Package ‘Omisc’

July 3, 2019

Type Package
Title Univariate Bootstrapping and Other Things
Version 0.1.2
Author Patrick O’Keefe
Maintainer Patrick O’Keefe <patrick.okeefe@vanderbilt.edu>
Description Primarily devoted to implementing the Univariate Bootstrap (as well as the Traditional Bootstrap). In addition there are multiple functions for DeFries-Fulker behavioral genetics models. The univariate bootstrapping functions, DeFries-Fulker functions, regression and traditional bootstrapping functions form the original core. Additional features may come online later, however this software is a work in progress. For more information about univariate bootstrapping see: Lee and Rodgers (1998) and Beasley et al (2007) <doi.org/10.1037/1082-989X.12.4.414>.
License GPL-3
Encoding UTF-8
LazyData true
RoxygenNote 6.1.1
Imports MASS, base, stats, psych, copula
Suggests lavaan
NeedsCompilation no
Repository CRAN
Date/Publication 2019-07-03 18:30:03 UTC

R topics documented:

aboot ................................................................. 2
aCalc ............................................................... 3
add ................................................................. 4
ajack ................................................................. 4
AllBootResults ..................................................... 5
BarebonesBetas ................................................... 6
BCa ................................................................. 6
<table>
<thead>
<tr>
<th>abootHbootI</th>
</tr>
</thead>
<tbody>
<tr>
<td>bias</td>
</tr>
<tr>
<td>bootAnalysis</td>
</tr>
<tr>
<td>bootsample</td>
</tr>
<tr>
<td>cent</td>
</tr>
<tr>
<td>centerData</td>
</tr>
<tr>
<td>cholcors</td>
</tr>
<tr>
<td>cholcovs</td>
</tr>
<tr>
<td>DFanalysis</td>
</tr>
<tr>
<td>DFSimulated</td>
</tr>
<tr>
<td>DFSimulatedChisq</td>
</tr>
<tr>
<td>DFSimulatedChisqNew</td>
</tr>
<tr>
<td>doubleEnter</td>
</tr>
<tr>
<td>findSa</td>
</tr>
<tr>
<td>Group_function</td>
</tr>
<tr>
<td>HoffPseudoStandard</td>
</tr>
<tr>
<td>jackknife</td>
</tr>
<tr>
<td>justBetas</td>
</tr>
<tr>
<td>lbind</td>
</tr>
<tr>
<td>leave1out</td>
</tr>
<tr>
<td>MyLM</td>
</tr>
<tr>
<td>NaiveBoot</td>
</tr>
<tr>
<td>NaiveBoot_dep</td>
</tr>
<tr>
<td>resample</td>
</tr>
<tr>
<td>RK</td>
</tr>
<tr>
<td>Sfunc</td>
</tr>
<tr>
<td>standardBootIntervals</td>
</tr>
<tr>
<td>TestData</td>
</tr>
<tr>
<td>uniboot</td>
</tr>
<tr>
<td>unibootsample</td>
</tr>
<tr>
<td>unibootVar</td>
</tr>
<tr>
<td>uniboot_dep</td>
</tr>
<tr>
<td>zScore</td>
</tr>
<tr>
<td>zScoreData</td>
</tr>
<tr>
<td>Index</td>
</tr>
</tbody>
</table>

### Description

**Title**

### Usage

```r
aboot(boot)
```
aCalc

Arguments

   boot          a vector of bootstrap resample statistics to use to calculate the accelleration parameter.

Value

   a vector of accelleration parameters for use in BCa bootstrap intervals

Examples

   data<-DFSImulated()
   boots<-NaiveBoot(data, groups="Rs", keepgroups=TRUE)
   boots<-bootAnalysis(boots, cbind, DFanalysis, 1,2,3, robust=FALSE)
   boots<-t(boots)
   aboot(boots)


---

Description

   This function calculates the actual "a" estimate from the jackknife approximation of a used in BCa CI's

Usage

   aCalc(x)

Arguments

   X          A vector of jackknife results

Value

   An estimate of a for use in BCa.

Examples

   X<-rchisq(100,2)
aCalc(X)
### `add` Function

**Description**

A function to sum a list of matrices.

**Usage**

```r
add(x)
```

**Arguments**

- `x`: a list to be summed. Useful for doing elementwise summation of a list of matrices.

**Value**

returns a single summed object (e.g., a matrix)

**Examples**

```r
x <- list(matrix(c(1:4), nrow=2), matrix(c(1:4), nrow=2))
add(x)
```

### `ajack` Function

**Description**

A function to get the bias parameter (a) for BCa bootstrap intervals.

**Usage**

```r
ajack(data, FUN, ...)
```

**Arguments**

- `data`: data to get the bias parameter (a) for
- `FUN`: a function to be applied to the data
- `...`: additional arguments passed to FUN

**Value**

a vector of acceleration parameters for use in BCa bootstrap intervals
AllBootResults

Examples

data<-DFSimulated()
ajack(data,DFanalysis, betasonly=TRUE, robust=FALSE)

Description

AllBootResults

Usage

AllBootResults(boot, lower = 0.025, upper = 0.975, data, FUN, ...)

Arguments

boot A matrix of bootstrap results
lower the lower alpha
upper the upper alpha
data the data used for analysis
FUN the function used for analysis
... additional arguments to pass to FUN

Value

a matrix of results. Includes the baseline results, all output from standardBootIntervals, all results from BCa for both the jackknife and bootstrap acceleration methods. The bootstrap acceleration method is experimental.

Examples

data<-DFSimulated()
boots<-NaiveBoot(data, groups="Rs", keepgroups=TRUE)
boots<-bootAnalysis(boots, cbind, DFanalysis, 1,2,3, robust=FALSE)
AllBootResults(boots, .025,.975, data, DFanalysis, 1,2,3, robust=FALSE)
BarebonesBetas

**Description**

Gives just the beta weights from a linear model.

**Usage**

BarebonesBetas(data, Y = NULL, RHS = NULL)

**Arguments**

- **data**: Data to be analyzed. Dependent variable MUST BE THE FIRST VARIABLE.
- **Y**: optional. The dependent variable
- **RHS**: option. The right hand side of the model, in R’s model formulation (i.e., ~ X1+X2+etc)

**Value**

A vector of beta coefficients

**Examples**

```r
Data<-TestData()
BarebonesBetas(Data)
```

BCa

**Description**

BCa

**Usage**

BCa(boot, data, alphalower = 0.025, alphappper = 0.975, accelleration = "jack", FUN, ...)

bias

Arguments

boot A vector of bootstrap estimates of Theta
data The data that was analyzed via the bootstrap
alphalower The lower alpha for CI creation
alphaupper The upper alpha for CI creation
acceleration can currently take two values, "jack" and "bootstrap". "jack" returns the jackknife estimate of the acceleration parameter. "boot" is an experimental function that uses the bootstrap estimates in the calculation of the acceleration parameter. "boot" is many times faster (approximately n times faster where n is the number of observations).
FUN The function used to get estimates of Theta
... Additional arguments to FUN

Value
A matrix of BCa bootstrap CI’s, the bias parameter and the acceleration parameter

Examples

```r
data<-DFSsimulated()
boot<-NaiveBoot(data, groups="Rs", keepgroups=TRUE)
boot<-bootAnalysis(boot, cbind, DFanalysis, 1,2,3, robust=FALSE)
BCa(boot, data, .025,.975, acceleration="bootstrap", DFanalysis, 1,2,3, robust=FALSE)
```

**Examples**

```r
data <- data.frame(rnorm(1000))
theta <- mean(X)
boot <- NaiveBoot(X)
boot <- lapply(boot, mean)
boot <- do.call(rbind, boot)
bias(boot, theta)
```

---

**Description**

bootAnalysis

**Usage**

```r
bootAnalysis(boot, collapse = cbind, FUN, ...)
```

**Arguments**

- `boot`: A list of bootstrap resamples from NaiveBoot or uniboot.
- `collapse`: Should the results be collapsed from list form. Can take values of NULL, cbind or rbind
- `FUN`: The function to apply to the bootstrap resamples
- `...`: additional arguments to be passed to FUN

**Value**

A list or matrix of results

**Examples**

```r
data <- DFSimulated()
data <- doubleEnter(data[,1], data[,2], data[,3])
boots <- uniboot(data, 1000, "Rs", TRUE, .5, NULL)
results <- bootAnalysis(boots, cbind, FUN=DFanalysis, 1,2,3,TRUE,FALSE,FALSE,TRUE,FALSE)
```
**bootsample**

---

**bootsample**

---

**Description**

bootsample

**Usage**

bootsample(data, size = 1)

**Arguments**

data : a dataset to be bootstrapped
size : the size of the bootstrap sample relative to the original sample

**Value**

a dataset

**Examples**

```r
X <- testData()
Y <- bootsample(X)
```

---

**cent**

---

**Description**

cent

**Usage**

cent(X)

**Arguments**

X : vector to be centered

**Value**

Returns a centered vector

**Examples**

```r
X <- c(1:10)
cent(X)
```
centerData

Description

CenterData

Usage

centerData(data)

Arguments

data The data to be centered

Value

The centered data

Examples

X<-data.frame(X=c(1:4),Y=c(6:9))
centerData(X)

cholcors

Description

cholcors

Usage

cholcors(X, use = "everything")

Arguments

X A matrix of data.
use the missing data type to use for the correlation. Default is R's default "everything".

Value

This function returns the cholesky decomposition of the correlation matrix of the data
cholcovs

Examples
X<-stats::rnorm(100)
Y<-stats::rnorm(100)+X
Z<-cbind(X,Y)
cholcors(Z)

cholcovs cholcovs

Description
cholcovs

Usage
cholcovs(X, use = "everything")

Arguments
X A matrix of data.
use the missing data type to use for the correlation. Default is R’s default "everything".

Value
This function returns the cholesky decomposition of the correlation matrix of the data

Examples
X<-stats::rnorm(100)
Y<-stats::rnorm(100)+X
Z<-cbind(X,Y)
cholcovs(Z)

DFanalysis

Description
DFanalysis

Usage
DFanalysis(data = NULL, proband, sibling, Rs, RK = T, robust = T,
DE = T, betasonicly = F, typicalSE = F)
Arguments

data A dataframe. This is not necessary as the variables can be passed directly via the other arguments.
proband Called "proband" for historical reasons this is the variable on the left hand side of the regression.
sibling The right hand side version of proband. This would be the matched sibling scores.
Rs This is the vector of relatedness coefficients
RK Use the Rodgers and Kohler simplified version of the DF model (recommended). Data should not be double entered prior to analysis.
robust Use the Kohler and Rodgers robust standard errors (recommended when using double entered data)
DE Will the data need to be double entered?
betasonly If TRUE only the beta weights from the regression analysis will be returned.
typicalSE Should the typical regression standard errors be used? Default is false.

Value
The results from MyLM

Examples

```r
twindata<-DFSimulatedHRPPPLRPPPLNSLNSI
p<-twindata[L1]
s<-twindata[LR]
r<-twindata[LS]
Dfanalysis(data=NULL, p, s, r)
```

<table>
<thead>
<tr>
<th>DFSimulated</th>
<th>DFSimulated</th>
</tr>
</thead>
</table>

Description

DFSimulated

Usage

DFSimulated(MZ = 250, DZ = 250, a2 = 0.3, c2 = 0.3)

Arguments

MZ Number of MZ twins to simulate
DZ Number of DZ twins to simulate
a2 Heritability (proportion of variance)
c2 Shared environment (proportion of variance)
DFSimulatedChisq

Value

A dataframe

Examples

TwinData<-DFSimulatedChisq(200,200,.3,.3)

Description

DFSimulatedChisq

Usage

DFSimulatedChisq(MZ = 250, DZ = 250, a2 = 0.3, c2 = 0.3, df = 10)

Arguments

MZ Number of MZ twins to simulate
DZ Number of DZ twins to simulate
a2 Heritability (proportion of variance)
c2 Shared environment (proportion of variance)
df Total degrees of freedom for the Chi-Square variable

Value

A dataframe of Chi-Square distributed outcome observations for MZ and DZ twins

Examples

TwinData<-DFSimulatedChisq(200,200,.3,.3,10)
### DFSimulatedChisqNew

**Description**

DFSimulatedChisqNew

**Usage**

`DFSimulatedChisqNew(MZ = 250, DZ = 250, a2 = 0.3, c2 = 0.3, df = 5)`

**Arguments**

- **MZ**: Number of MZ twins to simulate
- **DZ**: Number of DZ twins to simulate
- **a2**: Heritability (proportion of variance)
- **c2**: Shared environment (proportion of variance)
- **df**: Total degrees of freedom for the Chi-Square variable

**Value**

A dataframe of Chi-Square distributed outcome observations for MZ and DZ twins

**Examples**

```r
TwinData <- DFSimulatedChisqNew(200, 200, 0.3, 0.3, 10)
```

---

### doubleEnter

**Description**

DoubleEnter

**Usage**

`doubleEnter(proband, sibling, Rs)`

**Arguments**

- **proband**: The proband scores
- **sibling**: The matched sibling scores
- **Rs**: The relatedness coefficients
**Description**

This is an implementation of the YHY bootstrap covariance matrix.

**Usage**

```r
findSa(S, fitted, p, a = 0.5, df, n, tau = NULL, tol = 1e-07)
```

**Arguments**

- `S`: Sample covariance matrix
- `fitted`: The fitted covariance matrix
- `p`: the number of columns in the covariance matrix
- `a`: the starting value for the a parameter
- `df`: the degrees of freedom in the model
- `n`: the number of participants in the model
- `tau`: the population tau. If no tau is provided, the estimated tau from the model will be used
- `tol`: the difference between ga and tau at which the function will converge

**Value**

A list of the "a" adjusted covariance matrix, Sa, the tau, ga, and the number of iterations.

**Examples**

```r
require(Omisc)
require(lavaan)
set.seed(2^7-1)
modelTest<-'
LV1=-.7*x1+.8*x2+.75*x3+.6*x4
LV2=-.7*y1+.8*y2+.75*y3+.6*y4
LV1=-.3*LV2
LV1=-1*LV1
LV2=-1*LV2
```
modelFit<-'
LV1<- x1+x2+x3+x4
LV2<- y1+y2+y3+y4
LV1<-start(.5)*LV2
LV1<-1+LV1
LV2<-1+LV2
'

testdata<-simulateData(modelTest, sample.nobs = 250)
fit<-cfa(modelFit, testdata)

fitted<-fitted(fit)$cov
fitted<-fitted[,1:ncol(fitted)]
S<-cov(fitted)
p<-8
a<-.5
n<-250
df<-21
findSa(S, fitted, p, .5, df, n)

---

**Group_function**

**Grouping_function**

**Description**

originally from the ParallelTree package. If data argument is Null, takes a variable "x" and a matrix or dataframe of level identifiers (e.g., mother and then child IDs). Level variables should be included in order from highest level to the lowest. Listwise deletes missing data. Otherwise grabs variables from entered dataframe Group_function

**Usage**

Group_function(data = NULL, x, levels, func = mean, center = FALSE,
              nested = TRUE, append = FALSE, funcName = "Mean")

**Arguments**

data a data frame with the x and level variables included. Default is NULL.

x If data = NULL a dataframe of scores to have the function applied to. If data != NULL, a vector of string(s) naming the variable(s) in data to use.

levels If data = NULL, a dataframe of grouping variables. If data != NULL, a vector of strings naming the variables in data to use. levels should be ordered from the highest level to the lowest. Group and case identifiers should be unique, if they are not unique, cases with non-unique identifiers will be grouped together.

func A function to apply at each group. Default is mean.
**center**
If set to true variables will be group/person mean centered. Note that the grand mean remains unchanged by this operation. If this output is to be passed directly to Parallel_Tree the grand mean should be set to 0.

**nested**
Are level variables nested? Default is TRUE. If set to FALSE means will be calculated for level variable independently. FALSE may be useful in cases of crossed designs. Note that if data are nested but all identifiers are unique both within and across groups nested = FALSE and nested = TRUE will return the same result.

**append**
If set to true, the original data will be returned along with all created variables.

**funcName**
Provides way to name function used. This is used when creating names for created variables. Default is "Mean".

**Value**
This function returns a dataframe with variables labeled according to the level at which the function was applied. Assumed function is mean, and all variables are labeled accordingly. If an alternative function is used labels should be manually changed to reflect function used.

**Examples**

```r
#the ChickWeight data is from base R
#nested is set to false because Chick and Time are crossed
Means_Chick<-Group_function(data=ChickWeight,x="weight", levels=c("Diet","Chick","Time"), nested = FALSE, append=TRUE)
```

---

**HoffPseudoStandard**

**Description**
HoffPseudoStandard

**Usage**
HoffPseudoStandard(betas, SDX, interceptvar)

**Arguments**
- **betas**
  A vector of betas from a multilevel model
- **SDX**
  A vector of the standard deviations of the X value for each of the X’s associated with the bets
- **interceptvar**
  A vector of the intercept variances at the level associated with the betas

**Value**
A vector of pseudostandardized coefficients
justBetas

Examples

```r
print("none")
```

Description

jackknife

Usage

`jackknife(data)`

Arguments

- `data` (data) The data to jackknife

Value

a list of jackknife datasets

Examples

```r
data<-cbind(1:10,1:10)
result<-jackknife(data)
lapply(result,mean)
```

justBetas

Description

justBetas

Usage

`justBetas(data, Y, X)`

Arguments

- `data` (data) A data frame
- `Y` (Y) The name or column number of the Y variable
- `X` (X) The name(s) or column number(s) of the X variables
Value

A vector of unstandardized beta weights

Examples

\[
\begin{align*}
X & \leftarrow \text{stats::rnorm(100)} \\
Y & \leftarrow \text{stats::rnorm(100)} + 5 \times (X) \\
data & \leftarrow \text{cbind}(Y, X) \\
\text{justBetas(data, 1, 2)} & \\
\text{# if you want an intercept} & \\
Y & \leftarrow \text{stats::rnorm(100)} + 5 \times (X) + 5 \\
data & \leftarrow \text{cbind}(Y, X, 1) \\
\text{justBetas(data, 1, c(2:3))} & \\
\end{align*}
\]

Description

`lbind` is meant to be used in conjunction with `lapply` to combine elements of lists using `rbind`.

Usage

`lbind(index, alist, n)`

Arguments

- `index`: a list of indexes. This should count the number of items to return in the final list.
- `alist`: a list of objects to be passed to `rbind`. They should be grouped according to which objects will be combined (e.g., if 1, 2, 3 are to be passed to `cbind` then they should be adjacent to each other).
- `n`: The number of objects in each group. Currently each group must consist of the same number of objects.

Value

a list

Examples

```r
alist<-list(c(1,1),c(2,2),c(3,3))
index<-list(1)
n<-3
lapply(index,lbind,alist,3)
```


**leave1out**

**Description**

leave1out

**Usage**

`leave1out(x, data)`

**Arguments**

- `x` Which row(s) of data to leave out
- `data` A dataframe or matrix.

**Value**

The reduced dataframe or matrix

**Examples**

```r
data <- cbind(1:10, 1:10)
leave1out(5, data)
```

**MyLM**

**Description**

MyLM

**Usage**

`MyLM(Y, X, robust = F, betasonly = F, typicalSE = T)`

**Arguments**

- `Y` The Y variable
- `X` A matrix of X variables
- `robust` Should robust standard errors be calculated? Assumes a double entered twin dataset with twins evenly spaced in the dataset.
- `betasonly` Should only the betas be returned? Good for bootstrapping
- `typicalSE` Should the typical standard errors be included? Default is true. Can be true when robust is True.
The Naive Bootstrap

**Description**

The Naive Bootstrap

**Usage**

```r
NaiveBoot(data, B = 1000, groups = NULL, keepgroups = F, size = 1)
```

**Arguments**

- `data`: data to be bootstrapped
- `B`: number of bootstrap samples to take
- `groups`: grouping variable if there is one
- `keepgroups`: keep the grouping variable?
- `size`: size of the bootstrap resamples relative to the original sample

**Value**

a list of bootstrap resamples

**Examples**

```r
X <- DFSimulated(100, 100, .4, .4)
Y <- RK(X[, 1], X[, 2], X[, 3])
MyLM(Y[, 1], Y[, c(2:3)], TRUE)
```
The Naive Bootstrap

**Description**

The Naive Bootstrap

**Usage**

NaiveBoot_dep(data, B = 1000, groups = NULL, keepgroups = F, size = 1)

**Arguments**

- data: data to be bootstrapped
- B: number of bootstrap samples to take
- groups: grouping variable if there is one
- keepgroups: keep the grouping variable?
- size: size of the bootstrap resamples relative to the original sample

**Value**

a list of bootstrap resamples

**Examples**

X <- TestData()
Y <- NaiveBoot(X)

---

**resample**

**Description**

resample

**Usage**

resample(X, size)

**Arguments**

- X: A vector to be resampled
- size: The size of the resulting vector. Should be a number such that size*nrow(X) is a whole number
**Value**

A vector of resampled X values

**Examples**

```r
X<-c(1:10)
resample(X,.5)
```

---

**Description**

RK

**Usage**

```r
RK(proband, sibling, Rs, DE = T)
```

**Arguments**

- **proband**: column name or number of the proband
- **sibling**: column name or number of the siblings
- **Rs**: column name or number of the relatedness coefficients
- **DE**: Should the data be double entered?

**Value**

A dataframe

**Examples**

```r
X<-DFSImulated(100,100,.3,.3)
Y<-RK(X[,1],X[,2],X[,3])
```
**Sfunc**

**Description**

Function for calculating the matrices for the Kohler Rodgers SE

**Usage**

Sfunc(X, e)

**Arguments**

- **X**: A matrix of X variables
- **e**: A matrix of error terms

**Value**

A matrix

**Examples**

print("Nah")

---

**standardBootIntervals**

**Description**

This returns the quantiles of the bootstrap samples specified by the user. The quantiles uses the type=4 argument of the quantile function, which appears to function best.

**Usage**

standardBootIntervals(boot, lower = 0.025, upper = 0.975)

**Arguments**

- **boot**: A vector of bootstrap results
- **lower**: the lower alpha
- **upper**: the upper alpha

**Value**

A matrix of the mean, median, min, max, lower and upper CI values
Description

Simple function for creating a dataset of two related variables.

Usage

\[
\text{ TestData(nobs = 1000, intercept = 0, beta = 5, meanX = 0, } \\
\text{ sdX = 1, sdYerr = 1)}
\]

Arguments

- `nobs` Number of observations, defaults to 1000
- `intercept` Intercept of the regression. Defaults to 0
- `beta` Beta for the regression equation, defaults to 5
- `meanX` Mean of \( X \), defaults to 0
- `sdX` Standard deviation of \( X \), defaults to 1
- `sdYerr` Variance of the error term of \( Y \), defaults to 1

Value

A dataframe with an \( X \) and \( Y \) variable produced by the entered parameters

Examples

\[
X <- \text{TestData()}
\]
**Description**

**WARNING:** This function can’t be used with data that is already fed through the RK function. The correlation matrix will not be positive definite.

**Usage**

```r
uniboot(data, B = 1000, groups = NULL, keepgroups = F, size = 1,
         HICor = NULL, samplefrom = "group", use = "everything",
         standardized = T)
```

**Arguments**

- `data`: The data frame to be resampled
- `B`: The number of bootstrap samples.
- `groups`: A grouping variable name
- `keepgroups`: Should the grouping variable be kept in the final datasets?
- `size`: The size of the bootstrap sample to be returned. Should be as a proportion and must be evenly divided into nrow(data).
- `HICor`: If a hypothesis imposed correlation matrix is to be used, this argument takes a list of hypothesized correlation matrices. IT MUST BE A LIST OF ONE OR MORE MATRICES. Multiple matrices can be entered in the case of grouped data (one for each group). If the nil-null correlation is to be used an identity matrix can be entered here (the same size as the appropriate correlation matrix).
- `samplefrom`: Takes one of either "group" or "whole". When doing bootstrapping of grouped data this tells uniboot if the whole sample should be used as the sampling frame for each group (whole), or not (group). "group" should be used unless it is believed that all groups share the same underlying marginal distribution for each variable (e.g., the same mean and variance in the case of normally distributed data).
- `use`: The missing data method for cor. Default is R’s default "everything".
- `standardized`: should the resampled data be standardized? The default is TRUE. This is computationally more efficient (the data are standardized as a step during the diagonalization procedure).

**Value**

A list of bootstrap samples

**Examples**

```r
data<-TestData()
X<-uniboot(data,1000)
```
unibootsample

Description

unibootsample

Usage

unibootsample(data, size)

Arguments

data A dataframe or matrix to be univariately bootstrapped
size size of each bootstrap sample as a fraction of the total sample size. Total sample size must be evenly divisible by "size".

Value

A matrix or dataframe with nrow=nrow(X)*size

Examples

x=c(0:9)
y=c(20:29)
z=cbind(x,y)
unibootsample(z,1)

unibootVar

Description

unibootVar

Usage

unibootVar(x, times)

Arguments

x The variable
times The number of times the variable is repeated in the univariate sampling frame. This is going to be equal to the number of variables being univariately sampled.
Value
The variance of the variable in the univariate sampling frame

Examples
X<-c(1,2)
times<-100
unibootvar(X,times)
var(X)

---
uniboot_dep  Univariate Bootstrap
---

Description
WARNING: This function can’t be used with data that is already fed through the RK function. The correlation matrix will not be positive definite.

Usage
uniboot_dep(data, B = 1000, groups = NULL, keepgroups = F,
size = 1, HICor = NULL, samplefrom = "group", use = "everything")

Arguments
- **data**: The data frame to be resampled
- **B**: The number of bootstrap samples. Alternatively "sampleframe" which will return the univariate sampling frame. "samplefrom" is not advised when there are many observations and/or many variables as the returned dataframe will be quite large.
- **groups**: A grouping variable name
- **keepgroups**: Should the grouping variable be kept in the final datasets?
- **size**: The size of the bootstrap sample to be returned. Should be as a proportion and must be evenly divided into nrow(data).
- **HICor**: If a hypothesis imposed correlation matrix is to be used, this argument takes a list of hypothesized correlation matrices. IT MUST BE A LIST OF ONE OR MORE MATRICES. Multiple matrices can be entered in the case of grouped data (one for each group). If the nil-null correlation is to be used an identity matrix can be entered here (the same size as the appropriate correlation matrix).
- **samplefrom**: Takes one of either "group" or "whole". When doing bootstrapping of grouped data this tells uniboot if the whole sample should be used as the sampling frame for each group (whole), or not (group). "group" should be used unless it is believed that all groups share the same underlying marginal distribution for each variable (e.g., the same mean and variance in the case of normally distributed data).
- **use**: The missing data method for cor. Default is R’s default "everything".
Value

A list of bootstrap samples

Examples

data<-TestData()
X<-uniboot(data,1000)

zScore

Description

Title

Usage

zScore(X, reps = 1)

Arguments

x The vector to be turned into z scores
reps The number of reps the vector is to be repeated. This will only be used in univariate bootstrapping. The default is 1.

Value

A vector of z scores.

Examples

X<-c(1:10)
zScore(X)

zScoreData

Description

centerData

Usage

zScoreData(data)
Arguments

data  The data to be converted to z scores

Value

Data converted to z scores

Examples

```r
X <- data.frame(X = c(1:4), Y = c(6:9))
zScoreData(X)
```
Index

aboot, 2
aCalc, 3
add, 4
ajack, 4
AllBootResults, 5
BarebonesBetas, 6
BCa, 6
bias, 7
bootAnalysis, 8
bootsample, 9
cent, 9
centerData, 10
cholcors, 10
cholcovs, 11
DFanalysis, 11
DFSimulated, 12
DFSimulatedChisq, 13
DFSimulatedChisqNew, 14
doubleEnter, 14
findSa, 15
Group_function, 16
HoffPseudoStandard, 17
jackknife, 18
justBetas, 18
lbind, 19
leave1out, 20
MyLM, 20
NaiveBoot, 21
NaiveBoot_dep, 22
resample, 22
RK, 23
Sfunc, 24
standardBootIntervals, 24
TestData, 25
uniboot, 26
uniboot_dep, 28
unibootsample, 27
unibootVar, 27
zScore, 29
zScoreData, 29