

# Package ‘Omisc’

May 31, 2026

**Type** Package

**Title** DeFries-Fulker Analysis and Univariate Bootstrapping

**Version** 0.2.0

**Description** Implements the Univariate Bootstrap and the Traditional (Naive) Bootstrap for resampling multivariate data while preserving covariance structure. Also provides functions for DeFries-Fulker behavioral genetics models, including the Rodgers-Kohler formulation with robust standard errors.

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 8.0.0

**Imports** MASS, stats, parallel, psych

**Suggests** lavaan

**NeedsCompilation** no

**Author** Patrick O’Keefe [aut, cre]

**Maintainer** Patrick O’Keefe <pgok15@gmail.com>

**Repository** CRAN

**Date/Publication** 2026-05-30 22:41:08 UTC

## Contents

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
DFanalysis . . . . .	11
DFSimulated . . . . .	12
DFSimulatedChisq . . . . .	12
doubleEnter . . . . .	13
endparallel . . . . .	14
findSa . . . . .	14
HoffPseudoStandard . . . . .	15
jackknife . . . . .	16
justBetas . . . . .	17
leave1out . . . . .	17
MyLM . . . . .	18
NaiveBoot . . . . .	19
Parboot . . . . .	19
parUniboot . . . . .	20
resample . . . . .	21
RK . . . . .	22
Sfunc . . . . .	22
standardBootIntervals . . . . .	23
startparallel . . . . .	23
TestData . . . . .	24
uniboot . . . . .	25
uniboosample . . . . .	26
unibootVar . . . . .	26
zScore . . . . .	27
zScoreData . . . . .	28

**Index** **29**

---

aboot	<i>Title</i>
-------	--------------

---

**Description**

Title

**Usage**

aboot(boot)

**Arguments**

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

**Value**

a vector of acceleration 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)
```

---

aCalc	<i>aCalc</i>
-------	--------------

---

**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                      *add*

---

**Description**

add

**Usage**

```
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**

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

---

ajack                      *ajack*

---

**Description**

ajack

**Usage**

```
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

**Examples**

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

---

AllBootResults	<i>AllBootResults</i>
----------------	-----------------------

---

**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<-DFSsimulated()
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

*BarebonesBetas*

---

**Description**

Gives just the beta weights from a linear model.

**Usage**

```
BarebonesBetas(data)
```

**Arguments**

data                    Data to be analyzed. Dependent variable MUST BE THE FIRST VARIABLE.

**Value**

A vector of beta coefficients

**Examples**

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

---

BCa

*BCa*

---

**Description**

BCa

**Usage**

```
BCa(  
  Boot,  
  data,  
  alphalower = 0.025,  
  alphaupper = 0.975,  
  accelleration = "jack",  
  FUN,  
  ...  
)
```

**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 jack-knife 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**

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

---

bias	<i>Title</i>
------	--------------

---

**Description**

Title

**Usage**

```
bias(boot, theta)
```

**Arguments**

boot	A vector of bootstrap estimates of theta
theta	the sample estimate of theta

**Value**

z0 the bias parameter for BCa CI

**Examples**

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

---

bootAnalysis

*bootAnalysis*


---

**Description**

bootAnalysis

**Usage**

```
bootAnalysis(boot, collapse, 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**

```
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	<i>bootsample</i>
------------	-------------------

---

**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**

```
X<-TestData()  
Y<-bootsample(X)
```

---

cent	<i>cent</i>
------	-------------

---

**Description**

cent

**Usage**

```
cent(X)
```

**Arguments**

X	vector to be centered
---	-----------------------

**Value**

Returns a centered vector

**Examples**

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

centerData                    *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                    *cholcors*

---

**Description**

cholcors

**Usage**

```
cholcors(X)
```

**Arguments**

X                    A matrix of data.

**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)
cholcors(Z)
```

---

DFanalysis

*DFanalysis*


---

### Description

DFanalysis

### Usage

```
DFanalysis(
  data = NULL,
  proband,
  sibling,
  Rs,
  RK = TRUE,
  robust = TRUE,
  DE = TRUE,
  betasonly = FALSE,
  typicalSE = FALSE
)
```

### 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**

```
TwinData<-DFSImulated(2000,2000,.3,.3)
p<-TwinData[,1]
s<-TwinData[,2]
r<-TwinData[,3]
DFanalysis(data=NULL, p,s,r)
```

---

DFSImulated

*DFSImulated*


---

**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)

**Value**

A dataframe

**Examples**

```
TwinData<-DFSImulated(200,200,.3,.3)
```

---

DFSImulatedChisq

*DFSImulatedChisq*


---

**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)
```

---

doubleEnter	<i>DoubleEnter</i>
-------------	--------------------

---

**Description**

DoubleEnter

**Usage**

```
doubleEnter(proband, sibling, Rs)
```

**Arguments**

proband	The proband scores
sibling	The matched sibling scores
Rs	The relatedness coefficients

**Value**

A dataframe

**Examples**

```
X<-DFSImulated(10,10,.2,.2)
Y<-doubleEnter(X[, "proband"], X[, "sibling"], X[, "Rs"])
```

---

endparallel	<i>endparallel</i>
-------------	--------------------

---

**Description**

endparallel

**Usage**

endparallel(clust)

**Arguments**

clust                    dummy variable so that the function executes

**Value**

NA

**Examples**

```
print("NA")
```

---

findSa	<i>findSa</i>
--------	---------------

---

**Description**

This is an implementation of the YHY bootstrap covariance matrix.

**Usage**

```
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**

```
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(testdata)
p<-8
a<-.5
n<-250
df<-21
findSa(S, fitted, p, .5, df, n)
```

---

HoffPseudoStandard      *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

**Examples**

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

---

jackknife	<i>jackknife</i>
-----------	------------------

---

**Description**

jackknife

**Usage**

```
jackknife(data)
```

**Arguments**

data	The data to jackknife
------	-----------------------

**Value**

a list of jackknife datasets

**Examples**

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

---

justBetas	<i>justBetas</i>
-----------	------------------

---

**Description**

justBetas

**Usage**

```
justBetas(data, Y, X)
```

**Arguments**

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

**Value**

A vector of unstandardized beta weights

**Examples**

```
X<-stats::rnorm(100)
Y<-stats::rnorm(100)+5*(X)
data<-cbind(Y,X)
justBetas(data,1,2)
#if you want an intercept
Y<-stats::rnorm(100)+5*(X)+5
data<-cbind(Y,X,1)
justBetas(data,1,c(2:3))
```

---

leave1out	<i>leave1out</i>
-----------	------------------

---

**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**

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

---

MyLM

*MyLM*

---

**Description**

MyLM

**Usage**

```
MyLM(Y, X, robust = FALSE, betasonly = FALSE, typicalSE = TRUE)
```

**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.

**Value**

Returns a matrix of betas and standard errors

**Examples**

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

---

NaiveBoot

*The Naive Bootstrap*

---

### Description

The Naive Bootstrap

### Usage

```
NaiveBoot(data, B = 1000, groups = NULL, keepgroups = FALSE, 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)
```

---

Parboot

*Parboot*

---

### Description

Parboot

### Usage

```
Parboot(X, data, groups = NULL, keepgroups = FALSE, size = 1, HICor = NULL)
```

**Arguments**

X	A dummy variable to make parLapply happy
data	The data frame to be resampled
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 hypothesis imposed correlations are to be used, this is where the HI correlation matrix goes.

**Value**

A list of bootstrap samples

**Examples**

```
#A single univariate bootsrap sample
X<-TestData()
Y<-Parboot(data=X)
```

---

parUniboot

*parUniboot*

---

**Description**

parUniboot

**Usage**

```
parUniboot(
  data,
  B,
  clust,
  groups = NULL,
  keepgroups = FALSE,
  size = 1,
  HIcor = NULL,
  ...
)
```

**Arguments**

data	data to be bootstrapped
B	the number of bootstrap replications
clust	The list of clusters to use. Should be initialized using startparallel()
groups	Groups to be independently bootstrapped
keepgroups	Should the grouping variable be kept in the final dataset?
size	Size of the bootstrap sample relative to the original sample
HIcor	A hypothesis imposed correlation matrix to be used. Default is NULL
...	additional arguments to be passed. Currently does nothing

**Value**

A list of dataframes of size (size\*nrow(data))

**Examples**

```
#data<-TestData()
#clust<-startparallel("data")
#results<-parUniboot(data,1000,clust)
#endparallel(clust)
```

---

resample	<i>resample</i>
----------	-----------------

---

**Description**

resample

**Usage**

```
resample(X, size)
```

**Arguments**

X	A vector to be resamples
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**

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

---

RK	<i>RK</i>
----	-----------

---

**Description**

RK

**Usage**

```
RK(proband, sibling, Rs, DE = TRUE)
```

**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**

```
X<-DFSimulated(100,100,.3,.3)
Y<-RK(X[,1],X[,2],X[,3])
```

---

Sfunc	<i>Sfunc</i>
-------	--------------

---

**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 Title
```

---

**Description**

Title

**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

**Examples**

```
data<-DFSimulated()
boots<-NaiveBoot(data, groups="Rs", keepgroups=TRUE)
boots<-bootAnalysis(boots, cbind, DFanalysis,1,2,3,TRUE,FALSE,TRUE,TRUE,FALSE)
apply(boots,1, standardBootIntervals)
DFanalysis(data,1,2,3)
```

---

```
startparallel            startparallel
```

---

**Description**

startparallel

**Usage**

```
startparallel(data)
```

**Arguments**

data	data to pass to the clusters. Must be the name of the data in quotes
------	--

**Value**

NA

**Examples**

```
#data<-TestData()  
#clust<-startparallel("data")  
#endparallel(clust)
```

---

TestData

*TestData*

---

**Description**

Simple function for creating a dataset of two related variables.

**Usage**

```
TestData(nobs = 1000, intercept = 0, beta = 5, meanX = 0, 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<-TestData()
```

---

uniboot	<i>Univariate Bootstrap</i>
---------	-----------------------------

---

### 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(
  data,
  B = 1000,
  groups = NULL,
  keepgroups = FALSE,
  size = 1,
  HIcor = NULL,
  sampleframe = "group"
)
```

### Arguments

data	The data frame to be resampled
B	The number of bootstrap samples. Alternatively "sampleframe" which will return the univariate sampling frame. "sampleframe" 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. <b>IT MUST BE A LIST OF ONE OR MORE MATRICES.</b> 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).
sampleframe	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).

### Value

A list of bootstrap samples

**Examples**

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

---

unibootsample	<i>unibootsample</i>
---------------	----------------------

---

**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	<i>unibootVar</i>
------------	-------------------

---

**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)
```

---

zScore

*zScore*

---

**Description**

zScore

**Usage**

```
zScore(X, times)
```

**Arguments**

X	vector to be converted to z scores
times	exponent controlling the denominator scaling

**Value**

Returns a vector of z scores

**Examples**

```
X<-c(1:10)
zScore(X, times=1)
```

---

`zScoreData``centerData`

---

**Description**`centerData`**Usage**`zScoreData(data)`**Arguments**`data`            The data to be converted to z scores**Value**

Data converted to z scores

**Examples**

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

# Index

about, [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](#)  
  
DFanalysis, [11](#)  
DFSimulated, [12](#)  
DFSimulatedChisq, [12](#)  
doubleEnter, [13](#)  
  
endparallel, [14](#)  
  
findSa, [14](#)  
  
HoffPseudoStandard, [15](#)  
  
jackknife, [16](#)  
justBetas, [17](#)  
  
leave1out, [17](#)  
  
MyLM, [18](#)  
  
NaiveBoot, [19](#)  
  
Parboot, [19](#)  
parUniboot, [20](#)  
  
resample, [21](#)  
RK, [22](#)  
  
Sfunc, [22](#)  
standardBootIntervals, [23](#)  
startparallel, [23](#)  
  
TestData, [24](#)  
  
uniboot, [25](#)  
uniboosample, [26](#)  
unibootVar, [26](#)  
  
zScore, [27](#)  
zScoreData, [28](#)