Package ‘exuber’

December 18, 2020

Type Package

Title Econometric Analysis of Explosive Time Series

Version 0.4.2

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)
algorithm utilizes the matrix inversion lemma to avoid matrix
inversion which results in significant speed improvements. Simulation
of a variety of periodically-collapsing bubble processes.

License GPL-3

URL https://github.com/kvasilopoulos/exuber

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

Depends R (>= 3.2)

Imports cli (>= 1.1.0), doRNG (>= 1.8.2), doSNOW (>= 1.0.16), dplyr
(>= 1.0.0), foreach (>= 1.4.4), generics (>= 0.0.2), ggplot2
(>= 3.1.1), glue (>= 1.3.1), lubridate (>= 1.7.4), parallel,
purrr (>= 0.3.2), Rcpp (>= 0.12.17), rlang (>= 0.3.4), tibble
(>= 3.0.2), tidyr (>= 0.8.3), vctrs (>= 0.2.4), progress (>=
1.2.2)

Suggests magrittr (>= 1.5), clisymbols (>= 1.2.0), covr (>= 3.2.1),
exuberdata (>= 0.2.0), 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

Additional_repositories https://kvasilopoulos.github.io/drat

Encoding UTF-8

Language en-US

LazyData true
RoxygenNote 7.1.1

NeedsCompilation yes

Author Kostas Vasilopoulos [cre, aut],
Efthymios Pavlidis [aut],
Simon Spavound [aut],
Enrique Martínez-García [aut]

Maintainer Kostas Vasilopoulos <k.vasilopoulo@gmail.com>

Repository CRAN

Date/Publication 2020-12-18 07:30:19 UTC

R topics documented:

'autoplot.ds_radf' ........................................... 3
'autoplot.radf_distr' ...................................... 4
'autoplot.radf_obj' ....................................... 4
calculator_pvalue ......................................... 6
datestamp .................................................. 7
diagnostics .................................................. 8
index-rd ..................................................... 9
install_exuberdata ......................................... 9
psy_minw .................................................... 10
radf .......................................................... 11
radf_crit .................................................... 12
radf_mc_cv .................................................. 13
radf_sb_cv .................................................. 14
radf_wb_cv .................................................. 15
scale_exuber_manual ....................................... 17
series_names ............................................... 18
sim_blan .................................................... 19
sim_data .................................................... 20
sim_div ...................................................... 21
sim_evans ................................................... 23
sim_ps1 ....................................................... 24
sim_psy1 ..................................................... 26
sim_psy2 ..................................................... 28
summary.radf_obj ......................................... 30
tidy.ds_radf ............................................... 31
tidy.radf_cv ............................................... 31
tidy.radf_distr ............................................ 32
tidy.radf_obj ............................................... 33
tidy_join ..................................................... 34
tidy_join.radf_obj ......................................... 34

Index 35
Description

Takes a ds_radf object and returns a ggplot2 object, with a geom_segment() layer.

Usage

```r
## 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

A `ggplot2::ggplot()`

Examples

```r
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.radf_obj

### Description

Takes a radf_distr object and returns a ggplot2 object.

### Usage

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

### Arguments

- `object` - An object of class `radf_distr`.
- `...` - Further arguments passed to methods, used only in `wb_distr` facet options.

### Value

A `ggplot2::ggplot()`

---

autoplot.radf_obj

### Description

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

### Usage

```r
## S3 method for class 'radf_obj'
autoplot(
  object,
  cv = NULL,
  option = c("gsadf", "sadf"),
  min_duration = 0L,
  select_series = NULL,
  include_negative = FALSE,
  shade_opt = shade(),
  include = "DEPRECATED",
  select = "DEPRECATED",
  ...
)

shade(fill = "grey70", opacity = 0.5, ...)
```
Arguments

- **object**
  - An object of class `obj`.

- **cv**
  - An object of class `cv`.

- **option**
  - Whether to apply the "gsadf" or "sadf" methodology (default = "gsadf").

- **min_duration**
  - The minimum duration of an explosive period for it to be reported (default = 0).

- **select_series**
  - A vector of column names or numbers specifying the series to be used in plotting. Note that the order of the series does not alter the order used in plotting.

- **include_negative**
  - If TRUE, plot all variables regardless of rejecting the NULL at the 5 percent significance level.

- **shade_opt**
  - Shading options, typically set using `shade` function.

- **include**
  - Argument name is deprecated and substituted with `include_negative`.

- **select**
  - Argument name is deprecated and substituted with `select_series`.

- **...**
  - Further arguments passed to `ggplot2::facet_wrap` and `ggplot2::geom_rect` for `shade`.

- **fill**
  - The shade color that indicates the exuberance periods.

- **opacity**
  - The opacity of the shade color aka alpha.

Value

A `ggplot2::ggplot()`

Examples

```r
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))

# Or remove the shading completely
autoplot(rsim_data, shade_opt = shade(opacity = 0))

# 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_size_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")

---

**calc_pvalue**  
*Calculate p-values from distr object*

**Description**  
Calculate p-values from distr object

**Usage**  
```
calc_pvalue(x, distr = NULL)
```

**Arguments**  
- **x**  
  A radf_obj object.
- **distr**  
  A radf_distr object.

**Examples**

```r
## Not run:
radf_psy1 <- radf(sim_psy1(100))
calc_pvalue(radf_psy1)

# Using the Wild-Bootstrapped
wb_psy1 <- wb_distr(sim_psy1(100))
calc_pvalue(radf_psy1, wb_psy1)
## End(Not run)
```
**datestamp**

*Date-stamping periods of mildly explosive behavior*

### Description

Computes the origination, termination and duration of episodes during which the time series display explosive dynamics.

### Usage

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

### S3 method for class 'radf_obj'

```r
datestamp(
  object,
  cv = NULL,
  min_duration = 0L,
  option = c("gsadf", "sadf"),
  ...
)
```

### 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).
- `option` Whether to apply the "gsadf" or "sadf" methodology (default = "gsadf").

### Details

Datestamp also stores a vector whose elements take the value of 1 when there is a period of explosive behaviour and 0 otherwise. This output can serve as a dummy variable for the occurrence of exuberance.

### Value

Returns a list containing the estimated origination and termination dates of episodes of explosive behaviour and the corresponding duration.

### References

Examples

```r
sim_data <- radf(sim_data)

ds_data <- datestamp(rsim_data)
ds_data

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

autoplot(ds_data)
```

diagnostics

**Diagnostics on hypothesis testing**

**Description**

Provides information on whether the null hypothesis of a unit root is rejected against the alternative of explosive behaviour for each series in a dataset.

**Usage**

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

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

**Arguments**

- **object**: An object of class `obj`.
- **cv**: An object of class `cv`.
- **...**: Further arguments passed to methods.
- **option**: Whether to apply the "gsadf" or "sadf" methodology (default = "gsadf").

**Details**

Diagnostics also stores a vector whose elements take the value of 1 when there is a period of explosive behaviour and 0 otherwise.

**Value**

Returns a list with the series that reject (positive) and the series that do not reject (negative) the null hypothesis, and at what significance level.
Examples

```r
rsim_data <- radf(sim_data)
diagnostics(rsim_data)
diagnostics(rsim_data, option = "sadf")
```

Description

Retrieve or replace the index of an object.

Usage

```r
index(x, ...)  # Retrieve the index
index(x) <- value  # Replace the index
```

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 the user does not specify an index for the estimation a pseudo-index is generated which is a sequential numeric series. After the estimation, the user can use `index` to retrieve or `index<-` to replace the index. The index can be either numeric or Date.

install_exuberdata

Description

This function wraps the `install.packages` function and offers a faster and more convenient way to install exuberdata.

Usage

```r
install_exuberdata()
```
Examples

```r
if("exuberdata" %in% loadedNamespaces()) {
  exuberdata::radf_crit2
}
```

<table>
<thead>
<tr>
<th>psy_minw</th>
<th>Helper functions in accordance to PSY(2015)</th>
</tr>
</thead>
</table>

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

```r
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:

\[ rule_1 = \delta \log(n) \quad \text{and} \quad rule_2 = \delta \log(n)/n \]

\delta \text{ depends on the frequency of the data and the minimal duration condition.}

References


Examples

```r
psy_minw(100)
psy_ds(100)
```
**radf**  

**Recursive Augmented Dickey-Fuller Test**

**Description**

`radf` returns the recursive univariate and panel Augmented Dickey-Fuller test statistics.

**Usage**

```r
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. The object should not have any NA values.
- `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**

The `radf()` function is vectorized, i.e., it can handle multiple series at once, to improve efficiency. This property also enables the computation of panel statistics internally as a by-product of the univariate estimations with minimal additional cost incurred.

**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

**References**


Examples

```r
# 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_crit  

### Stored Monte Carlo Critical Values

**Description**

A dataset containing Monte Carlo critical values for up to 600 observations generated using the default minimum window. The critical values have been simulated and stored as data to save computation time for the user. The stored critical values can be obtained with the `radf_mc_cv()` function, using `nrep = 2000` and the `seed = 123`.

**Usage**

`radf_crit`

**Format**

A list with lower level lists that contain

- `adf_cv`: Augmented Dickey-Fuller
- `badf_cv`: Backward Augmented Dickey-Fuller
- `sadf_cv`: Supremum Augmented Dickey-Fuller
- `bsadf_cv`: Backward Supremum Augmented Dickey-Fuller
- `gsadf_cv`: Generalized Supremum Augmented Dickey Fuller
radf_mc_cv

Source
Simulated from exuber package function `radf_mc_cv()`.

Examples
## Not run:
all.equal(radf_crit[[50]], radf_mc_cv(50, nrep = 2000, seed = 123))
## End(Not run)

---

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 distribution.

Usage
```r
radf_mc_cv(n, minw = NULL, nrep = 1000L, seed = NULL)
radf_mc_distr(n, minw = NULL, nrep = 1000L, 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. Either NULL or an integer will be used in a call to `set.seed` before simulation. If set, the value is saved as "seed" attribute of the returned value. The default, NULL, will not change rng state, and return `.Random.seed` as the "seed" attribute. Results are different between the parallel and non-parallel option, even if they have the same seed.

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

See Also
- `radf_wb_cv` for wild bootstrap critical values
- `radf_sb_cv` for sieve bootstrap critical values
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)

---

radf_sb_cv

Panel Sieve Bootstrap Critical Values

Description

radf_sb_cv computes critical values for the panel recursive unit root test using the sieve bootstrap procedure outlined in Pavlidis et al. (2016). radf_sb_distr computes the distribution.

Usage

radf_sb_cv(data, minw = NULL, lag = 0L, nboot = 500L, seed = NULL)

radf_sb_distr(data, minw = NULL, lag = 0L, nboot = 500L, seed = NULL)

Arguments

data A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. The object should not have any NA values.

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).

seed An object specifying if and how the random number generator (rng) should be initialized. Either NULL or an integer will be used in a call to set.seed before simulation. If set, the value is saved as "seed" attribute of the returned value. The default, NULL, will not change rng state, and return .Random.seed as the "seed" attribute. Results are different between the parallel and non-parallel option, even if they have the same seed.
**radf_wb_cv**

**Value**

For `radf_sb_cv` A list that contains the critical values for the panel BSADF and panel GSADF test statistics. For `radf_wb_dist` a numeric vector that contains the distribution of the panel GSADF statistic.

**References**


**See Also**

`radf_mc_cv` for Monte Carlo critical values and `radf_wb_cv` for wild Bootstrap critical values

**Examples**

```r
rsim_data <- radf(sim_data, lag = 1)
# Critical values should have the same lag length with \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)
```

**radf_wb_cv**

Wild Bootstrap Critical Values

**Description**

`radf_wb_cv` performs the Harvey et al. (2016) wild bootstrap re-sampling scheme, which is asymptotically robust to non-stationary volatility, to generate critical values for the recursive unit root tests. `radf_wb_distr` computes the distribution.

**Usage**

```r
radf_wb_cv(data, minw = NULL, nboot = 500L, dist_rad = FALSE, seed = NULL)
radf_wb_distr(data, minw = NULL, nboot = 500L, dist_rad = FALSE, seed = NULL)
```
Arguments

data A univariate or multivariate numeric time series object, a numeric vector or matrix, or a data.frame. The object should not have any NA values.

minw A positive integer. The minimum window size (default = (0.01 + 1.8/√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.

seed An object specifying if and how the random number generator (rng) should be initialized. Either NULL or an integer will be used in a call to set.seed before simulation. If set, the value is saved as "seed" attribute of the returned value. The default, NULL, will not change rng state, and return .Random.seed as the "seed" attribute. Results are different between the parallel and non-parallel option, even if they have the same seed.

Details

This approach involves applying a wild bootstrap re-sampling scheme to construct the bootstrap analogue of the Phillips et al. (2015) test which is asymptotically robust to non-stationary volatility.

Value

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

References


See Also

radf_mc_cv for Monte Carlo critical values and radf_sb_cv for sieve bootstrap critical values.

Examples

# Default minimum window
wb <- radf_wb_cv(sim_data)
tidy(wb)

# Change the minimum window and the number of bootstraps
wb2 <- radf_wb_cv(sim_data, nboot = 600, minw = 20)
tidy(wb2)
# Simulate distribution
wdist <- radf wb_distr(sim_data)

autoplot(wdist)

---

**scale_exuber_manual**  
*Exuber scale and theme functions*

**Description**

*scale_exuber_manual* allows specifying the color, size and linetype in *autoplot.radf_obj* mappings. *theme_exuber* is a complete theme which control all non-data display.

**Usage**

```r
scale_exuber_manual(
  color_values = c("red", "blue"),
  linetype_values = c(2, 1),
  size_values = c(0.8, 0.7)
)

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.

- **size_values**  
a set of size values to map data values to.

- **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
series_names

Retrieve/Replace series names

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 n ordered vector of the same length as the "index" attribute of x.

Examples

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

rdf <- radf(dta)

series_names(rfd) <- c("OneBubble", "TwoBubbles")
Simulation of a Blanchard (1979) bubble process

*Description*

Simulation of a Blanchard (1979) rational bubble process.

*Usage*

```r
sim_blan(n, pi = 0.7, sigma = 0.03, r = 0.05, b0 = 0.1, seed = NULL)
```

*Arguments*

- `n` A positive integer specifying the length of the simulated output series.
- `pi` A positive value in (0, 1) which governs the probability of the bubble continuing to grow.
- `sigma` A positive scalar indicating the standard deviation of the innovations.
- `r` A positive scalar that determines the growth rate of the bubble process.
- `b0` The initial value of the bubble.
- `seed` An object specifying if and how the random number generator (rng) should be initialized. Either NULL or an integer will be used in a call to `set.seed` before simulation. If set, the value is saved as "seed" attribute of the returned value. The default, NULL, will not change rng state, and return `.Random.seed` as the "seed" attribute. Results are different between the parallel and non-parallel option, even if they have the same seed.

*Details*

Blanchard’s bubble process has two regimes, which occur with probability $\pi$ 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 $\pi$:

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

With probability $1 - \pi$:

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

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

*Value*

A numeric vector of length $n$.

*References*

See Also

sim_psy1, sim_psy2, sim_evans

Examples

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

---

**sim_data**

**Simulated dataset**

Description

An artificial dataset containing series simulated from data generating processes widely used in the literature on speculative bubbles.

Usage

```r
sim_data

sim_data_wdate
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 100 rows and 5 columns.

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

See Also

sim_psy1 sim_psy1 sim_evans sim_div sim_blan

Examples

```r
# Not run:
# The dataset can be easily replicated with the code below
library(tibble)
set.seed(1122)
sim_data <- tibble(
  sim_psy1 = sim_psy1(100),
  sim_psy2 = sim_psy2(100),
  sim_evans = sim_evans(100),
  sim_div = sim_div(100),
  sim_blan = sim_blan(100)
)
sim_data_wdate <- tibble(
  psy1 = sim_psy1(100),
  psy2 = sim_psy2(100),
  evans = sim_evans(100),
```
sim_div

    div = sim_div(100),
    blan = sim_blan(100),
    date = seq(as.Date("2000-01-01"), by = "month", length.out = 100)
)

## End(Not run)

__sim_div__

### Simulation of dividends

__Description__

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

__Usage__

```r
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. Either NULL or an integer will be used in a call to set. seed before simulation. If set, the value is saved as "seed" attribute of the returned value. The default, NULL, will not change rng state, and return .Random.seed as the "seed" attribute. Results are different between the parallel and non-parallel option, even if they have the same seed.
Details

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

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

where \( \epsilon \sim N(0, \sigma^2) \). 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:

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

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 \( \epsilon \sim N(0, \sigma^2) \). 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


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)
SimEvans

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. Either NULL or an integer will be used in a call to set.seed before simulation. If set, the value is saved as "seed" attribute of the returned value. The default, NULL, will not change rng state, and return .Random.seed as the "seed" attribute. Results are different between the parallel and non-parallel option, even if they have 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 $(1 + r)\pi^{-1}$:

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

where $\theta$ is a binary variable that takes the value 0 with probability $1 - \pi$ and 1 with probability $\pi$. 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**


**See Also**

`sim_psy1`, `sim_psy2`, `sim_blan`

**Examples**

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

---

**Description**

The new generating process considered here 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 process is modeled directly as a transient mildly integrated process that covers an explicit period of market collapse. Third, a market recovery date is introduced to capture the return to normal market behavior:*

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

In order to provide the duration of the collapse period $tr$ as $tr = tf + 0.2n$, you have to provide $tf$ as well.
Usage

```r
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
)
```

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. Either NULL or an integer will be used in a call to `set.seed` before simulation. If set, the value is saved as "seed" attribute of the returned value. The default, NULL, will not change rng state, and return .Random.seed as the "seed" attribute. Results are different between the parallel and non-parallel option, even if they have the same seed.

Value

A numeric vector of length `n`.

References

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

The following function generates a time series which 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
)

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. Either NULL or an integer will be used in a call to set.seed before simulation. If set, the value is saved as "seed" attribute of the returned value. The default, NULL, will not change rng state, and return .Random.seed as the "seed" attribute. Results are different between the parallel and non-parallel option, even if they have the same seed.

Details

The data generating process is described by the following equation:

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

where the autoregressive coefficient \( \delta_T \) 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 \( X' = O_p(1) \), \( \tau_e = [T \tau_e] \) dates the origination of the bubble, and \( \tau_f = [T \tau_f] \) 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 \( \delta_T \); 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


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
Simulation of a two-bubble process

Description

The following data generating process is similar to `sim_psy1`, with the difference that there are two episodes of mildly explosive dynamics.

Usage

```r
sim_psy2(n = 200, te1 = 100, tf1 = 150, seed = 123) %>% autoplot()
```

```r
sim_psy2(n = 200, 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. Either NULL or an integer will be used in a call to `set.seed` before simulation. If set, the value is saved as "seed" attribute of the returned value. The default, NULL, will not change rng state, and return .Random.seed as the "seed" attribute. Results are different between the parallel and non-parallel option, even if they have the same seed.
Details

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

\[ X_t = X_{t-1}1\{t \in N_0\} + \delta_T X_{t-1}1\{t \in B_1 \cup B_2\} + \left( \sum_{k=\tau_{1f}+1}^{t} \epsilon_k + X_{\tau_{1f}} \right)1\{t \in N_1\} \]

\[ + \left( \sum_{l=\tau_{2f}+1}^{t} \epsilon_l + X_{\tau_{2f}} \right)1\{t \in N_2\} + \epsilon_t1\{t \in N_0 \cup B_1 \cup B_2\} \]

where the autoregressive coefficient \( \delta_T \) is:

\[ \delta_T = 1 + eT^{-\alpha} \]

with \( e > 0, \alpha \in (0, 1), \epsilon \sim iid(0, \sigma^2) \), \( N_0 = [1, \tau_{1e}] \), \( B_1 = [\tau_{1e}, \tau_{1f}] \), \( N_1 = (\tau_{1f}, \tau_{2e}] \), \( B_2 = [\tau_{2e}, \tau_{2f}] \), \( N_2 = (\tau_{2f}, \tau] \), where \( \tau \) is the last observation of the sample. The observations \( \tau_{1e} = [Tr_{1e} \tau_{1f} \tau_{1e}] \) and \( \tau_{1f} = [Tr_{1f} \tau_{1f}] \) 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, \( X_t \) resums a martingale path until time \( \tau_{2e} - 1 \), and a second episode of exuberance begins at \( \tau_{2f} \). Exuberance lasts lasts until \( \tau_{2f} \) 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 \( \tau \). The duration of the first bubble is assumed to be longer than that of the second bubble, i.e. \( \tau_{1f} - \tau_{1e} > \tau_{2f} - \tau_{2e} \).


Value

A numeric vector of length \( n \).

References


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()
Summary: \texttt{radf_obj} 

\textbf{Summary method for \texttt{radf} models that consist of \texttt{radf_obj} and \texttt{radf_cv}.}

\section*{Usage}

\texttt{## S3 method for class 'radf_obj'
 summary(object, cv = NULL, ...)}

\section*{Arguments}

- \texttt{object}: An object of class \texttt{radf_obj}. The output of \texttt{radf()}.  
- \texttt{cv}: An object of class \texttt{radf_cv}. The output of \texttt{radf_mc_cv()}, \texttt{radf_wb_cv()} or \texttt{radf_sb_cv()}.  
- \texttt{...}: Further arguments passed to methods. Not used.

\section*{Value}

Returns a list of summary statistics, which include the estimated ADF, SADF, and GSADF test statistics and the corresponding critical values.

\section*{Examples}

```r
# 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)
```
### tidy.ds_radf

**Tidy a ds_radf object**

**Description**

Summarizes information about ds_radf object.

**Usage**

```r
## S3 method for class 'ds_radf'
tidy(x, ...)
```

**Arguments**

- `x` An object of class ds_radf.
- `...` Further arguments passed to methods. Not used.

### tidy.radf_cv

**Tidy a radf_cv object**

**Description**

Summarizes information about radf_cv object.

**Usage**

```r
## S3 method for class 'radf_cv'
tidy(x, format = c("wide", "long"), ...)
```

```r
## S3 method for class 'radf_cv'
augment(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

```r
mc <- radf_mc_cv(100)

# Get the critical values
tidy(mc)

# Get the critical value sequences
augment(mc)
```

tidy.radf_distr  
*Tidy a radf_distr object*

Description

Summarizes information about `radf_distr` object.

Usage

```r
## 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

```r
## Not run:
mc <- mc_cv(n = 100)

 tidy(mc)
```

## End(Not run)
tidy.radf_obj

Tidy a radf_obj object

Description
Summarizes information about radf_obj object.

Usage

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

Arguments

- `x` An object of class radf_obj.
- `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

```r
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**

```r
tidy_join(x, y, ...)

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

**Arguments**

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

---

tidy_join.radf_obj  
*Tidy into a joint model*

**Description**

Tidy or augment and then join objects of class `radf_obj` and `radf_cv`. The object of reference is the `radf_cv`. For example, if panel critical values are provided the function will return the panel test statistic.

**Usage**

```r
## S3 method for class 'radf_obj'
tidy_join(x, y = NULL, ...)

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

**Arguments**

- `x` An object of class `radf_obj`.
- `y` An object of class `radf_cv`. The output will depend on the type of critical value.
- `...` Further arguments passed to methods. Not used.

**Details**

`tidy_join` also calls `augment_join` when `cv` is of class `sb_cv`. 
## Index

* datasets
  - `radf_crit`, 12
  - `sim_data`, 20

  `augment.radf_cv(tidy.radf_cv)`, 31
  `augment.radf_obj(tidy.radf_obj)`, 33
  `augment_join(tidy_join)`, 34
  `augment_join.radf_obj(tidy_join.radf_obj)`, 34

  `autoplot.ds_radf`, 3
  `autoplot.radf_distr`, 4
  `autoplot.radf_obj`, 4

  `calc_pvalue`, 6

  `datestamp`, 7
  `datestamp()`, 3
  `diagnostics`, 8

  `geom_segment()`, 3
  `ggplot2::ggplot()`, 3–5

  `index(index-rd)`, 9
  `index-rd`, 9
  `index<- (index-rd)`, 9
  `install_exuberdata`, 9

  `psy_ds (psy_minw)`, 10
  `psy_minw`, 10

  `radf`, 11
  `radf()`, 30
  `radf_crit`, 12
  `radf.mc_cv`, 13, 15, 16
  `radf.mc_cv()`, 12, 13, 30
  `radf.mc_distr(radf.mc_cv)`, 13
  `radf.sb_cv`, 13, 14, 16
  `radf.sb_cv()`, 30
  `radf.sb_distr(radf.sb_cv)`, 14
  `radf.wb_cv`, 13, 15, 15
  `radf.wb_cv()`, 30

  `radf.wb_distr (radf.wb_cv)`, 15

  `scale_exuber_manual`, 17
  `series_names`, 18
  `series_names<- (series_names)`, 18
  `shade (autoplot.radf_obj)`, 4
  `sim_blan`, 19, 24, 27, 29
  `sim_data`, 20
  `sim_data.wdate (sim_data)`, 20
  `sim_div`, 21
  `sim_evans`, 20, 23, 27, 29
  `sim_psi1`, 24
  `sim_psi2`, 20, 24, 27, 28
  `summary.radf_obj`, 30

  `theme_exuber (scale_exuber_manual)`, 17
  `tibble::tibble()`, 31–33
  `tidy.ds_radf`, 31
  `tidy.radf_cv`, 31
  `tidy.radf_distr`, 32
  `tidy.radf_obj`, 33
  `tidy_join`, 34
  `tidy_join.radf_obj`, 34

  `tibble::tibble()`, 31–33
  `tidy.ds_radf`, 31
  `tidy.radf_cv`, 31
  `tidy.radf_distr`, 32
  `tidy.radf_obj`, 33
  `tidy_join`, 34
  `tidy_join.radf_obj`, 34