Package: maxbootR (via r-universe)

May 17, 2026

Type Package

Title Efficient Bootstrap Methods for Block Maxima

Version 1.0.0

Description Implements state-of-the-art block bootstrap methods for extreme value statistics based on block maxima. Includes disjoint blocks, sliding blocks, relying on a circular transformation of blocks. Fast C++ backends (via 'Rcpp') ensure scalability for large time series.

License GPL (>= 3)

Encoding UTF-8

URL https://torbenstaud.github.io/maxbootR/, https://github.com/torbenstaud/maxbootR

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

LinkingTo Rcpp

Imports Rcpp, evd

Suggests testthat (>= 3.0.0), covr, knitr, rmarkdown, ggplot2, dplyr, lubridate, tidyr

Config/testthat/edition 3

VignetteBuilder knitr

Depends R (>= 2.10)

LazyData true

BugReports https://github.com/torbenstaud/maxbootR/issues

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

Date/Publication 2025-05-03 16:43:35 UTC

RemoteUrl https://github.com/torbenstaud/maxbootr

RemoteRef HEAD

RemoteSha 3c2941d140d5d8c732cdb613795348d6c367b743

Contents

`blockmax` *Compute Block Maxima from a Time Series*

Description

Extracts block maxima from a univariate numeric vector or matrix using disjoint, sliding, or circular (k-dependent) block schemes.

Usage

```
blockmax(xx, block_size, type = "sb", k = 2)
```

Arguments

`xx`	A numeric vector or matrix. For matrix input, each row is treated as a separate univariate series.
`block_size`	Positive integer. Size of each block for maxima extraction.
`type`	Character. Type of block maxima to compute. One of: `"db"` (disjoint blocks), `"sb"` (sliding blocks), or `"cb"` (circular blocks with k offsets).
`k`	Integer (only used if `type = "cb"`). Blocking parameter which controls the number of blocks contained in a block of blocks. Must be an integer between 1 and `floor(length(xx) / block_size)`.

Value

A numeric vector (if `xx` is a vector) or a matrix (if `xx` is a matrix). Each entry contains block maxima computed according to the selected method.

Examples

```
if (requireNamespace("maxbootR", quietly = TRUE)) {
  set.seed(42)
  x <- rnorm(100)

  # Disjoint blocks of size 10
  bm_db <- blockmax(xx = x, block_size = 10, type = "db")

  # Sliding blocks of size 10
  bm_sb <- blockmax(xx = x, block_size = 10, type = "sb")
```

```
  # Circular blocks of size 10 with blocking parameter k = 2
  bm_cb <- blockmax(xx = x, block_size = 10, type = "cb", k = 2)
}
```

`logret_data` *Example Log Return Time Series*

Description

A tibble containing daily negative log returns of closing prices for the S&P 500 stock market index. The observation period spans 20 trading years: 1995-01-01 to 2024-12-31.

Usage

```
data(logret_data)
```

Format

A tibble with 7,550 rows and 2 columns:

day Date of observation (class `Date`)

neg_log_ret Negative log return (numeric)

Details

The data was obtained using the `quantmod` package with **Yahoo Finance** as the source.

`maxbootr` *Bootstrap Estimation for Block Maxima*

Description

Performs bootstrap resampling for various block maxima estimators (mean, variance, GEV parameters, quantile, return level) using either disjoint or sliding block methods.

Usage

```
maxbootr(
  xx,
  est,
  block_size,
  B = 1000,
  type = "sb",
  seed = 1,
  p = NULL,
  annuity = NULL
)
```

Arguments

xx	A numeric vector or array containing univariate samples. For multivariate cases, samples should be arranged in rows.
est	A string specifying the estimator to apply. Must be one of "mean", "var", "gev", "quantile", or "rl".
block_size	Integer. Size of each block used in the block maxima extraction.
B	Integer. Number of bootstrap replicates to generate.
type	Type of block bootstrapping: "db" for disjoint blocks or "sb" for sliding blocks (internally approximated via circular blocks).
seed	Integer. Seed for reproducibility.
p	Optional numeric value in (0,1). Required if est = "quantile".
annuity	Optional numeric value > 1. Required if est = "rl" for return level estimation.

Value

A numeric vector with B rows for scalar estimators. If est = "gev", a matrix with B rows is returned. Each row contains 3 estimated GEV parameters (location, scale, shape).

Examples

```
if (requireNamespace("maxbootR", quietly = TRUE)) {
  library(maxbootR)
  set.seed(123)
  x <- rnorm(100)

  # Bootstrap mean using sliding blocks
  boot_mean <- maxbootr(x, est = "mean", block_size = 10, B = 20, type = "sb")

  # Bootstrap variance using disjoint blocks
  boot_var <- maxbootr(x, est = "var", block_size = 10, B = 20, type = "db")

  # Bootstrap 95%-quantile of block maxima using sliding blocks
  boot_q <- maxbootr(x, est = "quantile", block_size = 10, B = 20, type = "db", p = 0.95)
}
```

---

temp_data *Example Temperature Time Series*

Description

This dataset contains daily temperature measurements in °C from the Hohenpeißenberg weather station in Germany, covering 145 years: 1878-01-01 to 2023-12-31.

Usage

```
data(temp_data)
```

Format

A tibble with 52,960 rows and 2 columns:

day Date of observation (class `Date`)

temp Temperature measured in °C (numeric)

Details

The data was obtained from the Open Data Server of the German Meteorological Service (Deutscher Wetterdienst, DWD): https://opendata.dwd.de/ and thus, is protected by law. It is reused under the Creative Commons license CC BY 4.0.

Index