Package: blocklength (via r-universe)

October 26, 2024

Type Package

Title Select an Optimal Block-Length to Bootstrap Dependent Data (Block Bootstrap)

Version 0.1.5.9000

Maintainer Alec Stashevsky <alec@alecstashevsky.com>

Description A set of functions to select the optimal block-length for a dependent bootstrap (block-bootstrap). Includes the Hall, Horowitz, and Jing (1995) <doi:10.1093/biomet/82.3.561> cross-validation method and the Politis and White (2004) <doi:10.1081/ETC-120028836> Spectral Density Plug-in method, including the Patton, Politis, and White (2009) <doi:10.1080/07474930802459016> correction with a corresponding set of S3 plot methods.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.1.2

Suggests testthat, covr, parallel, knitr, rmarkdown

Imports tseries, stats

URL https://alecstashevsky.com/r/blocklength, https://github.com/Alec-Stashevsky/blocklength

BugReports https://github.com/Alec-Stashevsky/blocklength/issues

VignetteBuilder knitr

Repository https://alec-stashevsky.r-universe.dev

RemoteUrl https://github.com/alec-stashevsky/blocklength

RemoteRef HEAD

RemoteSha 4c9596d59fcd25dca20af0e2b59b22366fc50309

Contents

---

hhj *Hall, Horowitz, and Jing (1995) "HHJ" Algorithm to Select the Optimal Block-Length*

Description

Perform the Hall, Horowitz, and Jing (1995) "HHJ" cross-validation algorithm to select the optimal block-length for a bootstrap on dependent data (block-bootstrap). Dependent data such as stationary time series are suitable for usage with the HHJ algorithm.

Usage

```
hhj(
  series,
  nb = 100L,
  n_iter = 10L,
  pilot_block_length = NULL,
  sub_sample = NULL,
  k = "two-sided",
  bofb = 1L,
  search_grid = NULL,
  grid_step = c(1L, 1L),
  cl = NULL,
  verbose = TRUE,
  plots = TRUE
)
```

Arguments

`series` a numeric vector or time series giving the original data for which to find the optimal block-length for.

`nb` an integer value, number of bootstrapped series to compute.

`n_iter` an integer value, maximum number of iterations for the HHJ algorithm to compute.

`pilot_block_length` a numeric value, the block-length ($l*$ in *HHJ*) for which to perform initial block bootstraps.

`sub_sample` a numeric value, the length of each overlapping subsample, m in *HHJ*.

`k`	a character string, either `"bias/variance"`, `"one-sided"`, or `"two-sided"` depending on the desired object of estimation. If the desired bootstrap statistic is bias or variance then select `"bias/variance"` which sets $k = 3$ per HHJ. If the object of estimation is the one-sided or two-sided distribution function, then set `k = "one-sided"` or `k = "two-sided"` which sets $k = 4$ and $k = 5$, respectively. For the purpose of generating symmetric confidence intervals around an unknown parameter, `k = "two-sided"` (the default) should be used.
`bofb`	a numeric value, length of the basic blocks in the *block-of-blocks* bootstrap, *see* `m =` for `tsbootstrap` and Kunsch (1989).
`search_grid`	a numeric value, the range of solutions around $l*$ to evaluate within the MSE function *after* the first iteration. The first iteration will search through all the possible block-lengths unless specified in `grid_step =`.
`grid_step`	a numeric value or vector of at most length 2, the number of steps to increment over the subsample block-lengths when evaluating the MSE function. If `grid_step = 1` then each block-length will be evaluated in the MSE function. If `grid_step > 1`, the MSE function will search over the sequence of block-lengths from `1` to `m` by `grid_step`. If `grid_step` is a vector of length 2, the first iteration will step by the first element of `grid_step` and subsequent iterations will step by the second element.
`cl`	a cluster object, created by package **parallel**, **doParallel**, or **snow**. If `NULL`, no parallelization will be used.
`verbose`	a logical value, if set to `FALSE` then no interim messages are output to the console. Error messages will still be output. Default is `TRUE`.
`plots`	a logical value, if set to `FALSE` then no interim plots are output to the console. Default is `TRUE`.

Details

The HHJ algorithm is computationally intensive as it relies on a cross-validation process using a type of subsampling to estimate the mean squared error (MSE) incurred by the bootstrap at various block-lengths.

Under-the-hood, `hhj()` makes use of `tsbootstrap`, *see* Trapletti and Hornik (2020), to perform the moving block-bootstrap (or the *block-of-blocks* bootstrap by setting `bofb > 1`) according to Kunsch (1989).

Value

an object of class 'hhj'

References

Adrian Trapletti and Kurt Hornik (2020). tseries: Time Series Analysis and Computational Finance. R package version 0.10-48.

Kunsch, H. (1989) The Jackknife and the Bootstrap for General Stationary Observations. The Annals of Statistics, 17(3), 1217-1241. Retrieved February 16, 2021, from http://www.jstor.org/stable/2241719

Peter Hall, Joel L. Horowitz, Bing-Yi Jing, On blocking rules for the bootstrap with dependent data, Biometrika, Volume 82, Issue 3, September 1995, Pages 561-574, DOI: doi:10.1093/biomet/82.3.561

Examples

```
# Generate AR(1) time series
sim <- stats::arima.sim(list(order = c(1, 0, 0), ar = 0.5),
                        n = 500, innov = rnorm(500))

# Calculate optimal block length for series
hhj(sim, sub_sample = 10)


# Use parallel computing
library(parallel)

# Make cluster object with 2 cores
cl <- makeCluster(2)

# Calculate optimal block length for series
hhj(sim, cl = cl)
```

`plot.hhj` *Plot MSE Function for HHJ Algorithm*

Description

S3 Method for objects of class 'hhj'

Usage

```
## S3 method for class 'hhj'
plot(x, iter = NULL, ...)
```

Arguments

`x`	an object of class 'hhj'
`iter`	a vector of `hhj()` iterations to plot. `NULL`. All iterations are plotted by default.
`...`	Arguments passed on to `base::plot` `y` the y coordinates of points in the plot, *optional* if `x` is an appropriate structure.

Value

No return value, called for side effects

Examples

```
# Generate AR(1) time series
sim <- stats::arima.sim(list(order = c(1, 0, 0), ar = 0.5),
                        n = 500, innov = rnorm(500))

# Generate 'hhj' class object of optimal block length for series
hhj <- hhj(sim, sub_sample = 10)

## S3 method for class 'hhj'
plot(hhj)
```

`plot.pwsd` *Plot Correlogram for Politis and White Auto−Correlation Implied Hypothesis Test*

Description

S3 Method for objects of class 'pwsd' *See* `?plot.acf` of the **stats** package for more customization options on the correlogram, from which `plot.pwsd` is based

Usage

```
## S3 method for class 'pwsd'
plot(x, c = NULL, main = NULL, ylim = NULL, ...)
```

Arguments

`x`	an of object of class 'pwsd' or 'acf'
`c`	a numeric value, the constant which acts as the significance level for the implied hypothesis test. Defaults to `qnorm(0.975)` for a two-tailed 95% confidence level. Politis and White (2004) suggest `c = 2`.
`main`	an overall title for the plot, if no string is supplied a default title will be populated. *See* `title`
`ylim`	a numeric of length 2 giving the y-axis limits for the plot
`...`	Arguments passed on to `base::plot`
	`y` the y coordinates of points in the plot, *optional* if `x` is an appropriate structure.

Value

No return value, called for side effects

Examples

```
# Use S3 Method

# Generate AR(1) time series
sim <- stats::arima.sim(list(order = c(1, 0, 0), ar = 0.5),
                        n = 500, innov = rnorm(500))

b <- pwsd(sim, round = TRUE, correlogram = FALSE)
plot(b)
```

---

pwsd — *Politis and White (2004) Spectral Density "PWSD" Automatic Block-Length Selection*

Description

Run the Automatic Block-Length selection method proposed by Politis and White (2004) and corrected in Patton, Politis, and White (2009). The method is based on spectral density estimation via flat-top lag windows of Politis and Romano (1995). This code was adapted from `b.star` to add functionality and include correlogram support including an S3 method, *see* Hayfield and Racine (2008).

Usage

```
pwsd(
  data,
  K_N = NULL,
  M_max = NULL,
  m_hat = NULL,
  b_max = NULL,
  c = NULL,
  round = FALSE,
  correlogram = TRUE
)
```

Arguments

`data` — an nxk data.frame, matrix, or vector (if $k = 1$) where the optimal block-length will be computed for each of the k columns.

`K_N` — an integer value, the maximum lags for the auto-correlation, rho_k, which to apply the *implied hypothesis* test. Defaults to `max(5, log(N))`. *See* Politis and White (2004) footnote c.

`M_max` — an integer value, the upper-bound for the optimal number of lags, M, to compute the auto-covariance for. *See* Theorem 3.3 (ii) of Politis and White (2004).

`m_hat`	an integer value, if set to `NULL` (the default), then `m_hat` is estimated as the smallest integer after which the correlogram appears negligible for `K_N` lags. In problematic cases, setting `m_hat` to an integer value can be used to override the estimation procedure.
`b_max`	a numeric value, the upper-bound for the optimal block-length. Defaults to `ceiling(min(3 * sqrt(n), n / 3))` per Politis and White (2004).
`c`	a numeric value, the constant which acts as the significance level for the implied hypothesis test. Defaults to `qnorm(0.975)` for a two-tailed 95% confidence level. Politis and White (2004) suggest `c = 2`.
`round`	a logical value, if set to `FALSE` then the final block-length output will not be rounded, the default. If set to `TRUE` the final estimates for the optimal block-length will be rounded to whole numbers.
`correlogram`	a logical value, if set to `TRUE` a plot of the correlogram (*i.e.* a plot of $R(k)$ vs. k) will be output to the console. If set to `FALSE`, no interim plots will be output to the console, but may be plotted later using the corresponding S3 method, plot.pwsd.

Value

an object of class 'pwsd'

References

Andrew Patton, Dimitris N. Politis & Halbert White (2009) Correction to "Automatic Block-Length Selection for the Dependent Bootstrap" by D. Politis and H. White, Econometric Review, 28:4, 372-375, DOI: doi:10.1080/07474930802459016

Dimitris N. Politis & Halbert White (2004) Automatic Block-Length Selection for the Dependent Bootstrap, Econometric Reviews, 23:1, 53-70, DOI: doi:10.1081/ETC120028836

Politis, D.N. and Romano, J.P. (1995), Bias-Corrected Nonparametric Spectral Estimation. Journal of Time Series Analysis, 16: 67-103, DOI: doi:10.1111/j.14679892.1995.tb00223.x

Tristen Hayfield and Jeffrey S. Racine (2008). Nonparametric Econometrics: The np Package. Journal of Statistical Software 27(5). DOI: doi:10.18637/jss.v027.i05

Examples

```
# Generate AR(1) time series
sim <- stats::arima.sim(list(order = c(1, 0, 0), ar = 0.5),
                        n = 500, innov = rnorm(500))

# Calculate optimal block length for series
pwsd(sim, round = TRUE)


# Use S3 Method
b <- pwsd(sim, round = TRUE, correlogram = FALSE)
plot(b)
```

Index