Package ‘resample’

April 12, 2015

Type Package
Title Resampling Functions
Version 0.4
Date 2015-04-25
Author Tim Hesterberg
Maintainer Tim Hesterberg <timhesterberg@gmail.com>
Depends R (>= 3.1.0), graphics, stats
Suggests splus2R
Description Bootstrap, permutation tests, and other resampling functions,
    featuring easy-to-use syntax.
License BSD_3_clause + file LICENSE
LazyLoad yes
ByteCompile yes
NeedsCompilation no
Repository CRAN
Date/Publication 2015-04-12 22:44:23

R topics documented:

  resample-package .................................................. 2
  bootstrap .......................................................... 5
  cat0 ................................................................. 8
  CI.percentile ...................................................... 9
  colVars ............................................................. 10
  deprecated.resample .............................................. 11
  ExpandProbs ....................................................... 12
  IfElse ............................................................... 14
  jackknife .......................................................... 15
  print.resample .................................................... 16
  Quantile ........................................................... 18
  resample .......................................................... 19
  resample-data ..................................................... 21
  samp.bootstrap ................................................... 22
Overview of the resample package

Description

Resampling functions, including one- and two-sample bootstrap and permutation tests, with an easy-to-use syntax.

Details

See `library(help = resample)` for version number, date, etc.

Data Sets

A list of datasets is at `resample-data`.

Main resampling functions

The main resampling functions are: `bootstrap, bootstrap2, permutationTest, permutationTest2`.

Methods

Methods for generic functions include: `print.resample, plot.resample, hist.resample, qqnorm.resample, and quantile.resample`.

Confidence Intervals

Functions that calculate confidence intervals for `bootstrap` and `bootstrap2` objects: `CI.bca, CI.bootstrapT, CI.percentile, CI.t`.

Samplers

Functions that generate indices for random samples: `samp.bootstrap, samp.permute`.

Low-level Resampling Function

This is called by the main resampling functions, but can also be called directly: `resample`.

New Versions

I will post the newest versions to `http://www.timhesterberg.net/r-packages`. See that page to join a list for announcements of new versions.

Author(s)

Tim Hesterberg <timhesterberg@gmail.com>,
`http://www.timhesterberg.net/bootstrap`
Examples

data(Verizon)
ILEC <- with(Verizon, Time[Group == "ILEC"])
CLEC <- with(Verizon, Time[Group == "CLEC"])

### Sections in this set of examples
### Different ways to specify the data and statistic
### Example with plots and confidence intervals.

### Different ways to specify the data and statistic
# This code is flexible; there are different ways to call it,
# depending on how the data are stored and on the statistic.

## One-sample Bootstrap

## Not run:
# Ordinary vector, give statistic as a function
bootstrap(CLEC, mean)

# Vector by name, give statistic as an expression
bootstrap(CLEC, mean(CLEC))

# Vector created by an expression, use the name 'data'
bootstrap(with(Verizon, Time[Group == "CLEC"]), mean(data))

# A column in a data frame; use the name of the column
temp <- data.frame(foo = CLEC)
bootstrap(temp, mean(foo))

# Put function arguments into an expression
bootstrap(CLEC, mean(CLEC, trim = .25))

# Put function arguments into a separate list
bootstrap(CLEC, mean, args.stat = list(trim = .25))

## End(Not run)

## One-sample jackknife

# Like bootstrap. E.g.
jackknife(CLEC, mean)

## One-sample permutation test

# To test H0: two variables are independent, exactly
# one of them just be permuted. For the CLEC data,
# we'll create an artificial variable.
CLEC2 <- data.frame(Time = CLEC, index = 1:length(CLEC))

## Not run:
permutationTest(CLEC2, cor(Time, index),
    resampleColumns = "index")
# Could permute "Time" instead.

# resampleColumns not needed for variables outside 'data'
permutationTest(CLEC, cor(CLEC, 1:length(CLEC)))

## End(Not run)

### Two-sample problems
### Different ways to specify data and statistic

### Two-sample bootstrap

# Two data objects (one for each group)
## Not run: bootstrap2(CLEC, data2 = ILEC, mean)

# data frame containing y variable(s) and a treatment variable
## Not run: bootstrap2(Verizon, mean(Time), treatment = Group)

# treatment variable as a separate object
temp <- Verizon$Group
## Not run: bootstrap2(Verizon$Time, mean, treatment = temp)

### Two-sample permutation test

# Like bootstrap2. E.g.
## Not run: permutationTest2(CLEC, data2 = ILEC, mean

### Example with plots and confidence intervals.
## Not run:
boot <- bootstrap2(CLEC, data2 = ILEC, mean)
perm <- permutationTest2(CLEC, data2 = ILEC, mean,
                         alternative = "greater")

## End(Not run)

## Not run:
par(mfrow = c(2,2))
hist(boot)
qqnorm(boot)
bootstrap

qline(boot$replicates)
hist(perm)

## End (Not run)

# P-value
perm

# Standard error, and bias estimate
boot

# Confidence intervals
CI.percentile(boot) # Percentile interval
CI.t(boot) # t interval using bootstrap SE
# CI.bootstrapT and CI.bca don't currently support two-sample problems.

# Statistic can be multivariate.
# For the bootstrap2, it must have the estimate first, and a standard
# error second (don't need to divide by sqrt(n), that cancels out).
bootC <- bootstrap(CLEC, mean, seed = 0)
bootC2 <- bootstrap(CLEC, c(mean = mean(CLEC), sd = sd(CLEC)), seed = 0)
identical(bootC$replicates[, 1], bootC2$replicates[, 1])

CI.percentile(bootC)
CI.t(bootC)
CI.bca(bootC)
CI.bootstrapT(bootC2)
# The bootstrapT is the most accurate for skewed data, especially
# for small samples.

# By default the percentile interval is "expanded", for better coverage
# in small samples. To turn this off:
CI.percentile(bootC, expand = FALSE)

---

**bootstrap**

*One and two sample bootstrap sampling and permutation tests.*

**Description**

Basic resampling. Supply the data and statistic to resample.

**Usage**

```
bootstrap(data, statistic, R = 10000,
    args.stat = NULL, seed = NULL, sampler = samp.bootstrap,
    label = NULL, statisticNames = NULL, block.size = 100,
    trace = FALSE)
bootstrap2(data, statistic, treatment, data2 = NULL, R = 10000,
    ratio = FALSE,
    args.stat = NULL, seed = NULL, sampler = samp.bootstrap,
```
Arguments

data
statistic
R
treatment
data2
alternative
ratio
resampleColumns
args.stat
paired
seed
sampler
label
statisticNames
block.size
trace
tolerance
Details

There is considerable flexibility in how you specify the data and statistic.

For the statistic, you may supply a function, or an expression. For example, if data = x, you may specify any of

- statistic = mean
- statistic = mean(x)
- statistic = mean(data)

If data is a data frame, the expression may refer to columns in the data frame, e.g.

- statistic = mean(x)
- statistic = mean(myData$x)
- statistic = mean(myData[, "x")

If data is not just the name of an object, e.g. data = subset(myData, age > 17), or if data2 is supplied, then use the name 'data', e.g.

- statistic = colMeans(data)

Value

A list with class "bootstrap", "bootstrap2", "permutationTest", or "permutationTest2", that inherits from "resample", with components:

- observed: the value of the statistic for the original data.
- replicates: a matrix with R rows and p columns.
- n: number of observations in the original data, or vector of length 2 in two-sample problems.
- p: length(observed).
- R: number of replications.
- seed: the value of the seed at the start of sampling.
- call: the matched call.
- statistics: a data frame with p rows, with columns "observed", "mean" (the mean of the replicates), and other columns appropriate to resampling; e.g. the bootstrap objects have columns "SE" and "Bias", while the permutation test objects have "Alternative" and "PValue".

The two-sample versions have an additional component:

- resultsBoth: containing resampling results from each data set. containing two components, the results from resampling each of the two samples. These are bootstrap objects; in the permutationTest2 case they are the result of sampling without replacement.

There are functions for printing and plotting these objects, in particular print, hist, qqnorm, plot (currently the same as hist), quantile.
Description

Call cat, with sep="" and/or newline at end.

Usage

```r
cat0(...)  
cat0n(...)  
catn(...)  
```

Arguments

```r  
...  
R objects, like for cat  
```

Details

`cat0` and `cat0n` call `cat` with sep = "". `catn` and `cat0n` print a final newline.

Value

None (invisible NULL).
CI.percentile

Author(s)

Tim Hesterberg <timhesterberg@gmail.com>,
http://www.timhesterberg.net/bootstrap

See Also

cat, paste0.

Examples

cat("Print this")
# That printed without a final newline.
catn("Print this")
catn("10", "000")

CI.percentile

Bootstrap confidence intervals

Description

Bootstrap confidence intervals - percentile method or t interval.

Usage

CI.percentile(x, probs = c(0.025, 0.975), expand = TRUE, ...)
CI.t(x, probs = c(0.025, 0.975))
CI.bca(x, probs = c(0.025, 0.975), expand = TRUE, L = NULL, ...)
CI.bootstrapT(x, probs = c(0.025, 0.975))

Arguments

x a bootstrap or bootstrap object.
probs probability values, between 0 and 1. The default vector c(0.025, 0.975) gives a 95% two-sided interval.
expand logical, if TRUE then use modified percentiles for better small-sample accuracy.
... additional arguments to pass to quantile.resample and quantile.
L vector of length n, empirical influence function values. If not supplied this is computed using jackknife.

Details

CI.bootstrapT assumes the first dimension of the statistic is an estimate, and the second is proportional to a SE for the estimate. E.g. for bootstrapping the mean, they could be the mean and s. This is subject to change.

CI.bca and CI.bootstrapT currently only support a single sample.
Value

A matrix with one column for each value in `probs` and one row for each statistic.

Author(s)

Tim Hesterberg <timhesterberg@gmail.com>,
http://www.timhesterberg.net/bootstrap

References


See Also

`bootstrap`, `bootstrap2`, `ExpandProbs` (for the expanded intervals).

Examples

```r
# See full set of examples in resample-package, including different ways to call all four functions depending on the structure of the data.
data(Verizon)
CLEC <- with(Verizon, Time[Group == "CLEC"])
bootC <- bootstrap(CLEC, mean, seed = 0)
bootC2 <- bootstrap(CLEC, c(mean = mean(CLEC), sd = sd(CLEC)), seed = 0)
CI.percentile(bootC)
CI.t(bootC)
CI.bca(bootC)
CI.bootstrapT(bootC2)

# End(Not run)
```

---

**colVars**

*Column variances and standard deviations for matrices.*

Description

Quick and dirty function for column variances and standard deviations.

Usage

```r
colVars(x, na.rm = FALSE)
colStdevs(x, ...)
```
Arguments

- **x**: data frame, matrix, or vector. These versions do not support higher-dimensional arrays.
- **na.rm**: logical. Should missing values (including NaN) be omitted from the calculations?
- **...**: other arguments passed to `colVars`.

Value

A numeric or complex array of suitable size, or a vector if the result is one-dimensional. The dimnames (or names for a vector result) are taken from the original array.

Note

There are better versions of these functions in the aggregate package [http://www.timhesterberg.net/r-packages](http://www.timhesterberg.net/r-packages).

Author(s)

Tim Hesterberg <timhesterberg@gmail.com>, [http://www.timhesterberg.net/](http://www.timhesterberg.net/)

See Also

colSums, var, sd.

Examples

```r
x <- matrix(rnorm(12), 4)
colVars(x)
colStdevs(x)
```

Description

Deprecated functions

Usage

- `limits.percentile(...)`
- `limits.t(...)`
- `limits.bootstrapT(...)`

Arguments

- **...**: arguments to pass to the replacement functions.
Details
These "limits.*" functions have been renamed "CI.*".

Value
See the replacement functions.

Author(s)
Tim Hesterberg <timhesterberg@gmail.com>, http://www.timhesterberg.net/bootstrap

See Also
CI.percentile, CI.t, CI.bootstrapT.

Examples
```r
## Not run:
# See full set of examples in resample-package, including different
# ways to call all four functions depending on the structure of the data.
data(Verizon)
CLEC <- with(Verizon, Time[Group == "CLEC"])
bootC <- bootstrap(CLEC, mean, seed = 0)
bootC2 <- bootstrap(CLEC, c(mean = mean(CLEC), sd = sd(CLEC)), seed = 0)
limits.percentile(bootC)
limits.t(bootC)
limits.bootstrapT(bootC2)

## End(Not run)
```

---

**ExpandProbs**

*Calculate modified probabilities for more accurate confidence intervals*

**Description**
Compute modified quantiles levels, for more accurate confidence intervals. Using these levels gives closer to desired coverage.

**Usage**
```r
ExpandProbs(probs, n)
```

**Arguments**
- `probs` vector of numerical values between 0 and 1.
- `n` number of observations.
ExpandProbs

Details

Bootstrap percentile confidence interval for a sample mean correspond roughly to

\[ \bar{x} \pm z_\alpha \hat{\sigma} \]

instead of

\[ \bar{x} \pm t_{\alpha, n-1} s \]

where

\[ \hat{\sigma} = \sqrt{\frac{(n-1)}{n}} s \]

is like \( s \) but computed using a divisor of \( n \) instead of \( n-1 \). Similarly for other statistics, the bootstrap percentile interval is too narrow, typically by roughly the same proportion.

This function finds modified probability levels \( \text{probs2} \), such that

\[ z_{\text{probs2}} \sqrt{\frac{(n-1)}{n}} = t_{\text{probs, n-1}} \]

\( z_{\text{probs2}} \sqrt{(n-1)/n} = t_{\text{probs, n-1}} \) so that for symmetric data, the bootstrap percentile interval approximately matches the usual \( t \) confidence interval.

Value

A vector like \( \text{probs} \), but with values closer to 0 and 1.

Author(s)

Tim Hesterberg <timhesterberg@gmail.com>,
http://www.timhesterberg.net/bootstrap

References


See Also

CI.percentile, CI.bca,

Examples

```r
probs <- c(0.025, 0.975)
n <- c(5, 10, 20, 40, 100, 200, 1000)
outer(probs, n, ExpandProbs)
```
**IfElse**

*Conditional Data Selection*

**Description**

This is equivalent to `{if(test) yes else no}`. The advantages of using this function are better formatting, and a more natural syntax when the result is being assigned; see examples below.

With 5 arguments, this is equivalent to `{if(test1) yes else if(test2) u else v}` (where arguments are given by name, not position).

**Usage**

`IfElse(test, yes, no, ...)`

**Arguments**

- **test**: logical value; if TRUE return yes.
- **yes**: any object; this is returned if test is TRUE.
- **no**: normally any object; this is returned if test is FALSE. If there are more than three arguments this should be logical.
- **...**: there should be 3, 5, 7, etc. arguments to this function; arguments 1, 3, 5, etc. should be logical values; the other arguments (even numbered, and last) are objects that may be returned.

**Details**

`test` should be a scalar logical, and only one of `yes` or `no` is evaluated, depending on whether `test = TRUE` or `test = FALSE`, and `yes` and `no` may be any objects. In contrast, for `ifelse`, `test` is normally a vector, both `yes` and `no` are evaluated, even if not used, and `yes` and `no` are vectors the same length as `test`.

**Value**

with three arguments, one of `yes` or `no`. With k arguments, one of arguments 2, 4, ..., k-1, k.

**Author(s)**

Tim Hesterberg <timhesterberg@gmail.com>,
[http://www.timhesterberg.net/bootstrap](http://www.timhesterberg.net/bootstrap)

**See Also**

`ifelse, if`.  

---

**ifelse**

*Conditional Data Selection*

**Description**

This is equivalent to `{if(test) yes else no}`. The advantages of using this function are better formatting, and a more natural syntax when the result is being assigned; see examples below.

With 5 arguments, this is equivalent to `{if(test1) yes else if(test2) u else v}` (where arguments are given by name, not position).

**Usage**

`ifelse(test, yes, no, ...)`

**Arguments**

- **test**: logical value; if TRUE return yes.
- **yes**: any object; this is returned if test is TRUE.
- **no**: normally any object; this is returned if test is FALSE. If there are more than three arguments this should be logical.
- **...**: there should be 3, 5, 7, etc. arguments to this function; arguments 1, 3, 5, etc. should be logical values; the other arguments (even numbered, and last) are objects that may be returned.

**Details**

`test` should be a scalar logical, and only one of `yes` or `no` is evaluated, depending on whether `test = TRUE` or `test = FALSE`, and `yes` and `no` may be any objects. In contrast, for `ifelse`, `test` is normally a vector, both `yes` and `no` are evaluated, even if not used, and `yes` and `no` are vectors the same length as `test`.

**Value**

with three arguments, one of `yes` or `no`. With k arguments, one of arguments 2, 4, ..., k-1, k.

**Author(s)**

Tim Hesterberg <timhesterberg@gmail.com>,
[http://www.timhesterberg.net/bootstrap](http://www.timhesterberg.net/bootstrap)

**See Also**

`ifelse, if`.
**Examples**

```r
IfElse(TRUE, "cat", "dog")
IfElse(FALSE, "one", TRUE, "two", "three")
IfElse(FALSE, "one", FALSE, "two", "three")
```

---

**Description**

Basic resampling. Supply the data and statistic to resample.

**Usage**

```r
jackknife(data, statistic, args.stat = NULL,
          label = NULL, statisticNames = NULL, trace = FALSE)
```

**Arguments**

- `data`: vector, matrix, or data frame.
- `statistic`: a function, or expression (e.g. `mean(myData, trim = .2)`).
- `args.stat`: a list of additional arguments to pass to `statistic`, if it is a function.
- `label`: used for labeling plots (in a future version).
- `statisticNames`: a character vector the same length as the vector returned by `statistic`.
- `trace`: logical, if TRUE an indication of progress is printed.

**Value**

A list with class "jackknife" that inherits from "resample", with components:

- `observed`: the value of the statistic for the original data.
- `replicates`: a matrix with R rows and p columns.
- `n`: number of observations in the original data, or vector of length 2 in two-sample problems.
- `p`: `length(observed)`.
- `R`: number of replications.
- `seed`: the value of the seed at the start of sampling.
- `call`: the matched call.
- `statistics`: a data frame with p rows, with columns "observed", "mean" (the mean of the replicates), and other columns appropriate to resampling; e.g. the bootstrap objects have columns "SE" and "Bias", while the permutation test objects have "Alternative" and "PValue".

There are functions for printing and plotting these objects, in particular `print`, `plot`, `hist`, `qqnorm`, `quantile`.
Note

The current version only handles a single sample.

Author(s)

Tim Hesterberg <timhesterberg@gmail.com>,
http://www.timhesterberg.net/bootstrap

See Also

resample-package.

Examples

### Not run:
# See full set of examples in resample-package
data(Verizon)
CLEC <- with(Verizon, Time[Group == "CLEC"])
jackknife(CLEC, mean)

### End(Not run)

print.resample  Methods for common generic functions for resample objects

Description

Methods for common generic functions. The methods operate primarily on the replicates (resampled statistics).

Usage

```r
## S3 method for class 'resample'
print(x, ...)
## S3 method for class 'resample'
hist(x, ..., resampleColumns = 1:x$p, xlim = NULL,
    xlab = NULL, main = "", col = "blue", border = 0,
    breaks = "FD", showObserved = TRUE,
    legend = TRUE, args.legend = NULL)
## S3 method for class 'resample'
plot(x, ...)
## S3 method for class 'resample'
qqnorm(y, ..., resampleColumns = 1:y$p, ylab = NULL,
    pch = if(y$R < 100) 1 else ".")
## S3 method for class 'resample'
quantile(x, ...)
```
Arguments

- `x`, `y`: a "resample" object, usually produced by one of `bootstrap`, `bootstrap2`, `permutationTest`, or `permutationTest2`.
- `...`: additional arguments passed to the corresponding generic function. For `plot.resample`, these are passed to `hist.resample`.
- `resampleColumns`: integer subscripts, or names of statistics. When a statistic is a vector, `resampleColumns` may be used to select which resampling distributions to plot.
- `xlim`: limits for the x axis.
- `xlab`, `ylab`: x and y axis labels.
- `main`: main title
- `col`: color used to fill bars, see `hist`.
- `border`: color of the order around the bars, see `hist`.
- `breaks`: method for computing breaks, see `hist`.
- `showObserved`: logical, if `TRUE` then vertical lines are shown at the observed statistic and mean of the bootstrap replicates.
- `legend`: logical, if `TRUE` a legend is added. Not used if `showObserved = FALSE`.
- `args.legend`: `NULL` or a list of arguments to pass to `legend`.
- `pch`: plotting character, see `par`.

Details

- `hist.resample` displays a histogram overlaid with a density plot, with the observed value of the statistic indicated.
- `plot.resample` currently just calls `hist.resample`.

Value

For `quantile.resample`, a matrix with one row for each statistic and one column for each value in `probs`. This uses `type=6` when calling `quantile`, for wider (more accurate) quantiles than the usual default.

The other functions are not called for their return values.

Author(s)

Tim Hesterberg <timhesterberg@gmail.com>,
http://www.timhesterberg.net/bootstrap

See Also

- `resample-package`, `bootstrap`, `bootstrap2`, `jackknife`, `permutationTest`,
- `permutationTest2`, `quantile`.
Examples

```r
## Not run:
# See full set of examples in resample-package
data(Verizon)
CLEC <- with(Verizon, Time[Group == "CLEC"])
bootC <- bootstrap(CLEC, mean, seed = 0)
print(bootC)
hist(bootC)
qnorm(bootC)
quantile(bootC, probs = c(.25, .975))
# That is the percentile interval with expand = FALSE
CI.percentile(bootC)

## End(Not run)
```

Quantile

*Compute quantiles using type = 6*

**Description**

Front end to `quantile`, using type = 6 (appropriate for resampling)

**Usage**

```r
Quantile(x, ..., type = 6)
```

**Arguments**

- `x`: resample object, numerical object, or other object with a method for `quantile`
- `...`: Other arguments passed to `quantile`
- `type`: With type=6 and 99 observations, the k% quantile is the k'th smallest observation; this corresponds to equal probability above the largest observation, below the smallest observation, and between each pair of adjacent observations.

**Details**

This is a front end to `quantile`.

**Value**

A vector or matrix of quantiles.

**Author(s)**

Tim Hesterberg <timhesterberg@gmail.com>,
[http://www.timhesterberg.net/bootstrap](http://www.timhesterberg.net/bootstrap)
resample

See Also
quantile

Examples
quantile(1:9, .2)
Quantile(1:9, .2)

resample Nonparametric resampling

Description
This function is called by bootstrap and other resampling functions to actually perform resampling, but may also be called directly.

Usage

resample(data, resampleFun, sampler, R = 10000, seed = NULL,
statisticNames = NULL, block.size = 100,
trace = FALSE, ..., observedIndices = 1:n,
call = match.call())

Arguments
data vector, matrix, or data frame.
resampleFun a function with argument data and ii, that calculates a statistic of interest for data[ii] or data[ii, , drop=FALSE], for a vector or matrix, respectively.
sampler a function like samp.bootstrap or samp.permute.
R number of resamples.
seed old value of .Random.seed, or argument to set.seed.
statisticNames a character vector the same length as the vector returned by statistic.
block.size integer. The R replicates are done this many at a time.
trace logical, if TRUE an indication of progress is printed.
... addition arguments passed to sampler.
observedIndices integer vector of indices, used for calculating the observed value. When this is called by bootstrap2 or permutationTest2, those should be indices corresponding to one sample in a merged data set.
call typically the call to bootstrap or another function that calls resample. This may be a character string, e.g. when called from bootstrap2.
Details

This is called by `bootstrap`, `bootstrap2`, `permutationTest`, and `permutationTest2` to actually perform resampling. The results are passed back to the calling function, which may add additional components and a class, which inherits from "resample".

This may also be called directly. In contrast to the other functions, where you have flexibility in how you specify the statistic, here `resampleFun` must be a function.

Value

an object of class "resample"; this is a list with components:

- `observed`: the observed statistic, length `p`.
- `replicates`: a matrix with `R` rows and `p` columns.
- `n`: number of observations
- `p`: the length of the statistic returned by `resampleFun`.
- `R`: number of resamples.
- `seed`: the value of `seed` when this function is called.

Author(s)

Tim Hesterberg <timhesterberg@gmail.com>,
http://www.timhesterberg.net/bootstrap

See Also

`bootstrap,bootstrap2,permutationTest,permutationTest2,samp.bootstrap,samp.permute.`

For an overview of all functions in the package, see `resample-package`.

Examples

```r
## Not run:
# See full set of examples in resample-package, including different
# ways to call all the functions depending on the structure of the data.
data(Verizon)
CLEC <- with(Verizon, Time[Group == "CLEC"])
bootC <- bootstrap(CLEC, mean, seed = 0)
bootC

## End(Not run)
```
resample-data  

Data sets for resampling examples

Description
Data sets for use in examples.

Details
TV has measurements of minutes of commercials per half-hour, for "Basic" and "Extended" (extra-cost) cable TV stations.

Verizon has repair times, with two groups, CLEC and ILEC, customers of the "Competitive" and "Incumbent" local exchange carriers.

DATA SETS
TV 10 observations: Time, Cable Verizon 1687 observations: Time, Group

Source
The TV and Verizon datasets are used in What Teachers Should Know about the Bootstrap: Resampling in the Undergraduate Statistics Curriculum

References

See Also
See resample-package for an overview of resampling functions.

Examples
```r
## Not run:
data(TV); summary(TV)
Basic <- with(TV, Time[Cable == "Basic"])
Extended <- with(TV, Time[Cable == "Extended"])

data(Verizon); summary(Verizon)
ILEC <- with(Verizon, Time[Group == "ILEC"])
CLEC <- with(Verizon, Time[Group == "CLEC"])

## End(Not run)
```
samp.bootstrap  
*Generate indices for resampling*

**Description**

Generate indices for resampling.

**Usage**

```r
samp.bootstrap(n, R, size = n - reduceSize, reduceSize = 0)
samp.permute(n, R, size = n - reduceSize, reduceSize = 0,
groupSizes = NULL, returnGroup = NULL)
```

**Arguments**

- `n` sample size. For two-sample permutation tests, this is the sum of the two sample sizes.
- `R` number of vectors of indices to produce.
- `size` size of samples to produce. For example, to do "what-if" analyses, to estimate the variability of a statistic had the data been a different size, you may specify the size.
- `reduceSize` integer; if specified, then size = n - reduceSize (for each sample or stratum). This is an alternate way to specify size. Typically bootstrap standard errors are too small; they correspond to using n in the divisor of the sample variance, rather than n-1. By specifying reduceSize = 1, you can correct for that bias. This is particularly convenient in two-sample problems where the sample sizes differ.
- `groupSizes` NULL, or vector of positive integers that add to n.
- `returnGroup` NULL, or integer from 1 to length(groupSizes). groupSizes and returnGroup must be supplied together; then full permutations are created, but only subsets of size groupSizes[returnGroup] is returned.

**Details**

To obtain disjoint samples without replacement, call this function multiple times, after setting the same random number seed, with the same groupSizes but different values of returnGroup. This is used for two-sample permutation tests.

If groupSizes is supplied then size is ignored.

**Value**

matrix with size rows and R columns (or groupSizes(returnGroup) rows). Each column contains indices for one bootstrap sample, or one permutation.

**Note**

The value passed as R to this function is typically the block.size argument to bootstrap and other resampling functions.
Author(s)

Tim Hesterberg <timhesterberg@gmail.com>,
http://www.timhesterberg.net/bootstrap

References


See Also

resample-package.

Examples

samp.bootstrap(7, 8)
samp.bootstrap(7, 8, size = 6)
samp.bootstrap(7, 8, reduceSize = 1)

# Full permutations
set.seed(0)
samp.permute(7, 8)

# Disjoint samples without replacement = subsets of permutations
set.seed(0)
samp.permute(7, 8, groupSizes = c(2, 5), returnGroup = 1)
set.seed(0)
samp.permute(7, 8, groupSizes = c(2, 5), returnGroup = 2)
Index

* Topic **algebra**
  colVars, 10
* Topic **arith**
  colVars, 10
* Topic **datasets**
  resampleMdata, 21
* Topic **htest**
  bootstrap, 5
  CI.percentile, 9
  deprecated.resample, 11
  ExpandProbs, 12
  print.resample, 16
  resample, 19
  resample-package, 2
  samp.bootstrap, 22
* Topic **nonparametric**
  bootstrap, 5
  CI.percentile, 9
  deprecated.resample, 11
  ExpandProbs, 12
  jackknife, 15
  print.resample, 16
  resample, 19
  resample-package, 2
  samp.bootstrap, 22
* Topic **univar**
  Quantile, 18
* Topic **utilities**
  cat, 8, 9
  IfElse, 14

  bootstrap, 2, 5, 9, 10, 17, 19, 20, 22
  bootstrap2, 2, 10, 17, 19, 20
  bootstrap2 (bootstrap), 5
  cat, 8, 9
  cat0, 8
  cat0n (cat0), 8
  catn (cat0), 8
  CI.bca, 2, 13

  CI.bca (CI.percentile), 9
  CI.bootstrapT, 2, 12
  CI.bootstrapT (CI.percentile), 9
  CI.percentile, 2, 8, 9, 12, 13
  CI.t, 2, 8, 12
  CI.t (CI.percentile), 9
  colStdevs (colVars), 10
  colSums, 11
  colVars, 10
  deprecated.resample, 11
  ExpandProbs, 10, 12
  hist, 17
  hist.resample, 2
  hist.resample (print.resample), 16
  if, 14
  IfElse, 14
  ifelse, 14
  jackknife, 9, 15, 17
  legend, 17
  limits.bootstrapT
    (deprecated.resample), 11
  limits.percentile
    (deprecated.resample), 11
  limits.t (deprecated.resample), 11
  par, 17
  paste0, 9
  permutationTest, 2, 17, 20
  permutationTest (bootstrap), 5
  permutationTest2, 2, 17, 19, 20
  permutationTest2 (bootstrap), 5
  plot.resample, 2
  plot.resample (print.resample), 16
  print.resample, 2, 16
qqnorm.resample, 2
qqnorm.resample(print.resample), 16
quantile, 18
quantile, 9, 17–19
quantile.resample, 2, 9
quantile.resample(print.resample), 16
resample, 2, 19
resample-data, 21
resample-package, 2
samp.bootstrap, 2, 6, 8, 19, 20, 22
samp.permute, 2, 19, 20
samp.permute(samp.bootstrap), 22
sd, 11
TV(resample-data), 21
var, 11
Verizon(resample-data), 21