Package 'boodd'

July 22, 2025

Type Package

Title Functions for the Book ``Bootstrap for Dependent Data, with an R Package"

Version 0.1

Date 2025-05-30

Maintainer Karolina Marek <karolina.marek10@gmail.com>

Description Companion package, functions, data sets, examples for the book Patrice Bertail and Anna Dudek (2025), Bootstrap for Dependent Data, with an R package (by Bernard Desgraupes and Karolina Marek) - submitted.
Kreiss, J.-P. and Paparoditis, E. (2003) <doi:10.1214/aos/1074290332>
Politis, D.N., and White, H. (2004) <doi:10.1081/ETC-120028836>
Patton, A., Politis, D.N., and White, H. (2009) <doi:10.1080/07474930802459016>
Tsybakov, A. B. (2018) <doi:10.1007/b13794>
Bickel, P., and Sakov, A. (2008) <doi:10.1214/18-AOS1803>
Götze, F. and Račkauskas, A. (2001) <doi:10.1214/lnms/1215090074>
Politis, D. N., Romano, J. P., & Wolf, M. (1999, ISBN:978-0-387-98854-2)
Carlstein E. (1986) <doi:10.1214/aos/1176350057>
Künsch, H. (1989) <doi:10.1214/aos/1176347265>
Liu, R. and Singh, K. (1992) <https://www.stat.purdue.edu/docs/research/tech-reports/1991/tr91-07.pdf>
Politis, D.N. and Romano, J.P. (1994) <doi:10.1080/01621459.1994.10476870>
Politis, D.N. and Romano, J.P. (1992) <https://www.stat.purdue.edu/docs/research/tech-reports/1991/tr91-07.pdf>
Patrice Bertail, Anna E. Dudek. (2022) <doi:10.3150/23-BEJ1683>
Dudek, A.E., Leśkow, J., Paparoditis, E. and Politis, D. (2014a) <https://ideas.repec.org/a/bla/jtsera/v35y2014i2p89-114.html>
Beran, R. (1997) <doi:10.1023/A:1003114420352>
B. Efron, and Tibshirani, R. (1993, ISBN:9780429246593)
Bickel, P. J., Götze, F. and van Zwet, W. R. (1997) <doi:10.1007/978-1-4614-1314-1_17>
A. C. Davison, D. Hinkley (1997) <doi:10.2307/1271471>
Falk, M., & Reiss, R. D. (1989) <doi:10.1007/BF00354758>
Lahiri, S. N. (2003) <doi:10.1007/978-1-4757-3803-2>
Shimizu, K. .(2017) <doi:10.1007/978-3-8348-9778-7>
Park, J.Y. (2003) <doi:10.1111/1468-0262.00471>
Kirch, C. and Politis, D. N. (2011) <doi:10.48550/arXiv.1211.4732>

Bertail, P. and Dudek, A.E. (2024) <doi:10.3150/23-BEJ1683>
Dudek, A. E. (2015) <doi:10.1007/s00184-014-0505-9>
Dudek, A. E. (2018) <doi:10.1080/10485252.2017.1404060>
Bertail, P., Clémençon, S. (2006a) <https:
//ideas.repec.org/p/crs/wpaper/2004-47.html>
Bertail, P. and Clémençon, S. (2006, ISBN:978-0-387-36062-1)
Radulović, D. (2006) <doi:10.1007/BF02603005>
Bertail, P. Politis, D. N. Rhomari, N. (2000) <doi:10.1080/02331880008802701>
Nordman, D.J. Lahiri, S.N.(2004) <doi:10.1214/009053604000000779>
Politis, D.N. Romano, J.P. (1993) <doi:10.1006/jmva.1993.1085>
Hurvich, C. M. and Zeger, S. L. (1987, ISBN:978-1-4612-0099-4)
Bertail, P. and Dudek, A. (2021) <doi:10.1214/20-EJS1787>
Bertail, P., Clémençon, S. and Tressou, J. (2015) <doi:10.1111/jtsa.12105>
Asmussen, S. (1987) <doi:10.1007/978-3-662-11657-9>
Efron, B. (1979) <doi:10.1214/aos/1176344552>
Gray, H., Schucany, W. and Watkins, T. (1972) <doi:10.2307/2335521>
Quenouille, M.H. (1949) <doi:10.1111/j.2517-6161.1949.tb00023.x>
Quenouille, M. H. (1956) <doi:10.2307/2332914>
Prakasa Rao, B. L. S. and Kulperger, R. J. (1989) <https:
//www.jstor.org/stable/25050735>
Rajarshi, M.B. (1990) <doi:10.1007/BF00050835>
Dudek, A.E. Maiz, S. and Elbadaoui, M. (2014) <doi:10.1016/j.sigpro.2014.04.022>
Beran R. (1986) <doi:10.1214/aos/1176349847>
Maritz, J. S. and Jarrett, R. G. (1978) <doi:10.2307/2286545>
Bertail, P., Politis, D., Romano, J. (1999) <doi:10.2307/2670177>
Bertail, P. and Clémençon, S. (2006b) <doi:10.1007/0-387-36062-X_1>
Radulović, D. (2004) <doi:10.1007/BF02603005>
Hurd, H.L., Miamee, A.G. (2007) <doi:10.1002/9780470182833>
Bühlmann, P. (1997) <doi:10.2307/3318584>
Choi, E., Hall, P. (2000) <doi:10.1111/1467-9868.00244>
Efron, B., Tibshirani, R. (1993, ISBN:9780429246593)
Bertail, P., Clémençon, S. and Tressou, J. (2009) <doi:10.1007/s10687-009-0081-y>
Bertail, P., Medina-Garay, A., De Lima-
Medina, F. and Jales, I. (2024) <doi:10.1080/02331888.2024.2344670>.

License GPL (>= 2)

Collate block.sub.R periodic.R utils.R jackVarField.R ppw.R boots.R
plot.boodd.R summary.boodd.R confint.boodd.R generate_process.R
bootsemi.R sieveboot.R blockboot.R blockboot.seasonal.R
regenboot.R freqboot.R aidedboot.R bootglm.R fieldboot.R
tboot_dist.R boot_dist.R pkc.R compute_power.R field.sub.R
findBestEpsilon.R fastNadaraya.R smallEnsemble.R jackVar.R
jackVarBlock.R jackVarRegen.R jackVarRegen.atom.R
jackVarRegen.smallEnsemblle.R jackFunc.R jackFuncBlock.R
jackFuncRegen.R qVar.R mark_boot.R fieldbootP.R
GetPseudoBlocks.R bandw1.R GetBlocks.R func_fdb.R naradamar.R
ftrunc.R thetaRB.R boot_wild.R boot_local.R boot_res.R
per_boo.R tft_boot.R para.boot.R rate.sub.R rate.block.sub.R
best.block.sub.size.R best.sub.size.iid.R f_PseudoBlocks.R

thetaARBB.R boodd-package.R boodd-class.R

Depends stats, tseries, grDevices, graphics, methods, fGarch, timeSeries, timeDate, fBasics, geoR

NeedsCompilation yes

RoxygenNote 7.3.2

Repository CRAN

Encoding UTF-8

Date/Publication 2025-06-05 11:00:02 UTC

Author Patrice Bertail [aut, ctb],
Bernard Desgraupes [aut],
Anna Dudek [aut, ctb],
Karolina Marek [aut, cre]

Contents

aidedboot . . . 4
b.star . . . 6
bandw1 . . . 7
best.block.sub.size . . . 8
best.sub.size.iid . . . 10
block.sub . . . 11
blockboot . . . 12
blockboot.seasonal . . . 14
boodd . . . 16
bootglm . . . 18
boots . . . 19
bootsemi . . . 22
boot_dist . . . 24
boot_local . . . 25
boot_res . . . 27
boot_wild . . . 28
bopt_circy . . . 29
class.boodd . . . 31
compute_power . . . 32
confint.boodd . . . 33
embb . . . 34
embb.sample . . . 36
fastNadaraya . . . 37
field.sub . . . 38
fieldboot . . . 39
fieldbootP . . . 41
findBestEpsilon . . . 42
freqboot . . . 43
ftrunc . . . 45
func_fdb . . . 46
f_PseudoBlocks . . . 48

genETARCH . . . 49
genMM1 . . . 50
GetBlocks . . . 51
GetPseudoBlocks . . . 53
jackFunc . . . 55
jackFuncBlock . . . 56
jackFuncRegen . . . 57
jackVar . . . 58
jackVarBlock . . . 60
jackVarField . . . 61
jackVarRegen . . . 62
jackVarRegen.atom . . . 64
jackVarRegen.smallEnsemble . . . 65
lam . . . 66
mark_boot . . . 67
meanCoeff, acfCoeff . . . 68
naradamar . . . 70
para.boot . . . 71
per_boo . . . 72
pkc . . . 74
plot.boodd . . . 75
qVar . . . 76
rate.block.sub . . . 77
rate.sub . . . 79
regenboot . . . 80
seasonalMean, seasonalVar, seasonalACF . . . 82
sieveboot . . . 84
smallEnsemble . . . 85
summary.boodd . . . 86
tboot_dist . . . 87
tft_boot . . . 89
thetaARBB . . . 90
thetaRB . . . 91
zi_inar_process . . . 92

Index **95**

`aidedboot` *Aided Frequency Bootstrap*

Description

The Aided Frequency Bootstrap (AFB) is a variation of the Frequency Domain Bootstrap (FDB). The idea is to fit a sieve AR(p) model and to generate the corresponding bootstrapped time series (by resampling centered residuals) with periodogram I^*_{AR}. Then the we estimate the quotient of the two spectral densities $q(\omega) = \frac{f(\omega)}{f_{AR}(\omega)}$ and generate bootstrap periodogram by multiplying I^*_{AR} by this quantity $q(\omega) = \frac{f(\omega)}{f_{AR}(\omega)}$.

Usage

```
aidedboot(x, XI, g, B, order = NULL, kernel = "normal", bandwidth)
```

Arguments

x	A numeric vector representing a time series.
XI	A list of functions defined on the interval $[0, \pi]$.
g	A numeric function taking `length(XI)` as arguments.
B	A positive integer; the number of bootstrap samples.
order	The order of the autoregressive sieve process (integer). If not specified, it is set by default as $\left\lfloor 4 * (length(x) \log(length(x)))^{1/4} \right\rfloor$.
kernel	A character string specifying the smoothing kernel. The possible choices are: `"normal"`, `"box"` and `"epanechnikov"`.
bandwidth	The kernel bandwidth smoothing parameter. If missing, the bandwidth is automaticly computed by `bandw1`.

Details

The idea underlying the Aided Frequency Bootstrap is importance sampling. It was introduced by Kreiss and Paparoditis (2003) and allows to better mimic the asymptotic covariance structure of the periodogram in the bootstrap world. Kreiss and Paparoditis (2003) considered a spectral density which is easy to estimate (typically based on a sieve AR representation of the time series), say $f_{AR}(\omega)$. The argument `x` is supposed to be a sample of a real valued zero-mean stationary time series.

The autoregressive sieve process of order $l = l(n)$ is modelled as

$$X_t = \sum_{k=1}^{l} \psi_k X_{t-k} + \epsilon_t$$

with $E(\epsilon_t) = 0, Var(\epsilon_t) = \sigma^2(l)$.

We estimate functionals of the spectral density $T(f)$ of the form

$$T(f) = g(A(\xi, f))$$

where g is a third order differentiable function,

$$A(\xi, f) = \left(\int_0^\pi \xi_1(\omega) f(\omega) d\omega, \int_0^\pi \xi_2(\omega) f(\omega) d\omega, \ldots, \int_0^\pi \xi_p(\omega) f(\omega) d\omega \right)$$

and

$$\xi = (\xi_1, \ldots, \xi_p) : [0, \pi] \to R^p.$$

If the `order` argument is not specified, its default value is $l = \left\lfloor (4 * (n \log(n))^{1/4}) \right\rfloor$, where `n` is the length of `x`.

The `kernel` argument has the same meaning as in the `freqboot` function.

Value

aidedboot returns an object of class boodd (see class.boodd).

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek)- submitted.

Kreiss, J.-P. and Paparoditis, E. (2003). Autoregressive aided periodogram bootstrap for time series. *Ann. Stat.* **31** 1923–1955.

See Also

freqboot.

Examples

```
n <- 200
x <- arima.sim(list(order=c(4,0,0),ar=c(0.7,0.4,-0.3,-0.1)),n=n)
B <- 299
one <- function(x) {1}
XI <- list(cos,one)
g <- function(x,y) {return(x/y)}
ord <- 2*floor(n^(1/3))
boo <- aidedboot(x,XI,g,B,order=ord)
plot(boo)
```

---

b.star *Bootstrap Block Length Choice in the Stationary Case*

Description

This function calculates the optimal bootstrap block lengths for both the Stationary Bootstrap, Nonoverlapping Block Bootstrap, Circular Block Bootstrap and Moving Block Bootstrap methods, based on the data provided.

Usage

```
b.star(data, mmax = NULL, Bmax = NULL, round = FALSE)
```

Arguments

data	A time series or a matrix of time series data.
mmax	An integer constant representing the maximum lag. By default determined by the data.
Bmax	An integer constant representing the maximum block length By default determined by the data.
round	A boolean indicating whether the resulting block lengths should be rounded.

Details

A `b.star` computes optimal block lengths for bootstrapping time series data, utilizing autocorrelation and autocovariance measures. The function incorporates several parameters, including `mmax`, and `Bmax`, to refine the block length calculations. The method involves a detailed analysis of the data's autocorrelation structure to identify the most suitable block lengths for bootstrapping procedures.

Value

A 2 x k matrix, where each column represents a time series and each row provides the optimal block length for the Moving Block Bootstrap or Circular Block Bootstrap (first row) and the Nonoverlapping Block Bootstrap or Stationary Bootstrap (second row).

Author(s)

Original code in Matlab by A. Patton. R translation and modifications by C. Parmeter and J. Racine, <racinej@mcmaster.ca>. We are grateful to Andrew Patton and Dimitris Politis for their assistance and feedback, and for allowing us to include this function in this package.

References

Politis, D.N., and White, H. (2004). Automatic block-length selection for the dependent bootstrap. *Econometric Reviews*, **23**, 53-70.

Patton, A., Politis, D.N., and White, H. (2009). Correction to 'Automatic Block-Length Selection for the Dependent Bootstrap' by D.N. Politis and H. White." *Econometric Reviews*, **28**, 372-375.

See Also

`blockboot`, `lam`.

Examples

```
# Simulate an ARIMA process
X <- arima.sim(n = 200, model = list(ar = c(0.5, 0.4), na = 0.5))
# Calculate the optimal bootstrap block lengths
optimal_choice <- b.star(X)
print(optimal_choice)
```

---

bandw1 *Computing the Bandwidth of the Kernel Density Estimators.*

Description

The function computes the bandwidth of a smoothed kernel density estimators using Silverman's rule.

Usage

```
bandw1(y)
```

Arguments

y	A numeric vector representing a Markov process of a vector of points.

Details

The function compute the bandwidth using the following formula: $h = 0.9An^{-1,5}$, where $A = \min\{\hat{\sigma}; IRQ/1.34\}$, where $\hat{\sigma}$ is the empirical standard deviation of the data and IRQ is the empirical interquartile range.

Value

Value of the Silverman's rule of thumb bandwidth.

References

Tsybakov, A. B. (2018). Introduction to Nonparametric Estimation. *SpringerLink*.

See Also

findBestEpsilon, fastNadaraya.

Examples

```
n=200# the length of the process
# Generating the AR(1) process
coeff=0.75
X = arima.sim(n=n, list(ar = c(coeff)))
h = bandw1(X)
```

---

best.block.sub.size *Optimal Block Subsampling Size*

Description

This function determines the optimal block size for subsampling using a distance-based method. It applies the Circular Block method and calculates Kolmogorov distances to select the most suitable subsampling size.

Usage

```
best.block.sub.size(X, func, PLT = TRUE, qq = 0.75, ...)
```

Arguments

X	A numeric vector or time series data.
func	A function applied to the blocks.
PLT	Logical. If TRUE (default), plots the Kolmogorov distances versus subsampling sizes.
qq	A numeric value in the interval $(0, 1)$. Determines the scaling factor for subsampling sizes. Higher values result in more subsampling distributions being computed. Default is 0.75.
...	Optional additional arguments passed to the func function.

Details

The procedure relies on the method proposed by Bickel and Sakov (2008) for determining optimal subsampling sizes. It computes a range of subsampling distributions for sizes proportional to powers of qq. The function then evaluates the Kolmogorov distance between consecutive subsampling distributions to identify the optimal block size. The function uses the Circular Block Bootstrap for generating subsamples. Ensure that qq is set such that $\lfloor -\log(length(X))/\log(qq) \rfloor \leq 3$; otherwise, the function will return an error.

Value

Returns the optimal block size for subsampling. If PLT = TRUE, a plot of Kolmogorov distances between consecutive subsampling distributions versus subsampling sizes is also displayed.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

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

See Also

block.sub, rate.sub, rate.block.sub.

Examples

```
set.seed(12345)
n = 1000 # sample size
# generating an AR(1) Gaussian process with variance 1
ts = arima.sim(n=n,model=list(ar=c(0.4)))*sqrt(1-0.4^2)
bopt1=best.block.sub.size(ts,mean)
```

`best.sub.size.iid` *Optimal Block Subsampling or MOON Bootstrap Sizes for I.I.D. Data*

Description

This function determines the optimal block size for subsampling or moon bootstrap sizes using a distance-based method, specifically applying undersampling techniques for independent and identically distributed (i.i.d.) data. It computes Kolmogorov distances between consecutive subsampling (or moon bootstrap) distributions to select the most suitable block size.

Usage

```
best.sub.size.iid(X, func, B = 999, PLT = TRUE, qq = 0.75, rep = FALSE, ...)
```

Arguments

`X`	A numeric vector or data representing i.i.d. observations.
`func`	A function pplied to the blocks.
`B`	An integer; the number of resampling replications. Default is 999.
`PLT`	Logical. If `TRUE` (default), plots the Kolmogorov distances versus subsampling sizes and intermediate regression results.
`qq`	A numeric value in the interval $(0, 1)$. Determines the scaling factor for subsampling sizes. Higher values result in more subsampling distributions being computed. Default is 0.75.
`rep`	Logical. If `TRUE`, performs moon bootstrap (subsampling with replacement). If `FALSE` (default), performs subsampling without replacement.
`...`	Optional additional arguments passed to the `func` function.

Details

This function implements a procedure based on the method proposed by Götze and Račkauskas (2001) and Bickel and Sakov (2008) for determining optimal subsampling sizes in i.i.d. case. It computes a range of subsampling distributions or moon bootstrap distribution for sizes proportional to powers of `qq`. The function then evaluates the Kolmogorov distance between consecutive distributions. The optimal block size is the value which minimises this distance.

Sometimes looking at the plot is more informative, especially when the distance does not vary very much. In this case, the largest value in a stable zone will be a better choice than the minimiser of the distance.

Value

Returns the optimal block size for subsampling or moon bootstrap.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

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

Götze, F. and Račkauskas, A. (2001). Adaptive choice of bootstrap sample sizes. In *State of the art in probability and statistics.* Institute of Mathematical Statistics, pp. 286-310.

See Also

`block.sub`, `rate.sub`, `rate.block.sub`, `best.block.sub.size`.

Examples

```
set.seed(12345)
n = 1000 # sample size
ts = rnorm(n)
bopt=best.sub.size.iid(ts,max)
```

`block.sub` *Block Subsampling*

Description

Block Subsampling for Time Series Using Blocks of Observations.

Usage

```
block.sub(x, func, length.block, method = c("movingblock", "circular"), ...)
```

Arguments

`x`	A numeric vector or time series data.
`func`	The function to apply to each block.
`length.block`	A scalar or a vector indicating the lengths of the blocks.
`method`	A character string specifying the type of block subsampling method. Can be `"movingblock"` or `"circular"`: `"movingblock"` divides the series into overlapping blocks. `"circular"` wraps the time series around a circle to create Circular Moving Blocks.
`...`	Optional additional arguments for the `func` function.

Details

This function performs block subsampling on time series data using the Moving or Circular Blocks methods (see Politis et al., 1999). It applies the specified function `func` to each subsample block whose length are defined by `length.block`. The function `func` is applied to the vector `x` and may return either a real number or a vector.

Value

A matrix containing, in rows, the value of `func` applied to each block. The number of columns in the matrix is equal to the length of the output of the `func` function.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Politis, D. N., Romano, J. P., & Wolf, M. (1999). Subsampling. *Springer N.Y.*.

See Also

`best.block.sub.size`, `rate.block.sub`.

Examples

```
set.seed(123)
data <- rnorm(100)
result <- block.sub(data, mean, length.block = 5, method="movingblock")
```

---

blockboot *Block Bootstrap*

Description

The function applies block bootstrap methods to a time series.

This function allows the following block bootstrap methods to be used: the Moving Block Bootstrap (Kunsch (1989), Liu and Singh (1992)), the Nonoverlapping Block Bootstrap (Carlstein (1986)), the Circular Block Bootstrap (Politis and Romano (1992)), and the Stationary Bootstrap (Politis and Romano (1994)).

Usage

```
blockboot(
  x,
  func,
  B,
  length.block = NULL,
  method = c("movingblock", "nonoverlapping", "circular", "stationary"),
  moon = NULL,
```

```
  replace = "TRUE",
  ...
)
```

Arguments

x	A vector or a time series.
func	The function to apply to each sample.
B	A positive integer; the number of bootstrap replications.
length.block	A positive integer; the length of the blocks.
method	The block bootstrap method. The possible values of the method argument are: "movingblock", "nonoverlapping", "circular" or "stationary". If it is not specified, the default method is *"movingblock"*. Method names may be abbreviated.
moon	Integer or NULL. When moon = NULL (default), blockboot performs a regular block bootstrap without subsampling. If moon is equal to some integer value, the function creates block bootstrap samples of size moon, drawing blocks of length length.block. Ensure that moon is less than n - 5, where n is the size of the data.
replace	Logical. If replace = TRUE (default), the function performs block bootstrap with replacement. If replace = FALSE, it performs Block Bootstrap Subsampling without replacement. In this case moon should be specified.
...	Optional additional arguments for the func function.

Details

Nonoverlapping Block Bootstrap (NBB) consists in cutting the original time series into nonoverlapping blocks of fixed length length.block and in resampling these blocks to reconstruct a bootstrap time series. Moving Block Bootstrap (MBB) consists in drawing independently overlapping blocks of fixed size length.block to reconstruct a bootstrap time series of length of the original process. Circular Block Bootstrap (CBB) consists in wrapping the data on a circle and to create the corresponding overlapping blocks, so that each value of the time series appears globally the same number of times in all the blocks. This generally reduce the bias of the bootstrap distribution. Stationary Bootstrap (SB) is based on blocks with random length, which ensure that the bootstrap sample is stationary.

Value

An object of class boodd.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Carlstein E. (1986). The use of subseries methods for estimating the variance of a general statistic from a stationary time series. *Annals of Statist.*, **14**, 1171-1179.

Künsch, H. (1989). The jackknife and the bootstrap for general stationary observations. *Ann. Statist.*, 17, 1217-1241.

Liu, R. and Singh, K. (1992). Moving block jackknife and bootstrap capture weak dependence. *Exploring the Limits of Bootstrap.*, Series in Probab. Math. Statist. Wiley, New York, pp 225-248.

Politis, D.N. and Romano, J.P. (1994). The stationary bootstrap. *J. Amer. Statist. Assoc.*, **89**, 1303–1313.

Politis, D.N. and Romano, J.P. (1992). A circular block-resampling procedure for stationary data. *Exploring the Limits of Bootstrap.*, Series in Probab. Math. Statist. Wiley, New York, pp 263-270.

See Also

boots, bootsemi, plot.boodd, confint.boodd, fieldboot, jackVarBlock.

Examples

```
B <- 999
data(airquality)
x <- airquality$Wind
n <- length(x)
b <- floor(sqrt(n))
boo1 <- blockboot(x,mean,B,b,method="moving")
plot(boo1,main="MBB", nclass=30)
confint(boo1, method="all")
```

---

blockboot.seasonal *Generalized Seasonal Block Bootstrap for Time Series.*

Description

Applies generalized block bootstrap methods, including the Generalized Seasonal Block Bootstrap (GSBB) and the Circular Generalized Seasonal Block Bootstrap (CGSBB), to periodically correlated (PC) processes in time series analysis.

Usage

```
blockboot.seasonal(
  x,
  func,
  B,
  length.block = NULL,
  ...,
  method = c("GSBB", "CGSBB"),
  d
)
```

Arguments

x	A vector or a time series.
func	A function to apply to each sample.
B	A positive integer; the number of bootstrap replications.
length.block	A positive integer; the length of the blocks; if NULL, a default value is used based on the time series period. The default value is d+1.
...	Optional additional arguments for the func function.
method	The bootstrap method to use; can be "GSBB" for Generalized Seasonal Block Bootstrap or "CGSBB" for Circular Generalized Seasonal Block Bootstrap.
d	The length of the seasonal period in the time series.

Details

This function is designed to handle the bootstrapping of time series data that exhibit periodic correlation. It allows the use of two methods: GSBB, which is based on moving blocks, and CGSBB, which uses a circular approach. The function selects blocks of data with a specified length and applies the user-defined function to each bootstrap sample. The user may calibrate the optimal size of the blocks by using the bopt_circy function.

The function adapts to the seasonal structure of the data, making it suitable for time series with inherent periodicity. The choice between GSBB and CGSBB methods can be made according to the specific characteristics of the time series.

Value

An object of class boodd containing the results of the bootstrap procedure and the statistic computed by the func function on the original data.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Patrice Bertail, Anna E. Dudek. (2022). Optimal choice of bootstrap block length for periodically correlated time series. *Bernoulli*, **30**, 2521-2545.

Dudek, A.E., Leśkow, J., Paparoditis, E. and Politis, D. (2014a). A generalized block bootstrap for seasonal time series. *J. Time Ser. Anal.*, **35**, 89-114.

See Also

blockboot, bopt_circy.

Examples

```
set.seed(54321)
n <- 1000 # sample size
 f0 <- 1/12
 u <- rnorm(n, 0, 0.4)
 coeff=0.2
```

```
b=rep(0,n)
for (i in 2:(n-1)){ b[i+1]=coeff*b[i]+rnorm(1,0,0.4)}
# Simulate a seasonal process with period 12
X <- 5*cos(2 * pi * f0 * (1:n))+5*b * cos(2 * pi * f0 * (1:n)) + u
X_ts <- ts(X)
b=15 # Block length
result = blockboot.seasonal(X_ts, mean, B=999, length.block=b, d=12, method="GSBB")
plot(result, nclass=30)
confint(result)
```

---

boodd *Package boodd: bootstrap for dependent data*

Description

The package boodd contains functions, datasets and examples to accompany the textbook by Patrice Bertail and Anna Dudek (2025), *Bootstrap for Dependent Data, with an R package (by Bernard Desgraupes and Karolina Marek)* – submitted.

Details

Version:	0.1
Date:	2025-06-03
License:	GPL (>= 2)

A list of functions:

- `boots` – Bootstrap in the i.i.d. case.
- `bootsemi` – Semi-parametric bootstrap for time series.
- `blockboot` – Block bootstrap for stationary time series.
- `regenboot` – Regenerative Bootstrap.
- `findBestEpsilon` – Calculating the optimal value of the epsilon.
- `aidedboot` – Aided Frequency Bootstrap.
- `b.star` – Optimal length of bootstrap blocks.
- `boodd` – Package overview: bootstrap for dependent data.
- `bootglm` – Bootstrap for Generalized Linear Model.
- `boot_dist` – Bootstrap distribution.
- `confint.boodd` – Confidence intervals for objects of class boodd.
- `class.boodd` – Objects of class boodd.
- `embb` – Characteristics for Extended Moving Block Bootstrap class.
- `embb.sample` – EMBB method.
- `fieldboot` – Random Field Bootstrap.

- field.sub – Field subsampling.
- freqboot – Frequency Domain Bootstrap.
- jackFunc – Create a function that calculates the statistic and the jackknife variance.
- jackVar – Jackknife Variance for statistics based on i.i.d data.
- jackVarBlock – Jackknife Variance for blocks.
- jackVarRegen – Jackknife Variance for regenerative processes.
- jackVarRegen.atom – Jackknife Variance for finite states Markov chains.
- jackVarRegen.smallEnsemble – Jackknife Variance for homogeneous Markov chains.
- jackFuncBlock – Jackknife Function for blocks.
- jackFuncRegen – Jackknife Function in regenerative case.
- jackVarField – Jackknife Variance for random fields.
- lam – Lag window.
- plot.boodd – Plot objects of class boodd.
- qVar – Variance estimator for quantile.
- seasonalMean – Characteristics of periodically correlated time series.
- sieveboot – Autoregressive Sieve Bootstrap.
- summary.boodd – Summary for objects of class boodd.
- tboot_dist – t-bootstrap distribution.
- pkc – Plot Kernel density.
- compute_power – Compute the power of the test.
- bopt_circy – Optimal choice of bootstrap block length for PC-process.
- genMM1 – Generate the queuing system process.
- genETARCH – Generate the ETAR-ARCH process.

Author(s)

Patrice Bertail <patrice.bertail@parisnanterre.fr>
University Paris Nanterre – Modal'X

Bernard Desgraupes
University Paris Nanterre – Modal'X

Anna Dudek
Department of Applied Mathematics
AGH University of Krakow, Poland

Karolina Marek <karolina.marek10@gmail.com>
Department of Applied Mathematics
AGH University of Krakow, Poland

References

P. Bertail and A. Dudek (2025), *Bootstrap for Dependent Data, with an R package (by Bernard Desgraupes and Karolina Marek)* – submitted.

`bootglm` *Bootstrap for Generalized Linear Model*

Description

Parametric bootstrap for generalized linear model.

Usage

```
bootglm(model, data, func, B, ...)
```

Arguments

`model`	An object of class `lm` or `glm`.
`data`	The dataframe used to fit the model.
`func`	The function to apply to each sample.
`B`	A positive integer; the number of bootstrap replications
`...`	Optional additional arguments for the `func` function.

Details

The parametric bootstrap simply consists in resampling data in the model with estimated parameters. `bootglm` uses this principle for Generalized Linear Models (GLM) conditionally to the explanatory variables (see Beran (1997)) for the conditions of validity of this method.

Value

`bootglm` returns an object of class `boodd` (see class.boodd).

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek)- submitted.

Beran, R. (1997). Diagnosing Bootstrap Success, *Annals of the Institute of Statistical Mathematics*, **49**, 1-24.

See Also

`bootsemi`.

Examples

```
B <- 999
x <- runif(100)
e <- rnorm(100)
y <- x + e>0
data <- data.frame(x,y)
glm_probit <- glm(y ~ x, family=binomial(link="probit"),data=data)
# Define the function to bootstrap: the mle in the probit model
coeff <- function(data){gg=glm(y ~ x, family=binomial(link="probit"),data=data)$coeff}
boo1 <- bootglm(glm_probit,data,coeff,B)
# parametric bootstrap of the coeffitients of a probit model
plot(boo1,main=c("Boostrap of GLM : probit model","Coefficient"))

# coeffv : a function to return coeff and variance of coefficients
coeffv <- function(data){
  gg <- glm(y~ x, family=binomial(link="probit"),data=data)
  var <- diag(summary(gg)$cov.u)
  c(gg$coeff,var)
}
# Parametric bootstrap of all coeff and variances
boo2 <- bootglm(glm_probit,data,coeffv,B)
# Construct all type of confidence intervals including bootstrap-t
# and symmetric bootstrap-t
confint(boo2,method="all")

# Family Poisson (Poisson regression)
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
data <- data.frame(treatment,outcome,counts)
gl <- glm(counts ~ outcome + treatment,family=poisson())
meancounts <- function(data) {mean(data$counts)}
boo3 <- bootglm(gl,data,meancounts,B)
confint(boo3,method="all")
plot(boo3)
```

---

boots *Bootstrap for the I.I.D. Case*

Description

Generic function to bootstrap in the iid case.

Usage

```
boots(x, func, B, smooth = FALSE, moonsize = NULL, mreplace = TRUE, ...)
```

Arguments

x	A vector or a matrix representing the data.
func	The function to apply to each sample.
B	A positive integer; the number of bootstrap replications.
smooth	Logical. If TRUE use a smooth bootstrap.
moonsize	A value of m in the 'm out of n' bootstrap.
mreplace	Logical. If TRUE, bootstrap is done with replacement. If FALSE the function performs subsampling. In that case, moonsize should be specified, else the function stops.
...	An optional additional arguments for the func function.

Details

The function boots performs different versions of the bootstrap in the iid case (see Davison and Hinkley (1997)). With default parameters it performs a naive bootstrap (see Efron and Tibshirani (1993)). If moonsize is specified, the function performs the moon bootstrap (see Bickel et al.(1997)) With smooth=TRUE the function performs a smooth bootstrap, based on a kernel density estimator of the data, with a bandwidth chosen by cross-validation (see Efron and Tibshirani (1993) and Falk and Reiss (1989)). Finally, when mreplace=FALSE, the function performs the subsampling method of the Politis and Romano (1994) with subsampling size equals to moonsize.

The func must be a function whose first argument is a vector and which returns either a single value or a vector.

The x argument can be a vector or a matrix. In the case of a matrix, the *rows* of the matrix are bootstrapped.

The moonsize and mreplace arguments concern *m out of n* bootstrap (aka *moon bootstrap*). The moonsize argument is an integer less than the length of x (or the number of rows if x is a matrix). The mreplace argument is a logical that indicates whether the bootstrap samples are drawn with or without replacement.

Value

An object of class boodd containing either a vector or a matrix, depending on whether the func function returns a single value or a vector. If the func function returns a vector of size n, then boots returns a matrix of size B x n.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

B. Efron, and Tibshirani, R. (1993). *An Introduction to the Bootstrap*, Chapman and Hall/CRC Monographs on Statistics and Applied Probability.

Bickel, P. J., Götze, F. and van Zwet, W. R. (1997). Resampling fewer than n observations: gains, losses, and remedies for losses. *Statistica Sinica*, **7**, 1-31

A. C. Davison, D. Hinkley (1997). *Bootstrap Methods and Their Application*, Cambridge Series in Statistical and Probabilistic Mathematics.

Falk, M., & Reiss, R. D. (1989). Bootstrapping the distance between smooth bootstrap and sample quantile distribution. *Probability Theory and Related Fields*, **82**, 177–186.

Politis, D. N., & Romano, J. P. (1994). Large Sample Confidence Regions Based on Subsamples under Minimal Assumptions. *The Annals of Statistics*, **22**. 2031-2050

See Also

plot.boodd, confint.boodd.

Examples

```
B <- 999
n <- 200
x <- rnorm(n)
# Naive bootstrap of the mean
boo1 <- boots(x,mean,B)
summary(boo1)
plot(boo1)
confint(boo1)
confint(boo1,method="bperc")

# Naive bootstrap of a multidimentional statistic
mv <- function(data) {c(mean(data),var(data))} # compute mean and variance
boo2 <- boots(x,mv,B)
# Confint can compute percentile and t-percentile confidence intervals
# when variance is bootstrapped
confint(boo2,method="all")

# Naive Bootstrap of the output parameters of lm (linear regression) function
sigma <- 0.2
y <- x+rnorm(n)
data <- as.matrix(data.frame(x,y))
nlm <- function(dat){lm(dat[,2]~dat[,1])$coefficient}
boo3 <- boots(data,nlm,B)

# Smoothed bootstrap for quantiles
boo4 <- boots(x,median,B) # without smoothing
plot(boo4)
boo5 <- boots(x,median,B,smooth=TRUE)
# with smoothing using a cross-validation estimator of the window
plot(boo5)

# Moon bootstrap
n <- 10000
x <- rnorm(n)
# i.i.d bootstrap is not consistent for the max
boo6 <- boots(x,max,B)
# Moon bootstrap of the max with a size equals to sqrt(n)
boo7 <- boots(x,max,B,moonsize=sqrt(n))
# Subsampling with the moonsize equals to sqrt(n)
boo8 <- boots(x,max,B,moonsize=sqrt(n),mreplace=TRUE)
```

`bootsemi` *Semiparametric Bootstrap*

Description

The function performs a semiparametric bootstrap for a general statistics, using a time-series model.

Usage

```
bootsemi(
  x,
  func,
  B,
  model = c("ARIMA", "GARCH"),
  params,
  model.fit = NULL,
  model.sim = NULL,
  H0 = FALSE,
  PAR0 = NULL,
  ...
)
```

Arguments

`x`	A vector or a time series representing the data.
`func`	The function to apply to each sample.
`B`	A positive integer; the number of bootstrap replications.
`model`	The chosen model to fit the time series. Either "ARIMA" or "GARCH", else `model.fit` and `model.sim` should be specified.
`params`	The parameters of the model (see below).
`model.fit`	A function fitting the parameters for the generic model (see below). By default `NULL`.
`model.sim`	A function simulating the data for the generic model (see below). By default `NULL`.
`H0`	Logical. Only implemented for the ARIMA and GARCH models. If `H0 = TRUE`, the value(s) of the parameter(s) under the null `PAR0` should be specified. Then, the model is simulated under the null hypothesis. By default it is `NULL`.
`PAR0`	The value of the parameter that we want to test under the null hypothesis. By default it is `FALSE`.
`...`	Optional additional arguments for the `func` function.

Details

The default basic models currently supported are : *ARIMA* and *GARCH*.

The argument `params` specifies the orders of the chosen model. In the case of *ARIMA*, it is a vector of the form *c(p,q)* or *c(p,d,q)*. In the case of *GARCH*, it is a vector of the form *c(q)* or *c(p,q)* corresponding to an *ARCH(q)* or *GARCH(p,q)* model, respectively.

Alternatively, one can specify two functions in the `model.fit` and `model.sim` arguments. They are used to implement a generic bootstrap procedure, which first estimates the model with `model.fit` and then simulates the bootstrap time series with `model.sim`. The `model.fit` function has the following prototype:

```
model.fit(x,params)
```

It receives the `params` argument specified in the `bootsemi` function. It should return an object describing the model (typically a list containing all the necessary components for the model). The `model.sim` function has the following prototype:

```
model.sim(model,innovations,params)
```

The `innovations` argument is the resampled vector of centered residuals. The function builds a new trajectory of the original process using the estimators provided by the `model.fit` function.

The Examples section below shows how this can be done in the case of a Threshold Autoregressive (*TAR*) process.

Parameters `H0` and `PAR0` may be used to generate the bootstrap distribution of the statistic of interest under a specific null hypothesis. This option may be useful, for instance, for ARIMA models with unit roots. In that case, the usual bootstrap dose not work, unless one simulates the process under the null hypothesis of unit root.

Value

`bootsemi` returns an object of class boodd (see class.boodd).

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Lahiri, S. N. (2003). *Resampling Methods for Dependent Data*. Springer N.Y..

Shimizu, K. .(2017). *Bootstrapping Stationary ARMA-GARCH Models*. Springer Fachmedien Wiesbaden.

Park, J.Y. (2003). Bootstrap unit-root tests, *Econometrica*, **77**, 1845-1895.

See Also

`blockboot`, `plot.boodd`, `confint.boodd`.

Examples

```
# An ARIMA(2,1) process
library(stats)
B <- 299
n <- 200
x <- arima.sim(model=list(ar=c(0.8,-0.4),ma=c(0.2)),n=n)
boo1 <- bootsemi(x,mean,B,model="ARIMA",params=c(2,1))
plot(boo1)
```

`boot_dist` *Bootstrap Distribution*

Description

This function computes the rescaled and recentered bootstrap distribution from a given bootstrap object. It optionally plots this distribution and superimposes a normal approximation for comparison.

Usage

```
boot_dist(
  boot_obj,
  comp = 1,
  PLT = TRUE,
  nn = TRUE,
  recenter = FALSE,
  return_values = FALSE
)
```

Arguments

`boot_obj`	A bootstrap object of type boodd, containing the bootstrap samples.
`comp`	A positive Integer; The index of the column of the bootstrap matrix to be plotted or analysed.
`PLT`	Logical. If `TRUE` (default), the function plots the distribution.
`nn`	Logical. If `TRUE` (default), a normal approximation is superimposed on the histogram.
`recenter`	Logical. If `TRUE`, the bootstrap distribution is recentered by the bootstrap mean, else if `FALSE` (default) it is recentered by the original value of the estimator.
`return_values`	Logical. If `TRUE` the function returns the calculated values for the rescaled and recentered distribution. `FALSE` by default.

Details

The function modifies the bootstrap distribution by recentering it, depending on value of `recenter` parameter and computes a standardized version of the selected component of the statistic. If `PLT` is `TRUE`, it generates a histogram of the standardized distribution with optional normal approximation overlay if `nn` is `TRUE`.

Value

A numeric vector representing the rescaled and optionally recentered bootstrap distribution of the selected component of the statistic.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

See Also

`plot.boodd`.

Examples

```
n=30
X=rnorm(n)
fun = function(x){
c(mean(x), var(x))}
boo=boots(X, fun, B=999)
# Recentered, rescaled bootstrap distribution of the first component - mean
boot_dist(boo, comp=1, recenter = TRUE)
# Rescaled bootstrap distribution of the first component - variance
boot_dist(boo, comp=2)
```

---

boot_local *TFT Local Bootstrap.*

Description

The local bootstrap is used to bootstrap the Fourier coefficients for the Time Frequency Toggle (TFT)-Bootstrap (see Kirch and Politis (2011)).

Usage

```
boot_local(X, n = length(X), h, kernel, t)
```

Arguments

X	A numeric vector representing a time series.
n	An integer; by default is the length of time series X but allow for a smaller sample size m to perform moon bootstrap.
h	A numeric value specifying the bandwidth used to compute the kernel estimator in case of local bootstrap. By default it is equal to $n^{-2/3}$.
kernel	An integer value indicating the kernel type. Use 0 for the Daniell kernel or any other value for the Bartlett-Priestley (Epanechnikov) kernel (by default).
t	An integer indicating the number of bootstrap replications.

Details

The function first centers X by subtracting its mean, calculates the Fourier coefficients using the Fast Fourier Transform (FFT), and computes resampling probabilities based on the specified kernel kernel and bandwidth h. These probabilities are used to select random Fourier coefficients, which are then transformed back to generate standardized bootstrap replicates.

Value

A matrix where each column contains a bootstrap replicate of the time series X.

Author(s)

We are grateful to Claudia Kirch for providing the original code in R.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Kirch, C. and Politis, D. N. (2011). TFT-Bootstrap: Resampling time series in the frequency domain to obtain replicates in the time domain, *Annals of Statistics*, vol.

See Also

tft_boot, boot_res, boot_wild.

Examples

```
# see the mother function tft_boot
```

`boot_res` *TFT Residual Bootstrap.*

Description

The residual bootstrap is used to bootstrap the Fourier coefficients for the Time Frequency Toggle (TFT)-Bootstrap (see Kirch and Politis (2011)).

Usage

```
boot_res(X, n = length(X), h, kernel, t)
```

Arguments

`X`	A numeric vector representing a time series.
`n`	An integer; by default is the length of time series `X` but allow for a smaller sample size m to perform moon bootstrap.
`h`	A positive numeric value specifying the bandwidth used to compute the kernel estimator in case of local bootstrap. By default it is equal to $n^{-2/3}$.
`kernel`	An integer value indicating the kernel type. Use `0` for the Daniell kernel or any other value for the Bartlett-Priestley (Epanechnikov) kernel (by default).
`t`	An integer indicating the number of bootstrap replications.

Details

The function first centers `X` and calculates its Fourier coefficients. It then estimates the spectral density using a kernel density estimator with parameters kernel `kernel` and bandwidth `h`, which standardizes the residuals accordingly. These standardized residuals are resampled with replacement to create bootstrap replicates.

Value

A matrix where each column contains a bootstrap replicate of the time series `X`.

Author(s)

We are grateful to Claudia Kirch for providing the original code in `R`.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Kirch, C. and Politis, D. N. (2011). TFT-Bootstrap: Resampling time series in the frequency domain to obtain replicates in the time domain, *Annals of Statistics*, vol.

See Also

tft_boot, boot_local, boot_wild.

Examples

```
# see the mother function tft_boot
```

---

boot_wild *TFT wild bootstrap.*

Description

The wild bootstrap is used to bootstrap the Fourier coefficients for the Time Frequency Toggle (TFT)-Bootstrap (see Kirch and Politis (2011)).

Usage

```
boot_wild(X, n = length(X), h, kernel, t)
```

Arguments

X	A numeric vector representing a time series.
n	An integer; by default is the length of time series X but allow for a smaller sample size m to perform moon bootstrap.
h	A positive numeric value specifying the bandwidth used to compute the kernel estimator in case of local bootstrap. By default it is equal to $n^{-2/3}$.
kernel	An integer value indicating the kernel type. Use 0 for the Daniell kernel or any other value for the Bartlett-Priestley (Epanechnikov) kernel (by default).
t	An integer indicating the number of bootstrap replications.

Details

The function centers process X by subtracting its mean, then computes the Fourier coefficients using the Fast Fourier Transform (FFT). These coefficients are used to compute periodograms, which are then smoothed using a spectral density estimation method based on the chosen kernel kernel and bandwidth h. Random normal samples are scaled by these smoothed spectral densities to generate bootstrapped replicates.

Value

A matrix where each column contains a bootstrap replicate of the time series X.

Author(s)

We are grateful to Claudia Kirch for providing the original code in R.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Kirch, C. and Politis, D. N. (2011). TFT-Bootstrap: Resampling time series in the frequency domain to obtain replicates in the time domain, *Annals of Statistics*, vol.

See Also

`tft_boot`, `boot_res`, `boot_local`.

Examples

```
# see the mother function tft_boot
```

`bopt_circy` *Optimal Bootstrap Block Length for Periodically Correlated Time Series.*

Description

Calculates the optimal block length for Generalized Seasonal Block Bootstrap (GSBB), Extension of Moving Block Bootstrap (EMBB), and their circular versions CGSBB and CEMBB for periodically correlated time series, in the problems of the overall mean and seasonal means estimation.

Usage

```
bopt_circy(
  x,
  period,
  PLT = FALSE,
  parameter = c("mean", "seasonal mean"),
  method = c("EMBB", "CEMBB", "GSBB", "CGSBB"),
  plot_range = NULL
)
```

Arguments

`x`	A numeric vector representing a periodically correlated time series.
`period`	An integer; period length of `x`.
`PLT`	Logical. If `TRUE` the function plots the Mean Square Error (MSE) of the bootstrap variance estimator for various block lengths. By default it is equal to `FALSE`.
`parameter`	The possible bopt_circy parameters are: • `"mean"`, • `"seasonal mean"`.

`method` A choice of the block bootstrap method:

- `"GSBB"` - Generalized Seasonal Block Bootstrap,
- `"CGSBB"` - Circular version of GSBB,
- `"EMBB"` - Extension of Moving Block Bootstrap,
- `"CEMBB"` - Circular version of EMBB.

`plot_range` If `PLT=TRUE`, a parameter changing the range of the x-axis in the plot of MSE. By default `plot_range = 10`.

Details

For each bootstrap method implemented here, the function `bopt_circy` computes the optimal block length for periodically correlated (PC) time series. The optimal block length is obtained by minimization of the MSE of the bootstrap variance estimator (see Bertail and Dudek (2024)). For the `GSBB` and `CGSBB` the optimal block length has the form $k \cdot d \pm 1$, where d is a period length and k is a positive integer. For the `EMBB` and `CEMBB` the optimal block length can be of any length.

Value

Returns the optimal block length (integer) for the chosen block bootstrap method.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Bertail, P. and Dudek, A.E. (2024). Optimal choice of bootstrap block length for periodically correlated time series, *Bernoulli*, **30**, 2521-2545.

See Also

`blockboot.seasonal`, `embb.sample`.

Examples

```
# Generate a periodically correlated time series
n=200
b <- arima.sim(n = n, model = list(ar = c(0.5, 0.4), na = 0.5))
period <- 12
x <- 5*cos(2 * pi /period * (1:n))+5*b * cos(2 * pi /period * (1:n))
# Calculate the optimal block length for GSBB
optimal_choice <- bopt_circy(x, period, parameter= "mean", method= "GSBB")
print(optimal_choice)
```

class.boodd *Objects of Class* boodd

Description

The bootstrap functions in this package return an object of class boodd, which contains the values of bootstrapped statistics and the statistic computed on the original data. Generic functions such as plot, confint, and summary can be applied to these objects.

Details

A boodd object has the following attributes:

- "class" – always "boodd".
- "kind" – the type of bootstrap applied, such as "iid", "block", "movingblock", "circular", "aided", "semi", "sub-block", "regenerative", "frequency", "sieve", "bootglm", "randomfield", "TFT", "parametric", "i.i.d. subsampling with estimated rate", "subsampling with rates estimation".
- "func" – the function that was bootstrapped.

Value

An object of class boodd, which is a list with at least:

- $s – a vector or matrix of bootstrap statistics.
- $Tn – the value of the statistic on the original data.

References

Bertail, P., & Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) – submitted.
Efron, B., & Tibshirani, R. (1993). *An Introduction to the Bootstrap*. Chapman and Hall.

See Also

boots, boot_dist, tboot_dist, plot.boodd, confint.boodd, summary.boodd, bootsemi, blockboot, regenboot, freqboot, aidedboot, sieveboot, bootglm, fieldboot, tft_boot, para.boot, mark_boot, rate.sub, rate.block.sub

Examples

```
set.seed(123)
data <- rnorm(100)
boot_result <- boots(data, mean, 1000)
summary(boot_result)
plot(boot_result)
```

`compute_power` *Compute the Power of a Statistical Test*

Description

This function computes the power of a statistical test given the distributions under the null and alternative hypotheses for a specified significance level.

Usage

```
compute_power(null_dist, alt_dist, alpha)
```

Arguments

`null_dist`	A numeric vector representing the distribution under the null hypothesis.
`alt_dist`	A numeric vector representing the distribution under an alternative hypothesis.
`alpha`	A numeric value in $(0, 1)$; type I error rate of the test.

Details

The function calculates the proportion of values of the alternative distribution that falls into the critical region determined by the distribution under the null for an error rate `alpha`.

Value

A numeric value estimating the power of the test.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Beran, R. (1986). Simulated Power Functions. *The Annals of Statistics*, **14**, 151-173.

See Also

`pkc`, `bootglm`, `bootsemi`.

Examples

```
# Generate two normally distributed samples as null and alternative distributions
set.seed(123)
null_dist <- rnorm(1000, mean = 0, sd = 1) # Null distribution
alt_dist <- rnorm(1000, mean = 0.5, sd = 1) # Alternative distribution
alpha <- 0.05 # Significance level
# Compute the power of the test
test_power <- compute_power(null_dist, alt_dist, alpha)
print(test_power)
```

confint.boodd *Calculate Confidence Intervals for* boodd *Objects.*

Description

Calculates confidence intervals for an object of class boodd returned by bootstrap functions such as boots, bootsemi, blockboot, regenboot, etc.

Usage

```
## S3 method for class 'boodd'
confint(
  object,
  parm = NULL,
  level = 0.95,
  method = c("perc", "bperc", "aboot", "tboot", "tsymboot", "all"),
  recenter,
  ...
)
```

Arguments

object	An object of class boodd.
parm	Not used. Included for consistency with the generic confint() function.
level	Confidence level. Default is 0.95.
method	Method used to build the confidence interval. Choices include: * perc - percentile, * bperc - basic percentile, * aboot - asymptotic bootstrap, * tboot - bootstrap-t, * tsymboot - symmetric bootstrap-t, * all - all the previous methods. Default is perc.
recenter	Logical. If TRUE it centers the intervals around the mean value of the bootstrap samples. Relevant only for tboot or tsymboot methods. By default it equals FALSE unless the attribute kind of object boodd is block.
...	Optional additional arguments.#' @details This function provides confidence intervals using several methods, see Efron and Tibshirani (1993), Bertail and Dudek (2025) for description. The tboot and tsymboot methods require the function to which the bootstrap method is applied to return an even number of values corresponding to parameter estimates (first k columns for the parameter of size k) and their variances (columns $(k+1)$ to $2k$).

Value

If the method argument is not all, the function confint.boodd returns a two-column matrix representing the lower and upper bounds of the interval. Each row of the matrix corresponds to the variable to which the interval applies. The default value of the method argument is perc. If the method argument is all, the function confint.boodd returns a list with the confidence intervals for all supported methods.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Efron, B., Tibshirani, R. (1993). *An Introduction to the Bootstrap*, Chapman and Hall.

See Also

plot.boodd,summary.boodd.

Examples

```
B <- 299
x <- round(rnorm(15),3)
boo1 <- boots(x,mean,B)
confint(boo1)
confint(boo1,method="bperc")

# bootstrap of several statistics
mv <- function(data) {c(mean(data),var(data)/length(data))} # compute both mean and variance
boo2 <- boots(x,mv,B)
# Compute both percentile and t-percentile confidence intervals when variance is bootstrapped
confint(boo2,method="all")
```

---

embb *Characteristics for Extension of Moving Block Bootstrap Class*

Description

The class of functions designed for bootstrap samples obtained using the Extension of Moving Block Bootstrap (EMBB) method or its circular version (CEMBB). These functions calculate seasonal means, seasonal variances, and seasonal autocovariances when a periodic time series with period length d is considered. For periodic and almost periodically correlated time series, the functions calculate the Fourier coefficients of the mean and autocovariance functions.

Usage

```
## S3 method for class 'embb'
seasonalMean(x, period, ...)
## S3 method for class 'embb'
seasonalVar(x, period, ...)
## S3 method for class 'embb'
seasonalACF(x, tau, period, ...)
## S3 method for class 'embb'
meanCoeff(x, period, freq, ...)
## S3 method for class 'embb'
acfCoeff(x, tau, period, freq, ...)
```

Arguments

x	An object of class embb.
period	An integer; period length of the original data.
tau	An integer or vector of integers; single lag or vector of lags.
freq	A vector of real numbers; vector of frequencies.
...	Additional arguments.

Details

These methods apply to objects of class embb typically obtained using the embb.sample function.

Value

- seasonalMean and seasonalVar return a vector of length period.
- seasonalACF returns either:
  - A vector of length period if a single lag tau is specified.
  - A matrix with length(tau) rows and period columns if tau is a vector.
- meanCoeff returns a vector of the same length as freq.
- acfCoeff returns either:
  - A vector of length length(freq) if a single lag tau is specified.
  - A matrix with length(tau) rows and length(freq) columns if tau is a vector.

References

Bertail, P., & Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.
Dudek, A. E. (2015). Circular block bootstrap for coefficients of autocovariance function of almost periodically correlated time series. *Metrika*, **78**, 313-335.
Dudek, A. E. (2018). Block bootstrap for periodic characteristics of periodically correlated time series. *Journal of Nonparametric Statistics*, **30**, 87-124.

See Also

embb.sample, seasonalMean.default, seasonalVar.default, seasonalACF.default

Examples

```
# Generate a periodically correlated time series
set.seed(123)
n=200
b <- arima.sim(n = n, model = list(ar = c(0.5, 0.4), na = 0.5))
period <- 12
x <- 5 * cos(2 * pi / period * (1:n)) + 5 * b * cos(2 * pi / period * (1:n))
X_ts <- ts(x)
bootstrapped_X <- embb.sample(X_ts, length.block = 15, method = "movingblock")
acf_results <- acfCoeff(bootstrapped_X, tau = 0, freq = 0)
mean_seasonal <- seasonalMean(bootstrapped_X, period = period)
```

`embb.sample` *EMBB Method*

Description

The function constructs a bootstrap sample using the Extension of Moving Block Bootstrap (EMBB) method or its circular version (CEMBB). The EMBB and CEMBB are suitable for periodically correlated and almost periodically correlated time series.

Usage

```
embb.sample(x,length.block,method=c("movingblock","circular"))
```

Arguments

`x`	An (almost) periodically correlated time series.
`length.block`	A positive integer; the number of bootstrap samples.
`method`	The bootstrap method: * `"movingblock"` - EMBB, * `"circular"` - CEMBB.

Details

The argument `method` can be set to `"movingblock"` (in the case of the `EMBB`) or to *"circular"* (in the case of `CEMBB`). Method names may be abbreviated.

Value

The `embb.sample` returns an object of type `embb` (not of type `boodd`), containing a matrix whose first column is the bootstrapped sample and second column contains the original time indices of the chosen observations.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

A.E. Dudek (2015). Circular block bootstrap for coefficients of autocovariance function of almost periodically correlated time series, *Metrika*, **78**, 313-335.

A.E. Dudek (2018). Block bootstrap for periodic characteristics of periodically correlated time series. *Journal of Nonparametric Statistics*, **30**, 87-124.

See Also

`embb`, `seasonalMean`,`seasonalVar`, `seasonalACF`, `meanCoeff`, `acfCoeff`.

Examples

```
# Generate a periodically correlated time series
n=200
b <- arima.sim(n = n, model = list(ar = c(0.5, 0.4), na = 0.5))
period <- 12
x <- 5*cos(2 * pi /period * (1:n))+5*b * cos(2 * pi /period * (1:n))
X_ts = ts(x)
bootstrapped_X <- embb.sample(X_ts, length.block = 15, method = "movingblock")
```

`fastNadaraya` *Nadaraya-Watson Estimator for Transition Densities for Markov chains.*

Description

Calculates the Nadaraya-Watson estimator for estimating the transition densities of a Markov chain. This function is particularly useful for approximating transition kernels in Markov chain analysis.

Usage

```
fastNadaraya(x, bandwidth)
```

Arguments

`x`	A numeric vector representing a Markov process.
`bandwidth`	A real number; the kernel bandwidth smoothing parameter

Details

The `fastNadaraya` function computes the estimated transition densities $p_n(X_i, X_{i+1})$ of a Markov chain. It is based on the Nadaraya kernel-type estimator for the transition density, with a bandwidth `bandwidth` provided by the user. This function is used in conjunction with `findBestEpsilon` to determine optimal small set for the Regenerative Block Bootstrap `regenboot`.

Value

Returns a numeric vector of size `length(x) - 1`, containing the estimated transition densities for each state transition in the Markov chain.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Bertail, P., Clémençon, S. (2006a). Regenerative Block Bootstrap for Markov Chains. *Bernoulli*, **12**, 689-712.

Bertail, P. and Clémençon, S. (2006). *Regeneration-based statistics for Harris recurrent Markov chains*, pages 1-54. Number 187 in Lecture notes in Statistics. Springer.

Radulović, D. (2006). Renewal type bootstrap for Markov chains. *Test*, **13**, 147-192.

See Also

findBestEpsilon, regenboot, bandw1.

Examples

```
x = arima.sim(1000, model=list(ar=c(0.4)))
h = bandw1(x)
fastNadaraya(x,h)
```

---

field.sub *Subsampling of Random Fields*

Description

Performs subsampling of a multidimensional array representing a random field on a lattice. This function applies a specified function to each subsample.

Usage

```
field.sub(arr, func, length.block, ...)
```

Arguments

arr	A multidimensional real-valued array; it represents a random field on a grid of dimension equals to dimension of the arr.
func	A function applied to each subsample. The function should accept the subsample as input and return a vector.
length.block	An integer or vector of integers; it specified the block lengths for subsampling. If a scalar is provided, the same block length is used for all dimensions
...	An optional additional arguments for the func function.

Details

The field.sub function is designed for subsampling a multidimensional array. The length.block argument defines the size of each subsample. If length.block is a scalar, it applies uniformly across all dimensions of the array. Otherwise, it must be a vector with a length equal to the number of dimensions in arr. The function func is applied to each subsample, and the results are returned in a matrix or vector, depending on the output of func.

Value

Returns a matrix or vector, depending on the output of the func function. Each row in the matrix corresponds to the result of applying func to a subsample. Since it is not a boodd object, one cannot apply, for example, the confint function to construct a confidence intervals (which depends on the rate of convergence of the statistic of interest), see example below.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Politis, D.N. Romano, J.P. Wolf, M. (1999). *Subsampling*, Springer, New York.

See Also

`fieldboot`, `fieldbootP`, `blockboot`, `jackVarField`.

Examples

```
set.seed(1)
# 2-dims array
dlens <- c(100,50)
blens <- c(6,3)
arr <- array(round(rnorm(prod(dlens)),2),dim=dlens)
resu<-field.sub(arr,mean,blens)
hist(resu,nclass=25)
N=length(resu)
lB=length(blens)
conf=mean(arr)-(quantile(resu,c(0.025,0.975))-mean(arr))*sqrt(lB)/sqrt(N)
conf[2:1]
```

`fieldboot` *Block Bootstrap of Random Field*

Description

Performs a bootstrap analysis of multidimensional array representing a random field on a lattice, using various block bootstrap methods such as moving block, circular block, or nonoverlapping block bootstrap.

Usage

```
fieldboot(
  arr,
  func,
  B,
  length.block,
  ...,
  method = c("movingblock", "nonoverlapping", "circular")
)
```

Arguments

`arr`	A multidimensional real-valued array; it represents a random field on a grid of dimension equals to dimension of the `arr`.
`func`	The function applied to each bootstrap sample.
`B`	A positive integer; the number of bootstrap samples.
`length.block`	A positive integer or vector of integers; it specified the block lengths for blocks. If a scalar is provided, the same block length is used for all dimensions.
`...`	Optional additional arguments for the `func` function.
`method`	The method for array reconstruction: • `"movingblock"` - Moving Block Bootstrap, • `"nonoverlapping"` - Nonoverlapping Block Bootstrap, • `"circular"` - Circular Block Bootstrap (obtained by wrapping the field on the torus). Default is `"movingblock"`.

Details

The `fieldboot` function resamples hyper-rectangles constructed using either moving blocks, nonoverlapping blocks or circular blocks to construct a bootstrap field of the same dimension as the original one. Then it applies the specified `func` to bootstrap samples of the provided data array. The `length.block` parameter determines the size of the blocks used in the bootstrap method. The `method` parameter specifies the type of block bootstrap to use. This function is useful for assessing the variability and distribution properties of a statistic in the context of random fields.

Value

Returns an object of class `boodd`.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Bertail, P. Politis, D. N. Rhomari, N. (2000). Subsampling continuous parameter random fields and a Bernstein inequality, *Statistics*, **33**, 367-392.

Nordman, D.J. Lahiri, S.N.(2004). On optimal spatial subsample size for variance estimation, *The Annals of Statistics*, **32**, 1981-2027.

Politis, D.N. Romano, J.P. (1993). Nonparametric Resampling for Homogeneous Strong Mixing Random Fields, *J. Multivar. Anal.*, **47**, 301-328.

See Also

`blockboot`, `jackVarField`, `field.sub`, `fieldbootP`.

Examples

```
set.seed(123)
arr <- array(rnorm(1000), dim = c(10, 10, 10))
res <- fieldboot(arr, mean, B = 100, length.block = c(2, 2, 2))
plot(res)
```

`fieldbootP` *Bootstrap Sample from the Random Field*

Description

Function returns a bootstrap sample of the random field on a lattice, using various block bootstrap methods such as moving block, circular block, or nonoverlapping block bootstrap.

Usage

```
fieldbootP(
  arr,
  length.block,
  method = c("movingblock", "nonoverlapping", "circular")
)
```

Arguments

`arr`	A multidimensional real-valued array; it represents a random field on a grid of dimension equals to dimension of the `arr`.
`length.block`	An integer or vector of integers; it specified the block lengths for blocks. If a scalar is provided, the same block length is used for all dimensions.
`method`	The method for array reconstruction: • `"movingblock"` - Moving Block Bootstrap, • `"nonoverlapping"` - Nonoverlapping Block Bootstrap, • `"circular"` - Circular Block Bootstrap (obtained by wrapping the field on the torus). Default is `"movingblock"`.

Details

In the case of random fields, the blocks B_i's are random matrices of size `length.block`. From the set of all blocks $\mathcal{B} = \{B_1, B_2, \ldots, B_q\}$, we select randomly with replacement blocks $B_1^*, \ldots, B_l^*$ and bind them together. The probability of choosing any block is $1/q$.

Value

Returns a bootstrap sample of given `arr`.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Bertail, P. Politis, D. N. Rhomari, N. (2000). Subsampling continuous parameter random fields and a Bernstein inequality, *Statistics*, **33**, 367-392.

Nordman, D.J. Lahiri, S.N.(2004). On optimal spatial subsample size for variance estimation, *The Annals of Statistics*, **32**, 1981-2027.

Politis, D.N. Romano, J.P. (1993). Nonparametric Resampling for Homogeneous Strong Mixing Random Fields, *J. Multivar. Anal.*, **47**, 301-328.

See Also

`blockboot`, `jackVarField`, `field.sub`, `fieldboot`.

Examples

```
library(geoR)
N=10
n=N^2
sim <- grf(n, grid="reg", cov.pars=c(1, .25), nsim=1)
image(sim)
arr=array(sim$data,dim=c(N,N))
mb=fieldbootP(arr,length.block=c(2,2),method="movingblock")
simb=sim
simb$data=as.vector(mb)
image(simb)
```

`findBestEpsilon` *Optimal Size of Small Sets*

Description

Determines the optimal radius ϵ of small sets, which maximizes the expected number of regeneration blocks. Useful for identifying the parameters for splitting a Markov chain into almost independent blocks.

Usage

```
findBestEpsilon(x, s = median(x), plotIt = FALSE)
```

Arguments

`x`	A numeric vector representing a Markov chain.
`s`	A real number specifying the center of the small set. Default is the median of `x`.
`plotIt`	Logical. If `TRUE` plots the estimated expected number of regenerations as a function of the radius ϵ.

Details

The `findBestEpsilon` calculates the optimal radius ϵ and the corresponding lower bound δ of the transition density for choosing the optimal small set of a Markov chain given the center `s` of a small set. It uses the `fastNadaraya` function to estimate the transition density $p_n(X_i, X_{i+1})$ taken at points (X_i, X_{i+1}).

Value

Returns an object of class `smallEnsemble`, which is a list containing:

- `s`: the chosen center of the small set,
- `epsilon`: estimated optimal radius of the small set,
- `delta`: estimated lower bound of the transition density over the optimal small set,
- `trans`: estimated values of $p_n(X_i, X_{i+1})$.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Bertail, P. and Clémençon, S. (2006). Regenerative block bootstrap for Markov chains. *Bernoulli*, **12**, 689-712.

See Also

`regenboot`, `fastNadaraya`, `naradamar`, `GetBlocks`, `GetPseudoBlocks`, `smallEnsemble`.

Examples

```
n=200 # the length of the process
# Generating the AR(1) process
sigma=1
coeff=0.75
X=arima.sim(n=n,list(ar=coeff, sd=sigma))
# Find the small ensemble with the largest number of regeneration
sm <- findBestEpsilon(X,s=0,plotIt=FALSE)
```

---

freqboot *Frequency Domain Bootstrap*

Description

Implements the Frequency Domain Bootstrap (FDB) for time series data.

Usage

```
freqboot(x, XI, g, B, kernel = "normal", bandwidth)
```

Arguments

x	A vector or time series.
XI	A list of functions defined on the interval $[0, \pi]$.
g	A numeric function accepting `length(XI)` arguments, used to compute the statistic of interest.
B	A positive integer; the number of bootstrap replications.
kernel	A character string specifying the smoothing kernel. The valid values are: • `"normal"` - default, • `"epanechnikov"`, • `"box"` - rectangular kernel.
bandwidth	A real number; the kernel bandwidth smoothing parameter. If unspecified, an optimal value is computed using formula $sd(x) * n^(- 1/3)$, which is smaller than the Silverman's rule-of-thumb bandwidth.

Details

The input series x is assumed to be a sample from a real-valued, zero-mean, stationary time series. The XI argument consists of functions ξ_i used to define linear functionals of the spectral density, say $A(\xi, f) = \int \xi_i(\omega) f(\omega) d\omega$. The statistic estimates $T(f) = g(A(\xi, f))$. The spectral density is estimated by smoothing the periodogram of the series, with the smoothing kernel specified by kernel and the smoothing parameter bandwidth. The FDB consists in resampling periodogram ordinates standardized by the spectral density estimates to recompute the bootstrap values of the statistics of interest.

Value

Returns an object of class boodd.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Hurvich, C. M. and Zeger, S. L. (1987). Frequency domain bootstrap methods for time series, Technical Report 87-115, Graduate School of Business Administration, New York Univ.

Bertail, P. and Dudek, A. (2021). Consistency of the Frequency Domain Bootstrap for differentiable functionals, *Electron. J. Statist.*, **15**, 1-36.

Lahiri, S.N. (2003). *Resampling Methods for Dependent Data*. Springer, New York.

See Also

aidedboot, func_fdb, per_boo, tft_boot.

Examples

```
set.seed(123)
n <- 120
x <- arima.sim(list(order=c(1,0,0),ar=0.7),n=n)
B <- 999
one <- function(x) {1}
XI <- list(cos,one)
g <- function(x,y) {return(x/y)}
# This gives an estimate for the autocorrelation of order 1
boo = freqboot(x,XI,g,B,"normal")
plot(boo)
```

`ftrunc` *Robust Estimators of the Mean Based on Regeneration Blocks.*

Description

The function calculates various statistics (mean, median, truncated, and Winsorized mean) based on regeneration blocks obtained from the data. It also computes the mean block size. It relies on block-based calculations for robust statistics by eliminating either too large blocks or too large values of the mean on a given block.

Usage

```
ftrunc(x, atom_f, m = quantile(x, 0.05), M = quantile(x, 0.95), trunc)
```

Arguments

`x`	A vector or time series.
`atom_f`	A numeric value specifying an atom of the Markov chain.
`m`	A numeric value; the lower truncation threshold Default is the 5th percentile of `x`.
`M`	A numeric value; the upper truncation threshold Default is the 95th percentile of `x`.
`trunc`	A numeric value specifying the truncation threshold for computing the truncated and Winsorized means of the block length.

Details

This function uses blocks obtained from the input data `x` to compute several descriptive statistics, including the mean size of blocks, overall mean, median, and robust estimates like truncated and Winsorized means. The function internally uses `GetBlocks` to divide the input data into regenerative blocks when the process hits the atom `atom_f`.

The parameters `m` and `M` represent the lower and upper truncation thresholds, respectively. By default, these are set to the 5th and 95th percentiles of the input data, but they can be manually adjusted by the user to perform customized truncation.

The parameter `trunc` is used to eliminate blocks which lengths are greater than `trunc`.

Value

A numeric vector containing the following elements:

- `"taumean"` - the mean block size,
- `"fmean"` - the mean of the input data,
- `"fmed"` - the median of the input data,
- `"tmean"` - the truncated mean of the blocks based on the truncation threshold,
- `"wmean"` - the Winsorized mean of the blocks based on the truncation threshold.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Bertail, P., Clémençon, S. and Tressou, J. (2015). Bootstrapping Robust Statistics for Markovian Data Applications to Regenerative R-Statistics and L-Statistics. *Journal of Time Series Analysis*, **36**, 462–480.

See Also

`GetBlocks`, `findBestEpsilon`, `GetPseudoBlocks`, `smallEnsemble`, `regenboot`.

Examples

```
n=500 # the length of the process
lambda=0.6 # arrival rate
mu=0.8 # departure rat
X = genMM1(n,lambda,mu) # generate MM1 queue
atom = 0 # specify the atom
trunc = 30 # set truncation threshold
result = ftrunc(x=X, atom_f=atom, m=0, trunc = trunc) # apply function
print(result)
```

---

func_fdb *Functional Bootstrap in the Frequency Domain (FDB)*

Description

Uses the Frequency Domain Bootstrap (FDB) to compute the bootstrap spectral density, cumulative distribution function for the estimated spectral density, and the quantiles of the standardized distribution.

Usage

```
func_fdb(
  x,
  B,
  kernel = "normal",
  bandwidth,
  p = 0.5,
  PLT = c("spec", "cdf", NULL)
)
```

Arguments

x	A vector or time series.
B	A positive integer; the number of bootstrap replications.
kernel	A character string specifying the smoothing kernel. The valid values are: • "normal" - default, • "epanechnikov", • "box" - rectangular kernel.
bandwidth	A real number; the kernel bandwidth smoothing parameter. If unspecified, an optimal value is computed using formula $sd(x) * n^(- 1/3)$, which is smaller than the Silverman's rule-of-thumb bandwidth.
p	A vector of the quantiles to be calculated. Default is p = 0.5.
PLT	An argument specifying what to plot: • "spec" - spectral density, • "cdf" - cumulative distribution, • NULL - no plots - default.

Details

The function performs a bootstrap in the frequency domain and computes the specified functionals. It estimates the spectral density using a periodogram smoothed by the specified kernel and bandwidth. The function then computes the cumulative spectral density and quantiles based on this estimation.

Value

The function returns a list of two components:

1. The boodd object:
   - s: A matrix of size $B \times (2n_0 + l_p)$, where $n_0 = \lfloor n/2 \rfloor$ and l_p is the length of vector p. The first n_0 columns contain bootstrap values of the spectral density at the frequency $2\pi n_0$. The columns $n_0 + 1$ up to $2n_0$ contain values of the cumulative distribution function corresponding to the bootstrap spectral density, and the last l_p columns contain the estimated quantiles.
   - Tn: Estimated values of the spectral density, the estimated cumulative distribution of the spectral density, and the quantiles.
2. freqs: The vector of frequencies at which the functionals are computed for the process x.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Bertail, P. and Dudek, A.E. (2021). Consistency of the Frequency Domain Bootstrap for differentiable functionals, *Electron. J. Statist.*, **15**, 1-36.

Hurvich, C. M. and Zeger, S. L. (1987). Frequency domain bootstrap methods for time series, Technical Report 87-115, Graduate School of Business Administration, New York Univ.

Lahiri, S.N. (2003). *Resampling Methods for Dependent Data*. Springer, New York.

See Also

aidedboot,tft_boot, aidedboot, freqboot, per_boo.

Examples

```
# Choice of sample size
n <- 1000
# Simulate AR(1) model with parameter 0.6
 x <- arima.sim(list(order=c(1,0,0),ar=0.6),n=n)
 n0=floor(n/2)
 bf=func_fdb(x,199, PLT="spec")
```

`f_PseudoBlocks` *Compute the Value of the Function on a (Pseudo)-Regenerative Blocks.*

Description

Function is an adaptation of `GetPseudoBlocks` to compute the value of any function on pseudo-regenerative blocks.

Usage

```
f_PseudoBlocks(x, s, eps_opt, delta_opt, p_XiXip1, func = sum)
```

Arguments

`x`	A vector or time series.
`s`	A real number specifying the center of the small set.
`eps_opt`	A numeric value for the size of the small set.
`delta_opt`	A numeric value for the lower bound in the minorization condition.
`p_XiXip1`	A numeric value representing the estimator of the transition density.
`func`	A function to apply to each block. Default is `sum`.

Details

This function computes the value of a specified function on pseudo-regenerative blocks of a time series. It uses parameters such as the central value (`s`), the size of the small set (`eps_opt`), and the lower bound in the minorization condition (`delta_opt`). Robustification is not proposed here due to the complexity of the pseudo-regenerative procedure.

Value

A matrix with two columns:

- `Subf` - the value of the function `func` on each sub-block,
- `lB` - the length of the sub-blocks on which the function is computed.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Bertail, P. and Clémençon, S. (2006). Regenerative block bootstrap for Markov chains. *Bernoulli*, **12**, 689-712.

See Also

`GetPseudoBlocks`, `regenboot`, `findBestEpsilon`, `GetBlocks`, `smallEnsemble`.

Examples

```
n=1000
coeff=0.75
X = arima.sim(n=n, list(ar = c(coeff)))
sm <- findBestEpsilon(X,s=0,plotIt=FALSE)
eps = sm$epsilon
delta = sm$delta
m = sm$s
f = sm$trans
result <- f_PseudoBlocks(X, 0, eps, delta, f, func = max)
print(result)
```

---

genETARCH *Generate an Exponential TAR-ARCH Process*

Description

Simulates an Exponential Threshold Autoregressive (ETAR)-ARCH process.

Usage

```
genETARCH(n, alpha1, alpha2, beta)
```

Arguments

n	An integer value; the length of the process to simulate.
alpha1	A numeric value; parameter of the process such that $\|alpha1\| < 1$.
alpha2	A numeric value; parameter of the process driving the behaviour of the process for small values of x. $\|alpha1 + alpha2\|$ may be bigger than 1.
beta	A positive numeric value; parameter of the process driving the ARCH effect.

Details

The ETAR-ARCH process is defined by the equation

$$X_{t+1} = \left(\alpha_1 + \alpha_2 \exp(-X_t^2)\right) X_t + \sqrt{1 + \beta X_t^2} Z_t$$

, where Z_t is an i.i.d. error sequence.

Value

A numeric vector of length n+1, containing the simulated values of the ETAR-ARCH process.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

See Also

genMM1, zi_inar_process.

Examples

```
etarch = genETARCH(100,0.3,0.8,0.5)
plot(etarch, type="l")
```

---

genMM1 *Generate an M/M/1 Queue Process*

Description

Simulates an M/M/1 queue process. The M/M/1 queue is a single-server queue with Poisson arrivals and exponential service times. Allows to introduce an artificial outlier.

Usage

```
genMM1(n, lambda, mu, out = NULL)
```

Arguments

n	An integer value; the length of the process to simulate.
lambda	A positive numeric value; rate parameter for the Poisson arrival process.
mu	A positive numeric value; rate parameter for the exponential service times.
out	A positive numeric value; the value of an outlier appearing at time $floor(n/2)+1$. Default is NULL, this means that the process is generated without outlier.

Details

The M/M/1 waiting time is defined as follows

$$X_{t+1} = \max(0, (X_t + U_t - V_t))$$

, where U_t and V_t are exponentially distributed random variables with parameters lambda and mu, respectively.

Value

A numeric vector of length n+1, representing the waiting time of customer i arriving at time T_i.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Asmussen, S. (1987). *Applied Probability and Queues*. Springer N.Y.

See Also

genETARCH, zi_inar_process.

Examples

```
queue = genMM1(100,1,2)
plot(queue, type="l")
```

GetBlocks *Compute Block Splitting for Atomic Markov Chains*

Description

Computes regenerative blocks for atomic Markov chains.

Usage

```
GetBlocks(X, atom, m = min(X), M = max(X), func = sum, ...)
```

Arguments

X	A numeric vector representing a Markov chain.
atom	A numeric value; an atom of the Markov chain.
m	A numeric value; the lower truncation threshold Default is the 5th percentile of X.
M	A numeric value; the upper truncation threshold Default is the 95th percentile of X.
func	A function to apply to each block. Default is sum.
...	Additional arguments passed to the function func.

Details

Identifies values in X equal to atom to determines regeneration times and creates regeneration blocks (or cycles). The function then assigns block numbers, counts observations in each block, and calculates various statistics for each block.

Value

Returns a list containing:

1. A data frame with the following columns:
   - Time - the index of each observation,
   - X - values of the process,
   - Bnumber - block number assigned to each observation,
   - regen - indicator (1 or 0) of regeneration times. 1 corresponds to the regeneration time.
2. A matrix summarizing block characteristics with the following columns:
   - Block number - the block index,
   - Block length - number of observations in the block,
   - Truncated sum - the value of func applied to truncated observations in the block,
   - Valid points - number of observations within the truncation thresholds,
   - Winsorized value - the Winsorized value of func applied to the block,
   - Start index - the starting index of the block,
   - End index - the ending index of the block.
3. Total blocks - the total number of regeneration blocks.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

See Also

findBestEpsilon, ftrunc, regenboot, smallEnsemble.

Examples

```
X = genMM1(1000, 1, 2)
blocks = GetBlocks(X, 0, func=sum) # compute sum over all blocks (without truncation)
# compute sum over all blocks (with truncation over quantiles of order 5% and 95%)
blocks = GetBlocks(X, 0, func=sum,m=quantile(X,0.05),M=quantile(X,0.95))
```

GetPseudoBlocks *Computing Pseudo-regenerative Blocks*

Description

The function computes pseudo-regenerative blocks for general Markov chains.

Usage

```
GetPseudoBlocks(
  x,
  s,
  eps_opt,
  delta_opt,
  p_XiXip1,
  m = min(x),
  M = max(x),
  func = sum,
  ...
)
```

Arguments

x	A numeric vector representing a Markov chain.
s	A real number specifying the center of the small set.
eps_opt	A numeric value for the size of the small set.
delta_opt	A numeric value for the lower bound in the minorization condition.
p_XiXip1	A numeric value representing the estimator of the transition density.
m	A numeric value; the lower truncation threshold Default is the 5th percentile of x.
M	A numeric value; the upper truncation threshold Default is the 95th percentile of x.
func	A function to apply to each block. Default is sum.
...	Additional arguments passed to the function func.

Details

The function begins by determining which elements of x are within an interval $[s - esp_opt, s + eps_opt]$. Then an estimated Nummelin splitting trick is performed using the estimators $p_n(X_i, X_{i+1})$.

Value

Returns a list containing:

1. A data frame with the following columns:
   - `Time` - the index of each observation,
   - `x` - values of the process,
   - `Bnumber` - block number assigned to each observation,
   - `regen` - indicator (1 or 0) of regeneration times. 1 corresponds to the regeneration time.
2. A matrix summarizing block characteristics with the following columns:
   - `Block number` - the block index,
   - `Block length` - number of observations in the block,
   - `Truncated sum` - the value of `func` applied to truncated observations in the block,
   - `Valid points` - number of observations within the truncation thresholds,
   - `Winsorized value` - the Winsorized value of `func` applied to the block,
   - `Start index` - the starting index of the block,
   - `End index` - the ending index of the block.
3. `Total blocks` - the total number of regeneration blocks.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Bertail, P. and Clémençon, S. (2006). Regenerative block bootstrap for Markov chains. *Bernoulli*, **12**, 689-712.

See Also

`findBestEpsilon`, `ftrunc`, `regenboot`, `smallEnsemble`.

Examples

```
n=200# the length of the process
# Generating the AR(1) process
coeff=0.75
X = arima.sim(n=n, list(ar = c(coeff)))
# Find the small ensemble with the largest number of regeneration
sm <- findBestEpsilon(X,s=0,plotIt=FALSE)
f =sm$trans
eps = sm$epsilon
delta = sm$delta
m = sm$s
Pseudo_blocks=GetPseudoBlocks(X, m, eps_opt = eps, delta_opt = delta, p_XiXip1 = f,func=sum)
```

jackFunc *Jackknife Variance Function*

Description

Creates a vector-valued function for computing both the statistic defined by `func` and the estimated jackknife variance of the statistic.

Usage

```
jackFunc(func, ...)
```

Arguments

`func`	The function used to compute the statistic on each sample.
`...`	Optional additional arguments for the `func` function.

Details

The `jackFunc` function constructs a new function that, when applied to a data sample, calculates both the statistic specified by `func` and its associated jackknife variance. This newly created function is useful in generic bootstrap procedures, particularly for constructing bootstrap-t confidence intervals.

Value

Returns an object, which is a function.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Efron, B. (1979). Bootstrap methods: another look at the jackknife. *Ann. Statist.*, **7**, 1-26.

Gray, H., Schucany, W. and Watkins, T. (1972). *The Generalized Jackknife Statistics*. Marcel Dekker, New York.

Quenouille, M.H. (1949). Approximate tests of correlation in time-series. *J. Roy. Statist. Soc., Ser. B*, **11**, 68-84.

See Also

`jackVar`, `boots`, `jackVarBlock`, `jackFuncBlock`, `jackFuncRegen`.

Examples

```
# Create a function to compute the empirical skewness
func <- function(x) { mean((x - mean(x))^3) / (mean((x - mean(x))^2)^(3/2)) }
x <- rnorm(100)
# Create a function to compute the empirical skewness and its variance
jf <- jackFunc(func)
# Bootstrapping of the skewness and its variance allows to construct
# bootstrap-t confidence intervals
boo1 <- boots(x, jf, 299)
confint(boo1, method="all")
```

---

jackFuncBlock *Jackknife Variance Function Using Blocks of Fixed Length*

Description

Creates a vector-valued function for computing both the statistic defined by `func` and the estimated jackknife variance based on the blocks.

Usage

```
jackFuncBlock(func, length.block = NULL, ...)
```

Arguments

`func`	The function used to compute the statistic on each sample.
`length.block`	An integer; the block length. If not provided, a default value equals $\lfloor (length(X))^{\frac{1}{3}} \rfloor$.
`...`	Optional additional arguments for the `func` function.

Details

The `jackFuncBlock` function constructs a new function that, when applied to a data sample, calculates both the statistic specified by `func` and its associated jackknife variance based on non-overlapping blocks of fixed length equals to `length.block`.

The block jackknife method is an extension of the jackknife resampling technique, used to estimate the bias and variance of a statistical estimate in the presence of dependent data.

Value

Returns an object which is a function.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Carlstein, E. (1986). The use of subseries methods for estimating the variance of a general statistic from a stationary time series. *Annals of Statist.*, **14**, 1171-1179.

Gray, H., Schucany, W. and Watkins, T. (1972). *The Generalized Jackknife Statistics*. Marcel Dekker, New York.

Quenouille, M.H. (1949). Approximate tests of correlation in time-series. *J. Roy. Statist. Soc., Ser. B*, **11**, 68-84.

See Also

jackVar, jackFunc, blockboot, jackVarBlock, jackFuncRegen.

Examples

```
# Create a function to compute the empirical skewness
func <- function(x) { mean((x - mean(x))^3) / (mean((x - mean(x))^2)^(3/2)) }
x <- arima.sim(list(order = c(1, 0, 4), ar = 0.5, ma = c(0.7, 0.4, -0.3, -0.1)), n = 100)
# Create a function to compute the empirical skewness and its variance based on blocks of size 5
jfb <- jackFuncBlock(func, length.block = 5)
# Bootstrapping of the skewness and its variance allows to construct
# bootstrap-t confidence intervals
boo2 <- blockboot(x, jfb, 299, length.block=7, method="circular")
confint(boo2, method="all")
```

---

jackFuncRegen — *Jackknife Variance Function for Markov Chains Using Regenerative Blocks*

Description

Creates a function that calculates both a specified statistic and its jackknife variance estimator based on regenerative blocks for Markov chains.

Usage

```
jackFuncRegen(func, atom = atom, small = NULL, ...)
```

Arguments

func	A function for which the statistic and the jackknife variance are to be calculated.
atom	A numeric value or a string; an atom of the Markov chain..
small	An optional object of class smallEnsemble. It can be created optimally using findBestEpsilon.
...	Optional additional arguments for the func function.

Details

This function is designed for use with regenerative data, such as Markov chains. It employs regeneration-based methods to estimate the jackknife variance of a given statistic by omitting alternatively each block to compute the function of interest. This function is particularly useful in conjunction with functions like `regenboot` and provides an alternative to block-based methods like `jackFuncBlock`.

Value

Returns an object which is a function.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Quenouille, M.H. (1949). Approximate tests of correlation in time-series. *J. Roy. Statist. Soc., Ser. B*, **11**, 68-84.

See Also

`jackVar`, `jackFunc`, `regenboot`, `jackVarBlock`, `jackFuncBlock`.

Examples

```
x=genMM1(100,1,2)
#' # A function to compute the mean of strictly positive values and its variance based on
# regenerative blocks
func <- function(x) {mean(x*(x>0))}
jfb <- jackFuncRegen(func, atom=0)
# Regenerative bootstrap of the mean of strictly positive values and its variance allows
# to construct bootstrap-t confidence intervals
boo <- regenboot(x,jfb,99, atom=0)
confint(boo, method="all")
```

---

jackVar *Jackknife Variance Estimator*

Description

Estimates the variance of a statistic using the jackknife-variance procedure in the i.i.d case.

Usage

```
jackVar(x, func, ...)
```

Arguments

x	A vector or a matrix representing the data.
func	The function used to compute the statistic on each sample.
...	Optional additional arguments for the func function.

Details

When x is a vector of length *n* or a matrix with *n* rows, the function func, having output size equal to *p*, is applied to x with each *i*-th row removed, resulting in

$$T_{n-1}^{i} = func(x[-i]).$$

The jackknife variance is computed based on these recalculated statistics and the original statistic

$$T_n = func(x).$$

The covariance matrix is calculated according to the jackknife formula.

This method is used to estimate the variance of a statistic that is potentially biased due to the finite sample size.

Value

Returns a scalar or a covariance matrix, depending on whether the function func is univariate or multivariate. For a function returning a vector of length *p*, the output will be a covariance matrix of size *p x p*.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Efron, B. (1979). Bootstrap methods: another look at the jackknife. *Ann. Statist.*, **7**, 1-26.

Gray, H., Schucany, W. and Watkins, T. (1972). *The Generalized Jackknife Statistics*. Marcel Dekker, New York.

Quenouille, M.H. (1949). Approximate tests of correlation in time-series. *J. Roy. Statist. Soc., Ser. B*, **11**, 68-84.

See Also

jackFunc, boots, jackVarBlock, jackFuncBlock, jackFuncRegen.

Examples

```
set.seed(1)
x <- rnorm(101)
func <- function(x) { mean(x^2) }
jackVar(x, func)
# Function returning a vector with the mean and standard deviation of x
mfunc <- function(x) { c(mean(x), sd(x)) }
jackVar(x, mfunc)
```

```
# Function to compute the moment of order p with p as additional argument
funca <- function(x, p) { mean((x-mean(x))^p)}
jackVar(x, funca, 3)
```

`jackVarBlock` *Jackknife Variance Estimator Based on Fixed Length Blocks*

Description

Estimates the variance of a statistic applied to a vector or a matrix using a block jackknife procedure.

Usage

```
jackVarBlock(x, func, length.block, ...)
```

Arguments

`x`	A vector or a matrix representing the data.
`func`	The function used to compute the statistic on each sample.
`length.block`	An integer; the block length.
`...`	Optional additional arguments for the `func` function.

Details

The `jackVarBlock` function extends the jackknife variance estimation to block data. The data `x` is divided into non-overlapping blocks of length `length.block`. The function `func`, having output size equal to p, is applied to the data with each block removed in turn to finally compute an empirical variance of the obtained values. This approach is particularly useful for dependent data, where traditional jackknife methods may not be appropriate.

Value

Returns a scalar or a covariance matrix, depending on whether the function `func` is univariate or multivariate. For a function returning a vector of length p, the output will be a covariance matrix of size $p \times p$.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Carlstein, E. (1986). The use of subseries methods for estimating the variance of a general statistic from a stationary time series. *Annals of Statist.*, **14**, 1171-1179.

Gray, H., Schucany, W. and Watkins, T. (1972). *The Generalized Jackknife Statistics*. Marcel Dekker, New York.

Quenouille, M.H. (1949). Approximate tests of correlation in time-series. *J. Roy. Statist. Soc., Ser. B*, **11**, 68-84.

See Also

jackVar, jackFunc, blockboot, jackFuncBlock, jackFuncRegen.

Examples

```
set.seed(1)
x <- arima.sim(list(order = c(1, 0, 4), ar = 0.5, ma = c(0.7, 0.4, -0.3, -0.1)), n = 101)
# Jackknife variance estimator of 'func' with blocks of length length.block
length.block <- 10
V1 <- jackVarBlock(x, mean, length.block)
```

---

jackVarField *Jackknife Variance for Random Fields Based on Blocks*

Description

Estimates the variance of a statistic applied to random fields using non-overlapping multidimensional blocks.

Usage

```
jackVarField(arr, func, length.block, ...)
```

Arguments

arr	A multidimensional real-valued array; it represents a random field on a grid of dimension equal to dimension of the arr.
func	The function applied to each bootstrap sample.
length.block	An integer or vector of integers; it specified the block lengths for blocks. If a scalar is provided, the same block length is used for all dimensions.
...	Optional additional arguments for the func function.

Details

The jackVarField function computes the jackknife variance estimator for random fields. It involves dividing the array into non-overlapping blocks of size specified by length.block and applying the function func, having output size equal to p, to the data with each block removed in turn. Finally, an empirical variance of the obtained values is computed. This method is particularly useful for data with spatial or multidimensional structure.

Value

Returns a scalar or a covariance matrix, depending on whether the function func is univariate or multivariate. For a function returning a vector of length p, the output will be a covariance matrix of size p x p.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Carlstein, E. (1986). The use of subseries methods for estimating the variance of a general statistic from a stationary time series. *Annals of Statist.*, **14**, 1171-1179.

Gray, H., Schucany, W. and Watkins, T. (1972). *The Generalized Jackknife Statistics*. Marcel Dekker, New York.

Quenouille, M.H. (1949). Approximate tests of correlation in time-series. *J. Roy. Statist. Soc., Ser. B*, **11**, 68-84.

See Also

jackVar, jackFunc, blockboot, jackVarBlock, jackFuncRegen.

Examples

```
arr <- array(rnorm(1000), dim = c(10, 10, 10))
func <- function(x) { mean(x) }
length.block <- c(2, 2, 2)
result <- jackVarField(arr, func, length.block)
```

---

jackVarRegen *Jackknife Variance Estimator for Regenerative Processes*

Description

Estimates the variance of a statistic applied to a vector or a matrix using a jackknife procedure tailored for regenerative processes, in particular recurrent Markov chains.

Usage

```
jackVarRegen(x, func, ..., atom, small = NULL, s = median(x))
```

Arguments

x	A vector or a matrix representing the data.
func	The function applied to each sample.
...	Optional additional arguments for the func function.
atom	A numeric value or a string; an atom of the Markov chain in the atomic case.
small	An optional object of class smallEnsemble. It can be created optimally using findBestEpsilon.
s	A real number specifying the center of the small set. Default is the median of x.

Details

The jackVarRegen function is a versatile tool for estimating the jackknife variance in cases of statistics based on regenerative blocks. It accommodates variable length blocks and is effective for both finite state and general Markov chains. It calls jackVarRegen.atom and jackVarRegen.smallEnsemble, respectively in the atomic and the general case.

Value

Returns a scalar or a covariance matrix, depending on whether the function func is univariate or multivariate. For a function returning a vector of length *p*, the output will be a covariance matrix of size *p x p*.

References

Bertail, P. and Clémençon. S. (2006). *Regeneration-based statistics for Harris recurrent Markov chains*, 1-54. Lecture notes in Statistics 187. Springer.

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Quenouille, M.H. (1949). Approximate tests of correlation in time-series. *J. Roy. Statist. Soc., Ser. B*, **11**, 68-84.

Quenouille, M. H. (1956). Notes on bias in estimation , *Biometrika*, **43**, 353–360.

See Also

jackVar, jackFunc, regenboot, jackFuncRegen, jackFuncBlock, jackVarRegen.atom, jackVarRegen.smallEnsemble.

Examples

```
acgt <- c("A", "C", "G", "T")
probs <- c(.3, .1, .3, .3)
n <- 100
atom <- "A"
set.seed(1)
y <- sample(acgt, n, prob=probs, repl=TRUE)
propAtom <- function(x) {
  tbl <- as.vector(table(x))
  prop <- tbl[1] / length(x)
  return(prop)
}
jackVarRegen(y, propAtom, atom=atom)
```

`jackVarRegen.atom` *Jackknife Variance Estimator for Atomic Markov Chains*

Description

Provides a regenerative jackknife estimator of the variance of a function applied to atomic Markov chains.

Usage

```
jackVarRegen.atom(x, func, atom, ...)
```

Arguments

`x`	A vector or a matrix representing the Markov chain.
`func`	The function to apply to each sample.
`atom`	A numeric value or a string; an atom of the Markov chain in the atomic case.
`...`	Optional additional arguments for the `func` function.

Details

This function uses a regenerative approach to estimate the jackknife variance of a statistic for atomic Markov chains. It segments the chain at the specified `atom` into independent blocks. The function `func`, having output size equal to p, is applied to the data with each regenerative block removed in turn to finally compute an empirical variance of the obtained values. This approach is particularly useful for atomic Markov chains.

Value

Returns a scalar or a covariance matrix, depending on whether the function `func` is univariate or multivariate. For a function returning a vector of length p, the output will be a covariance matrix of size p x p.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Quenouille, M.H. (1949). Approximate tests of correlation in time-series. *J. Roy. Statist. Soc., Ser. B*, **11**, 68-84.

Quenouille, M. H. (1956). Notes on bias in estimation. *Biometrika*, **43**, 353–360.

See Also

`jackVar`, `jackFunc`, `regenboot`, `jackFuncRegen`, `jackFuncBlock`, `jackVarRegen`.

Examples

```
B=1000
set.seed(5)
bb=0*(1:B)
cc=0*(1:B)
for (i in 1:B) {
  ts=genMM1(100,2,4)
  vv=function(ts){as.numeric(var(ts))}
  bb[i]=mean(ts)
  cc[i]=jackVarRegen.atom(ts,mean,atom=0)}
var(bb)  # true variance of the mean (evaluated by Monte-Carlo)
mean(cc) # mean of the variance estimators over the Monte-Carlo simulations
```

---

jackVarRegen.smallEnsemble

Jackknife Variance Estimation for General Harris Markov Chains

---

Description

Estimates the jackknife variance of a function applied to general Harris Markov chains using a regenerative approach and a `smallEnsemble` object.

Usage

```
jackVarRegen.smallEnsemble(x, func, small, ...)
```

Arguments

`x`	A vector or matrix representing the data from a general Harris Markov chain.
`func`	The function to apply to each sample.
`small`	An object of class `smallEnsemble`. It can be created optimally using `findBestEpsilon`.
`...`	Optional additional arguments for the `func` function.

Details

`The function uses a regenerative approach to estimate the`

jackknife variance for functions applied to general Harris Markov chains. It relies on a `smallEnsemble` object to define the regenerative structure of the data. It segments the chain using an estimated Nummelin splitting trick to create almost independent blocks. The function `func`, having output size equal to p, is applied to the data with each approximate regenerative block removed in turn to finally compute an empirical variance of the obtained values.

Value

Returns a scalar or a covariance matrix, depending on whether the function `func` is univariate or multivariate. For a function returning a vector of length p, the output will be a covariance matrix of size p *x* p.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Quenouille, M.H. (1949). Approximate tests of correlation in time-series. *J. Roy. Statist. Soc., Ser. B*, **11**, 68-84.

See Also

jackVar, jackFunc, regenboot, jackFuncRegen, jackFuncBlock, jackVarRegen.

Examples

```
B=10
bb=0*(1:B)
cc=0*(1:B)
dd=0*(1:B)
for (i in 1:B) {
  ts=arima.sim(list(ar=0.4),200)
   vv=function(ts){as.numeric(var(ts))}
   bb[i]=mean(ts)
   cc[i]=jackVarRegen.smallEnsemble(ts,mean, small= findBestEpsilon(ts))}
var(bb)
mean(cc)
# Monte Carlo simulations
mean(dd)
```

---

lam *Lag window*

Description

The function lam is used to construct a "flat-top" lag window for spectral estimation based on Politis, D.N. and J.P. Romano (1995), "Bias-Corrected Nonparametric Spectral Estimation", Journal of Time Series Analysis, vol. 16, No. 1.

Usage

```
lam(s)
```

Arguments

s	a time series.

Details

This function constructs a lag window used in spectral estimation. More details about the lag window and its usage can be found in the referenced papers.

Value

A lag window for spectral estimation.

Author(s)

Original code in Matlab by A. Patton, R translation and modifications by C. Parmeter and J. Racine. We are grateful to Andrew Patton and Dimitris Politis for their assistance and feedback. Kindly report features, deficiencies, and improvements to <racinej@mcmaster.ca>.

References

Patton, A., Politis, D.N. and White, H. (2009). Correction to "Automatic Block-Length Selection for the Dependent Bootstrap" by D. Politis and H. White, *Econometric Reviews*, **28**, 372-375.

See Also

blockboot, b.star

Examples

```
# Generate a sequence for testing
s <- seq(-1, 1, by = 0.1)
# Calculate the lag window using the lam function
lag_window <- lam(s)
# Plot the generated lag window
plot(lag_window, type="l")
```

---

mark_boot *Bootstraping Markov chain*

Description

The function fo bootstrap the Markov chain using estimator of the transition kernel

Usage

```
mark_boot(X, func, B, ...)
```

Arguments

X	A numeric vector representing a Markov chain.
func	The function to apply to each sample.
B	A positive integer; the number of bootstrap samples.
...	Optional additional arguments for the func function.

Details

The method is based on estimating the transition kernel of the chain, which is used to generate the bootstrap time series. The transition density is estimated using some Gaussian kernel.

Value

Returns an object of class boodd.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted..

Prakasa Rao, B. L. S. and Kulperger, R. J. (1989). Bootstrapping a finite state Markov chain. *Sankhya - Series A*, **51**, 178-191.

Rajarshi, M.B. (1990). Bootstrap in Markov-Sequences Based on Estimates of Transition Density. *Annals of the Institute of Statistical Mathematics*, **42**, 253-268.

See Also

blockboot, regenboot, findBestEpsilon.

Examples

```
set.seed(12345)
phi=0.6
n=200
X <- arima.sim(list(order=c(1,0,0),ar=phi),n=n)
 boo1=mark_boot(X,mean,199)
 boot_dist(boo1)
 # Compute confidence intervals
 confint(boo1,method="all")
```

---

meanCoeff, acfCoeff *Fourier Coefficients Estimation of the Mean and Autocovariance Functions.*

Description

For both periodically (PC) and almost periodically correlated (APC) data, the functions calculate the Fourier coefficients of the mean and autocovariance functions. The function can also be used for bootstrap samples obtained with the EMBB, CEMBB, GSBB, CGSBB.

Usage

```
    meanCoeff(x, period, freq, ...)
    acfCoeff(x, tau, period, freq, ...)

    ## Default S3 method:
meanCoeff(x, period, freq, ...)
    ## Default S3 method:
acfCoeff(x, tau, period, freq, ...)

    ## S3 method for class 'ts'
meanCoeff(x, period=frequency(x), freq, ...)
    ## S3 method for class 'ts'
acfCoeff(x, tau, period=frequency(x), freq, ...)
```

Arguments

x	A vector or time series representing a periodically or almost periodically correlated time series.
period	A positive integer; the period length. By default it is frequency(x).
tau	A vector of integers; a single lag or vector of lags.
freq	A numeric vector of frequencies.
...	Optional additional arguments for the function.

Details

If the freq argument is not specified, the Fourier frequencies are used: $2 * k * pi/period$ for *k=0,1,...,period*, where period is the frequency of the time series.

The meanCoeff function implements the estimator of the Fourier coefficient of the mean at frequency γ:

$$\widehat{b}(\gamma) = \frac{1}{n}\sum_{t=1}^{n} X_t e^{-i\gamma t}.$$

The acfCoeff function implements the estimator of the Fourier coefficient of the autocovariance for given lag tau at frequency λ:

$$\widehat{a}(\lambda,\tau) = \frac{1}{n}\sum_{t=1-\min\{\tau,0\}}^{n-\max\{\tau,0\}} (X_{t+\tau} - \widehat{\mu}_n(t+\tau))(X_t - \widehat{\mu}_n(t))e^{-i\lambda t}.$$

Value

meanCoeff returns a vector of the same length as freq.

acfCoeff returns either a vector of length length(freq) if a single lag tau is specified, or a matrix with length(tau) rows and length(freq) columns if tau is a vector.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted..

Dudek, A.E. (2015). Circular block bootstrap for coefficients of autocovariance function of almost periodically correlated time series. *Metrika*, **78**, 313-335.

Dudek, A.E. Maiz, S. and Elbadaoui, M. (2014). Generalized Seasonal Block Bootstrap in frequency analysis of cyclostationary signals. *Signal Process.*, **104C**, 358-368.

See Also

seasonalMean, seasonalVar, seasonalACF

Examples

```
# Fourier frequencies for the data nottem (temperatures at Nottingham Castle)
meanCoeff(nottem)
acfCoeff(nottem, 5)

# Given frequencies
freq <- 2 * (0:5) * pi / 12
meanCoeff(nottem, freq = freq)
```

---

naradamar *Nadaraya-Watson Estimator for Transition Densities.*

Description

Calculates the Nadaraya-Watson estimator for estimating the transition densities of a Markov chain.

Usage

```
naradamar(Sx, Sy, x, bandwidth)
```

Arguments

Sx	A vector of the first coordinate of the grid for which the Nadaraya-Watson kernel estimator will be computed.
Sy	A vector of the second coordinate of the grid for which the Nadaraya-Watson kernel estimator will be computed.
x	A numeric vector representing a Markov chain.
bandwidth	A positive numeric value representing the kernel bandwidth smoothing parameter.

Details

The naradamar function computes the estimated transition densities $f(x, y)$ of a process x on a grid given by the Sx and Sy values. It utilizes a Nadaraya kernel-type estimator for the transition density, with a bandwidth provided by the user.

Value

Returns a matrix of size Sx·Sy, containing the estimated transition densities on the grid.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted..

See Also

bandw1.

Examples

```
set.seed(12345)
phi=0.4
n=200
X <- arima.sim(list(ar=phi),n=n)
s=seq(quantile(X,0.01),quantile(X,0.99), length.out=50)
h=bandw1(X)
res=naradamar(s,s,X,h)
persp(s,s,res)
```

---

para.boot *Parametric Bootstrap for i.i.d. Data*

Description

This function performs a parametric bootstrap, a technique that resamples data based on an assumed distribution with estimated parameter, rather than resampling the original data directly.

Usage

```
para.boot(X, func, rdist, param, B = 999, ...)
```

Arguments

X	A numeric vector representing the data.
func	A function taking X as an argument, representing the statistic of interest to be bootstrapped. It should returns a vector of size p >= 1.
rdist	A parametric distribution generator that produces bootstrap data based on the data size and param. It should be a function with two arguments n - the size of the bootstrap sample and par - a vector of the parameters.
param	Numeric vector. Values of parameters used to generate the bootstrap data. These can be either the true parameter values for Monte Carlo approximation of the true distribution, or the estimated parameters, typically obtained by the maximum likelihood method.
B	A positive integer; the number of bootstrap replications. By default it is 999.
...	Optional additional arguments for the func function.

Details

para.boot is a flexible function for bootstrapping a specified function func using a parametric distribution rdist generated with estimated or true parameters param. The function returns a boodd object containing the values of the function over all bootstrap samples and the statistic computed over the original sample.

Value

A boodd object containing:

- s: Matrix of bootstrap replicates of func, with dimensions B x p.
- Tn: Vector of func evaluated on the original data X.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted..

Efron, B., Tibshirani, R. (1993). *An Introduction to the Bootstrap*, Chapman and Hall.

See Also

bootglm, boots.

Examples

```
rn<-function(n,par) {rnorm(n,mean=par[1],sd=par[2])}
set.seed(5)
# Parametric bootstrap of the mean in a gaussian family
X=rnorm(n=100,mean=2,sd=1)
# simulate distribution with true parameter values (and a Monte-Carlo size 9999
true1<-para.boot(X,mean,rn,param=c(2,1),B=9999)
pb1<-para.boot(X,mean,rn,param=c(mean(X),sd(X)))
plot(pb1)
lines(density(true1$s),col="red")
confint(true1,method="bperc")
confint(pb1, method="all")
```

---

per_boo *Bootstrap of Periodogram*

Description

Computes periodogram values at Fourier frequencies for a time series, smooths the periodogram to estimate the spectral density, and generates the bootstrap version of the periodogram.

Usage

```
per_boo(x, B, taper0 = 0)
```

Arguments

x	A vector or a time series.
B	A positive integer; the number of bootstrap replications.
taper0	A numeric value; specifies the proportion of data to taper. The default value is 0, that is there is no tapering.

Details

The function first centers the input time series and calculates the values of the periodogram at Fourier frequencies using spec.pgram. Spectral density is then estimated by applying a kernel smoother to the periodogram values, with the smoothing bandwidth computed as $sd(x) * n^(-1/3)$. Bootstrap is then performed by resampling periodogram ordinates.

The function outputs a graph of the histogram of the periodogram ordinates which should be close to an exponential density.

Value

A list containing:

- obj: A list of class "boodd" containing the bootstrap periodograms.
- freqs: A vector of Fourier frequencies used in the periodogram estimation.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted..

Bertail, P. and Dudek, A.E. (2021). Consistency of the Frequency Domain Bootstrap for differentiable functionals, *Electron. J. Statist.*, **15**, 1-36.

Hurvich, C. M. and Zeger, S. L. (1987). Frequency domain bootstrap methods for time series, Technical Report 87-115, Graduate School of Business Administration, New York Univ.

Lahiri, S.N. (2003). *Resampling Methods for Dependent Data*. Springer, New York.

See Also

tft_boot, func_fdb, freqboot.

Examples

```
set.seed(12345)
x=arima.sim(model=list(ar=0.8),n=200)
boo1=per_boo(x,99)

fn=length(boo1[[2]])
spec.pgram(x, plot=TRUE)
# Superimposed plots of 99 bootstrap periodograms
for ( i in (1:99)) {
lines(boo1[[2]],t(boo1[[1]]$s)[,i], type="l", col=i)
}
```

pkc *Plot Kernel Density Estimates for Null and Alternative Distributions.*

Description

Plots kernel density estimates for null and alternative distributions, showing the acceptance region for a hypothesis test and highlighting the type II error against an alternative hypothesis.

Usage

```
pkc(nul_dist, alt_dist, alpha)
```

Arguments

`nul_dist`	Numeric vector representing the distribution under the null hypothesis.
`alt_dist`	Numeric vector representing the distribution under an alternative hypothesis.
`alpha`	Numeric value between $[0, 0.5]$; the significance level of the test (type I error rate).

Details

This function visualizes the kernel density estimates of two distributions: one under the null hypothesis and the other one under an alternative hypothesis. It highlights the acceptance region of the test (using the significance level `alpha`) and the region corresponding to type II error. This visual representation can be useful for understanding the behaviour of bootstrapped test statistics and the trade-off between type I and type II errors.

Value

Creates a plot showing the kernel density estimates of the null and alternative distributions with the relevant regions highlighted. The function does not return any values.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted..

Beran R. (1986). Simulated Power Functions. *Ann. Statist.*, **14**, 151 - 173.

See Also

`compute_power`.

Examples

```
# Example: Comparing null and alternative distributions
# Generate two normally distributed samples
set.seed(123)
null_dist <- rnorm(1000, mean = 0, sd = 1) # Null distribution
alt_dist <- rnorm(1000, mean = 0.5, sd = 1) # Alternative distribution
alpha <- 0.05 # Significance level

# Plot kernel density estimates
pkc(null_dist, alt_dist, alpha)
```

`plot.boodd` *Plot an Object of Class* boodd

Description

Plots histograms or density estimates for objects of class boodd, which are returned by bootstrap functions such as boots, `bootsemi`, `blockboot`, `regenboot`, etc.

Usage

```
## S3 method for class 'boodd'
plot(x, with.density = TRUE, which, byrow = FALSE, ...)
```

Arguments

`x`	An object of class boodd.
`with.density`	Logical value indicating whether to plot the estimated density of the bootstrap distribution (default is TRUE).
`which`	Specifies which columns of the data to plot.
`byrow`	Logical value indicating whether to display the matrix of histograms by row. By default it is `FALSE`.
`...`	Optional additional arguments for the `plot` function.

Details

The function `plot.boodd` plots histograms or density estimates of the output data contained in an object of class boodd. When the data have more than one column, the function can display a matrix of (at most 6) histograms. The argument `which` allows the selection of specific columns to plot, while `byrow` controls the layout of the matrix display.

Value

The function returns an invisible list containing the output of the `hist` function. In the case of multiple histograms, it returns a list of lists.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Efron, B., Tibshirani, R. (1993). *An Introduction to the Bootstrap*, Chapman and Hall.

See Also

`confint.boodd`, `summary.boodd`, `class.boodd`.

Examples

```
B <- 299
x <- round(rnorm(15),3)
boo1 <- boots(x, mean, B)
plot(boo1)

# Bootstrap of several statistics
mv <- function(data) {c(mean(data), var(data))} # compute both mean and variance
boo2 <- boots(x, mv, B)
plot(boo2)
```

---

qVar *Estimating Variance of a Quantile*

Description

This function calculates the quantile variance using kernel density estimation.

Usage

```
qVar(
  x,
  alpha,
  bandwidth = NULL,
  kernel = c("gaussian", "epanecbandwidthikov", "rectangular")
)
```

Arguments

`x`	A numeric vector.
`alpha`	A numeric value from the interval $[0, 1]$ or a vector of probabilities.
`bandwidth`	A positive numeric value representing the kernel bandwidth smoothing parameter. If `NULL`, bandwidth is estimated using unbiased cross-validation method.
`kernel`	A character string specifying the smoothing kernel to use: `"gaussian"`, `"epanechnikov"`, or `"rectangular"`.

Details

If qa is the quantile of order alpha, the quantile variance v is given by:

$$v = \frac{\alpha(1-\alpha)}{\hat{f}_h(qa)^2}$$

where

- $\hat{f}_h(x) = \frac{1}{n \cdot h} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right)$,
- h is the bandwidth,
- n is the sample size.

Value

A numeric vector containing the quantile variances.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted..

Maritz, J. S. and Jarrett, R. G. (1978). A note on estimating the variance of the sample median. *Journal of the American Statistical Association*, **73**, 194-196.

See Also

boots, confint.

Examples

```
# Example usage of qVar function
data <- rnorm(100)
qVar(data, 0.5)
```

---

rate.block.sub *Block Subsampling for Time Series with Convergence Rate Estimation*

Description

Performs block subsampling for a time series with an estimation of the convergence rate of a given statistic. The function constructs subsampling distributions of specified sizes, estimates the convergence rate and asymptotic bias, and provides optional diagnostic plots.

Usage

```
rate.block.sub(X, func, n_b = 99, q = 0.75, PLT = TRUE, ...)
```

Arguments

X	A numeric vector or a time series.
func	The function to be estimated using block subsampling.
n_b	A positive integer; the number of subsampling distributions to be constructed, with block sizes ranging from $\lfloor n^{1/3} \rfloor$ to $\rfloor n^{3/4} \rfloor$, where n is the sample size.
q	A numeric value in the interval $(0.5, 1)$; it is used to calculate the interquantile range which estimates the rate of convergence.
PLT	A logical value; By default it is TRUE, plots graphics and intermediate regression results.
...	Additional arguments for the func function.

Details

The function rate.block.sub performs block subsampling, specifically suited for time series data, estimates the convergence rate $n^{-\alpha}$ and the approximation rate $n^{-\beta}$ between the true distribution and the asymptotic one.

Value

A list containing:

alpha	The estimated rate of convergence.
beta	The estimated approximation rate.
obj	An object of class boodd with the optimized subsampling distribution.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted..

Bertail, P., Politis, D., Romano, J. (1999). On Subsampling Estimators with Unknown Rate of Convergence. *Journal of the American Statistical Association*, **94**, 569-579.

See Also

rate.sub, best.block.sub.size, best.sub.size.iid, block.sub, boots.

Examples

```
set.seed(12345)
n = 500 # sample size
ts = arima.sim(n=n,model=list(ar=c(0.4)))*sqrt(1-0.4^2) # Generating an AR(1)
# process with variance 1
# Moon MBB without replacement with estimated rate
reb=rate.block.sub(ts,mean,PLT=TRUE)
```

`rate.sub` *Subsampling for i.i.d. Data with Convergence Rate Estimation*

Description

Performs subsampling for an i.i.d. data with an estimation of the convergence rate of a given statistic. The function constructs subsampling distributions of specified sizes, estimates the convergence rate and asymptotic bias, and provides optional diagnostic plots.

Usage

```
rate.sub(X, func, B = 999, n_b = 99, q = 0.75, PLT = TRUE, ...)
```

Arguments

`X`	A numeric vector.
`func`	The function to be estimated using subsampling.
`B`	A positive integer; the number of subsampling replications.
`n_b`	A positive integer; the number of subsampling distributions to be constructed, with subsampling sizes ranging from $\lfloor n^{1/3} \rfloor$ to $\rfloor n^{3/4} \rfloor$, where n is the sample size.
`q`	A numeric value in the interval $(0.5, 1)$; it is used to calculate the interquantile range which estimates the rate of convergence.
`PLT`	A logical value; By default it is `TRUE`, plots graphics and intermediate regression results.
`...`	Additional arguments for the `func` function.

Details

The function `rate.block` performs subsampling, specifically suited for i.i.d. data, estimates the convergence rate $n^{-\alpha}$ and the approximation rate $n^{-\beta}$ between the true distribution and the asymptotic one.

Value

A list containing:

`alpha`	The estimated rate of convergence.
`beta`	The estimated approximation rate.
`obj`	An object of class `boodd` with the optimized subsampling distribution.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted..

Bertail, P., Politis, D., Romano, J. (1999). On Subsampling Estimators with Unknown Rate of Convergence.

Politis, D. N., Romano, J. P., & Wolf, M. (1999). *Subsampling*. Springer N.Y..

See Also

rate.block.sub, best.block.sub.size, best.sub.size.iid, block.sub, boots.

Examples

```
#' set.seed(12345)
n = 500 # sample size
x = rnorm(n)
# Subsampling with estimated rate
reb=rate.sub(x,mean,PLT=TRUE, B=99)
```

---

regenboot *Regenerative and Approximative Regenerative Block Bootstrap.*

Description

Performs regenerative block bootstrap and approximately regenerative block bootstrap on a Markov chain, either in the atomic case or in the general Harris case.

Usage

```
regenboot(
  x,
  func,
  B,
  ...,
  atom,
  small = NULL,
  s = median(x),
  plotIt = FALSE,
  moon = length(x)
)
```

Arguments

x	A numeric vector representing a Markov process.
func	The function to apply to each sample.
B	A positive integer; the number of bootstrap replications.

...	Optional additional arguments for the func function.
atom	A numeric value or a string; an atom of the Markov chain in the atomic case.
small	An object of class smallEnsemble. It can be created optimally using the function findBestEpsilon.
s	A real number specifying the center of the small set. Default is the median of x.
plotIt	Logical. If TRUE then the function returns a plot of the time series with the approximative regenerative blocks. Does not plot anything in the atomic case. Default is FALSE.
moon	A positive integer. Default is length of x. moon should be smaller than the length of x, then it creates bootstrap samples of size moon.

Details

This function regenboot implements two different kinds of regenerative bootstrap:

- *A regenerative block bootstrap* used for atomic Markov chains.
- *An approximate regenerative block bootstrap* used to bootstrap Harris recurrent Markov chains based on a given small set of the form $[s - eps, s + eps]$ where s is the center and *eps* the radius.

One must specify either the atom argument or the small argument. In the first case, atom is the state used to split the Markov chain into blocks ending with the atom. In the second case, small is an object of class smallEnsemble representing the small ensemble. Such objects are typically obtained using the findBestEpsilon function but may also be constructed manually using the smallEnsemble function.

Value

returns an object of class boodd.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Bertail, P., Clémençon, S. (2006a). Regenerative Block Bootstrap for Markov Chains. *Bernoulli*, **12**, 689-712.

Bertail, P. and Clémençon, S. (2006b). Regeneration-based statistics for Harris recurrent Markov chains. *Lecture notes in Statistics*, vol. **187**, pp. 1-54, Springer.

Radulović, D. (2004). Renewal type bootstrap for Markov chains. *Test*, **13**, 147-192.

See Also

boots, blockboot, plot.boodd, confint.boodd, findBestEpsilon,smallEnsemble.

Examples

```
B <- 299
n <- 200

# Atomic Boostrap
acgt <- c("A","C","G","T")
probs <- c(.3,.1,.3,.3)
atom <- "C"
set.seed(1)
x <- sample(acgt,n,prob=probs,repl=TRUE)
propAtom <- function(x) {
  tbl <- as.vector(table(x))
  prop <- tbl[3]/length(x)
   return(prop)
}
boo <- regenboot(x,propAtom,B,atom=atom)
plot(boo)

# Approximate regenerative bootstrap with estimated small set
ar <- arima.sim(list(c(1,0,0),ar=0.6),n=500)
# Find the small ensemble with the largest number of regenerations
sm <- findBestEpsilon(ar,s=0,plotIt=TRUE)
# Approximate regenerative bootstrap of the mean
rboo <- regenboot(ar,mean,small=sm,B=999, plotIt=TRUE)
# Plot the corresponding bootstrap distribution
plot(rboo)
# Compute the bootstrap percentile confidence interval
confint(rboo)
```

---

seasonalMean, seasonalVar, seasonalACF

Computes time domain characteristics of periodically correlated time series

---

Description

Calculate estimates of the seasonal means, variances and autocovariances of a periodically correlated time series.

Usage

```
seasonalMean(x,period,...)
seasonalVar(x,period,...)
seasonalACF(x,tau,period,...)

## Default S3 method:
seasonalMean(x,period,...)
## Default S3 method:
```

```
seasonalVar(x,period,...)
## Default S3 method:
seasonalACF(x,tau,period,...)

## S3 method for class 'ts'
seasonalMean(x,period=frequency(x),...)
## S3 method for class 'ts'
seasonalVar(x,period=frequency(x),...)
## S3 method for class 'ts'
seasonalACF(x,tau,period=frequency(x),...)
```

Arguments

x	A vector or time series representing a periodically correlated time series.
period	A positive integer; the period length. By default it is `frequency(x)`.
tau	A vector of integers; a single lag or vector of lags.
...	Optional additional arguments for the function.

Details

The functions `seasonalMean` and `seasonalVar` calculate estimates of the seasonal means and variances respectively. The function `seasonalACF` calculates an estimator of the autocovariance for the given lags.

Value

The `seasonalMean` and `seasonalVar` functions return a vector of length `period`.

`seasonalACF` returns either a vector of length `period` if a single lag `tau` is specified, or a matrix with `length(tau)` rows and `period` columns if `tau` is a vector.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted..

Hurd, H.L., Miamee, A.G. (2007). *Periodically Correlated Random Sequences: Spectral. Theory and Practice*. Wiley.

See Also

`blockboot.seasonal`, `meanCoeff`, `acfCoeff`, `embb`, `embb.sample`, `bopt_circy`.

Examples

```
# Means
seasonalMean(nottem)  # The period is already in the time-series object nottem
# Variances
seasonalVar(nottem)
# Autocovariances
seasonalACF(nottem,c(0,1))
```

`sieveboot` *Autoregressive Sieve Bootstrap*

Description

Applies autoregressive sieve bootstrap to stationary time series.

Usage

```
sieveboot(x, func, B, order = NULL, ...)
```

Arguments

`x`	A vector or time series.
`func`	The function applied to each bootstrap sample.
`B`	A positive integer; the number of bootstrap replications.
`order`	A positive integer; represents the order of the sieve autoregressive process. If not provided, it is automatically determined (see details below).
`...`	Optional additional arguments for the `func` function.

Details

The sieve bootstrap estimates an AR(p) model, with a large `order` equal to p, resamples the centered estimated residuals, and reconstructs an AR(p) bootstrap time series to compute a given statistic. The default `order`, if not specified, is set to $4 * (n^{1/4})/\log(n)^{1/2}$, where n is the sample size.

Value

Returns an object of class `boodd`.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Bühlmann, P. (1997). Sieve Bootstrap for time series. *Bernoulli,* **3**, 123-148.

Choi, E., Hall, P. (2000). Bootstrap confidence regions computed from autoregressions of arbitrary order. *Journal of the Royal Statistical Society*, Series B, **62**, 461-477.

See Also

`aidedboot`.

Examples

```
n <- 200
B <- 599
x <- arima.sim(list(order=c(0,0,4),ma=c(0.7,0.4,-0.3,-0.1)),n=n)
b1 <- sieveboot(x,mean,B,order=10)
plot(b1)
```

`smallEnsemble` *Class* `smallEnsemble`

Description

Creates an object of class `smallEnsemble`, typically used in regenerative bootstrap methods for Markov chains. This function allows manual specification of parameters for the small ensemble.

Usage

```
smallEnsemble(s, eps, delta, trans)
```

Arguments

`s`	A numeric value specifying the center of the small set.
`eps`	A positive numeric value; the radius of the small set.
`delta`	A positive numeric value; a lower bound for the transition density over the small set, controlling the Nummelin splitting trick.
`trans`	A vector representing the estimated transition density between X_i and X_{i+1}. This can be calculated using `fastNadaraya`.

Details

The `smallEnsemble` object is used in regenerative bootstrap methods, particularly in the context of Markov chains. It defines a small set, characterized by a central point `s`, a radius `eps`, and a lower bound `delta`.

This function is useful for users who wish to manually specify the parameters of the small set, as opposed to using automated methods like `findBestEpsilon`, which returns a `smallEnsemble` object.

Value

Returns an object of class `smallEnsemble`, containing the specified parameters for the small set and the transition density.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Bertail, P., Clémençon, S. (2006a). Regenerative Block Bootstrap for Markov Chains. *Bernoulli*, **12**, 689-712.

See Also

regenboot, findBestEpsilon, fastNadaraya.

Examples

```
# Example of creating a smallEnsemble object
# Assuming a Markov chain data
data <- rnorm(100)
s <- median(data)   # Middle point of the small set
eps <- 0.5          # Width of the small set
delta <- 0.1        # Parameter of the small set
# Define a simple transition kernel function
trans <- fastNadaraya(data, 1/10)
# Create the smallEnsemble object
small_ensemble <- smallEnsemble(s, eps, delta, trans)
```

---

summary.boodd *Summary for Objects of Class boodd*

Description

The bootstrap functions return an object of class boodd containing the generated data. The generic function summary may be applied to these objects.

Usage

```
## S3 method for class 'boodd'
summary(object, ...)
```

Arguments

object	An object of class boodd.
...	Additional arguments affecting the summary produced.

Details

The function summary.boodd displays basic information about the values of the samples computed by the bootsrap functions.

Value

The function summary.boodd returns the kind attribute of the function that produced the boodd object and a (possibly multicolumn) table with the quartiles, mean, min and max of the computed values of the statistic. The number of columns is the size of the return value of the function func.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Efron, B., Tibshirani, R. (1993). *An Introduction to the Bootstrap*, Chapman and Hall.

See Also

confint.boodd, boots, class.boodd.

Examples

```
B <- 299
x <- round(rnorm(15),3)
boo1 <- blockboot(x,mean,B, method="circular")
summary(boo1)


# bootstrap of several statistics
mv <- function(data) {c(mean(data),var(data))} # compute both mean and variance
boo2 <- boots(x,mv,B)
summary(boo2)
```

---

tboot_dist *Computation of the Bootstrap-t Distribution*

Description

Computes and plots the bootstrap-t distribution of a statistic when an estimator of the variance is available.

Usage

```
tboot_dist(
  boot_obj,
  comp = 1,
  PLT = TRUE,
  nn = TRUE,
  recenter = FALSE,
  return_values = FALSE,
  ...
)
```

Arguments

boot_obj	A boodd object containing the bootstrap values of eventually a multivariate statistic of size p (first p columns) and its variance (last p columns).
comp	The index of the component of the statistic to be plotted.
PLT	Logical. If TRUE (default), then the function plots the bootstrap-t distribution, otherwise it only returns the computed distribution.
nn	Logical. If TRUE (default), then a normal approximation is superimposed on the histogram.
recenter	Logical. If TRUE, then the data is recentered by the bootstrap mean of the statistic. By default it is FALSE.
return_values	Logical. If TRUE, the function returns the values obtained using tboot_dist. By default it is FALSE.
...	Additional arguments affecting the plot produced (e.g., nclass, main, ylim,...).

Details

The tboot_dist function is designed to work with bootstrap samples that include estimates of both the parameters of interest and their variances. The function calculates a studentized version of the bootstrap distribution (the bootstrap-t distribution) and optionally plots this distribution and the normal approximation.

Value

Optionally returns a numeric vector representing the bootstrap-t distribution of the selected component.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Efron, B., Tibshirani, R. (1993). *An Introduction to the Bootstrap*, Chapman and Hall.

See Also

boots, class.boodd, jackVar, confint.boodd.

Examples

```
set.seed(5)
m=0
n=100
X=rnorm(n,mean=m)
mv=function(X){
  c(mean(X), var(X)/length(X))}
# the function mv computes the estimates of the mean and variance of the mean
resb=boots(X,mv,B=999)
# compare distributions
tboot_dist(resb,1)
```

tft_boot *TFT Bootstrap.*

Description

This function implements the Time Frequency Toggle (TFT)-Bootstrap introduced in Kirch and Politis (2011).

Usage

```
tft_boot(
  X,
  n = length(X),
  func,
  h = length(X)^{
     -2/3
 },
  kernel = 1,
  B,
  option = "local"
)
```

Arguments

X	A numeric vector representing a time series.
n	An integer; by default is the length of time series X but allow for a smaller sample size m to perform moon bootstrap.
func	A function to be applied to each column of the TFT bootstrap samples.
h	A numeric value specifying the bandwidth used to compute the kernel estimator in case of local bootstrap. By default it is equal to $n^{-2/3}$.
kernel	An integer value indicating the kernel type. Use 0 for the Daniell kernel or any other value for the Bartlett-Priestley (Epanechnikov) kernel (by default).
B	An integer indicating the number of bootstrap replications.
option	A character string specifying the bootstrap method: "local" for local bootstrap, "wild" for wild bootstrap, and "res" for residual bootstrap.

Details

The function performs the Time Frequency Toggle (TFT)-Bootstrap. Its basic idea is to bootstrap the Fourier coefficients of the observed time series, and then back-transforming them to obtain a bootstrap sample in the time domain (see Kirch and Politis (2011)). Then it computes the specified function, func, on each bootstrap replicate. Depending on the option, different bootstrap methods are used to construct bootstrap samples:

- "local" - local bootstrap.
- "wild"- wild bootstrap.
- "res" - residual bootstrap.

Value

An object of class "boodd".

Author(s)

We are grateful to Claudia Kirch for providing the original code in R.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Kirch, C. and Politis, D. N. (2011). TFT-Bootstrap: Resampling time series in the frequency domain to obtain replicates in the time domain. *Annals of Statistics*, vol. **39**, pp. 1427-1470

See Also

boot_local, boot_res, boot_wild.

Examples

```
n=2000
phi0=0.5
p0=length(phi0)
theta0=0
q0=length(theta0)
sim <- arima.sim(n , model=list(ar = phi0, ma = theta0))
# The function to be bootstrapped using tft_boot (both variance and acf of order 1)
vc<-function(X){return(c(var(X),acf(X,plot=FALSE)$acf[2]))}
coeff=vc(sim)
bb1=tft_boot(sim,n=n,func=vc,B=999,option="local")
bb2=tft_boot(sim,n=n,func=vc,B=999,option="res")
bb3=tft_boot(sim,n=n,func=vc,B=999,option="wild")
```

---

thetaARBB *Compute the Extremal Index for Non-Atomic Markov Chains Using Pseudo-Regenerative Blocks*

Description

This function divides the input dataset into pseudo-blocks for a non-atomic Markov chain using a Nummelin splitting trick with estimated parameters. We use the optimal small set computed by findBestEpsilon function and calculates the extremal index using quantile-based thresholds.

Usage

```
thetaARBB(X)
```

Arguments

X	A numeric vector representing the Markov chain.

Details

The function uses GetPseudoBlocks to divide X into blocks using to the estimated Nummelin splitting trick. High quantiles from X are generated and used as thresholds to compute statistics on each block, including sub-maximums and block indices. These statistics are then used to calculate the extremal index, a measure of extreme value clustering, across the blocks.

Value

A numeric vector representing the extremal index at various quantile thresholds.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Bertail, P. and Clémençon, S. (2006). Regeneration-based statistics for Harris recurrent Markov chains. *Lecture notes in Statistics*, vol. **187**, pp. 1-54, Springer.

See Also

GetPseudoBlocks, fastNadaraya, regenboot, smallEnsemble,findBestEpsilon.

Examples

```
coeff=0.75
X = arima.sim(n=200, list(ar = c(coeff)))
thetaB <- thetaARBB(X)
plot(thetaB)
```

---

thetaRB — *Compute the Extremal Index for Atomic Markov Chains Using Regenerative Blocks*

Description

This function divides the input dataset into blocks for an atomic Markov chain and calculates the extremal index using quantile-based thresholds.

Usage

```
thetaRB(X, ato)
```

Arguments

X	A numeric vector representing the Markov chain.
ato	A numeric value; the atom of the Markov chain.

Details

The function uses GetBlocks to divide X into blocks according to the atom ato. High quantiles from X are generated and used as thresholds to compute statistics on each block, including sub-maximums and block indices. These statistics are then used to calculate the extremal index, a measure of extreme value clustering, across the blocks.

Value

A numeric vector representing the extremal index at various quantile thresholds.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Bertail, P. and Clémençon, S. (2006). Regeneration-based statistics for Harris recurrent Markov chains. *Lecture notes in Statistics*, vol. **187**, pp. 1-54, Springer.

Bertail, P., Clémençon, S. and Tressou, J. (2009). Extreme values statistics for Markov chains via the (pseudo-) regenerative method. *Extremes*, **12**, 327–360.

See Also

GetBlocks, regenboot.

Examples

```
X = genMM1(1000, 1, 2)
thetaB <- thetaRB(X, ato=0)
plot(thetaB)
```

---

zi_inar_process *Generate a ZI-INAR Process*

Description

Simulates a zero-inflated Poisson distributed (ZI)-INAR(p) process.

Usage

```
zi_inar_process(n, p, alpha, Y = 1, p2, lamb)
```

Arguments

n	A positive integer; the length of the simulated process.
p	A positive integer; the order of the simulated process.
alpha	A numeric vector whose components belong to the interval $(0, 1)$; parameters of the thinning operator (see details below).
Y	A non-negative integer; the initial value of the process.
p2	A numeric value between 0 and 1; parameter of the Bernoulli distribution (see details below).
lamb	A positive numeric value; parameter of the Poisson distribution (see details below).

Details

The ZI-INAR process is defined by the equation

$$Y_t = \sum_{i=1}^{p} \alpha_i \circ Y_{t-i} + V_t, \qquad t \in \mathbb{Z},$$

where the so called thinning operator is defined as

$$\alpha \circ Y = \sum_{i=1}^{Y} Z_i,$$

where $\{Z_i\}_{i\in\mathcal{Z}}$ is an i.i.d. sequence of random variables with Bernoulli distribution with parameter α with $\alpha_i \in [0, 1]$ for $i \in 1, \ldots, p$ and independent of Y. Additionally, $\{V_t\}_{t\in\mathcal{Z}}$ is an i.i.d. non-negative and integer valued sequence of random variables.

Let V be a non-negative discrete random variable with parameters $p2$ and λ. We said that V follows a zero-inflated distribution and we denote it by $V \sim ZI(p2, \lambda)$, if it can be expressed as

$$V = BU, \qquad \text{with} \qquad B \perp U,$$

where B has Bernoulli distribution with mean $1-p2$, for $p2 \in [0, 1)$ and U is a non-negative discrete valued random variable with finite variance and parameter vector λ. We consider the zero-inflated Poisson distribution (ZI-INAR(p))), when U has Poisson distribution with mean λ.

Value

A numeric vector of length n.

References

Bertail, P. and Dudek, A. (2025). *Bootstrap for Dependent Data, with an R package* (by Bernard Desgraupes and Karolina Marek) - submitted.

Bertail, P., Medina-Garay, A., De Lima-Medina, F. and Jales, I. (2024). A maximum likelihood and regenerative bootstrap approach for estimation and forecasting of inar (p) processes with zero-inated innovations. *Statistics*, **58**, 336-363.

See Also

regenboot, genETARCH, genMM1.

Examples

```
X = zi_inar_process(70, 2, c(0.1,0.1), Y = 1, p2 = 0.2, lamb = 0.7)
plot(X, type="l")
```

Index

∗ **Almost periodically correlated time series**
embb, 34
embb.sample, 36
∗ **Approximate**
thetaARBB, 90
∗ **Approximative**
regenboot, 80
∗ **Atomic**
jackVarRegen.atom, 64
∗ **Atom**
GetBlocks, 51
∗ **Autoregressive**
sieveboot, 84
∗ **Block Bootstrap**
embb.sample, 36
∗ **Block bootstrap**
embb, 34
∗ **Blocks**
jackFuncBlock, 56
jackVarBlock, 60
∗ **Block**
f_PseudoBlocks, 48
fastNadaraya, 37
findBestEpsilon, 42
ftrunc, 45
GetBlocks, 51
GetPseudoBlocks, 53
regenboot, 80
∗ **Bootstrap**
aidedboot, 4
boot_dist, 24
boot_local, 25
boot_res, 27
boot_wild, 28
bootsemi, 22
compute_power, 32
field.sub, 38
pkc, 74
plot.boodd, 75
summary.boodd, 86
tft_boot, 89
∗ **Carlo**
para.boot, 71
∗ **Conditional**
genETARCH, 49
∗ **Confidence**
confint.boodd, 33
∗ **Convergence**
rate.block.sub, 77
rate.sub, 79
∗ **Dependent**
class.boodd, 31
∗ **Domain**
freqboot, 43
func_fdb, 46
∗ **Extremal**
thetaARBB, 90
thetaRB, 91
∗ **Fast**
boot_local, 25
boot_res, 27
boot_wild, 28
tft_boot, 89
∗ **Fourier coefficients estimators**
meanCoeff, acfCoeff, 68
∗ **Fourier**
boot_local, 25
boot_res, 27
boot_wild, 28
tft_boot, 89
∗ **Frequency domain**
meanCoeff, acfCoeff, 68
∗ **Frequency**
aidedboot, 4
boot_local, 25
boot_res, 27
boot_wild, 28
freqboot, 43

func_fdb, 46
per_boo, 72
tft_boot, 89
* **GLM.**
bootglm, 18
* **Harris**
jackVarRegen.smallEnsemble, 65
* **Integer**
zi_inar_process, 92
* **Jackknife**
jackFunc, 55
jackFuncBlock, 56
jackFuncRegen, 57
jackVar, 58
jackVarBlock, 60
jackVarField, 61
jackVarRegen, 62
jackVarRegen.atom, 64
jackVarRegen.smallEnsemble, 65
* **Kernel**
fastNadaraya, 37
* **Markov**
bandw1, 7
fastNadaraya, 37
findBestEpsilon, 42
ftrunc, 45
GetBlocks, 51
GetPseudoBlocks, 53
jackVarRegen.atom, 64
jackVarRegen.smallEnsemble, 65
mark_boot, 67
naradamar, 70
regenboot, 80
smallEnsemble, 85
thetaARBB, 90
thetaRB, 91
* **Monte**
para.boot, 71
* **Moving**
block.sub, 11
blockboot, 12
* **Multidimensional**
jackVarField, 61
* **Normal**
boot_dist, 24
* **Nummelin**
GetPseudoBlocks, 53
* **Optimal**
bopt_circy, 29
f_PseudoBlocks, 48
findBestEpsilon, 42
lam, 66
* **Parametric**
bootglm, 18
para.boot, 71
* **Periodic time series**
embb, 34
embb.sample, 36
meanCoeff, acfCoeff, 68
* **Periodically**
blockboot.seasonal, 14
bopt_circy, 29
* **Periodogram**
aidedboot, 4
per_boo, 72
* **Plotting**
plot.boodd, 75
* **Power.**
compute_power, 32
* **Power**
pkc, 74
* **Pseudo-regenerative**
f_PseudoBlocks, 48
* **Quantile**
qVar, 76
* **Queueing**
genMM1, 50
* **Random**
field.sub, 38
fieldboot, 39
fieldbootP, 41
* **Recentering**
boot_dist, 24
* **Regenerative**
bandw1, 7
f_PseudoBlocks, 48
fastNadaraya, 37
findBestEpsilon, 42
ftrunc, 45
GetBlocks, 51
GetPseudoBlocks, 53
jackVarRegen, 62
jackVarRegen.atom, 64
jackVarRegen.smallEnsemble, 65
regenboot, 80
thetaARBB, 90

thetaRB, 91
* **Resampling**
jackFunc, 55
jackFuncBlock, 56
jackVar, 58
jackVarBlock, 60
jackVarField, 61
* **Rescaling**
boot_dist, 24
* **Robust**
ftrunc, 45
* **Seasonal autocovariances**
seasonalMean, seasonalVar, seasonalACF, 82
* **Seasonal means**
seasonalMean, seasonalVar, seasonalACF, 82
* **Seasonal variances**
seasonalMean, seasonalVar, seasonalACF, 82
* **Semiparametric**
bootsemi, 22
class.boodd, 31
* **Sieve**
sieveboot, 84
* **Small**
bandw1, 7
GetPseudoBlocks, 53
regenboot, 80
thetaARBB, 90
* **Spectral**
freqboot, 43
func_fdb, 46
* **Stationary**
freqboot, 43
lam, 66
* **Statistical**
jackFunc, 55
jackFuncBlock, 56
jackVar, 58
jackVarBlock, 60
jackVarField, 61
* **Studentization**
tboot_dist, 87
* **Subsampling**
field.sub, 38
rate.block.sub, 77
rate.sub, 79
* **Test**
compute_power, 32
* **Thresholds**
genETARCH, 49
* **Time**
class.boodd, 31
func_fdb, 46
sieveboot, 84
* **Transition**
fastNadaraya, 37
mark_boot, 67
naradamar, 70
* **Variance**
jackFunc, 55
jackVar, 58
jackVarRegen, 62
jackVarRegen.atom, 64
jackVarRegen.smallEnsemble, 65
qVar, 76
* **ZI-INAR**
zi_inar_process, 92
* **and**
boot_local, 25
boot_res, 27
boot_wild, 28
tft_boot, 89
* **autoregressive**
zi_inar_process, 92
* **blocks**
b.star, 6
* **block**
bopt_circy, 29
lam, 66
* **bootstrap**
best.block.sub.size, 8
best.sub.size.iid, 10
* **density**
fastNadaraya, 37
mark_boot, 67
naradamar, 70
* **package**
boodd, 16
* **process**
zi_inar_process, 92
* **rate**
rate.block.sub, 77
rate.sub, 79
* **small**

f_PseudoBlocks, 48
findBestEpsilon, 42
∗ **splitting**
GetPseudoBlocks, 53
∗ **the**
bootsemi, 22
∗ **time**
boot_local, 25
boot_res, 27
boot_wild, 28
freqboot, 43
lam, 66
tft_boot, 89
∗ **under**
bootsemi, 22

acfCoeff, *36*, *83*
acfCoeff *(*meanCoeff, acfCoeff*)*, 68
acfCoeff.embb *(*embb*)*, 34
aidedboot, 4, *16*, *31*, *44*, *48*, *84*

b.star, 6, *16*, *67*
bandw1, *5*, 7, *38*, *71*
best.block.sub.size, 8, *11*, *12*, *78*, *80*
best.sub.size.iid, 10, *78*, *80*
block.sub, *9*, *11*, 11, *78*, *80*
blockboot, *7*, 12, *15*, *16*, *23*, *31*, *33*, *39*, *40*, *42*, *57*, *61*, *62*, *67*, *68*, *75*, *81*
blockboot.seasonal, 14, *30*, *83*
boodd, *16*, 16
boodd-package *(*boodd*)*, 16
boot_dist, *16*, 24, *31*
boot_local, 25, *28*, *29*, *90*
boot_res, *26*, 27, *29*, *90*
boot_wild, *26*, *28*, 28, *90*
bootglm, *16*, 18, *31*, *32*, *72*
boots, *14*, *16*, 19, *31*, *33*, *55*, *59*, *72*, *77*, *78*, *80*, *81*, *87*, *88*
bootsemi, *14*, *16*, *18*, 22, *31*–*33*, *75*
bopt_circy, *15*, *17*, 29, *83*

class.boodd, *6*, *16*, *18*, *23*, 31, *76*, *87*, *88*
compute_power, *17*, 32, *74*
confint, *77*
confint.boodd, *14*, *16*, *21*, *23*, *31*, 33, *76*, *81*, *87*, *88*

embb, *16*, 34, *36*, *83*
embb.sample, *16*, *30*, *35*, 36, *83*

f_PseudoBlocks, 48
fastNadaraya, *8*, 37, *43*, *85*, *86*, *91*
field.sub, *17*, 38, *40*, *42*
fieldboot, *14*, *16*, *31*, *39*, 39, *42*
fieldbootP, *39*, *40*, 41
findBestEpsilon, *8*, *16*, *37*, *38*, 42, *46*, *49*, *52*, *54*, *57*, *62*, *65*, *68*, *81*, *86*, *90*, *91*
freqboot, *5*, *6*, *17*, *31*, 43, *48*, *73*
ftrunc, 45, *52*, *54*
func_fdb, *44*, 46, *73*

genETARCH, *17*, 49, *51*, *94*
genMM1, *17*, *50*, 50, *94*
GetBlocks, *43*, *45*, *46*, *49*, 51, *92*
GetPseudoBlocks, *43*, *46*, *48*, *49*, 53, *91*

jackFunc, *17*, 55, *57*–*59*, *61*–*64*, *66*
jackFuncBlock, *17*, *55*, 56, *58*, *59*, *61*, *63*, *64*, *66*
jackFuncRegen, *17*, *55*, *57*, 57, *59*, *61*–*64*, *66*
jackVar, *17*, *55*, *57*, *58*, 58, *61*–*64*, *66*, *88*
jackVarBlock, *14*, *17*, *55*, *57*–*59*, 60, *62*
jackVarField, *17*, *39*, *40*, *42*, *61*, 61
jackVarRegen, *17*, 62, *64*, *66*
jackVarRegen.atom, *17*, *63*, 64
jackVarRegen.smallEnsemble, *17*, *63*, 65

lam, *7*, *17*, 66

mark_boot, *31*, 67
meanCoeff, *36*, *83*
meanCoeff *(*meanCoeff, acfCoeff*)*, 68
meanCoeff, acfCoeff, 68
meanCoeff.embb *(*embb*)*, 34

naradamar, *43*, 70

para.boot, *31*, 71
per_boo, *44*, *48*, 72
pkc, *17*, *32*, 74
plot.boodd, *14*, *17*, *21*, *23*, *25*, *31*, *34*, 75, *81*

qVar, *17*, 76

rate.block.sub, *9*, *11*, *12*, *31*, 77, *80*
rate.sub, *9*, *11*, *31*, *78*, 79
regenboot, *16*, *31*, *33*, *37*, *38*, *43*, *46*, *49*, *52*, *54*, *58*, *63*, *64*, *66*, *68*, *75*, 80, *86*, *91*, *92*, *94*

seasonalACF, *36*, *70*

seasonalACF *(*seasonalMean, seasonalVar, seasonalACF*)*, 82
seasonalACF.default, *35*
seasonalACF.embb *(*embb*)*, 34
seasonalMean, *17*, *36*, *70*
seasonalMean *(*seasonalMean, seasonalVar, seasonalACF*)*, 82
seasonalMean, seasonalVar, seasonalACF, 82
seasonalMean.default, *35*
seasonalMean.embb *(*embb*)*, 34
seasonalVar, *36*, *70*
seasonalVar *(*seasonalMean, seasonalVar, seasonalACF*)*, 82
seasonalVar.default, *35*
seasonalVar.embb *(*embb*)*, 34
sieveboot, *17*, *31*, 84
smallEnsemble, *43*, *46*, *49*, *52*, *54*, *57*, *62*, *65*, *81*, 85, *91*
summary.boodd, *17*, *31*, *34*, *76*, 86

tboot_dist, *17*, *31*, 87
tft_boot, *26*, *28, 29*, *31*, *44*, *48*, *73*, 89
thetaARBB, 90
thetaARRB *(*thetaARBB*)*, 90
thetaRB, 91

zi_inar_process, *50, 51*, 92