radf_wb_ps_distr`, a list with the ADF, SADF and GSADF distributions.

References

Phillips, P. C., & Shi, S. (2020). Real time monitoring of asset markets: Bubbles and crises. In *Handbook of Statistics* (Vol. 42, pp. 61-80). Elsevier.

Phillips, P. C. B., Shi, S., & Yu, J. (2015). Testing for Multiple Bubbles: Historical Episodes of Exuberance and Collapse in the S&P 500. *International Economic Review*, 56(4), 1043-1078. [doi:10.1111/iere.12132](https://doi.org/10.1111/iere.12132)

See Also

[radf_wb_cv](#) for the Harvey et al. (2016) wild bootstrap, [radf_mc_cv](#) for Monte Carlo critical values and [radf_sb_cv](#) for sieve bootstrap critical values.

Other critical values: [radf_common_cv\(\)](#), [radf_mc_cv\(\)](#), [radf_recovery_cv\(\)](#), [radf_sb_cv\(\)](#), [radf_sbz_cv\(\)](#), [radf_sign_cv\(\)](#), [radf_sign_dm_cv\(\)](#), [radf_tt_cv\(\)](#), [radf_wb_cv\(\)](#)

Examples

```
# Default minimum window
wb <- radf_wb_ps_cv(sim_data)

tidy(wb)

# Change the minimum window and the number of bootstraps
wb2 <- radf_wb_ps_cv(sim_data, nboot = 600, minw = 20)

tidy(wb2)
```

```
# Simulate distribution
wdist <- radf_wb_ps_distr(sim_data)

autoplot(wdist)

# Apply the critical values to actual data
rsim_data <- radf(sim_data, minw = 20)
autoplot(rsim_data, cv = wb2)
```

rootstamp

Confidence Interval and Doubling Time for an Explosive Root

Description

Fits a no-intercept AR(1) regression $y_t = \rho y_{t-1} + \epsilon_t$ (Phillips & Magdalinos 2007, who omit the intercept in their eq. 58 "to exclude the presence of a deterministically explosive component") and reports $\hat{\rho}$ together with a confidence interval and the implied **doubling time** $\log(2)/\log(\hat{\rho})$, which is the number of periods the series needs to double in magnitude at the estimated growth rate.

Usage

```
rootstamp(object, ...)

## Default S3 method:
rootstamp(object, sig_lvl = 95, type = c("normal", "cauchy"), ...)

## S3 method for class 'radf_obj'
rootstamp(object, ds, sig_lvl = 95, type = c("normal", "cauchy"), ...)
```

Arguments

object	For the default method, a numeric vector with the sub-sample to fit, already sliced to the episode of interest. For the <code>radf_obj</code> method, the <code>radf_obj</code> on which <code>ds</code> was computed.
...	Further arguments passed to methods.
sig_lvl	Confidence level of the interval on the 0 to 100 scale used throughout the package (default 95). Any value in $[50, 100)$ is allowed.
type	"normal" (default) for the normal-t interval of Guo, Sun & Wang, or "cauchy" for the fixed-root Cauchy interval of Phillips & Magdalinos.
ds	(<code>radf_obj</code> method only) A datestamp result computed on object. Root inference on a very short episode is statistically meaningless. Set <code>min_duration</code> in that datestamp call to exclude episodes that are too short for reliable root inference, because this method does not decide what counts as too short.

Details

Guo, Sun & Wang (2019) show that the ordinary t-statistic for $\hat{\rho}$, estimated by OLS with no intercept, is asymptotically **standard normal** under i.i.d. errors, and also under weakly dependent errors when a HAC standard error is used. This differs from the classical stationary and unit-root cases. An ordinary-looking Wald interval, $\hat{\rho} \pm z_{\alpha/2} \cdot se(\hat{\rho})$, is therefore asymptotically valid here even though $\hat{\rho} > 1$. The interval has the same form as a classical normal-theory interval, which would be invalid for an explosive root, but the justification is different: it rests on the explosive-root central limit theorem of Guo, Sun & Wang and not on the classical stationary one.

type = "cauchy" instead uses the fixed-root result of Phillips & Magdalinos (2007, their eq. 27, which restates White 1958). For an explosive root that does not drift, $\frac{\hat{\rho}^n}{\rho^2 - 1}(\hat{\rho} - \rho)$ converges to a standard Cauchy variate. We replace the unknown ρ in the normalization with $\hat{\rho}$, as is usual for this kind of self-normalized pivot, and obtain $\hat{\rho} \pm q_{\alpha/2} \cdot (\hat{\rho}^2 - 1)/\hat{\rho}^n$, where $q_{\alpha/2}$ is a standard-Cauchy quantile. This interval assumes a *fixed* explosive root, with no drift and no unknown localizing rate. When that assumption is in doubt, the default "normal" type is safer, because the result of Guo, Sun & Wang allows for drift and weak dependence.

There are two methods, for two different starting points:

- **Default:** object is a numeric vector, the sub-sample to fit. It can be an episode that you sliced out by hand or by position from a [datestamp](#) result (y[from: to]). The method fits the sub-sample once and returns one confidence interval.
- **radf_obj:** object is the radf_obj on which a [datestamp](#) result ds was computed. The method runs the default method once for each datestamped episode of each series and slices the data of object itself, so no manual loop is needed.

Value

The default method returns a rootstamp_est object, which is a list with rho, se, t_stat, n, rho_ci, doubling_time and doubling_time_ci and has its own print() method.

The radf_obj method returns a rootstamp_episodes object, which is a named list with one element for each series in ds. Each element is a data frame with one row for each datestamped episode and the columns Start, End, rho, rho_lower, rho_upper, doubling_time, doubling_time_lower and doubling_time_upper. The object has its own print() method. The panel sieve-bootstrap case has a ds entry named "panel" that corresponds to no single series, and it is dropped with a warning.

Status

[Experimental]

Note

Neither method returns the radf_obj class. Even the radf_obj method, which dispatches on that class for its *input*, returns its own rootstamp_episodes class. rootstamp() therefore does not work with summary(), \link{datestamp}, tidy and autoplot. See vignette("naming-and-analysis", package = "exuber") for which functions fit that pipeline and which do not.

References

- Guo, G., Sun, Y., & Wang, S. (2019). Testing for moderate explosiveness. *The Econometrics Journal*, 22(3), 279-303.
- Phillips, P. C. B., & Magdalinos, T. (2007). Limit theory for moderate deviations from a unit root. *Journal of Econometrics*, 136(1), 115-130.

See Also

Other dating: [dating_hls\(\)](#), [dating_hlw\(\)](#), [dating_knp\(\)](#), [dating_pdc\(\)](#), [radf_recovery\(\)](#)

Examples

```
# The martingale-to-explosive process of sim_psy1(), explosive through the sample end
y <- sim_psy1(n = 100, te = 60, tf = 100, seed = 2026)

r <- radf(y, minw = 20)
cv <- radf_mc_cv(length(y), minw = 20, nrep = 300, seed = 4)
ds <- datestamp(r, cv = cv, min_duration = 3)

# default method: one episode, sliced by hand
ep <- y[ds[["series1"]]]$Start[1]:ds[["series1"]]]$End[1]]
fit <- rootstamp(ep) # recovers the DGP's explosive AR coefficient
fit
rootstamp(ep, type = "cauchy")

# Plot the episode with the fitted explosive-root path overlaid
autoplot(fit)

# radf_obj method: every datestamped episode at once
res_all <- rootstamp(r, ds)
res_all

# Plot the estimated rho (with its CI) for every episode
autoplot(res_all)
```

scale_exuber_manual *Exuber scale and theme functions*

Description

scale_exuber_manual sets the color, linewidth and linetype in the mappings of autoplot.radf_obj. theme_exuber is a complete theme that controls all non-data display.

Usage

```
scale_exuber_manual(  
  color_values = c("red", "blue"),  
  linetype_values = c(2, 1),  
  linewidth_values = c(0.8, 0.7),  
  size_values = "DEPRECATED"  
)  
  
theme_exuber(  
  base_size = 11,  
  base_family = "",  
  base_line_size = base_size/22,  
  base_rect_size = base_size/22  
)
```

Arguments

`color_values` a set of color values to map data values to.

`linetype_values` a set of linetype values to map data values to.

`linewidth_values` a set of linewidth values to map data values to.

`size_values` **[Deprecated]** use `linewidth_values`.

`base_size` base font size, given in pts.

`base_family` base font family

`base_line_size` base size for line elements

`base_rect_size` base size for rect elements

Value

A list of three ggplot2 scales (`scale_exuber_manual`) or a ggplot2 theme object (`theme_exuber`), which you add to a plot with `+`.

Examples

```
rsim <- radf(sim_psy1(100))  
autoplot(rsim, cv = radf_mc_cv(100, nrep = 100)) +  
  scale_exuber_manual(color_values = c("black", "black")) +  
  theme_exuber(base_size = 9)
```

series_names	<i>Retrieve/Replace series names</i>
--------------	--------------------------------------

Description

Retrieve or replace the series names of an object.

Usage

```
series_names(x, ...)

series_names(x) <- value

## S3 replacement method for class 'radf_obj'
series_names(x) <- value

## S3 replacement method for class 'wb_cv'
series_names(x) <- value

## S3 replacement method for class 'sb_cv'
series_names(x) <- value
```

Arguments

x	An object.
...	Further arguments passed to methods.
value	An ordered vector of the same length as the "index" attribute of x.

Value

A character vector of series names. The replacement form returns the modified object.

Examples

```
rfd <- radf(sim_data)
series_names(rfd)

# Rename the series, which carries through to the facet labels of tidy() and autoplot()
series_names(rfd) <- c("Bubble A", "Bubble B", "Bubble C", "Bubble D", "Bubble E")
series_names(rfd)
tidy(rfd)
autoplot(rfd)

# Simulate bubble processes
dta <- data.frame(psy1 = sim_psy1(n = 100), psy2 = sim_psy2(n = 100))

rfd <- radf(dta)
```

```
series_names(rfd) <- c("OneBubble", "TwoBubbles")
```

sim_blan	<i>Simulation of a Blanchard (1979) / Rotermann-Wilfling (2018) bubble process</i>
----------	--

Description

Simulates the rational bubble process of Blanchard (1979) or, with `type = "rotermann_wilfling"`, the lognormal-mixture extension of Rotermann & Wilfling (2018).

Usage

```
sim_blan(
  n,
  pi = 0.7,
  sigma = 0.03,
  r = 0.05,
  b0 = 0.1,
  type = c("blanchard", "rotermann_wilfling"),
  delta = 0.984,
  rw_sigma = 0.05,
  seed = NULL
)
```

Arguments

<code>n</code>	A positive integer specifying the length of the simulated output series.
<code>pi</code>	A positive value in (0, 1) which governs the probability of the bubble continuing to grow.
<code>sigma</code>	A positive scalar indicating the standard deviation of the innovations.
<code>r</code>	A positive scalar that determines the growth rate of the bubble process.
<code>b0</code>	The initial value of the bubble.
<code>type</code>	"blanchard" (default) or "rotermann_wilfling". <code>r</code> is used only by "blanchard", and <code>delta</code> and <code>rw_sigma</code> only by "rotermann_wilfling" (see Details).
<code>delta</code>	A scalar in (0, 1), the Rotermann-Wilfling deflation parameter. Only used for <code>type = "rotermann_wilfling"</code> .
<code>rw_sigma</code>	A positive scalar, the standard deviation (on the log scale) of the Rotermann-Wilfling multiplicative lognormal shock. Only used for <code>type = "rotermann_wilfling"</code> .
<code>seed</code>	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

Blanchard's bubble process (type = "blanchard") has two regimes, which occur with probability π and $1 - \pi$. In the first regime, the bubble grows exponentially, whereas in the second regime, the bubble collapses to a white noise.

With probability π :

$$B_{t+1} = \frac{1+r}{\pi} B_t + \epsilon_{t+1}$$

With probability $1 - \pi$:

$$B_{t+1} = \epsilon_{t+1}$$

where r is a positive constant and $\epsilon \sim iid(0, \sigma^2)$.

The bubble of Rotermann & Wilfling (2018) (type = "rotermann_wilfling") replaces the "collapse to white noise" regime with a partial, stochastically evolving deflation. The trajectories recur periodically and deflate gradually instead of collapsing abruptly in one period:

$$B_t = \frac{B_{t-1} u_t}{\delta}$$

with probability π , or

$$B_t = \frac{1 - \pi\delta}{1 - \pi} B_{t-1} u_t$$

with probability $1 - \pi$, where $u_t \sim iid LN(-rw_sigma^2/2, rw_sigma^2)$ (so $E[u_t] = 1$). $\delta \in (0, 1)$ ensures that the bubble never collapses to exactly zero and can inflate again.

Value

A numeric vector of length n .

References

- Blanchard, O. J. (1979). Speculative bubbles, crashes and rational expectations. *Economics letters*, 3(4), 387-389.
- Rotermann, B. & Wilfling, B. (2018). "A new stochastic bubble process: Theoretical properties and empirical tests." *Applied Economics Letters*, 25(15), 1091-1096. As used for Monte Carlo power analysis in Monschang, V. & Wilfling, B. (2021). "Sup-ADF-style bubble-detection methods under test." *Empirical Economics*, 61, 145-172.

See Also

[sim_psy1](#), [sim_psy2](#), [sim_evans](#)

Examples

```
sim_blan(n = 100, seed = 123) %>%
  autoplot()
```

```
sim_blan(n = 250, type = "rotermann_wilfling", delta = 0.984, seed = 123) %>%
  autoplot()
```

sim_coexplosive

*Simulation of a bivariate co-explosive pair***Description**

Simulates the co-explosive process of Evripidou, Harvey, Leybourne & Sollis (2022): an explosive series x (from [sim_psy1](#)) and a second series y that is linked to a lead or lagged copy of it, and optionally to a third, independent explosive series z , $y_t = \mu_y + \phi_x x_{t-i} + \phi_z z_t + \epsilon_{y,t}$. $i > 0$ means that the explosive episode of x leads that of y , and $i < 0$ means that it lags.

Usage

```
sim_coexplosive(
  n,
  lag = 0,
  phi_x = 1,
  phi_z = 0,
  mu_y = 0,
  sigma_y = 6.79,
  x_args = list(),
  z_args = list(),
  seed = NULL
)
```

Arguments

<code>n</code>	A positive integer specifying the length of the simulated output series.
<code>lag</code>	Integer lead (> 0) or lag (< 0) of x relative to y ; $0 =$ contemporaneous.
<code>phi_x, phi_z, mu_y</code>	Linkage coefficients and intercept for y .
<code>sigma_y</code>	A non-negative scalar, the standard deviation of y 's own noise.
<code>x_args, z_args</code>	Named lists of extra arguments passed to the sim_psy1 calls generating x and (if $\text{phi}_z \neq 0$) z .
<code>seed</code>	An object specifying if and how the random number generator (rng) should be initialized. It is either <code>NULL</code> or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, <code>NULL</code> , leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Value

A data.frame with columns x and y (length n).

References

Evripidou, C., Harvey, D.I., Leybourne, S.J. & Sollis, R. (2022). "Co-explosive behaviour in explosive financial bubbles." OBES.

See Also

[sim_psy1](#)

Examples

```
x <- sim_coexplosive(n = 100, lag = 5, seed = 123)
x

# x's explosive episode should visibly lead y's by 5 periods
matplot(x, type = "l", lty = 1, col = c("steelblue", "tomato"),
  xlab = "t", ylab = "value", main = "sim_coexplosive(): x leads y by 5 periods")
legend("topleft", legend = c("x", "y"), col = c("steelblue", "tomato"),
  lty = 1, bty = "n")
```

sim_common

Simulation of a latent common-factor bubble across multiple series

Description

Simulates the common-bubble process of Chen, Phillips & Shi (2023): n_series observed series driven by one latent PSY-style bubble factor plus idiosyncratic noise, $X_t = \Lambda f_t + e_t$, with loadings $\Lambda \sim U[0, 2]$. Unlike every other `sim_*` function, it returns a `data.frame` with one column for each series, the standard panel input shape of `exuber`, and not a single numeric vector.

Usage

```
sim_common(
  n_series,
  n,
  te = 0.4 * n,
  tf = 0.15 * n + te,
  c = 1,
  alpha = 0.6,
  sigma = 6.79,
  sigma_e = 0.1,
  seed = NULL
)
```

Arguments

n_series	A positive integer, the number of observed series.
n	A positive integer specifying the length of the simulated output series.
te	A scalar in (0, tf) specifying the observation in which the bubble originates.
tf	A scalar in (te, n) specifying the observation in which the bubble collapses.
c	A positive scalar determining the autoregressive coefficient in the explosive regime.
alpha	A positive scalar in (0, 1) determining the value of the expansion rate in the autoregressive coefficient.
sigma	A positive scalar indicating the standard deviation of the innovations.
sigma_e	A non-negative scalar, the idiosyncratic-noise standard deviation.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Value

A data.frame with n_series numeric columns of length n, plus a "factor" attribute holding the latent bubble factor itself.

References

Chen, Y., Phillips, P.C.B. & Shi, S. (2023). "Common Bubble Detection in Large Dimensional Financial Systems." Cowles Foundation Discussion Paper.

See Also

[sim_psy1](#)

Examples

```
x <- sim_common(n_series = 5, n = 100, seed = 123)
x

# Plot every observed series (grey) against the shared latent factor (red)
matplot(x, type = "l", col = "grey60", lty = 1, xlab = "t", ylab = "value",
  main = "sim_common(): observed series driven by one latent bubble factor")
lines(attr(x, "factor"), col = "red", lwd = 2)
legend("topleft", legend = c("observed series", "latent factor"),
  col = c("grey60", "red"), lty = 1, bty = "n")
```

sim_data	<i>Simulated dataset</i>
----------	--------------------------

Description

An artificial dataset containing series simulated from data generating processes widely used in the literature on speculative bubbles. `sim_data` holds five series of 100 observations each; `sim_data_wdate` is the same data with an additional monthly date column. Both were generated by `data-raw/sim-data.R` with `set.seed(1122)`.

Usage

```
sim_data  
  
sim_data_wdate
```

Format

A tibble with 100 rows and columns `psy1` ([sim_psy1\(\)](#)), `psy2` ([sim_psy2\(\)](#)), `evans` ([sim_evans\(\)](#)), `div` ([sim_div\(\)](#)), `blan` ([sim_blan\(\)](#)); `sim_data_wdate` adds a date column (monthly from 2000-01-01).

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 100 rows and 6 columns.

See Also

[sim_psy1\(\)](#), [sim_psy2\(\)](#), [sim_evans\(\)](#), [sim_div\(\)](#), [sim_blan\(\)](#) for the generating processes.

Examples

```
# Explore the five bundled series before running any test  
matplot(sim_data, type = "l", lty = 1, xlab = "t", ylab = "value",  
        main = "sim_data: the five bundled example series")  
legend("topleft", legend = colnames(sim_data), col = 1:5, lty = 1, bty = "n")  
  
# The usual next step: run radf() and plot with the default critical values  
autoplot(radf(sim_data))
```

sim_div

*Simulation of dividends***Description**

Simulate (log) dividends from a random walk with drift.

Usage

```
sim_div(
  n,
  mu,
  sigma,
  r = 0.05,
  log = FALSE,
  output = c("pf", "d"),
  seed = NULL
)
```

Arguments

n	A positive integer specifying the length of the simulated output series.
mu	A scalar indicating the drift.
sigma	A positive scalar indicating the standard deviation of the innovations.
r	A positive value indicating the discount factor.
log	Logical. If true dividends follow a lognormal distribution.
output	A character string giving the fundamental price("pf") or dividend series("d"). Default is pf.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

If log is set to FALSE (default value) dividends follow:

$$d_t = \mu + d_{t-1} + \epsilon_t$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. The default parameters are $\mu = 0.0373$, $\sigma^2 = 0.1574$ and $d_0 = 1.3$ (the initial value of the dividend sequence). The above equation can be solved to yield the fundamental price:

$$F_t = \mu(1+r)r^{-2} + r^{-1}d_t$$

If log is set to TRUE then dividends follow a lognormal distribution or log(dividends) follow:

$$\ln(d_t) = \mu + \ln(d_{t-1}) + \epsilon_t$$

where $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Default parameters are $\mu = 0.013$, $\sigma^2 = 0.16$. The fundamental price in this case is:

$$F_t = \frac{1+g}{r-g}d_t$$

where $1+g = \exp(\mu + \sigma^2/2)$. All default parameter values are those suggested by West (1988).

Value

A numeric vector of length n.

References

West, K. D. (1988). Dividend innovations and stock price volatility. *Econometrica: Journal of the Econometric Society*, p. 37-61.

Examples

```
# Price is the sum of the bubble and fundamental components
# 20 is the scaling factor
pf <- sim_div(100, r = 0.05, output = "pf", seed = 123)
pb <- sim_evans(100, r = 0.05, seed = 123)
p <- pf + 20 * pb

autoplot(p)
```

sim_evans

Simulation of an Evans (1991) bubble process

Description

Simulation of an Evans (1991) rational periodically collapsing bubble process.

Usage

```
sim_evans(
  n,
  alpha = 1,
  delta = 0.5,
  tau = 0.05,
  pi = 0.7,
  r = 0.05,
  b1 = delta,
  seed = NULL
)
```

Arguments

n	A positive integer specifying the length of the simulated output series.
alpha	A positive scalar, with restrictions (see details).
delta	A positive scalar, with restrictions (see details).
tau	The standard deviation of the innovations.
pi	A positive value in (0, 1) which governs the probability of the bubble continuing to grow.
r	A positive scalar that determines the growth rate of the bubble process.
b1	A positive scalar, the initial value of the series. Defaults to delta.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to set.seed before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns .Random.seed as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

delta and alpha are positive parameters which satisfy $0 < \delta < (1 + r)\alpha$. delta represents the size of the bubble after collapse. The default value of r is 0.05. The function checks whether alpha and delta satisfy this condition and will return an error if not.

The Evans bubble has two regimes. If $B_t \leq \alpha$ the bubble grows at an average rate of $1 + r$:

$$B_{t+1} = (1 + r)B_t u_{t+1},$$

When $B_t > \alpha$ the bubble expands at the increased rate of $(1 + r)\pi^{-1}$:

$$B_{t+1} = [\delta + (1 + r)\pi^{-1}\theta_{t+1}(B_t - (1 + r)^{-1}\delta)]u_{t+1},$$

where θ is a binary variable that takes the value 0 with probability $1 - \pi$ and 1 with probability π . In the second phase, there is a $(1 - \pi)$ probability of the bubble process collapsing to delta. By modifying the values of delta, alpha and pi the user can change the frequency at which bubbles appear, the mean duration of a bubble before collapse and the scale of the bubble.

Value

A numeric vector of length n .

References

Evans, G. W. (1991). Pitfalls in testing for explosive bubbles in asset prices. *The American Economic Review*, 81(4), 922-930.

See Also

[sim_psy1](#), [sim_psy2](#), [sim_blan](#)

Examples

```
sim_evans(100, seed = 123) %>%
  autoplot()
```

sim_falsebubble	<i>Simulation of a deterministic technology-adoption "false bubble" null</i>
-----------------	--

Description

Simulates the false-bubble process of Chen, Chen, Huang, Li & Zhang (2026): a hump-shaped, *deterministic* technology-adoption shock embedded in dividend growth. It is engineered so that a Campbell-Shiller present-value fundamental alone, with no bubble component at all, displays a price path that looks locally explosive. It is useful as a null (no-bubble) stress test that differs from a plain random walk.

Usage

```
sim_falsebubble(
  n,
  t1 = floor(0.3 * n),
  t2 = floor(0.7 * n),
  kappa = floor((t2 - t1)/2),
  shape = c("triangular", "gaussian"),
  amplitude = 1,
  mu = 0.02,
  sigma_d = 0.05,
  r = 0.05,
  d0 = 0,
  seed = NULL
)
```

Arguments

n	A positive integer specifying the length of the simulated output series.
t1	Adoption (ramp-up start) date, in 1:n.
t2	Maturation (shock end) date, in t1:n.
kappa	Peak lag (time from t1 to the hump's peak), in 0:(t2 - t1).
shape	"triangular" (default) or "gaussian".
amplitude	A positive scalar scaling the hump's peak height.
mu	A scalar, the baseline dividend-growth drift.
sigma_d	A positive scalar, the dividend-growth innovation standard deviation.
r	A positive scalar, the discount rate.
d0	Starting (log) dividend level.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to set.seed before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns .Random.seed as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

Dividends follow a random walk with drift plus the technology hump: $d_t = d_{t-1} + \mu + \tau_t + \eta_t$. The hump τ_t either rises linearly from t1 to t1 + kappa and then falls linearly to t2 (shape = "triangular", the worked example of the source, eq. 4), or follows a Gaussian bump centered at t1 + kappa (shape = "gaussian"). Because τ_t is deterministic and known in advance, its contribution to the price is an exact forward-looking discounted sum, $T_t = \sum_{s>t} \beta^{s-t} \tau_s$ with $\beta = 1/(1+r)$, which is added to the fundamental pricing formula that [sim_div](#) uses. This is a simplified single-shock reproduction of the mechanism in the source (a deterministic hump gives a hump-shaped fundamental price and no bubble). It does not include the full DOLS and multiple-functional-form robustness machinery of the source.

Value

A numeric vector of length n (the price), with "dividend" and "technology" attributes.

References

Chen, H., Chen, L., Huang, D., Li, Y. & Zhang, Z. (2026). "Technology Fundamentals and False Bubble Detection: Evidence from Dot-Com and AI Episodes." arXiv:2604.25826.

See Also

[sim_div](#), [sim_evans](#)

Examples

```
sim_falsebubble(200, seed = 123) %>%
  autoplot()
```

sim_fi

*Simulate fractionally-integrated (long-memory) innovations***Description**

Generates $u_t = \Delta^{-d}\epsilon_t$, with ϵ_t i.i.d. $(0, \sigma^2)$, through a truncated $MA(\infty)$ expansion of the fractional-differencing operator, for use as `sim_psy1(..., e = sim_fi(...))`.

Usage

```
sim_fi(n, d = 0.2, sigma = 1, seed = NULL)
```

Arguments

n	Number of innovations to generate.
d	Long-memory (fractional differencing) parameter, in $(0, 0.5)$ so that u_t itself is stationary.
sigma	A positive scalar indicating the standard deviation of the innovations.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

$$\Delta^{-d} = \sum_{j=0}^{\infty} \psi_j L^j, \quad \psi_0 = 1, \quad \psi_j = \psi_{j-1} \frac{j-1+d}{j}$$

The expansion is truncated at $\max(200, n)$ lags with a matching burn-in, which is dropped before the function returns, to limit the truncation bias in the early observations.

Value

A numeric vector of length n.

References

Lui, Y.L., Phillips, P.C.B. & Yu, J. (2024). "Robust testing for explosive behavior with strongly dependent errors."

See Also

[sim_psy1](#)

Examples

```
sim_fi(199, d = 0.2, seed = 1) %>%
  autoplot()
```

 sim_innov

Simulate innovations with heavy-tailed/skewed marginal distributions

Description

Generates a shock sequence for use with the PSY-style mean equations ([sim_psy1](#) and [sim_psy2](#)) but with a non-Gaussian marginal distribution. The sequence is standardized to mean 0 and variance σ^2 , so it fits directly into `sim_psy1(..., e = sim_innov(...))`.

Usage

```
sim_innov(
  n,
  dist = c("normal", "t", "skew_t"),
  sigma = 6.79,
  df = 5,
  xi = 0,
  seed = NULL
)
```

Arguments

<code>n</code>	Number of innovations to generate.
<code>dist</code>	One of "normal", "t", "skew_t".
<code>sigma</code>	A positive scalar indicating the standard deviation of the innovations.
<code>df</code>	Degrees of freedom for "t"/"skew_t" (> 2).
<code>xi</code>	Skewness parameter for "skew_t" (any real; 0 = symmetric).
<code>seed</code>	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

`dist = "t"` rescales a Student-t(df) draw to variance 1 before scaling by `sigma`. This is exact and closed form, because $\text{Var}(t_{df}) = df / (df - 2)$. `dist = "skew_t"` combines two independent standardized Student-t draws in the manner of Azzalini, $\delta * \text{abs}(T_0) + \sqrt{1 - \delta^2} * T_1$ with $\delta = xi / \sqrt{1 + xi^2}$, and then standardizes the result with the closed-form mean and variance of that combination ($E|T_0|$ is itself closed form through the Beta function). $xi > 0$ skews the distribution to the right, $xi < 0$ skews it to the left, and $xi = 0$ gives the symmetric t case.

Value

A numeric vector of length n .

References

Wu, R., Shi, S. & Wu, J. (2025). "Quantile analysis for financial bubble detection and surveillance." JTSA, 46(5), 908-931 (uses $N(0,1)/t(3)/\text{skewed-}t(3, -0.75)/\text{skewed-}t(3, 0.75)$ innovations in their Monte Carlo design, eq. 6).

See Also

[sim_psy1](#)

Examples

```
sim_innov(199, dist = "skew_t", df = 3, xi = -0.75, seed = 1) %>%
  autoplot()

# Feed skew-t innovations into sim_psy1() in place of i.i.d. Gaussian ones
sim_psy1(n = 200, seed = 123, e = sim_innov(199, dist = "skew_t", df = 3, xi = -0.75, seed = 1)) %>%
  autoplot()
```

sim_mar

Simulation of a mixed causal-noncausal AR(1,1) bubble

Description

Simulates the mixed causal-noncausal autoregressive (MAR) bubble process of Blasques, Koopman, Mingoli & Telg (2025). Transient, self-terminating local bubbles arise on their own from the *noncausal* (forward-looking) component. Unlike [sim_psy1](#), no origination or collapse dates are scripted.

Usage

```
sim_mar(
  n,
  phi1 = 0.7,
  psi1 = 0.7,
  dist = c("cauchy", "t"),
  df = 2,
  burn = 100,
  seed = NULL
)
```

Arguments

n	A positive integer specifying the length of the simulated output series.
phi1	Causal AR coefficient, in (0, 1).
psi1	Noncausal AR coefficient, in (0, 1).
dist	Innovation distribution: "cauchy" or "t" (with df degrees of freedom).
df	Degrees of freedom if dist = "t".
burn	Non-negative burn-in length applied at <i>both</i> ends (see Details).
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to set.seed before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns .Random.seed as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

$$(1 - \phi_1 L)(1 - \psi_1 L^{-1})y_t = \epsilon_t$$

The function uses the standard two-sided filtering method for MAR processes (Lanne & Saikkonen 2011; Gouriéroux & Zakoian 2017). It generates the noncausal component by running $u_t = \psi_1 u_{t+1} + \epsilon_t$ *backward* from a zero boundary burn observations past the end of the sample. It then generates the causal component by running $y_t = \phi_1 y_{t-1} + u_t$ *forward* from a zero boundary burn observations before the start. Both burn-in windows are then dropped.

Value

A numeric vector of length n.

References

Blasques, F., Koopman, S.J., Mingoli, G. & Telg, S. (2025). "A Novel Test for the Presence of Local Explosive Dynamics." JTSA, 46(5), 966-980.

See Also

[sim_psy1](#)

Examples

```
sim_mar(200, seed = 123) %>%
  autoplot()
```

sim_msbubble

*Simulation of a Markov-switching present-value bubble***Description**

Simulation of Chan & Santi (2021)'s bubble component of a present-value state-space model: an AR(1) whose persistence switches between a "surviving" (explosive) and a "collapsing" (mean-reverting) regime under a first-order Markov chain, rather than at deterministic dates ([sim_psy1](#)) or a fixed-probability mixture ([sim_blan](#)).

Usage

```
sim_msbubble(
  n,
  p11 = 0.98,
  p22 = 0.9,
  lambda1 = 0.98,
  lambda2 = 1.03,
  sigma_b = 0.05,
  b0 = 0,
  s0 = 1L,
  seed = NULL
)
```

Arguments

n	A positive integer specifying the length of the simulated output series.
p11, p22	Regime-1-to-1 and regime-2-to-2 transition probabilities, in (0, 1).
lambda1, lambda2	Regime persistence parameters (lambda1 < 1 explosive, lambda2 > 1 mean-reverting).
sigma_b	A positive scalar, the bubble-innovation standard deviation.
b0	Starting value.
s0	Starting regime, 1L or 2L.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

$$b_t = \frac{1}{\lambda_{S_t}} b_{t-1} + \epsilon_t^b, \quad \epsilon_t^b \sim iid N(0, \sigma_b^2)$$

with $S_t \in \{1, 2\}$ a Markov chain with transition probabilities $p11 = P(S[t]=1 | S[t-1]=1)$, $p22 = P(S[t]=2 | S[t-1]=2)$. Regime 1 ("surviving") uses $\lambda_1 < 1$ (so $1/\lambda_1 > 1$, explosive); regime 2 ("collapsing") uses $\lambda_2 > 1$ (mean-reverting). Note: the source's own eq. 16 indexes the coefficient by S_{t+1} ; this implementation uses the contemporaneous S_t instead (an indexing-convention simplification, not a change to the qualitative Markov-switching mechanism).

Value

A numeric vector of length n , with a "regime" attribute (the simulated S_t path).

References

Chan, J.C.C. & Santi, C. (2021). "Speculative Bubbles in Present-Value Models: A Bayesian Markov-Switching State Space Approach." *Journal of Economic Dynamics and Control*, 127, 104101.

See Also

[sim_psy1](#), [sim_blan](#)

Examples

```
sim_msbubble(200, seed = 123) %>%
  autoplot()
```

sim_psl	<i>Simulation of a single-bubble process with multiple forms of collapse regime</i>
---------	---

Description

The process differs from the `sim_psy1` model in three respects (Phillips and Shi 2018). First, it includes an asymptotically negligible drift in the martingale path during normal periods. Second, the collapse is modeled directly as a transient mildly integrated process that covers an explicit period of market collapse. Third, it introduces a market recovery date to capture the return to normal market behavior. Three forms of collapse are available:

- sudden: with $\beta = 0.1$ and $tr = tf + 0.01 \cdot n$
- disturbing: with $\beta = 0.5$ and $tr = tf + 0.1 \cdot n$
- smooth: with $\beta = 0.9$ and $tr = tf + 0.2 \cdot n$

To set the duration of the collapse period through $tr = tf + 0.2n$, you must also provide tf .

Usage

```
sim_ps1(
  n,
  te = 0.4 * n,
  tf = te + 0.2 * n,
  tr = tf + 0.1 * n,
  c = 1,
  c1 = 1,
  c2 = 1,
  eta = 0.6,
  alpha = 0.6,
  beta = 0.5,
  sigma = 6.79,
  seed = NULL,
  e = NULL
)
```

Arguments

n	A positive integer specifying the length of the simulated output series.
te	A scalar in (0, tf) specifying the observation in which the bubble originates.
tf	A scalar in (te, n) specifying the observation in which the bubble collapses.
tr	A scalar in (tf, n) specifying the observation in which market recovers
c	A positive scalar determining the drift in the normal market periods.
c1	A positive scalar determining the autoregressive coefficient in the explosive regime.
c2	A positive scalar determining the autoregressive coefficient in the collapse regime.
eta	A positive scalar (>0.5) determining the drift in the normal market periods.
alpha	A positive scalar in (0, 1) determining the autoregressive coefficient in the bubble period.
beta	A positive scalar in (0, 1) determining the autoregressive coefficient in the collapse period.
sigma	A positive scalar indicating the standard deviation of the innovations.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.
e	An optional numeric vector of length $n - 1$ with innovations to use in place of <code>rnorm(n - 1, sd = sigma)</code> . It lets the plain PSY equation above be driven by a shock sequence that is non-Gaussian, heteroskedastic or dependent instead of i.i.d. Gaussian noise. The generators sim_innov (heavy-tailed or skewed), sim_vol_break (permanent volatility break), sim_vol_garch (GARCH/TGARCH),

[sim_vol_cir](#) and [sim_vol_sv](#) (stochastic volatility) and [sim_fi](#) (long memory) produce suitable sequences. The default NULL reproduces the plain i.i.d. Gaussian process exactly.

Value

A numeric vector of length n.

References

Phillips, Peter CB, and Shu-Ping Shi. "Financial bubble implosion and reverse regression." *Econometric Theory* 34.4 (2018): 705-753.

See Also

[sim_psy1](#)

Examples

```
# Disturbing collapse (default)
disturbing <- sim_psy1(100)
autoplot(disturbing)

# Sudden collapse
sudden <- sim_psy1(100, te = 40, tf = 60, tr = 61, beta = 0.1)
autoplot(sudden)
```

sim_psy1

Simulation of a single-bubble process

Description

Generates a time series that switches from a martingale to a mildly explosive process and then back to a martingale.

Usage

```
sim_psy1(
  n,
  te = 0.4 * n,
  tf = 0.15 * n + te,
  c = 1,
  alpha = 0.6,
  sigma = 6.79,
  seed = NULL,
  e = NULL,
  shifts = NULL,
```

```

    coef_noise = NULL,
    coef_a = 1
)

```

Arguments

n	A positive integer specifying the length of the simulated output series.
te	A scalar in (0, tf) specifying the observation in which the bubble originates.
tf	A scalar in (te, n) specifying the observation in which the bubble collapses.
c	A positive scalar determining the autoregressive coefficient in the explosive regime.
alpha	A positive scalar in (0, 1) determining the value of the expansion rate in the autoregressive coefficient.
sigma	A positive scalar indicating the standard deviation of the innovations.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.
e	An optional numeric vector of length $n - 1$ with innovations to use in place of <code>rnorm(n - 1, sd = sigma)</code> . It lets the plain PSY equation above be driven by a shock sequence that is non-Gaussian, heteroskedastic or dependent instead of i.i.d. Gaussian noise. The generators sim_innov (heavy-tailed or skewed), sim_vol_break (permanent volatility break), sim_vol_garch (GARCH/TGARCH), sim_vol_cir and sim_vol_sv (stochastic volatility) and sim_fi (long memory) produce suitable sequences. The default NULL reproduces the plain i.i.d. Gaussian process exactly.
shifts	An optional data frame or list with an integer element or column date (in 2:n) and a numeric element or column size. It adds a one-period deterministic level shift of magnitude size at each date, which is the level-shift process of Harvey, Leybourne, Tatlow & Zu (2025). The default NULL adds no shifts.
coef_noise	An optional numeric vector of length $n - 1$ with zero mean and unit variance. It perturbs the coefficient of the explosive regime as $\delta + \text{coef_a} * \text{coef_noise}[t] / \sqrt{n}$ instead of the fixed δ , which gives the stochastically varying explosive coefficient of Kurozumi & Nishi (2025). The default NULL keeps δ fixed.
coef_a	A positive scalar scaling <code>coef_noise</code> . Ignored if <code>coef_noise</code> is NULL.

Details

The data generating process is described by the following equation:

$$X_t = X_{t-1}1\{t < \tau_e\} + \delta_T X_{t-1}1\{\tau_e \leq t \leq \tau_f\} + \left(\sum_{k=\tau_f+1}^t \epsilon_k + X_{\tau_f} \right) 1\{t > \tau_f\} + \epsilon_t 1\{t \leq \tau_f\}$$

where the autoregressive coefficient δ_T is given by:

$$\delta_T = 1 + cT^{-\alpha}$$

with $c > 0$, $\alpha \in (0, 1)$, $\epsilon \sim iid(0, \sigma^2)$ and $X_{\tau_f} = X_{\tau_e} + X'$ with $X' = O_p(1)$, $\tau_e = [Tr_e]$ dates the origination of the bubble, and $\tau_f = [Tr_f]$ dates the collapse of the bubble. During the pre- and post- bubble periods, $[1, \tau_e)$, X_t is a pure random walk process. During the bubble expansion period $[\tau_e, \tau_f]$ becomes a mildly explosive process with expansion rate given by the autoregressive coefficient δ_T ; and, finally during the post-bubble period, $(\tau_f, \tau]$ X_t reverts to a martingale.

For further details see Phillips et al. (2015) p. 1054.

Value

A numeric vector of length n.

References

Phillips, P. C. B., Shi, S., & Yu, J. (2015). Testing for Multiple Bubbles: Historical Episodes of Exuberance and Collapse in the S&P 500. *International Economic Review*, 56(4), 1043-1078. [doi:10.1111/iere.12132](https://doi.org/10.1111/iere.12132)

See Also

[sim_psy2](#), [sim_blan](#), [sim_evans](#)

Examples

```
# 100 periods with bubble origination date 40 and termination date 55
sim_psy1(n = 100, seed = 123) %>%
  autoplot()

# 200 periods with bubble origination date 80 and termination date 110
sim_psy1(n = 200, seed = 123) %>%
  autoplot()

# 200 periods with bubble origination date 100 and termination date 150
sim_psy1(n = 200, te = 100, tf = 150, seed = 123) %>%
  autoplot()

# Same DGP, driven by GARCH(1,1) innovations instead of i.i.d. Gaussian
sim_psy1(n = 200, seed = 123, e = sim_vol_garch(199, seed = 123)) %>%
  autoplot()

# Same DGP, with two deterministic level shifts
sim_psy1(n = 200, seed = 123, shifts = list(date = c(50, 150), size = c(20, -20))) %>%
  autoplot()
```

sim_psy2

*Simulation of a two-bubble process***Description**

This data generating process is similar to [sim_psy1](#), with the difference that there are two episodes of mildly explosive dynamics.

Usage

```
sim_psy2(
  n,
  te1 = 0.2 * n,
  tf1 = 0.2 * n + te1,
  te2 = 0.6 * n,
  tf2 = 0.1 * n + te2,
  c = 1,
  alpha = 0.6,
  sigma = 6.79,
  seed = NULL
)
```

Arguments

n	A positive integer specifying the length of the simulated output series.
te1	A scalar in (0, n) specifying the observation in which the first bubble originates.
tf1	A scalar in (te1, n) specifying the observation in which the first bubble collapses.
te2	A scalar in (tf1, n) specifying the observation in which the second bubble originates.
tf2	A scalar in (te2, n) specifying the observation in which the second bubble collapses.
c	A positive scalar determining the autoregressive coefficient in the explosive regime.
alpha	A positive scalar in (0, 1) determining the value of the expansion rate in the autoregressive coefficient.
sigma	A positive scalar indicating the standard deviation of the innovations.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

The two-bubble data generating process is given by (see also sim_psy1):

$$X_t = X_{t-1}1\{t \in N_0\} + \delta_T X_{t-1}1\{t \in B_1 \cup B_2\} + \left(\sum_{k=\tau_{1f}+1}^t \epsilon_k + X_{\tau_{1f}} \right) 1\{t \in N_1\} \\ + \left(\sum_{l=\tau_{2f}+1}^t \epsilon_l + X_{\tau_{2f}} \right) 1\{t \in N_2\} + \epsilon_t 1\{t \in N_0 \cup B_1 \cup B_2\}$$

where the autoregressive coefficient δ_T is:

$$\delta_T = 1 + cT^{-\alpha}$$

with $c > 0$, $\alpha \in (0, 1)$, $\epsilon \sim iid(0, \sigma^2)$, $N_0 = [1, \tau_{1e})$, $B_1 = [\tau_{1e}, \tau_{1f}]$, $N_1 = (\tau_{1f}, \tau_{2e})$, $B_2 = [\tau_{2e}, \tau_{2f}]$, $N_2 = (\tau_{2f}, \tau]$, where τ is the last observation of the sample. The observations $\tau_{1e} = [Tr_{1e}]$ and $\tau_{1f} = [Tr_{1f}]$ are the origination and termination dates of the first bubble; $\tau_{2e} = [Tr_{2e}]$ and $\tau_{2f} = [Tr_{2f}]$ are the origination and termination dates of the second bubble. After the collapse of the first bubble, X_t resumes a martingale path until time $\tau_{2e} - 1$, and a second episode of exuberance begins at τ_{2e} . Exuberance lasts until τ_{2f} , at which point the process collapses to a value of $X_{\tau_{2f}}$. The process then continues on a martingale path until the end of the sample period τ . The duration of the first bubble is assumed to be longer than that of the second bubble, i.e. $\tau_{1f} - \tau_{1e} > \tau_{2f} - \tau_{2e}$.

For further details you can refer to Phillips et al., (2015) p. 1055.

Value

A numeric vector of length n.

References

Phillips, P. C. B., Shi, S., & Yu, J. (2015). Testing for Multiple Bubbles: Historical Episodes of Exuberance and Collapse in the S&P 500. *International Economic Review*, 56(4), 1043-1078. [doi:10.1111/iere.12132](https://doi.org/10.1111/iere.12132)

See Also

[sim_psy1](#), [sim_blan](#), [sim_evans](#)

Examples

```
# 100 periods with bubble origination dates 20/60 and termination dates 40/70
sim_psy2(n = 100, seed = 123) %>%
  autoplot()

# 200 periods with bubble origination dates 40/120 and termination dates 80/140
sim_psy2(n = 200, seed = 123) %>%
  autoplot()
```

sim_tree

*Simulation of a stochastic branching-tree bubble***Description**

Simulates the stochastic-tree bubble process of Gouriéroux & Jasiak (2025). It is a positive stationary submartingale generated by a binomial tree with *stochastic* branching intensity, which makes it a random-coefficient autoregression, in contrast to the deterministic branches of Cox-Ross-Rubinstein. The bubble of Blanchard & Watson (1982) ([sim_blan](#)) is the special case of *constant* intensity.

Usage

```
sim_tree(
  n,
  a = 0.95,
  eta = 1,
  mu = -1,
  rho = 0.7,
  sigma = 4,
  y0 = eta/(1 - a),
  seed = NULL
)
```

Arguments

n	A positive integer specifying the length of the simulated output series.
a	A scalar in (0, 1) (note: $1/a > 1$ is the growth rate).
eta	A positive scalar setting the price floor $\eta / (1 - a)$.
mu, rho, sigma	Parameters of the latent Gaussian AR(1) intensity process (rho in (-1, 1), sigma > 0).
y0	Starting value. Defaults to the price floor $\eta / (1 - a)$.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

The stochastic intensity is $p_t = \Phi(X_t)$, with X_t a latent stationary Gaussian AR(1):

$$X_t = \mu + \rho(X_{t-1} - \mu) + \sigma\sqrt{1 - \rho^2}u_t, \quad u_t \sim iid N(0, 1)$$

Given p_t , draw $Z_t \sim \text{Bernoulli}(p_t)$ and set (the paper's eq. 5-6):

$$Y_t = \xi_{1t} Y_{t-1} + \epsilon_t, \quad \xi_{1t} = \frac{1}{a} \frac{Z_t}{p_t}, \quad \epsilon_t = \frac{\eta}{1-a} (1 - \xi_{1t}) + \frac{\eta}{a} \frac{1 - Z_t}{1 - p_t}$$

$a > 1$ controls the growth rate in the active phase of a branch, $\eta > 0$ sets the price floor $\eta/(1-a)$ (Corollary 1 in the source: $Y_t \geq \eta/(1-a)$), ρ controls the persistence of the bubble-growth phase, and σ controls the frequency of bubbles. The process has no finite mean (Proposition 3 of the source), so occasional very large values are a feature of the model and not a bug.

The default parameters ($\mu = -1, \eta = 1, a = 0.95, \sigma = 4, \rho = 0.7$) reproduce the illustrative example of the source (Section 2.3, Figure 2).

Value

A numeric vector of length n.

References

Gourieroux, C. & Jasiak, J. (2025). "A Stochastic Tree for Bubble Asset Modelling and Pricing." JTSA, 46(5), 932-944.

See Also

[sim_blan](#)

Examples

```
sim_tree(100, seed = 123) %>%
  autoplot()
```

sim_vol_break

Simulate innovations with a permanent volatility break

Description

Generates i.i.d. Gaussian shocks whose standard deviation shifts permanently from σ to $\sigma * \text{ratio}$ at observation $\tau * n$, for use as `sim_psy1(..., e = sim_vol_break(...))`. This is the *non-stationary* volatility process (the single break of Cavaliere & Taylor 2007) under which the standard critical values of `radf` lose size control. The volatility-robust tests (`radf_tt`, `radf_kp`, `radf_sbz`, `radf_sign` and `radf_wb_cv`) are designed for it. Stationary conditional heteroskedasticity (`sim_vol_garch`) is different, because its variance profile is asymptotically flat.

Usage

```
sim_vol_break(n, tau = 0.5, ratio = 3, sigma = 6.79, seed = NULL)
```

Arguments

n	Number of innovations to generate.
tau	Break fraction in (0, 1): the shift happens after observation <code>floor(tau * n)</code> .
ratio	Positive ratio of the post-break to the pre-break standard deviation. <code>ratio > 1</code> is an upward break, the case in which <code>radf()</code> over-rejects the most, and <code>ratio < 1</code> is a downward one.
sigma	A positive scalar indicating the standard deviation of the innovations.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Value

A numeric vector of length n.

References

Cavaliere, G. & Taylor, A.M.R. (2007). "Testing for unit roots in time series models with non-stationary volatility." *Journal of Econometrics*, 140, 919-947. Harvey, D.I., Leybourne, S.J., Sollis, R. & Taylor, A.M.R. (2016). "Tests for explosive financial bubbles in the presence of non-stationary volatility." *Journal of Empirical Finance*, 38, 548-574.

See Also

[sim_psy1](#), [sim_vol_garch](#)

Examples

```
sim_vol_break(199, seed = 1) %>%
  autoplot()
# Volatility triples half-way through a PSY bubble series
sim_psy1(n = 200, seed = 123, e = sim_vol_break(199, seed = 123)) %>%
  autoplot()
```

sim_vol_cir

Simulate CIR-type stochastic-volatility innovations

Description

Generates shocks driven by a Cox-Ingersoll-Ross (square-root) stochastic variance process, discretized with the Euler-Maruyama scheme, for use as `sim_psy1(..., e = sim_vol_cir(...))`.

Usage

```
sim_vol_cir(
  n,
  kappa = 0.03,
  theta = 0.25,
  xi = 0.1,
  sigma0_sq = theta,
  seed = NULL
)
```

Arguments

n	Number of innovations to generate.
kappa, theta, xi	Positive CIR parameters (speed of mean reversion, long-run variance and volatility of volatility).
sigma0_sq	Non-negative starting variance. Defaults to theta.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

$$d\sigma^2(r) = \kappa(\theta - \sigma^2(r))dr + \xi\sigma(r)dB(r)$$

discretized over n steps of r in $[0, 1]$. The variance is reflected at zero if a step would take it negative. The default parameters ($\kappa = 0.03$, $\theta = 0.25$, $\xi = 0.1$) match the robustness design of Harvey, Leybourne & Zu (2019), which is "representative of Bollerslev and Zhou (2002)".

Value

A numeric vector of length n .

References

Harvey, D.I., Leybourne, S.J. & Zu, Y. (2019). "Testing explosive bubbles with time-varying volatility." *Econometric Reviews*, 38(10), 1131-1151.

See Also

[sim_psy1](#), [sim_vol_sv](#)

Examples

```
sim_vol_cir(199, seed = 1) %>%
  autoplot()
```

sim_vol_garch

*Simulate GARCH(1,1)/TGARCH(1,1) innovations***Description**

Generates shocks $z_t = \sqrt{h_t} * \epsilon_t$ from a GARCH(1,1) recursion with an optional threshold (leverage) term, for use as `sim_psy1(..., e = sim_vol_garch(...))`.

Usage

```
sim_vol_garch(n, omega = 0.1, alpha = 0.1, beta = 0.8, gamma = 0, seed = NULL)
```

Arguments

n	Number of innovations to generate.
omega, alpha, beta	Positive GARCH(1,1) parameters. The defaults (omega = 0.1, alpha = 0.1, beta = 0.8) match Whitehouse, Harvey & Leybourne (2025) and Harvey, Leybourne, Taylor & Zu (2024).
gamma	Non-negative TGARCH leverage parameter. The NASDAQ calibration of Monschang & Wilfling (2021) is omega = 0.4387, alpha = 0, beta = 0.9319, gamma = 0.1306.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

$$z_t = \sqrt{h_t} \epsilon_t, \quad h_t = \omega + \alpha z_{t-1}^2 + \beta h_{t-1} + \gamma z_{t-1}^2 1\{z_{t-1} < 0\}$$

with $\epsilon_t \sim NIID(0, 1)$ and $h_0 = z_0 = 0$. `gamma = 0` (the default) gives plain GARCH(1,1), and `gamma > 0` adds the TGARCH leverage effect, a larger response to negative shocks.

Value

A numeric vector of length n.

References

Whitehouse, E.J., Harvey, D.I. & Leybourne, S.J. (2025). "Real-time monitoring of explosive financial bubbles." Monschang, V. & Wilfling, B. (2021). "Sup-ADF-style bubble-detection methods under test." *Empirical Economics*, 61, 145-172.

See Also[sim_psy1](#)**Examples**

```

sim_vol_garch(199, seed = 1) %>%
  autoplot()
# NASDAQ-calibrated TGARCH (Monschang & Wilfling 2021)
sim_vol_garch(199, omega = 0.4387, alpha = 0, beta = 0.9319, gamma = 0.1306, seed = 1) %>%
  autoplot()

```

sim_vol_sv

*Simulate AR(1) lognormal stochastic-volatility innovations***Description**

Generates shocks $z_t = \sigma_t * \epsilon_t$ with a persistent AR(1) log-variance, for use as `sim_psy1(..., e = sim_vol_sv(...))`.

Usage

```
sim_vol_sv(n, phi = 0.98, tau = 0.1, log_sigma0_sq = 0, seed = NULL)
```

Arguments

n	Number of innovations to generate.
phi	AR(1) log-variance persistence, in (0, 1).
tau	Positive standard deviation of the log-variance innovations.
log_sigma0_sq	Starting value of $\log(\sigma^2)$. Defaults to 0.
seed	An object specifying if and how the random number generator (rng) should be initialized. It is either NULL or an integer, which is passed to <code>set.seed</code> before the simulation. If you set it, the value is saved as the "seed" attribute of the returned value. The default, NULL, leaves the state of the rng unchanged and returns <code>.Random.seed</code> as the "seed" attribute. Results are reproducible across the parallel and the non-parallel option when you use the same seed.

Details

$$\log \sigma_t^2 = \phi \log \sigma_{t-1}^2 + \eta_t, \quad \eta_t \sim iid N(0, \tau^2)$$

with phi close to, but below, 1 for the "double local-to-unity" near-integrated-variance case studied in the source.

Value

A numeric vector of length n.

References

Sarkar, A. & Wells, M.T. (2025). "Double Local-to-Unity." arXiv:2512.06823.

See Also

[sim_psy1](#), [sim_vol_cir](#)

Examples

```
sim_vol_sv(199, seed = 1) %>%
  autoplot()
```

ssu_test	<i>Stochastic Unit Root Bubble Test (Kurozumi & Nishi 2025)</i>
----------	---

Description

ssu_test implements the SSU and GSSU statistics of Kurozumi & Nishi (2025). They are sup-type tests for a bubble that test for a stochastic, and not deterministic, unit root in the *squared* first differences, $(\Delta y_t)^2 = \mu_2 + \omega y_{t-1}^2 + \eta_t$. The statistic is bias-corrected for the dependence on the correlation between the innovations of this regression and those of the plain ADF regression.

Usage

```
ssu_test(
  data,
  minw = NULL,
  sig_lvl = 95,
  type = c("ssu", "gssu"),
  union = FALSE,
  cv = NULL
)
```

Arguments

data	A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. A column may have leading or trailing NA values, which describes an unbalanced panel in which series enter or exit the sample at different times. Those periods are filled with NA in badf and bsadf and excluded from the adf, sadf and gsadf of that series. Interior NA values (a gap in the middle of a series) are not supported. When any series is padded in this way, the panel statistics (bsadf_panel and gsadf_panel) are not available, and the function returns NA for them with a warning.
minw	Minimum window. The default is psy_minw for "ssu" and the value of the paper, $\text{floor}(n * (-0.004 + 2.24/\sqrt{n}))$, for "gssu". Table I is computed at these values.

sig_lvl	Significance level on the 0 to 100 scale used throughout the package, one of 90, 95 or 99. Table I of Kurozumi & Nishi tabulates these levels.
type	"ssu" or "gssu".
union	Logical. Also run the union-of-rejections procedure with SADF ("ssu") or GSADF ("gssu").
cv	Critical values for the SADF or GSADF side of the union, for example from radf_mc_cv with lag = 0. The default is the precomputed critical values, which are fetched on first use.

Details

This test generalizes the framework differently from the other volatility-robustness tests in *exuber*. It does not touch the innovation variance at all. It allows the explosive AR coefficient itself to vary stochastically over time, $1 + c1/T + a*u_t/\sqrt{T}$, where every recursive-ADF-family statistic in this package assumes the deterministic coefficient $1 + c/T^\alpha$.

type = "ssu" is the single recursion, with the start fixed at the beginning of the sample, which has the shape of SADF. type = "gssu" also takes the supremum over window starts, which has the shape of GSADF, with the minimum window $r0 = -0.004 + 2.24/\sqrt{n}$ from the paper. The paper finds that GSSU is no more powerful than SSU.

union = TRUE adds the union-of-rejections procedure that the paper recommends: $UR = \max(SADF / cv_sadf, SSU / cv_ssu)$ (or GUR with GSADF and GSSU), compared with the published scaling constant ur (gur). Neither SADF nor SSU dominates. SSU wins when the explosive coefficient is stochastic, SADF wins when it is deterministic, and the union stays close to the better of the two. The SADF or GSADF side is [radf](#) with its default minimum window and lag = 0, compared with `cv` (default: the precomputed critical values).

Value

An object of class `ssu_test_obj`: a list with the statistic path (`stat`, one value for each candidate end point from `minw` to `n`. For GSSU it is the sup over window starts at each end point), the constant `crit` from Table I, `sadf` (the maximum, which is compared with `crit`) and `detected`. With `union = TRUE` the list also contains `adf_stat` (SADF or GSADF), `union_stat`, `union_crit` and `union_detected`.

Status

[Experimental]

Note

The SSU and GSSU critical values and the union constants are published asymptotic values (Table I of Kurozumi & Nishi 2025), so no simulation is needed. The union constant is valid only at the level for which the statistic was built.

The function returns its own class and not `radf_obj`, so it does not work with `summary()`, `\link{datestamp}` and `tidy`. It has its own `print()` and `autoplot()` methods instead. `print()` shows the statistic and the critical value. See `vignette("naming-and-analysis", package = "exuber")` for which functions fit the shared pipeline and which do not.

References

Kurozumi, E., & Nishi, M. (2025). Bubble testing with stochastically varying explosive coefficient. *Journal of Time Series Analysis*, 46(5), 945-965.

See Also

[radf](#) for the recursive ADF-family alternative with a deterministic coefficient, which this test complements.

Other volatility-robust tests: [cusum_test\(\)](#), [radf_kp\(\)](#), [radf_sbz\(\)](#), [radf_sbz_union\(\)](#), [radf_sign\(\)](#), [radf_sign_dm\(\)](#), [radf_tt\(\)](#)

Examples

```
# A stochastically varying explosive root, rho_t = 1 + 3/n + 4 * u_t / sqrt(n).
# This is the alternative that ssu_test() is built for, and lbi_test() is built
# for a fixed root
y <- sim_psy1(n = 150, te = 75, tf = 150, c = 3, alpha = 1, seed = 2001,
  coef_noise = rnorm(149), coef_a = 4)
res <- ssu_test(y, sig_lvl = 95)
print(res)

# The double-recursion version
ssu_test(y, type = "gssu")

# Plot the recursive SSU statistic path against its critical value
autoplot(res)
```

summary.radf_obj	<i>Summarizing radf models</i>
------------------	--------------------------------

Description

summary method for radf models, which consist of a radf_obj and a radf_cv.

Usage

```
## S3 method for class 'radf_obj'
summary(object, cv = NULL, ...)
```

Arguments

object	An object of class radf_obj. The output of radf .
cv	An object of class radf_cv. The output of radf_mc_cv , radf_wb_cv or radf_sb_cv .
...	Further arguments passed to methods. Not used.

Value

A list of summary statistics, which includes the estimated ADF, SADF and GSADF test statistics and the corresponding critical values.

Examples

```
# Simulate bubble processes, compute the test statistics and critical values
rsim_data <- radf(sim_data)

# Summary, diagnostics and datestamp (default)
summary(rsim_data)

# Summary, diagnostics and datestamp (wild bootstrap critical values)

wb <- radf_wb_cv(sim_data)

summary(rsim_data, cv = wb)

# summary() reports the same numbers autoplot() draws
autoplot(rsim_data, cv = wb)
```

tidy.ds_radf	<i>Tidy a ds_radf object</i>
--------------	------------------------------

Description

Summarizes information about ds_radf object.

Usage

```
## S3 method for class 'ds_radf'
tidy(x, ...)
```

Arguments

x	An object of class ds_radf.
...	Further arguments passed to methods. Not used.

Examples

```
rsim_data <- radf(sim_data)
ds_data <- datestamp(rsim_data)

# One row per detected explosive episode, across all series
tidy(ds_data)

# Pass the table to ggplot2 if the default layout of autoplot() does not suit you
library(ggplot2)
```

```
tidy(ds_data) %>%
  ggplot(aes(y = id)) +
  geom_segment(aes(x = Start, xend = End, yend = id), linewidth = 3)
```

tidy.radf_cv

Tidy a radf_cv object

Description

Summarizes the information in a radf_cv object.

Usage

```
## S3 method for class 'radf_cv'
tidy(x, format = c("wide", "long"), ...)
```

Arguments

x	An object of class radf_cv.
format	Long or wide format (default = "wide").
...	Further arguments passed to methods. Not used.

Value

A `tibble::tibble()`

- id: The series names.
- sig: The significance level.
- name: The name of the series (when format is "long").
- crit: The critical value (when format is "long").

Examples

```
mc <- radf_mc_cv(100)

# Get the critical values
tidy(mc)

# Get the critical value sequences
augment(mc)
```

tidy.radf_distr	<i>Tidy a radf_distr object</i>
-----------------	---------------------------------

Description

Summarizes the information in a radf_distr object.

Usage

```
## S3 method for class 'radf_distr'  
tidy(x, ...)
```

Arguments

x	An object of class radf_distr.
...	Further arguments passed to methods. Not used.

Value

A `tibble::tibble()`

Examples

```
mdist <- radf_mc_distr(n = 100, nrep = 1000)  
  
tidy(mdist)  
  
# Plot the resulting statistic distributions  
autoplot(mdist)
```

tidy.radf_obj	<i>Tidy a radf_obj object</i>
---------------	-------------------------------

Description

Summarizes the information in a radf_obj object.

Usage

```
## S3 method for class 'radf_obj'  
tidy(x, format = c("wide", "long"), panel = FALSE, ...)
```

Arguments

x	An object of class <code>radf_obj</code> .
format	Long or wide format (default = "wide").
panel	If TRUE then returns the panel statistics
...	Further arguments passed to methods. Not used.

Value

A `tibble::tibble()`

Examples

```
dta <- data.frame(psy1 = sim_psy1(n = 100), psy2 = sim_psy2(n = 100))

rfd <- radf(dta)

# Get the test statistic
tidy(rfd)

# Get the test statistic sequences
augment(rfd)

# Get the panel test statistic
tidy(rfd, panel = TRUE)
```

tidy_join

Tidy into a joint model

Description

Tidy or augment and then join objects.

Usage

```
tidy_join(x, y, ...)
```

Arguments

x	An object of class <code>obj</code> .
y	An object of class <code>cv</code> .
...	Further arguments passed to methods.

Value

A tibble joining the tidied statistics of x with the tidied critical values of y; see the methods for the exact columns.

Examples

```
rsim <- radf(sim_data)
mc <- radf_mc_cv(nrow(sim_data), nrep = 200)
tidy_join(rsim, mc)
augment_join(rsim, mc)
```

tidy_join.radf_obj	<i>Tidy into a joint model</i>
--------------------	--------------------------------

Description

Tidy or augment and then join objects of class `radf_obj` and `radf_cv`. The `radf_cv` is the object of reference. For example, if you provide panel critical values, the function returns the panel test statistic.

Usage

```
## S3 method for class 'radf_obj'
tidy_join(x, y = NULL, ...)
```

Arguments

<code>x</code>	An object of class <code>radf_obj</code> .
<code>y</code>	An object of class <code>radf_cv</code> . The output will depend on the type of critical value.
<code>...</code>	Further arguments passed to methods. Not used.

Details

`tidy_join` also calls `augment_join` when `cv` is of class `sb_cv`.

Examples

```
rsim_data <- radf(sim_data, minw = 20)
cv <- radf_wb_cv(sim_data, minw = 20)

# One row per series/statistic, statistic and critical value side by side
tidy_join(rsim_data, cv)

# summary() and diagnostics() are themselves built on top of tidy_join()
summary(rsim_data, cv = cv)
```

Index

- * **alternative tests**
 - lbi_test, [41](#)
 - quantile_test, [55](#)
- * **critical values**
 - radf_common_cv, [60](#)
 - radf_mc_cv, [63](#)
 - radf_recovery_cv, [67](#)
 - radf_sb_cv, [68](#)
 - radf_sbz_cv, [72](#)
 - radf_sign_cv, [78](#)
 - radf_sign_dm_cv, [81](#)
 - radf_tt_cv, [84](#)
 - radf_wb_cv, [85](#)
 - radf_wb_ps_cv, [88](#)
- * **datasets**
 - sim_data, [100](#)
- * **dating**
 - dating_hls, [30](#)
 - dating_hlw, [31](#)
 - dating_knp, [33](#)
 - dating_pdc, [35](#)
 - radf_recovery, [65](#)
 - rootstamp, [90](#)
- * **monitoring**
 - monitor, [42](#)
 - monitor_cusum, [45](#)
 - monitor_lbi, [48](#)
 - monitor_quantile, [50](#)
- * **multivariate**
 - cobubble_test, [22](#)
 - contagion_reg, [24](#)
 - radf_common, [59](#)
- * **volatility-robust tests**
 - cusum_test, [26](#)
 - radf_kp, [62](#)
 - radf_sbz, [70](#)
 - radf_sbz_union, [74](#)
 - radf_sign, [76](#)
 - radf_sign_dm, [79](#)
 - radf_tt, [82](#)
 - ssu_test, [124](#)
- augment.radf_cv, [4](#)
- augment.radf_obj, [5](#)
- augment_join, [5](#)
- augment_join.radf_obj, [6](#)
- autoplot.cobubble_test_obj, [7](#)
- autoplot.contagion_reg_obj, [7](#)
- autoplot.cusum_test_obj, [8](#)
- autoplot.dating_hls_obj, [9](#)
- autoplot.dating_hlw_obj, [9](#)
- autoplot.dating_knp_obj, [10](#)
- autoplot.dating_pdc_obj, [10](#)
- autoplot.ds_radf, [11](#)
- autoplot.lbi_test_obj, [12](#)
- autoplot.monitor_cusum_obj, [12](#)
- autoplot.monitor_lbi_obj, [13](#)
- autoplot.monitor_obj, [14](#)
- autoplot.monitor_quantile_obj, [14](#)
- autoplot.quantile_test_obj, [15](#)
- autoplot.radf_distr, [16](#)
- autoplot.radf_obj, [6](#), [16](#)
- autoplot.radf_recovery_obj, [19](#)
- autoplot.radf_sbz_union_obj, [20](#)
- autoplot.rootstamp_episodes, [20](#)
- autoplot.rootstamp_est, [21](#)
- autoplot.ssu_test_obj, [22](#)
- autoplot2.radf_obj (autoplot.radf_obj),
[16](#)
- cobubble_test, [7](#), [22](#), [25](#)
- cobubble_test(), [25](#), [60](#)
- contagion_reg, [7](#), [8](#), [24](#)
- contagion_reg(), [24](#), [60](#)
- cusum_test, [8](#), [26](#)
- cusum_test(), [63](#), [71](#), [75](#), [77](#), [80](#), [83](#), [126](#)
- datestamp, [11](#), [17](#), [23](#), [28](#), [30–34](#), [36](#), [37](#), [39](#),
[44](#), [60](#), [61](#), [63](#), [66](#), [70](#), [72](#), [75](#), [77](#), [80](#),
[83](#), [85](#), [90](#), [91](#)

- dating_hls, [9](#), [30](#), [31–33](#), [35](#)
- dating_hls(), [33](#), [35](#), [37](#), [67](#), [92](#)
- dating_hlw, [9](#), [10](#), [31](#)
- dating_hlw(), [31](#), [35](#), [37](#), [67](#), [92](#)
- dating_knp, [10](#), [33](#)
- dating_knp(), [31](#), [33](#), [37](#), [67](#), [92](#)
- dating_pdc, [11](#), [30](#), [31](#), [35](#), [35](#)
- dating_pdc(), [31](#), [33](#), [35](#), [67](#), [92](#)
- diagnostics, [38](#)
- exuber_functions, [39](#)
- geom_segment, [11](#)
- ggplot, [7–15](#), [18–22](#)
- ggplot2::ggplot(), [16](#)
- index (index-rd), [40](#)
- index-rd, [40](#)
- index<- (index-rd), [40](#)
- lbi_test, [12](#), [41](#), [48](#), [49](#)
- lbi_test(), [56](#)
- monitor, [14](#), [42](#), [46](#), [47](#), [49](#), [52](#), [53](#), [88](#)
- monitor(), [47](#), [49](#), [52](#)
- monitor_cusum, [13](#), [27](#), [45](#), [48](#), [49](#)
- monitor_cusum(), [44](#), [49](#), [52](#)
- monitor_lbi, [13](#), [48](#)
- monitor_lbi(), [44](#), [47](#), [52](#)
- monitor_quantile, [14](#), [15](#), [50](#)
- monitor_quantile(), [44](#), [47](#), [49](#)
- ps_tb, [53](#)
- psy_ds, [32](#)
- psy_ds (psy_minw), [54](#)
- psy_minw, [32](#), [54](#), [124](#)
- quantile_test, [15](#), [50–52](#), [55](#)
- quantile_test(), [42](#)
- radf, [23](#), [31](#), [32](#), [42](#), [50](#), [55](#), [56](#), [57](#), [59](#), [60](#),
[62–64](#), [67](#), [69](#), [70](#), [76](#), [77](#), [80](#), [82](#), [83](#),
[119](#), [125](#), [126](#)
- radf_common, [59](#), [60](#), [61](#)
- radf_common(), [24](#), [25](#)
- radf_common_cv, [59](#), [60](#), [60](#)
- radf_common_cv(), [64](#), [68](#), [70](#), [73](#), [79](#), [82](#), [85](#),
[87](#), [89](#)
- radf_kp, [36](#), [62](#), [119](#)
- radf_kp(), [27](#), [71](#), [75](#), [77](#), [80](#), [83](#), [126](#)
- radf_mc_cv, [51](#), [56](#), [59–62](#), [63](#), [63](#), [67](#), [68](#), [70](#),
[71](#), [78](#), [81](#), [84](#), [85](#), [87](#), [89](#), [125](#), [126](#)
- radf_mc_cv(), [61](#), [68](#), [70](#), [73](#), [79](#), [82](#), [85](#), [87](#),
[89](#)
- radf_mc_distr (radf_mc_cv), [63](#)
- radf_recovery, [19](#), [65](#), [67](#), [68](#)
- radf_recovery(), [31](#), [33](#), [35](#), [37](#), [92](#)
- radf_recovery_cv, [65](#), [67](#)
- radf_recovery_cv(), [61](#), [64](#), [70](#), [73](#), [79](#), [82](#),
[85](#), [87](#), [89](#)
- radf_sb_cv, [64](#), [68](#), [87](#), [89](#), [126](#)
- radf_sb_cv(), [61](#), [64](#), [68](#), [73](#), [79](#), [82](#), [85](#), [87](#),
[89](#)
- radf_sb_distr (radf_sb_cv), [68](#)
- radf_sbz, [70](#), [72](#), [73](#), [75](#), [119](#)
- radf_sbz(), [27](#), [63](#), [75](#), [77](#), [80](#), [83](#), [126](#)
- radf_sbz_cv, [36](#), [62](#), [70](#), [71](#), [72](#), [75](#)
- radf_sbz_cv(), [61](#), [64](#), [68](#), [70](#), [79](#), [82](#), [85](#), [87](#),
[89](#)
- radf_sbz_union, [20](#), [71](#), [73](#), [74](#)
- radf_sbz_union(), [27](#), [63](#), [71](#), [77](#), [80](#), [83](#), [126](#)
- radf_sign, [76](#), [78–80](#), [119](#)
- radf_sign(), [27](#), [63](#), [71](#), [75](#), [80](#), [83](#), [126](#)
- radf_sign_cv, [76](#), [77](#), [78](#), [80](#), [81](#)
- radf_sign_cv(), [61](#), [64](#), [68](#), [70](#), [73](#), [82](#), [85](#),
[87](#), [89](#)
- radf_sign_dm, [77](#), [79](#), [81](#)
- radf_sign_dm(), [27](#), [63](#), [71](#), [75](#), [77](#), [83](#), [126](#)
- radf_sign_dm_cv, [80](#), [81](#)
- radf_sign_dm_cv(), [61](#), [64](#), [68](#), [70](#), [73](#), [79](#),
[85](#), [87](#), [89](#)
- radf_tt, [63](#), [75](#), [82](#), [84](#), [85](#), [119](#)
- radf_tt(), [27](#), [63](#), [71](#), [75](#), [77](#), [80](#), [126](#)
- radf_tt_cv, [78](#), [83](#), [84](#)
- radf_tt_cv(), [61](#), [64](#), [68](#), [70](#), [73](#), [79](#), [82](#), [87](#),
[89](#)
- radf_wb_cv, [32](#), [62–64](#), [70–75](#), [77](#), [78](#), [83](#), [84](#),
[85](#), [88](#), [89](#), [119](#), [126](#)
- radf_wb_cv(), [61](#), [64](#), [68](#), [70](#), [73](#), [79](#), [82](#), [85](#),
[89](#)
- radf_wb_distr (radf_wb_cv), [85](#)
- radf_wb_ps_cv, [43](#), [44](#), [88](#)
- radf_wb_ps_cv(), [61](#), [64](#), [68](#), [70](#), [73](#), [79](#), [82](#),
[85](#), [87](#)
- radf_wb_ps_distr (radf_wb_ps_cv), [88](#)
- rootstamp, [21](#), [90](#)
- rootstamp(), [31](#), [33](#), [35](#), [37](#), [67](#)
- scale_exuber_manual, [92](#)

series_names, 94
series_names<- (series_names), 94
shade (autoplot.radf_obj), 16
sim_blan, 95, 104, 110, 111, 115, 117–119
sim_blan(), 100
sim_coexplosive, 97
sim_common, 98
sim_data, 100
sim_data_wdate (sim_data), 100
sim_div, 101, 105
sim_div(), 100
sim_evans, 96, 102, 105, 115, 117
sim_evans(), 100
sim_falsebubble, 104
sim_fi, 106, 113, 114
sim_innov, 107, 112, 114
sim_mar, 108
sim_msbubble, 110
sim_ps1, 111
sim_psy1, 96–99, 104, 106–111, 113, 113,
116, 117, 120, 121, 123, 124
sim_psy1(), 100
sim_psy2, 96, 104, 107, 115, 116
sim_psy2(), 100
sim_tree, 118
sim_vol_break, 112, 114, 119
sim_vol_cir, 113, 114, 120, 124
sim_vol_garch, 112, 114, 119, 120, 122
sim_vol_sv, 113, 114, 121, 123
ssu_test, 22, 27, 124
ssu_test(), 27, 63, 71, 75, 77, 80, 83
summary.radf_obj, 126
suppressMessages, 51, 66

theme_exuber (scale_exuber_manual), 92
tibble::tibble(), 4–6, 128–130
tidy.ds_radf, 127
tidy.radf_cv, 4, 128
tidy.radf_distr, 129
tidy.radf_obj, 5, 129
tidy_join, 130
tidy_join.radf_obj, 131