Package 'moonboot'

July 23, 2025

Title m-Out-of-n Bootstrap Functions

Version 1.0.1

Date 2025-02-21

Description Functions and examples based on the m-out-of-n bootstrap suggested by Politis, D.N. and Romano, J.P. (1994) <doi:10.1214/aos/1176325770>. Additionally there are functions to estimate the scaling factor tau and the subsampling size m. For a detailed description and a full list of references, see Dalitz, C. and Lögler, F. (2024) <doi:10.48550/arXiv.2412.05032>.

License BSD 2-clause License + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

Imports methods (>= 3.5.0), stats (>= 3.5.0)

Suggests testthat (>= 2.3.0)

Config/testthat/edition 3

NeedsCompilation no

Author Christoph Dalitz [aut, cre], Felix Lögler [aut]

Maintainer Christoph Dalitz <christoph.dalitz@hs-niederrhein.de>

Repository CRAN

Date/Publication 2025-02-21 14:00:01 UTC

Contents

distPower 2
estimate.m 3
estimate.tau 5
mboot 6
mboot.ci 8
shorth 9

Index **11**

distPower *Ditribution with a Power Law*

Description

Density, distribution function, quantile function and random generation for a continuous distribution with the density `(pow+1)*(x-min)^pow/(max-min)^(pow+1)` for `x` in the range `[min,max]` and `pow > -1`.

Usage

```
dpower(x, pow, min = 0, max = 1)

ppower(x, pow, min = 0, max = 1)

qpower(p, pow, min = 0, max = 1)

rpower(n, pow, min = 0, max = 1)
```

Arguments

`x`	vector of values where to evaluate the denisty or CDF.
`pow`	degree of the power law.
`min`	minimum value of the support of the distribution.
`max`	maximum value of the support of the distribution.
`p`	vector of probabilities.
`n`	number of observations. If `length(n) > 1`, the length is taken to be the number required.

Value

`dpower` gives the density, `ppower` gives the cumulative distribution function (CDF), `qpower` gives the quantile function (i.e., the inverse of the CDF), and `rpower` generates random numbers.

The length of the result is determined by `n` for `rpower`, and is the length of `x` or `p` for the other functions.

`estimate.m` *Estimating a Subsample Size m*

Description

Estimates `m` using the selected `method`. Additional parameters can be passed to the underlying methods using `params`. It is also possible to pass parameters to the statistic using '...'.

Usage

```
estimate.m(
  data,
  statistic,
  tau = NULL,
  R = 1000,
  replace = FALSE,
  min.m = 3,
  method = "bickel",
  params = NULL,
  ...
)
```

Arguments

`data`	The data to be bootstrapped.
`statistic`	The estimator of the parameter.
`tau`	The convergence rate.
`R`	The amount of bootstrap replicates. Must be a positive integer.
`replace`	If the sampling should be done with replacement. Setting this value to true requires a sufficient smooth estimator.
`min.m`	Minimum subsample size to be tried. Should be the minimum size for which the statistic make sense.
`method`	The method to be used, one of `c("goetze","bickel","politis", "sherman")`.
`params`	Additional parameters to be passed to the internal functions, see details for more information.
`...`	Additional parameters to be passed to the statistic.

Details

The different methods have different parameters. Therefore, this wrapper method has been given the `params` parameter, which can be used to pass method-specific arguments to the underlying methods. The specific parameters are described below. Most of the provided methods need `tau`. If not provided, it will be estimated using `estimate.tau`. Note that method 'sherman' is using an alternative approach without using the scalation factor and therefore `tau` will not be computed if

selecting 'sherman' as method. Any non NULL values will be ignored when selecting the method 'sherman'.

Possible methods are:

goetze: The method from Goetze and Rackauskas is based on minimizing the distance between the CDF of the bootstrap distributions of different subsampling sizes 'm'. As distance measurement the 'Kolmogorov distance' is used. The method uses the pairs 'm' and 'm/2' to be minimized. As this would involve trying out all combinations of 'm' and 'm/2' this method has a running time of order Rn^2. To reduce the runtime in practical use, params can be used to pass a search.value, which is a list of the smallest and largest value for m to try.

bickel: This method works similary to the previous one. The difference here is that the subsample sizes to be compared are consecutive subsample sizes generated by q^j*n for j = seq(2,n) and a chosen q value between zero and one. The parameter q can be selected using params. The default value is q=0.75, as suggested in the corresponding paper.

politis: This method is also known as the 'minimum volatility method'. It is based on the idea that there should be some range for subsampling sizes, where its choice has little effect on the estimated confidence points. The algorithm starts by smoothing the endpoints of the intervals and then calculates the standard deviation. The h.ci parameter is used to select the number of neighbors used for smoothing. The h.sigma parameter is the number of neighbors used in the standard deviation calculation. Both parameters can be set by using params. Note that the h.* neigbors from each side are used. To use five elements for smoothing, h.ci should therefore be set to 2.

sherman: This method is based on a 'double-bootstrap' approach. It tries to estimate the coverage error of different subsampling sizes and chooses the subsampling size with the lowest one. As estimating the coverage error is highly computationally intensive, it is not practical to try all m values. Therefore, the beta parameter can be used to control which m values are tried. The values are then calculated by ms = n^beta. The default value is a sequence between 0.3 and 0.9 out of 15 values. This parameter can be set using params.

Value

Subsampling size m choosen by the selected method.

References

Götze F. and Rackauskas A. (2001) Adaptive choice of bootstrap sample sizes. *Lecture Notes-Monograph Series*, 36(State of the Art in Probability and Statistics):286-309

Bickel P.J. and Sakov A. (2008) On the choice of m in the m out of n bootstrap and confidence bounds for extrema. *Statistic Sinica*, 18(3):967-985.

Politis D.N. et al. (1999) *Subsampling*, Springer, New York.

Sherman M. and Carlstein E. (2004) Confidence intervals based on estimators with unknown rates of convergence. *Computional statistics & data analysis*, 46(1):123-136.

See Also

mboot estimate.tau

Examples

```
data <- runif(1000)
estimate.max <- function(data, indices) {return(max(data[indices]))}
tau <- function(n){n} # convergence rate (usually sqrt(n), but n for max)
choosen.m <- estimate.m(data, estimate.max, tau, R = 1000, method = "bickel")
print(choosen.m)
```

`estimate.tau` *Estimating the convergence rate*

Description

This function estimates the convergence rate of the bootstrap estimator and returns it as a function of the form `tau_n = n^a`, where `n` is the input parameter.

Usage

```
estimate.tau(
  data,
  statistic,
  R = 1000,
  replace = FALSE,
  min.m = 3,
  beta = seq(0.2, 0.7, length.out = 5),
  method = "variance",
  ...
)
```

Arguments

`data`	The data to be bootstrapped.
`statistic`	The estimator of the parameter.
`R`	Amount of bootstrap replicates used to estimate tau.
`replace`	If sampling should be done with replacement.
`min.m`	Minimal subsampling size used to estimate tau. Should be set to the minimum size for which the statistic makes sense.
`beta`	The tested subsample sizes m are `n^beta`.
`method`	Method to estimate tau, can be one of `c("variance", "quantile")`.
`...`	Additional parameters to be passed to the `mboot` function.

Details

There are two methods to choose from, `variance` and `quantile`. The provided `beta` values are used to select subsample sizes `m` by using `ms = n^beta`. Note that the choice of the `beta` values can impact the accuracy of the estimated `tau` (Dalitz & Lögler, 2024). For each selected subsample size `m` a bootstrap with `R` replications is performed. The method 'variance' then fits a linear function to log(variance) of the bootstrap statistics as function of log(m). The method 'quantile' averages over multiple quantile ranges Q and fits a linear function to log(Q) as a function of log(m).

Value

A function for the square root of the convergence rate of the variance, i.e., `f(n) = tau_n`. This function can directly be passed to `mboot.ci`.

References

Bertail P. et al. (1999) On subsampling estimators with unknown rate of convergence. *Journal of the American Statistical Association*, 94(446):568-579.

Politis D.N. et al. (1999) *Subsampling*, Springer, New York.

Dalitz, C, and Lögler, F. (2024) *moonboot: An R Package Implementing m-out-of-n Bootstrap Methods*. doi:10.48550/arXiv.2412.05032

See Also

mboot.ci

Examples

```
data <- runif(1000)
estimate.max <- function(data, indices) {return(max(data[indices]))}
estimated.tau <- estimate.tau(data, estimate.max)
boot.out <- mboot(data, estimate.max, R = 1000, m = 2*sqrt(NROW(data)), replace = FALSE)
cis <- mboot.ci(boot.out, 0.95, estimated.tau, c("all"))
ci.basic <- cis$basic
print(ci.basic)
```

`mboot` *m-Out-of-n Bootstrap Implementation*

Description

Generate `R` bootstrap replicates of the given `statistic` applied to the `data`. Sampling can be done with or without replacement. The subsample size m can either be chosen directly or estimated with `estimate.m()`.

Usage

```
mboot(data, statistic, m, R = 1000, replace = FALSE, ...)
```

Arguments

`data`	The data to be bootstrapped. If it is multidimensional, each row is considered as one observation passed to the `statistic`.
`statistic`	A function returing the statistic of interest. It must take two arguments. The first argument passed will be the original data, the second will be a vector of indicies. Any further arguments can be passed through the `...` argument.
`m`	The subsampling size.
`R`	The number of bootstrap replicates.
`replace`	Whether sampling should be done with replacement or without replacement (the default).
`...`	Additional parameters to be passed to the `statistic`.

Details

`m` needs to be a numeric value meeting the condition `2<=m<=n`. It must be chosen such that m goes to infinity as n goes to infinits, but the ratio m/n must go to zero. The m-out-of-n Bootstrap without replacement, known as subsampling, was introduced by Politis and Romano (1994).

Value

The returned value is an object of the class `"mboot"` containing the following components:

- t0: The observed value of `statistic` applied to the `data`.
- t: A matrix with `R` rows where each is a bootstrap replicate of the result of calling `statistic`.
- m,n: Selected subsample size and data size.
- data: The `data` passed to `mboot`.
- statistic: The `statistic` passed to `mboot`.
- replace: Whether the bootstrap replicates were done with or without replacement.

References

Politis D.N. and Romano J.P. (1994) Large sample confidence regions based on subsamples under minimal assumptions. *The Annals of Statistics*, 22(4):2031-2050, doi:10.1214/aos/1176325770

See Also

mboot.ci estimate.m estimate.tau

Examples

```
data <- runif(1000)
estimate.max <- function(data, indices) {return(max(data[indices]))}
boot.out <- mboot(data, estimate.max, R = 1000, m = 2*sqrt(NROW(data)), replace = FALSE)
```

mboot.ci *m-Out-of-n Bootstrap Confidence Intervals*

Description

Estimates the confidence interval using the methods provided by types. tau must be a function that calculates teh scaling factor tau(n) for a given n. If tau is not provided, it is estimated with estimate.tau using the default settings of this function.

Usage

```
mboot.ci(boot.out, conf = 0.95, tau = NULL, types = "all", ...)
```

Arguments

boot.out	The simulated bootstrap distribution from the mboot call.
conf	The confidence level.
tau	Function that returns the scaling factor tau in dependence of n. If NULL, estimate.tau is used to estimate tau.
types	The types of confidence intervals to be calculated. The value can be 'all' for all types, or a subset of c("basic", "norm", "sherman").
...	When tau is omitted, the additional parameters are passed to statistic during estimation of tau.

Details

As estimating the scaling factor tau(n) can be unreliable, it is recommended to explicitly provide tau. Otherwise it is estimated with estimate.tau. To specify additional arguments for estimate.tau, call this function directly and use its return value as tau argument. For the type sherman, tau is not needed and its value is ignored.

The following methods to compute teh confidence intervals are supported through the parameter type:

basic: This method works for all estimators and computes the interval directly from the quantiles of the m-out-of-n bootstrap distribution.

norm: This method only works for normally distributed estimators. It estimates the variance with the m-out-of-n bootstrap and then computes te interval with the quantiles of teh standard normal distribution.

sherman: This method does not scale the interval with tau(m)/tau(n) and thus is too wide. To avoid over-coverage, this is compensated by centering it randomly around the point estimators of one of the m-out-of-n bootstrap samples. Although this results on average in the nominal coverage probability, the interval is less accurate than the other intervals and should be used only as a last resort if the scaling factor tau is neither known, nor estimatable.

Value

A list of confidence intervals for the given types.

References

Politis D.N. and Romano J.P. (1994) Large sample confidence regions based on subsamples under minimal assumptions. *The Annals of Statistics*, 22(4):2031-2050, doi:10.1214/aos/1176325770

Sherman M. and Carlstein E. (2004) Confidence intervals based on estimators with unknown rates of convergence. *Computional statistics & data analysis*, 46(1):123-136.

Dalitz C. and Lögler M. (2024) moonboot: An R Package Implementing m-out-of-n Bootstrap Methods doi:10.48550/arXiv.2412.05032

See Also

mboot estimate.tau

Examples

```
data <- runif(1000)
estimate.max <- function(data, indices) {return(max(data[indices]))}
tau <- function(n){n} # convergence rate (usually sqrt(n), but n for max)
boot.out <- mboot(data, estimate.max, R = 1000, m = 2*sqrt(NROW(data)), replace = FALSE)
cis <- mboot.ci(boot.out, 0.95, tau, c("all"))
ci.basic <- cis$basic
print(ci.basic)
```

---

shorth *Mean of the Shorthest Half*

Description

Calculates the mean of the data points in the shortest interval containing half of the data. The arguments of the function are such that it directly can be used as a statistic in the `mboot()` function.

Usage

```
shorth(data, indices = NULL)
```

Arguments

`data`	the data as a numeric vector.
`indices`	the selected indices of `data`, by default `seq_along(data)`.

Value

The mean of the data points in the shortest interval containing half of the data.

References

Andrews D.F. et al. (1972) *Robust Estimates of Location* Princeton University Press, Princeton.

Examples

```
data <- rnorm(100)
shorth(data)
shorth(data, sample(1:100, size = 20))

# Calculating a CI for shorth using [mboot()]
data <- rnorm(100)
boot.out <- mboot(data, shorth, m = sqrt(length(data)))
basic.ci <- mboot.ci(boot.out, conf =0.95, tau = function(n) return(n^(1/3)), types = "basic")$basic
```

Index

∗ **~htest**
    mboot, 6
    mboot.ci, 8
∗ **~nonparametric**
    mboot, 6

distPower, 2
dpower *(*distPower*)*, 2

estimate.m, 3
estimate.m(), *6*
estimate.tau, 5

mboot, 6
mboot(), *9*
mboot.ci, 8

ppower *(*distPower*)*, 2

qpower *(*distPower*)*, 2

rpower *(*distPower*)*, 2

shorth, 9