

# Package ‘meboot’

November 20, 2009

**Version** 1.0-2

**Date** 2009-11-18

**Title** Maximum Entropy Bootstrap for Time Series

**Author** Hrishikesh D. Vinod <Vinod@fordham.edu> and Javier LÃ³pez-de-Lacalle

**Maintainer** Javier LÃ³pez-de-Lacalle <javlacalle@yahoo.es>

**Encoding** latin1

**Depends** R (>= 2.0.0), ConvergenceConcepts, dynlm, kinship, survival, splines, nlme, lattice, plm, zoo

**Suggests** boot, car, geepack, hdrcte, lmtest, strucchange, tcltk

**Description** This package performs maximum entropy density based dependent data bootstrap. An algorithm is provided to create a population of time series (ensemble) without assuming stationarity. The reference paper (Vinod, H.D., 2004) explains how the algorithm satisfies the ergodic theorem and the central limit theorem.

**License** GPL (>= 2)

**Repository** CRAN

**Date/Publication** 2009-11-20 07:53:25

## R topics documented:

checkConv	2
expand.sd	4
force.clt	5
meboot	6
meboot.default	8
meboot.pdata.frame	9
null.ci	11
olsHALL.b	12
slider.mts	13
ullwan	13

USconsum . . . . .	14
USfygt . . . . .	15
zero.ci . . . . .	16

<b>Index</b>	<b>17</b>
--------------	-----------

---

checkConv	<i>Check Convergence</i>
-----------	--------------------------

---

## Description

This function generates a 3D array giving  $(X_n - X)$  in the notation of the `ConvergenceConcepts` package by Lafaye de Micheaux and Liqueur for sample paths with dimensions = `n999` as first dimension, `nover` = range of `n` values as second dimension and number of items in `key` as the third dimension. It is intended to be used for checking convergence of `meboot` in the context of a specific real world time series regression problem.

## Usage

```
checkConv (y, bigx, trueb = 1, n999 = 999, nover = 5,
           seed1 = 294, key = 0, trace = FALSE)
```

## Arguments

<code>y</code>	vector of data containing the dependent variable.
<code>bigx</code>	vector of data for all regressor variables in a regression or <code>ts</code> object. <code>bigx</code> should not include column of ones for the intercept.
<code>trueb</code>	true values of regressor coefficients for simulation. If <code>trueb=0</code> then use OLS coefficient values rounded to 2 digits as true values of beta for simulation purposes.
<code>n999</code>	number of replicates to generate in a simulation.
<code>nover</code>	number of values of <code>n</code> over which convergence calculated.
<code>seed1</code>	seed for the random number generator.
<code>key</code>	the subset of key regression coefficient whose convergence is studied if <code>key=0</code> all coefficients are studied for convergence.
<code>trace</code>	logical. If <code>TRUE</code> , tracing information on the process is printed.

## Details

Use this only when lagged dependent variable is absent.

Warning: `key=0` might use up too much memory for large regression problems.

The algorithm first creates data on the dependent variable for a simulation using known true values denoted by `trueb`. It proceeds to create `n999` regression problems using the seven-step algorithm in `meboot` creating `n999` time series for all variable in the simulated regression. It then creates sample paths over a range of `n` values for coefficients of interest denoted as `key` (usually a subset

of original coefficients). For each key coefficient there are  $n999$  paths as  $n$  increases. If `meboot` algorithm is converging to true values, the value of  $(X_n - X)$  based criteria for "convergence in probability" and "almost sure convergence" in the notation of the `ConvergenceConcepts` package should decline. The decline can be plotted and/or tested to check if it is statistically significant as sample size increases. This function permits the user of `meboot` working with a short time series to see if the `meboot` algorithm is working in his or her particular situation.

### Value

A 3 dimensional array giving  $(X_n - X)$  for sample paths with dimensions =  $n999$  as first dimension, `nover` = range of  $n$  values as second dimension and number of items in `key` as the third dimension ready for use in `ConvergenceConcepts` package.

### References

- Lafaye de Micheaux, P. and Liquet, B. (2009), Understanding Convergence Concepts: a Visual-Minded and Graphical Simulation-Based Approach, *The American Statistician*, **63**(2) pp. 173-178.
- Vinod, H.D. (2006), Maximum Entropy Ensembles for Time Series Inference in Economics, *Journal of Asian Economics*, **17**(6), pp. 955-978
- Vinod, H.D. (2004), Ranking mutual funds using unconventional utility theory and stochastic dominance, *Journal of Empirical Finance*, **11**(3), pp. 353-377.

### See Also

`meboot`, `criterion`.

### Examples

```
library(strucchange)
data("PhillipsCurve")

uk <- window(PhillipsCurve, start = 1948)
attach(data.frame(uk))
y <- uk[,5]
bigx <- cbind(dp1,du,u1) # collect all regressors
reg1 <- lm(y ~ bigx)
coef(reg1)
nover <- 16 #choose range allowing n=25, 26, .. , 40

# the following step will take some time to run

sC <- checkConv(y, bigx, trueb=0, n999=999, nover=nover, seed1=294)

dim(sC) # n999 x nover x length(key)
j <- 3 # choose key coefficient no. 3 for lagged income
epsilon <- 0.1 #needed for p.as.plot command below
nb.sp <- 10 # needed for p.as.plot command below
mode <- "p" # needed for p.as.plot command below

# criterion function in ConvergenceConcepts package wants 999 rows
dat <- sC[, , j] # 999 sample paths for diff inflation n=25,26,..40
```

```

critp <- criterion(data = dat, epsilon = epsilon, mode = "p")$crit
critas <- criterion(data = dat, epsilon = epsilon, mode = "as")$crit

p.as.plot(dat, critp, critas, epsilon, nb.sp, mode = mode)
nstart <- length(y) - nover + 1
nn <- seq(nstart, length(y)) # choose the range of n the sample size

opar <- par(mfrow = c(2,1)) #plot 2 plots in one
plot(nn,critp, typ="l",
     main="Convergence in probability: Diff(Inflation) Coefficient",
     xlab="Sample size", ylab="Criterion using 999 sample paths")
plot(nn,critas, typ="l",
     main="Almost sure convergence: Diff(Inflation) Coefficient",
     xlab="Sample size", ylab="Criterion using 999 sample paths")

par(opar) # reset

regp <- lm(critp ~ nn) # OLS of conv. in prob. criterion
sup <- summary(regp) # regressed on sample size
sup$coef
# slope coeff. should be negative in sign for convergence
# the t statistic on the slope coefficient should be large
regas <- lm(critas ~ nn) #OLS of almost sure conv criterion
suas <- summary(regas)#regressed on sample size
suas$coef
# slope coeff. should be negative in sign for convergence
# the t statistic on the slope coefficient should be large

```

---

expand.sd

*Expand the Standard Deviation of Resamples*

---

## Description

This function expands the standard deviation of the simulated data. Expansion is needed since some of the ratios of the actual standard deviation to that of the original data are lower than 1 due to attenuation.

## Usage

```
expand.sd (x, ensemble, fiv=5)
```

## Arguments

x	a vector of data or a time series object.
ensemble	a matrix or 'mts' object containing resamples of the original data in 'x'.
fiv	reference value for the upper limit of a uniform distribution used in expansion. For example, if equal to 5 the standard deviation of each resample is expanded through a value from a uniform random distribution with lower limit equal to 1 and upper limit equal to $1+(5/100)=1.05$ .

**Value**

Resamples (by columns) with expanded standard deviations.

**Examples**

```
set.seed(345)
out <- meboot(x=AirPassengers, reps=100, trim=0.10, reachbnd=FALSE, elaps=TRUE)
exp.ens <- expand.sd(x=AirPassengers, out$ensemble)
```

---

force.clt

*Enforce Central Limit Theorem*

---

**Description**

Function to enforce the maximum entropy bootstrap resamples to satisfy the central limit theorem.

**Usage**

```
force.clt (x, ensemble)
```

**Arguments**

**x** a vector of data or a time series object.  
**ensemble** a matrix or 'mts' object containing resamples of the original data in 'x'.

**Value**

Revised matrix satisfying the central limit theorem.

**Examples**

```
set.seed(345)
out <- meboot(x=AirPassengers, reps=100, trim=0.10, reachbnd=FALSE, elaps=TRUE)
cm1 <- colMeans(out$ensemb)
# Note that the column means are somewhat non-normal
qqnorm(cm1)

clt.ens <- force.clt(x=AirPassengers, ensemble=out$ensemble)
cm2 <- colMeans(clt.ens)
# Note that the columns are closer to being normal
qqnorm(cm2)
```

meboot

*Generate Maximum Entropy Bootstrapped Time Series Ensemble***Description**

Generates maximum entropy bootstrap replicates for dependent data. (See details.)

**Usage**

```
meboot (x, reps=999, trim=0.10, reachbnd=TRUE,
        expand.sd=TRUE, force.clt=TRUE, elaps=FALSE, colsubj, coldata, coltimes, ...)
```

**Arguments**

<code>x</code>	vector of data, <code>ts</code> object or <code>pdata.frame</code> object.
<code>reps</code>	number of replicates to generate.
<code>trim</code>	the trimming proportion.
<code>reachbnd</code>	logical. If TRUE potentially reached bounds ( <code>xmin</code> = smallest value - trimmed mean and <code>xmax</code> =largest value + trimmed mean) are given when the random draw happens to be equal to 0 and 1, respectively.
<code>expand.sd</code>	logical. If TRUE the standard deviation in the ensemble is expanded. See <a href="#">expand.sd</a> .
<code>force.clt</code>	logical. If TRUE the ensemble is forced to satisfy the central limit theorem. See <a href="#">force.clt</a> .
<code>elaps</code>	logical. If TRUE elapsed time during computations is displayed.
<code>colsubj</code>	the column in <code>x</code> that contains the individual index. It is ignored if the input data <code>x</code> is not a <code>pdata.frame</code> object.
<code>coldata</code>	the column in <code>x</code> that contains the data of the variable to create the ensemble. It is ignored if the input data <code>x</code> is not a <code>pdata.frame</code> object.
<code>coltimes</code>	an optional argument indicating the column that contains the times at which the observations for each individual are observed. It is ignored if the input data <code>x</code> is not a <code>pdata.frame</code> object.
<code>...</code>	possible argument <code>fiv</code> to be passed to <a href="#">expand.sd</a> .

**Details**

Seven-steps algorithm:

1. Sort the original data in increasing order and store the ordering index vector.
2. Compute intermediate points on the sorted series.
3. Compute lower limit for left tail (`xmin`) and upper limit for right tail (`xmax`). This is done by computing the `trim` (e.g. 10

4. Compute the mean of the maximum entropy density within each interval in such a way that the *mean preserving constraint* is satisfied. (Denoted as  $m_t$  in the reference paper.) The first and last interval means have distinct formulas. See Theil and Laitinen (1980) for details.
5. Generate random numbers from the [0,1] uniform interval and compute sample quantiles at those points.
6. Apply to the sample quantiles the correct order to keep the dependence relationships of the observed data.
7. Repeat the previous steps several times (e.g. 999).

### Value

x	original data provided as input.
ensemble	maximum entropy bootstrap replicates.
xx	sorted order stats (x[1] is minimum value).
z	class intervals limits.
dv	deviations of consecutive data values.
dvtrim	trimmed mean of dv.
xmin	data minimum for ensemble=x[1]-dvtrim.
xmax	data x maximum for ensemble=x[n]+dvtrim.
desintxb	desired interval means.
ordxx	ordered x values.
elaps	elapsed time.

### References

Vinod, H.D. (2006), Maximum Entropy Ensembles for Time Series Inference in Economics, *Journal of Asian Economics*, **17**(6), pp. 955-978

Vinod, H.D. (2004), Ranking mutual funds using unconventional utility theory and stochastic dominance, *Journal of Empirical Finance*, **11**(3), pp. 353-377.

### See Also

[slider.mts](#).

### Examples

```
## Ensemble for the AirPassenger time series data
set.seed(345)
out <- meboot(x=AirPassengers, reps=100, trim=0.10, elaps=TRUE)

## Ensemble for T=5 toy time series used in Vinod (2004)
set.seed(345)
out <- meboot(x=c(4, 12, 36, 20, 8), reps=999, trim=0.25, elaps=TRUE)
mean(out$sens) # ensemble mean should be close to sample mean 16
```

---

meboot.default      *Generate Maximum Entropy Bootstrapped Time Series Ensemble*

---

### Description

This function generates maximum entropy bootstrap replicates for dependent data. (See details.)

### Usage

```
meboot.default(x, reps=999, trim=0.10, reachbnd=TRUE,
expand.sd=TRUE, force.clt=TRUE, elaps=FALSE, ...)
```

### Arguments

<code>x</code>	a vector of data or a time series object.
<code>reps</code>	number of replicates to generate.
<code>trim</code>	the trimming proportion.
<code>reachbnd</code>	logical. If TRUE potentially reached bounds ( <code>xmin</code> = smallest value - trimmed mean and <code>xmax</code> =largest value + trimmed mean) are given when the random draw happens to be equal to 0 and 1, respectively.
<code>expand.sd</code>	logical. If TRUE the standard deviation in the ensemble is expanded. See <a href="#">expand.sd</a> .
<code>force.clt</code>	logical.If TRUE the ensemble is forced to satisfy the central limit theorem. See <a href="#">force.clt</a> .
<code>elaps</code>	logical. If TRUE elapsed time during computations is displayed.
<code>...</code>	possible argument <code>fiv</code> to be passed to <a href="#">expand.sd</a> .

### Details

Seven-steps algorithm:

1. Sort the original data in increasing order and store the ordering index vector.
2. Compute intermediate points on the sorted series.
3. Compute lower limit for left tail (`xmin`) and upper limit for right tail (`xmax`). This is done by computing the `trim` (e.g. 10
4. Compute the mean of the maximum entropy density within each interval in such a way that the *mean preserving constraint* is satisfied. (Denoted as  $m_t$  in the reference paper.) The first and last interval means have distinct formulas. See Theil and Laitinen (1980) for details.
5. Generate random numbers from the [0,1] uniform interval and compute sample quantiles at those points.
6. Apply to the sample quantiles the correct order to keep the dependence relationships of the observed data.
7. Repeat the previous steps several times (e.g. 999).

**Value**

x	original data provided as input.
ensemble	maximum entropy bootstrap replicates.
xx	sorted order stats (x[1] is minimum value).
z	class intervals limits.
dv	deviations of consecutive data values.
dvtrim	trimmed mean of dv.
xmin	data minimum for ensemble=x[1]-dvtrim.
xmax	data x maximum for ensemble=x[n]+dvtrim.
desintxb	desired interval means.
ordxx	ordered x values.
elaps	elapsed time.

**References**

Vinod, H.D. (2006), Maximum Entropy Ensembles for Time Series Inference in Economics, *Journal of Asian Economics*, **17**(6), pp. 955-978

Vinod, H.D. (2004), Ranking mutual funds using unconventional utility theory and stochastic dominance, *Journal of Empirical Finance*, **11**(3), 353-377.

**See Also**

[slider.mts](#).

**Examples**

```
## Ensemble for the AirPassenger time series data
set.seed(345)
out <- meboot(x=AirPassengers, reps=100, trim=0.10, elaps=TRUE)

## Ensemble for T=5 toy time series used in Vinod (2004)
set.seed(345)
out <- meboot(x=c(4, 12, 36, 20, 8), reps=999, trim=0.25, elaps=TRUE)
mean(out$sens) # ensemble mean should be close to sample mean 16
```

---

meboot.pdata.frame *Maximum Entropy Bootstrap for Panel Time Series Data*

---

**Description**

This function applies the maximum entropy bootstrapped in a panel of time series data.

**Usage**

```
meboot.pdata.frame (x, reps=999, trim=0.10, reachbnd=TRUE,
  expand.sd=TRUE, force.clt=TRUE, elaps=FALSE,
  colsubj, coldata, coltimes, ...)
```

**Arguments**

<code>x</code>	a <code>pdata.frame</code> object containing by columns: the individual index, an optional time index and a panel of time series data.
<code>reps</code>	number of replicates to generate for each subject in the panel.
<code>trim</code>	the trimming proportion.
<code>reachbnd</code>	logical. If <code>TRUE</code> potentially reached bounds ( <code>xmin</code> = smallest value - trimmed mean and <code>xmax</code> =largest value + trimmed mean) are given when the random draw happens to be equal to 0 and 1, respectively.
<code>expand.sd</code>	logical. If <code>TRUE</code> the standard deviation in the ensemble is expanded. See <a href="#">expand.sd</a> .
<code>force.clt</code>	logical. If <code>TRUE</code> the ensemble is forced to satisfy the central limit theorem. See <a href="#">force.clt</a> .
<code>elaps</code>	logical. If <code>TRUE</code> elapsed time during computations is displayed.
<code>colsubj</code>	the column in <code>x</code> that contains the individual index.
<code>coldata</code>	the column in <code>x</code> that contains the data of the variable to create the ensemble.
<code>coltimes</code>	an optional argument indicating the column that contains the times at which the observations for each individual are observed.
<code>...</code>	possible argument <code>fiv</code> to be passed to <a href="#">expand.sd</a> .

**Details**

The observations in `x` should be arranged by individuals. The observations for each individual must be sorted by time.

The argument `colsubj` can be either a numeric or a character index indicating the individual or the time series to which each observation is related.

Only one variable can be replicated at a time, `coldata` must be of length one.

If the times at which observations are observed is provided specifying the column with the times through the argument `coltimes`, these times are used only to label the rows of the `data.frame` returned as output.

**Value**

A `data.frame` object of dimension: number of rows of `x` times number of replicates indicated in `reps`. The replicates for the panel of data are arranged by columns. Each replicate in each column is sorted with the same order established in the input `x`.

**See Also**

[meboot](#).

**Examples**

```
## Ensemble for a panel of series of stock prices
data("ullwan")
out <- meboot(ullwan, reps=99, colsubj=2, coldata=4)
```

---

`null.ci`*Get Confidence Interval Around Specified NullZero Total*

---

**Description**

Function to get two sided confidence interval around zero as the true value. Confidence interval is adjusted so that it covers the true zero  $(1-\text{level}) \times 100$  times. Symmetry is not assumed.

**Usage**

```
null.ci (x, level=0.95, null.value=0, type=8, ...)
```

**Arguments**

<code>x</code>	a vector of data.
<code>level</code>	confidence level.
<code>null.value</code>	a specified value of the null, e.g., 0.
<code>type</code>	type of quantile, a number between 1 and 9. See <a href="#">quantile</a> .
<code>...</code>	further arguments passed to or from other methods.

**Value**

Lower limit and upper limit of the confidence interval.

**Examples**

```
x <- runif(25, 0, 1)
null.ci(x)
```

---

olsHALL.b

*OLS regression model for consumption*


---

### Description

Compute OLS coefficients in a regression model for the consumption variable. See details.

### Usage

```
olsHALL.b (y, x)
```

### Arguments

y	dependent variable (consumption).
x	regressor variable (disposable income).

### Details

The regression model is:  $c(t) = b_1 + b_2*c(t) + b_3*y(t-1) + u(t)$ , where 'c' is consumption and 'y' is disposable income.

This function is intended to speed up the ME bootstrap procedure for inference. Instead of using the `lm` or `dynlm` interfaces the function calls directly to the Fortran procedure 'dqrls'.

### Value

Coefficient estimates by OLS.

### Examples

```
data("USconsum")
USconsum <- log(USconsum)

# lm interface
lmcf1 <- lm(USconsum[-1,1] ~ USconsum[-51,1] + USconsum[-51,2])
coefficients(lmcf1)

# dynlm interface
library("dynlm")
lmcf2 <- dynlm(consum ~ L(consum, 1) + L(dispinc, 1), data=USconsum)
coefficients(lmcf2)

# olsHALL.b
olsHALL.b(y=USconsum[,1], x=USconsum[,2])
```

---

`slider.mts`*Slider for Selecting a Single Time Series to Plot from a 'mts' Object*

---

**Description**

This function displays single plots from a 'mts' object. The time series to plot is selected through a slider scaled from 1 to the number total of times series in the object.

**Usage**

```
slider.mts (x, ...)
```

**Arguments**

`x` an object of class `mts`.  
`...` further arguments passed to `plot`.

**Details**

Requires the `tcltk` package.

**See Also**

[meboot](#).

**Examples**

```
## Ensemble for the AirPassenger time series data
set.seed(345)
out <- meboot(x=AirPassengers, reps=100)
slider.mts(ts.union(AirPassengers, out$ensemble),
           ylim=range(out$ensemble))
```

---

`ullwan`*Data about Some of the S&P 500 Stock Prices*

---

**Description**

This data set collects information about seven S&P 500 stocks whose market capitalization exceeds \$27 billion. The seven companies are labelled as ABT, AEG, ATI, ALD, ALL, AOL and AXP. For each company data from May 1993 to November 1997 are available (469 observations).

**Usage**

```
data (ullwan)
```

**Format**

The data are stored in an object of classes `pdata.frame` (a `data.frame` class with further attributes useful for panel data from the `plm` package) and `data.frame`.

**Details**

The following information is contained by columns:

- `Subj`: Company index.
- `Tim`: Times at which the data were observed (on a monthly basis).
- `MktVal`: Market capitalization.
- `Price`: Stock prices.
- `Pupdn`: Binary variable, takes the value 1 if there is a turning point (a switch from a bull to a bear market or vice versa) and 0 otherwise.
- `Tb3`: Interest on 3-month Treasury bills.

**Source**

Compustat database.

**References**

Yves Croissant (2005). `plm`: Linear models for panel data. R package version 0.1-2.

---

USconsum

*Consumption and Disposable Income Data (Annual 1948-1998)*

---

**Description**

Data set employed in Murray (2006, pp.46-47, 799-801) to discuss the Keynesian consumption function on the basis of the Friedman's permanent income hypothesis and Robert Hall's model.

**Usage**

```
data (USconsum)
```

**Format**

A `.rda` file storing the data as an `mts` object.

**Details**

Annual data. Available time series: (Each corresponding label in the list object appears in quotes.)

- `consum`: consumption per capita in thousands of dollars (1948-1998). Log of this variable is the dependent variable and its lagged value is a regressor in Murray's Table 18.1.
- `dispincl`: disposable income per capita in thousands of dollars (1948-1998). Lagged value of the Log of this variable is the second regressor (proxy for permanent income) in Murray's Table 18.1.

**Source**

Robert Hall's data for 1948 to 1977 extended by Murray to 1998 by using standard sources for US macroeconomic data from government publications (U.S. Bureau of Labor Statistics for population data, U.S. Bureau of Economic Analysis for income data, U.S. Bureau of Labor Statistics for consumption data).

**References**

Murray, M.P. (2006), *Econometrics. A modern introduction*, New York: Pearson Addison Wesley.

---

USfygt

*Long-term Treasury Bond Rates and Deficit Data Set (Annual 1948-200)*

---

**Description**

Data set employed in Murray (2006, pp.795-797) to test the null hypothesis that per capita federal deficits explain long-term Treasury bond interest rates based on the Stock and Watson's dynamic OLS model.

**Usage**

```
data (USfygt)
```

**Format**

A `.rda` file storing the data as an `mts` object.

**Details**

Annual data. Available time series: (Each corresponding label in the list object appears in quotes.)

- `"dy"`: mean changes in real per capita income (1949-1998).
- `"fygt1"`: shorth-term (one-year) Treasury bond interest rates (1953-1998).
- `"fygt10"`: long-term (ten-year) Treasury bond interest rates (1953-2000).

- "infl": inflation (1949-2000).
- "usdef": per capita real federal deficit (1948-2000).
- "reallir": real long term interest rates (not used in Murray's Table 18.12).
- "realsir": real short term interest rates (not used in Murray's Table 18.12).

### Source

Data was made available by James Stock and Mark Watson to readers of their famous *Econometrica* paper, 1993, 61, pp 783-820, who in turn used standard sources for US macroeconomic data from government publications.

### References

Murray, M.P. (2006), *Econometrics. A modern introduction*, New York: Pearson Addison Wesley.

---

zero.ci

*Get Confidence Interval Around Zero*

---

### Description

Function to get two sided confidence interval around zero as the true value. Confidence interval is adjusted so that it covers the true zero  $(1 - \text{confl}) * 100$  times. Symmetry is not assumed.

### Usage

```
zero.ci (x, confl=0.05)
```

### Arguments

x	a vector of data.
confl	confidence level.

### Value

bnlo	count of number of items below lower limit.
bnup	count of number of items above upper limit.
lolim	lower limit of the confidence interval.
uplim	upper limit of the confidence interval.

### Examples

```
x <- runif(25, 0, 1)
zero.ci(x)
```

# Index

## \*Topic **datagen**

ullwan, [13](#)  
USconsum, [14](#)  
USfygt, [15](#)

## \*Topic **ts**

checkConv, [2](#)  
expand.sd, [4](#)  
force.clt, [5](#)  
meboot, [6](#)  
meboot.default, [8](#)  
meboot.pdata.frame, [9](#)  
null.ci, [11](#)  
slider.mts, [13](#)  
zero.ci, [16](#)

checkConv, [2](#)  
criterion, [3](#)

expand.sd, [4](#), [6](#), [8](#), [10](#)

force.clt, [5](#), [6](#), [8](#), [10](#)

meboot, [2](#), [3](#), [6](#), [10](#), [13](#)  
meboot.default, [8](#)  
meboot.pdata.frame, [9](#)

null.ci, [11](#)

olsHALL.b, [12](#)

plot, [13](#)

quantile, [11](#)

slider.mts, [7](#), [9](#), [13](#)

ullwan, [13](#)  
USconsum, [14](#)  
USfygt, [15](#)

zero.ci, [16](#)