Package: modernBoot (via r-universe)

October 8, 2026

Type Package

Title Modern Resampling Methods: Bootstraps, Wild, Block, Permutation, and Selection Guidance

Version 0.1.1

Author Ibrahim Kholil Rakib [aut, cre]

Maintainer Ibrahim Kholil Rakib <ikrakib1010@gmail.com>

Description Implements modern resampling and permutation methods for robust statistical inference without restrictive parametric assumptions. Provides bias-corrected and accelerated (BCa) bootstrap (Efron and Tibshirani (1993) <doi:10.1201/9780429246593>), wild bootstrap for heteroscedastic regression (Liu (1988) <doi:10.1214/aos/1176351062>, Davidson and Flachaire (2008) <doi:10.1016/j.jeconom.2008.08.003>), block bootstrap for time series (Politis and Romano (1994) <doi:10.1080/01621459.1994.10476870>), and permutation-based multiple testing correction (Westfall and Young (1993) <ISBN:0-471-55761-7>). Methods handle non-normal data, heteroscedasticity, time series correlation, and multiple comparisons.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.0)

Imports stats, boot, future, future.apply

Suggests testthat (>= 3.0.0), covr, pkgdown, rhub

URL https://github.com/ikrakib/modernBoot

BugReports https://github.com/ikrakib/modernBoot/issues

Config/testthat/edition 3

RoxygenNote 7.3.3

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

Date/Publication 2026-08-02 07:36:12 UTC

RemoteUrl https://github.com/ikrakib/modernboot

RemoteRef HEAD

RemoteSha d22177995503a47fac3e3215d2d7c3a197fa21c2

Contents

auto_select_method 2
bca_ci 3
bs_mean 4
compare_methods_sim 5
moving_block_boot 6
perm_maxT 7
perm_test_2sample 8
stationary_boot 9
studentized_ci 10
wild_boot_lm 12

Index **14**

`auto_select_method` *Automatic Resampling Method Selection*

Description

Inspects data structure and recommends an appropriate resampling method.

Usage

```
auto_select_method(data, cluster = NULL)
```

Arguments

`data`	numeric vector, matrix, or time series object representing data to analyze. For vectors: univariate sample. For matrices/data.frames: multivariate data with rows=observations, columns=variables. For ts objects: automatically detected as time series.
`cluster`	optional vector of cluster IDs (integer or factor). Length must equal nrow(data) for matrices or length(data) for vectors. Identifies hierarchical grouping (e.g., repeated measures within subjects, multiple samples from same source). If NULL (default), clustering is not considered. If provided, triggers clustered bootstrap recommendation.

Details

Decision logic: - If clusters provided: use clustered bootstrap - If time series detected: use block or stationary bootstrap - If multivariate matrix: use permutation maxT - Otherwise (IID data): use standard nonparametric bootstrap

Value

A list with two character string elements:

`method`	recommended resampling method name. Values: "clustered_boot" (if cluster provided), "block_boot" (if time series detected), "perm_maxT" (if multivariate matrix detected), "nonparametric_boot" (default for IID univariate data)
`rationale`	human-readable explanation of recommendation. Describes why chosen method is appropriate and what structure the data exhibits.

Examples

```
# IID data
x <- rnorm(50)
auto_select_method(x)

# Time series
ts_data <- arima.sim(n = 100, list(ar = 0.7))
auto_select_method(ts_data)
```

---

bca_ci *Bias-Corrected and Accelerated (BCa) Bootstrap Confidence Interval*

Description

Computes BCa confidence interval for the mean using the boot package.

Usage

```
bca_ci(x, R = 2000, conf = 0.95)
```

Arguments

`x`	numeric vector of data.
`R`	integer number of bootstrap replicates (default 2000).
`conf`	numeric confidence level between 0 and 1 (default 0.95).

Value

A list with elements boot (object) and ci (matrix).

Examples

```
set.seed(42)
x <- rnorm(50, mean = 5, sd = 2)
result <- bca_ci(x, R = 500)
result$ci
```

---

bs_mean *Nonparametric Bootstrap Confidence Interval for the Mean*

Description

Compute a bootstrap distribution for the sample mean.

Usage

```
bs_mean(x, R = 2000, conf = 0.95)
```

Arguments

x	numeric vector of data.
R	integer number of bootstrap replicates (default 2000).
conf	numeric confidence level between 0 and 1 (default 0.95).

Value

A list with elements stat (mean), boot (replicates), and ci (interval).

Examples

```
set.seed(42)
x <- rnorm(50, mean = 5, sd = 2)
result <- bs_mean(x, R = 500)
result$ci
```

`compare_methods_sim` *Simulation Study: Compare Bootstrap Methods*

Description

Runs a simulation study comparing different bootstrap and permutation methods. Useful for empirical evaluation of method performance and coverage.

Usage

```
compare_methods_sim(data_generator, Rsim = 100, Rboot = 1000, parallel = TRUE)
```

Arguments

`data_generator`	function taking no arguments and returning numeric vector of data. Called Rsim times to generate independent datasets. Example: `function() rnorm(50, mean=5, sd=2)` generates normal data. Function must not have side effects; should be deterministic given seed.
`Rsim`	integer number of simulated datasets to generate (default 100). Larger values (500+) provide more stable coverage rate estimates but increase computation time. Each simulation applies both bs_mean and bca_ci methods independently. Must be >= 1.
`Rboot`	integer number of bootstrap replicates used within each method per dataset (default 1000). Matches R parameter passed to bs_mean and bca_ci. Larger values (2000-5000) improve interval accuracy but increase total computation (total bootstrap samples = Rsim * Rboot). Must be >= 1.
`parallel`	logical enable parallel computation across simulations (default TRUE). Uses `multisession` plan to distribute simulations across available cores. User can override with `plan(future::sequential)` before calling for sequential execution.

Details

This function repeatedly generates data from data_generator and applies bs_mean and bca_ci methods. Results can be used to study coverage rates, interval width, and method robustness.

Value

A list of length Rsim. Each element is a list with two components:

`bs`	result from `bs_mean` applied to simulated dataset. Is NA if generation or computation failed.
`bca`	result from `bca_ci` applied to same dataset. Is NA if generation or computation failed.

Use to assess coverage rates: did true parameter fall within returned CIs? Interval width variation indicates method stability.

Examples

```
# Fast example (runs in < 5 sec) - UNWRAPPED
set.seed(42)
generator <- function() rnorm(20)
# Very small simulation for demonstration
results_fast <- compare_methods_sim(generator, Rsim = 5, Rboot = 100)


# Realistic simulation (takes > 5 sec) - WRAPPED in \donttest
set.seed(42)
generator <- function() rnorm(50, mean = 5, sd = 2)
sim_results <- compare_methods_sim(generator, Rsim = 50, Rboot = 500)

# Analyze coverage: does true mean (5) fall in each CI?
coverage_bs <- mean(sapply(sim_results, function(res) {
  res$bs$ci[1] <= 5 && 5 <= res$bs$ci[2]
}))
coverage_bs
```

`moving_block_boot` *Moving Block Bootstrap for Time Series*

Description

Performs moving block bootstrap resampling for dependent data (time series). Preserves temporal dependence structure within blocks.

Usage

```
moving_block_boot(x, block_size = 10, R = 1000)
```

Arguments

`x`	numeric vector or time series object. Should be univariate time series with temporal dependence (autocorrelation). If using ts object, inherits frequency information. Length n >= block_size required.
`block_size`	integer length of consecutive observations to keep together in bootstrap samples (default 10). Rule of thumb: approximately sqrt(n) where n is series length. Must be >= 1 and <= length(x). Larger blocks preserve longer-range dependence; smaller blocks reduce distortion but may not capture autocorrelation structure.
`R`	integer number of bootstrap replicates (default 1000). Each replicate is a complete time series of length n obtained by concatenating randomly selected blocks. Must be >= 1.

Details

The moving block bootstrap divides the time series into overlapping blocks of length block_size and resamples these blocks with replacement. This preserves short-range dependence while allowing the empirical sampling distribution to reflect dependence.

Value

A list of length R. Each element is a numeric vector of length n (same as original series length), representing one bootstrap replicate of the time series. Replicates preserve block structure and local dependence within blocks, though global autocorrelation structure may be altered.

Examples

```
set.seed(42)
x <- arima.sim(n = 100, list(ar = 0.7))
result <- moving_block_boot(x, block_size = 10, R = 100)
length(result)  # 100 bootstrap replicates
```

---

perm_maxT *Permutation maxT for Multiple Testing*

Description

Performs permutation-based multiple testing correction using the maxT method. Controls family-wise error rate (FWER) while testing multiple hypotheses.

Usage

```
perm_maxT(data_matrix, groups, R = 2000)
```

Arguments

data_matrix	numeric matrix of dimensions (n x p) where n = number of observations (samples), p = number of variables (features/genes/voxels). Rows are observations, columns are variables. No NAs allowed; remove or impute before calling. Example: gene expression matrix with n = 50 samples, p = 10000 genes.
groups	factor or vector of group labels (length n). Should have exactly 2 unique levels representing group membership. Numeric (0/1) or character ("control"/"treatment") both acceptable. Order corresponds to data_matrix rows.
R	integer number of permutation replicates (default 2000). Larger values (5000-10000) recommended for stable p-values. Computational cost scales linearly with R. Must be >= 1.

Details

The maxT method conducts individual t-tests for each variable, then corrects for multiple comparisons using the distribution of the maximum absolute t-statistic under permutation. This maintains FWER at the specified level while preserving power.

Value

A list with two elements:

`obs`	numeric vector of length p, observed t-statistics for each variable. Positive/negative values indicate direction of difference. Names preserved from column names of data_matrix if present.
`p.values`	numeric vector of length p, FWER-adjusted p-values. Computed as proportion of permutation replicates where max(|t_permuted|) >= |t_observed|. Controls family-wise error rate at level approximately R/(R+1). Values automatically sorted to match obs vector.

Examples

```
set.seed(42)
data <- matrix(rnorm(200), nrow = 50, ncol = 4)
groups <- rep(0:1, each = 25)
result <- perm_maxT(data, groups, R = 500)
result$p.values
```

`perm_test_2sample` *Two-Sample Permutation Test*

Description

Performs exact or approximate permutation test for comparing two independent samples. Tests null hypothesis that two groups have equal distributions.

Usage

```
perm_test_2sample(x, y, R = 5000, stat = function(a, b) mean(a) - mean(b))
```

Arguments

`x`	numeric vector, first sample (group A). May contain NAs which are removed. Minimum 2 observations required. Length need not equal length(y).
`y`	numeric vector, second sample (group B). May contain NAs which are removed. Minimum 2 observations required. Compared against x for differences.
`R`	integer number of permutation replicates (default 5000). Larger values (e.g., 10000) improve p-value accuracy. Must be >= 1. Exact test feasible when R = C(n1+n2, n1) (factorial complexity), otherwise uses approximate permutation distribution.

stat	function taking two arguments (a, b) and returning numeric scalar test statistic. Default: `function(a, b) mean(a) - mean(b)` tests difference of means. Alternative: `function(a, b) median(a) - median(b)` for medians, or custom statistics. Function must handle NAs gracefully.

Details

Under the null hypothesis of equal distributions, exchanging group labels should be equally likely. The p-value is the proportion of permutations with test statistic at least as extreme as observed (in absolute value).

Value

A list with three elements:

obs	numeric scalar, the observed test statistic computed on original data
reps	numeric vector of length R, test statistics under random permutations of group labels. Represents null distribution assuming equal distributions between groups.
p.value	numeric scalar between 0 and 1, two-sided p-value computed as proportion of permutation replicates with |stat| >= |obs|. Value 1/R is smallest achievable p-value.

Examples

```
set.seed(42)
group1 <- rnorm(20, mean = 0)
group2 <- rnorm(20, mean = 0.5)
result <- perm_test_2sample(group1, group2, R = 1000)
result$p.value
```

---

stationary_boot *Stationary Bootstrap (Politis & Romano)*

Description

Performs stationary bootstrap for dependent data with random block lengths. More flexible than fixed-block bootstrap for time series with variable dependence.

Usage

```
stationary_boot(x, p = 0.1, R = 1000)
```

Arguments

x	numeric vector or time series object. Should be univariate time series with temporal dependence. Length n >= 2 required. If ts object, frequency information is not preserved in output.
p	numeric probability parameter controlling average block length (default 0.1). Must satisfy 0 < p <= 1. Average block length approximately 1/p: set p = 0.1 for average blocks of ~10 observations, p = 0.05 for ~20 observations. Smaller p values preserve longer-range dependence; larger p values reduce distortion.
R	integer number of bootstrap replicates (default 1000). Each replicate is complete time series of length n with random block structure. Must be >= 1.

Details

The stationary bootstrap uses random block lengths drawn from a geometric distribution. This avoids artificial periodicity inherent in fixed-block methods. Set p = 1/m to have average block length approximately m.

Value

A list of length R. Each element is a numeric vector of length n (matching original series). Unlike moving block bootstrap, block lengths vary randomly following geometric distribution, avoiding artificial periodicity. Replicates preserve local dependence structure more flexibly than fixed-block methods.

Examples

```
set.seed(42)
x <- arima.sim(n = 100, list(ar = 0.7))
# Average block length of 10: p = 0.1
result <- stationary_boot(x, p = 0.1, R = 100)
length(result)  # 100 bootstrap replicates
```

---

studentized_ci *Studentized Bootstrap Confidence Interval for Quantiles*

Description

Computes a studentized (t-based) bootstrap confidence interval for quantiles. More accurate than percentile intervals, especially for extreme quantiles.

Usage

```
studentized_ci(x, q = 0.5, R = 1000, Rinner = 200, conf = 0.95)
```

Arguments

x	numeric vector of data. May contain NAs which are removed. Minimum 2 observations required. Works well for skewed distributions and extreme quantiles where percentile bootstrap performs poorly.
q	numeric quantile level between 0 and 1 (default 0.5 for median). 0.25 = first quartile, 0.75 = third quartile, 0.95 = 95th percentile. Method particularly effective for extreme quantiles (q < 0.1 or q > 0.9).
R	integer number of outer bootstrap replicates (default 1000). This is primary bootstrap sample for t-distribution estimation. Recommended: 500-1000 for exploration, 2000+ for publication. Must be >= 1.
Rinner	integer number of inner bootstrap replicates for standard error estimation of each outer replicate (default 200). Larger values increase accuracy but also computation time (total iterations = R * Rinner). Recommended: 100-200. Must be >= 1.
conf	numeric confidence level between 0 and 1 (default 0.95). Standard: 0.95 or 0.99. Higher values give wider intervals.

Details

Studentized bootstrap uses the bootstrap t-distribution to construct intervals. This method often provides better coverage than percentile bootstrap, especially for skewed distributions or extreme quantiles.

Computation is intensive: O(R * Rinner) bootstrap samples are generated.

Value

A numeric vector of length 2 with names c("lower", "upper"):

lower	numeric, lower confidence limit for the q-th quantile
upper	numeric, upper confidence limit for the q-th quantile

Uses studentized bootstrap t-distribution to construct interval, providing better coverage probability than percentile method, especially for skewed distributions and extreme quantiles.

Examples

```
set.seed(42)
x <- rexp(30)  # Smaller sample: 30 instead of 50

# Fast example with reduced replications (< 5 sec) - UNWRAPPED
studentized_ci(x, q = 0.75, R = 100, Rinner = 20)


# Larger, more accurate example (takes > 5 sec) - WRAPPED in \donttest
set.seed(42)
x <- rexp(50)
studentized_ci(x, q = 0.75, R = 1000, Rinner = 200)
```

`wild_boot_lm` *Wild Bootstrap for Linear Model Coefficients*

Description

Performs wild bootstrap resampling for linear regression models to handle heteroscedasticity. Supports Rademacher and Mammen weight schemes.

Usage

```
wild_boot_lm(fit, R = 2000, type = c("rademacher", "mammen"))
```

Arguments

`fit`	an object of class 'lm' from `lm`. Should be fitted linear regression model. Heteroscedasticity is allowed and expected - this method is specifically designed to handle non-constant variance in residuals. Model should have at least 1 predictor.
`R`	integer number of bootstrap replicates (default 2000). Larger values (5000-10000) recommended for confidence intervals in publications. Must be >= 1 and a whole number.
`type`	character string specifying weight distribution scheme. Options: "rademacher" (default, faster, robust) generates weights as +1/-1 with equal probability. "mammen" (asymptotically optimal) uses empirically calibrated Mammen distribution with golden ratio. Choice has minimal practical impact on results.

Details

The wild bootstrap works by resampling residuals with random signs/weights while keeping predictors fixed. This is particularly useful for heteroscedastic data.

Value

A list with two elements:

`coef`	numeric vector of original fitted model coefficients, including intercept if present. Names preserved from original model.
`boot`	numeric matrix of dimensions (p x R) where p is number of coefficients. Each column contains one bootstrap replicate of coefficients. Row names are coefficient names; column names are bootstrap iteration numbers.

Examples

```
set.seed(42)
x <- rnorm(50)
y <- 2 + 1.5 * x + rnorm(50, sd = abs(x))  # heteroscedastic
fit <- lm(y ~ x)
```

```
result <- wild_boot_lm(fit, R = 500, type = "rademacher")
head(result$boot, 3)
```

Index

auto_select_method, 2

bca_ci, 3, *5*
bs_mean, 4, *5*

compare_methods_sim, 5

lm, *12*

moving_block_boot, 6
multisession, *5*

perm_maxT, 7
perm_test_2sample, 8
plan, *5*

stationary_boot, 9
studentized_ci, 10

wild_boot_lm, 12