

# Package ‘RcmdrMisc’

August 13, 2020

**Version** 2.7-1

**Date** 2020-08-12

**Title** R Commander Miscellaneous Functions

**Depends** R (>= 3.5.0), utils, car (>= 3.0-0), sandwich

**Imports** abind, colorspace, Hmisc (>= 4.1-0), MASS, e1071, foreign,  
haven, readstata13, readxl, graphics, grDevices, stats,  
nortest, lattice

**Suggests** boot, datasets, carData

**ByteCompile** yes

**Description** Various statistical, graphics, and data-  
management functions used by the Rcmdr package in the R Commander GUI for R.

**License** GPL (>= 2)

**URL** <https://www.r-project.org>,  
<https://socialsciences.mcmaster.ca/jfox/>

**Author** John Fox [aut, cre],  
Robert Muenchen [ctb],  
Dan Putler [ctb]

**Maintainer** John Fox <jfox@mcmaster.ca>

**Repository** CRAN

**Repository/R-Forge/Project** rcmdr

**Repository/R-Forge/Revision** 1022

**Repository/R-Forge/DateTimeStamp** 2020-08-12 22:29:12

**Date/Publication** 2020-08-13 05:40:02 UTC

**NeedsCompilation** no

## R topics documented:

assignCluster . . . . .	2
Barplot . . . . .	3

binnedCounts	5
binVariable	6
colPercents	7
DeltaMethod	8
discreteCounts	9
discretePlot	10
Dotplot	11
Gumbel	12
Hist	13
indexplot	14
KMeans	15
lineplot	16
mergeRows	17
normalityTest	18
numSummary	19
partial.cor	20
piechart	21
plotBoot	22
plotDistr	23
plotMeans	24
rcorr.adjust	25
readSAS	26
readSPSS	27
readStata	28
readXL	28
reliability	29
repeatedMeasuresPlot	30
reshapeL2W	32
reshapeW2L	34
stepwise	36
summarySandwich	37

## Index 39

---

assignCluster	<i>Append a Cluster Membership Variable to a Dataframe</i>
---------------	--

---

### Description

Correctly creates a cluster membership variable that can be attached to a dataframe when only a subset of the observations in that dataframe were used to create the clustering solution. NAs are assigned to the observations of the original dataframe not used in creating the clustering solution.

### Usage

```
assignCluster(clusterData, origData, clusterVec)
```

**Arguments**

<code>clusterData</code>	The data matrix used in the clustering solution. The data matrix may have only a subset of the observations contained in the original dataframe.
<code>origData</code>	The original dataframe from which the data used in the clustering solution were taken.
<code>clusterVec</code>	An integer variable containing the cluster membership assignments for the observations used in creating the clustering solution. This vector can be created using <code>cutree</code> for clustering solutions generated by <code>hclust</code> or the <code>cluster</code> component of a list object created by <code>kmeans</code> or <code>KMeans</code> .

**Value**

A factor (with integer labels) that indicate the cluster assignment for each observation, with an NA value given to observations not used in the clustering solution.

**Author(s)**

Dan Putler

**See Also**

[hclust](#), [cutree](#), [kmeans](#), [KMeans](#)

**Examples**

```
data(USArrests)
USArrkm3 <- KMeans(USArrests[USArrests$UrbanPop<66, ], centers=3)
assignCluster(USArrests[USArrests$UrbanPop<66, ], USArrests, USArrkm3$cluster)
```

---

Barplot

*Bar Plots*

---

**Description**

Create bar plots for one or two factors scaled by frequency or percentages. In the case of two factors, the bars can be divided (stacked) or plotted in parallel (side-by-side). This function is a front end to [barplot](#) in the **graphics** package.

**Usage**

```
Barplot(x, by, scale = c("frequency", "percent"), conditional=TRUE,
  style = c("divided", "parallel"),
  col = if (missing(by)) "gray" else rainbow_hcl(length(levels(by))),
  xlab = deparse(substitute(x)), legend.title = deparse(substitute(by)),
  ylab = scale, main=NULL, legend.pos = "above", label.bars=FALSE, ...)
```

**Arguments**

x	a factor (or character or logical variable).
by	optionally, a second factor (or character or logical variable).
scale	either "frequency" (the default) or "percent".
conditional	if TRUE then percentages are computed separately for each value of x (i.e., conditional percentages of by within levels of x); if FALSE then total percentages are graphed; ignored if scale="frequency".
style	for two-factor plots, either "divided" (the default) or "parallel".
col	if by is missing, the color for the bars, defaulting to "gray"; otherwise colors for the levels of the by factor in two-factor plots, defaulting to colors provided by <a href="#">rainbow_hcl</a> in the <b>colorspace</b> package.
xlab	an optional character string providing a label for the horizontal axis.
legend.title	an optional character string providing a title for the legend.
ylab	an optional character string providing a label for the vertical axis.
main	an optional main title for the plot.
legend.pos	position of the legend, in a form acceptable to the <a href="#">legend</a> function; the default, "above", puts the legend above the plot.
label.bars	if TRUE (the default is FALSE) show values of frequencies or percents in the bars.
...	arguments to be passed to the <a href="#">barplot</a> function.

**Value**

Invisibly returns the horizontal coordinates of the centers of the bars.

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**

[barplot](#), [legend](#), [rainbow\\_hcl](#)

**Examples**

```
with(Mroz, {
  Barplot(wc)
  Barplot(wc, col="lightblue", label.bars=TRUE)
  Barplot(wc, by=hc)
  Barplot(wc, by=hc, scale="percent", label.bars=TRUE)
  Barplot(wc, by=hc, style="parallel",
    scale="percent", legend.pos="center")
})
```

**Description**

Bins a numeric variable, as for a histogram, and reports the count and percentage in each bin. The computations are done by the [hist](#) function, but no histogram is drawn. If supplied a numeric matrix or data frame, the distribution of each column is printed.

**Usage**

```
binnedCounts(x, breaks="Sturges", round.percents=2,  
             name=deparse(substitute(x)))
```

**Arguments**

`x` a numeric vector, matrix, or data frame.  
`breaks` specification of the breaks between bins, to be passed to the [hist](#) function.  
`round.percents` number of decimal places to round percentages; default is 2.  
`name` name for the variable; only used for vector argument `x`.

**Value**

For a numeric vector, invisibly returns the vector of counts, named with the end-points of the corresponding bins. For a matrix or data frame, invisibly returns NULL.

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**

[hist](#), [discreteCounts](#)

**Examples**

```
with(Prestige, binnedCounts(income))  
binnedCounts(Prestige[, 1:4])
```

---

`binVariable`*Bin a Numeric Variable*

---

**Description**

Create a factor dissecting the range of a numeric variable into bins of equal width, (roughly) equal frequency, or at "natural" cut points. The `cut` function is used to create the factor. `bin.var` is a synonym for `binVariable`, retained for backwards compatibility.

**Usage**

```
binVariable(x, bins = 4, method = c("intervals", "proportions", "natural"),
           labels = FALSE)
```

```
bin.var(...)
```

**Arguments**

<code>x</code>	numeric variable to be binned.
<code>bins</code>	number of bins.
<code>method</code>	one of "intervals" for equal-width bins; "proportions" for equal-count bins; "natural" for cut points between bins to be determined by a k-means clustering.
<code>labels</code>	if FALSE, numeric labels will be used for the factor levels; if NULL, the cut points are used to define labels; otherwise a character vector of level names.
<code>...</code>	arguments to be passed to <code>binVariable</code> .

**Value**

A factor.

**Author(s)**

Dan Putler, slightly modified by John Fox <jfox@mcmaster.ca> with the original author's permission.

**See Also**

[cut](#), [kmeans](#).

**Examples**

```
summary(binVariable(rnorm(100), method="prop", labels=letters[1:4]))
```

---

`colPercents`*Row, Column, and Total Percentage Tables*

---

**Description**

Percentage a matrix or higher-dimensional array of frequency counts by rows, columns, or total frequency.

**Usage**

```
colPercents(tab, digits=1)
rowPercents(tab, digits=1)
totPercents(tab, digits=1)
```

**Arguments**

`tab` a matrix or higher-dimensional array of frequency counts.  
`digits` number of places to the right of the decimal place for percentages.

**Value**

Returns an array of the same size and shape as `tab` percentaged by rows or columns, plus rows or columns of totals and counts, or by the table total.

**Author(s)**

John Fox <jfox@mcmaster.ca>

**Examples**

```
if (require(car)){
  data(Mroz) # from car package
  cat("\n\n column percents:\n")
  print(colPercents(xtabs(~ lfp + wc, data=Mroz)))
  cat("\n\n row percents:\n")
  print(rowPercents(xtabs(~ hc + lfp, data=Mroz)))
  cat("\n\n total percents:\n")
  print(totPercents(xtabs(~ hc + wc, data=Mroz)))
  cat("\n\n three-way table, column percents:\n")
  print(colPercents(xtabs(~ lfp + wc + hc, data=Mroz)))
}
```

## Description

DeltaMethod is a wrapper for the [deltaMethod](#) function in the **car** package. It computes the asymptotic standard error of an arbitrary, usually nonlinear, function of model coefficients, which are named  $b_0$  (if there is an intercept in the model),  $b_1$ ,  $b_2$ , etc., and based on the standard error, a confidence interval based on the normal distribution.

## Usage

```
DeltaMethod(model, g, level = 0.95)
## S3 method for class 'DeltaMethod'
print(x, ...)
```

## Arguments

model	a regression model; see the <a href="#">deltaMethod</a> documentation.
g	the expression — that is, function of the coefficients — to evaluate, as a character string.
level	the confidence level, defaults to 0.95.
x	an object of class "DeltaMethod".
...	optional arguments to pass to print to show the results.

## Value

DeltaMethod returns an objects of class "DeltaMethod", for which a print method is provided.

## Author(s)

John Fox <jfox@mcmaster.ca>

## See Also

[deltaMethod](#) function in the **car** package

## Examples

```
if (require(car)){
  DeltaMethod(lm(prestige ~ income + education, data=Duncan), "b1/b2")
}
```



---

discreteCounts      *Frequency Distributions of Numeric Variables*

---

### Description

Computes the frequency and percentage distribution of a discrete numeric variable or the distributions of the variables in a numeric matrix or data frame.

### Usage

```
discreteCounts(x, round.percents=2, name=deparse(substitute(x)),
  max.values=min(round(2*sqrt(length(x))), round(10*log10(length(x))), 100))
```

### Arguments

x	a discrete numeric vector, matrix, or data frame.
round.percents	number of decimal places to round percentages; default is 2.
name	name for the variable; only used for vector argument x.
max.values	maximum number of unique values (default is the smallest of twice the square root of the number of elements in x, 10 times the log10 of the number of elements, and 100); if exceeded, an error is reported.

### Value

For a numeric vector, invisibly returns the table of counts. For a matrix or data frame, invisibly returns NULL

### Author(s)

John Fox <jfox@mcmaster.ca>

### See Also

[binnedCounts](#)

### Examples

```
set.seed(12345) # for reproducibility
discreteCounts(data.frame(x=rpois(51, 2), y=rpois(51, 10)))
```

---

`discretePlot`*Plot Distribution of Discrete Numeric Variable*

---

**Description**

Plot the distribution of a discrete numeric variable, optionally classified by a factor.

**Usage**

```
discretePlot(x, by, scale = c("frequency", "percent"),
             xlab = deparse(substitute(x)), ylab = scale, main = "")
```

**Arguments**

<code>x</code>	a numeric variable.
<code>by</code>	optionally a factor (or character or logical variable) by which to classify <code>x</code> .
<code>scale</code>	either "frequency" (the default) or "percent".
<code>xlab</code>	optional character string to label the horizontal axis.
<code>ylab</code>	optional character string to label the vertical axis.
<code>main</code>	optional main label for the plot (ignored if the <code>by</code> argument is specified).

**Details**

If the `by` argument is specified, then one plot is produced for each level of `by`; these are arranged vertically and all use the same scale for `x`.

**Value**

Returns NULL invisibly.

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**

[Hist](#), [link{Dotplot}](#).

**Examples**

```
if (require(datasets)){
  data(mtcars)
  mtcars$cyl <- factor(mtcars$cyl)
  with(mtcars, {
    discretePlot(carb)
    discretePlot(carb, scale="percent")
    discretePlot(carb, by=cyl)
  })
}
```

```
    })  
  }
```

---

Dotplot

*Dot Plots*

---

### Description

Dot plot of numeric variable, either using raw values or binned, optionally classified by a factor. Dot plots are useful for visualizing the distribution of a numeric variable in a small data set.

### Usage

```
Dotplot(x, by, bin = FALSE, breaks, xlim,  
        xlab = deparse(substitute(x)))
```

### Arguments

x	a numeric variable.
by	optionally a factor (or character or logical variable) by which to classify x.
bin	if TRUE (the default is FALSE), the values of x are binned, as in a histogram, prior to plotting.
breaks	breaks for the bins, in a form acceptable to the <a href="#">hist</a> function; the default is "Sturges".
xlim	optional 2-element numeric vector giving limits of the horizontal axis.
xlab	optional character string to label horizontal axis.

### Details

If the `by` argument is specified, then one dot plot is produced for each level of `by`; these are arranged vertically and all use the same scale for `x`. An attempt is made to adjust the size of the dots to the space available without making them too big.

### Value

Returns NULL invisibly.

### Author(s)

John Fox <jfox@mcmaster.ca>

### See Also

[hist](#)

**Examples**

```

if (require(car)){
  data(Duncan)
  with(Duncan, {
    Dotplot(education)
    Dotplot(education, bin=TRUE)
    Dotplot(education, by=type)
    Dotplot(education, by=type, bin=TRUE)
  })
}

```

Gumbel

*The Gumbel Distribution***Description**

Density, distribution function, quantile function and random generation for the Gumbel distribution with specified location and scale parameters.

**Usage**

```

dgumbel(x, location = 0, scale = 1)
pgumbel(q, location=0, scale=1, lower.tail=TRUE)
qgumbel(p, location=0, scale=1, lower.tail=TRUE)
rgumbel(n, location=0, scale=1)

```

**Arguments**

x, q	vector of quantiles (values of the variable).
p	vector of probabilities.
n	number of observations. If <code>length(n) &gt; 1</code> , the length is taken to be the number required.
location	location parameter (default 0); potentially a vector.
scale	scale parameter (default 1); potentially a vector.
lower.tail	logical; if TRUE (the default) probabilities and quantiles correspond to $P(X \leq x)$ , if FALSE to $P(X > x)$ .

**Author(s)**

John Fox <jfox@mcmaster.ca>

**References**

See [https://en.wikipedia.org/wiki/Gumbel\\_distribution](https://en.wikipedia.org/wiki/Gumbel_distribution) for details of the Gumbel distribution.

### Examples

```
x <- 100 + 5*c(-Inf, -1, 0, 1, 2, 3, Inf, NA)
dgumbel(x, 100, 5)
pgumbel(x, 100, 5)

p <- c(0, .25, .5, .75, 1, NA)
qgumbel(p, 100, 5)

summary(rgumbel(1e5, 100, 5))
```

---

Hist

*Plot a Histogram*

---

### Description

This function is a wrapper for the [hist](#) function in the base package, permitting percentage scaling of the vertical axis in addition to frequency and density scaling.

### Usage

```
Hist(x, groups, scale=c("frequency", "percent", "density"), xlab=deparse(substitute(x)),
     ylab=scale, main="", breaks = "Sturges", ...)
```

### Arguments

x	a vector of values for which a histogram is to be plotted.
groups	a factor (or character or logical variable) to create histograms by group with common horizontal and vertical scales.
scale	the scaling of the vertical axis: "frequency" (the default), "percent", or "density".
xlab	x-axis label, defaults to name of variable.
ylab	y-axis label, defaults to value of scale.
main	main title for graph, defaults to empty.
breaks	see the breaks argument for <a href="#">hist</a> .
...	arguments to be passed to <a href="#">hist</a> .

### Value

This function is primarily called for its side effect — plotting a histogram or histograms — but it also invisibly returns an object of class [hist](#) or a list of [hist](#) objects.

### Author(s)

John Fox <jfox@mcmaster.ca>

### See Also

[hist](#)

**Examples**

```
data(Prestige, package="car")
Hist(Prestige$income, scale="percent")
with(Prestige, Hist(income, groups=type))
```

indexplot

*Index Plots***Description**

Index plots with point identification.

**Usage**

```
indexplot(x, groups, labels = seq_along(x), id.method = "y", type = "h",
          id.n = 0, ylab, legend="topright", title, col=palette(), ...)
```

**Arguments**

x	a numeric variable, a matrix whose columns are numeric variables, or a numeric data frame; if x is a matrix or data frame, plots vertically aligned index plots for the columns.
labels	point labels; if x is a data frame, defaults to the row names of x, otherwise to the case index.
groups	an optional grouping variable, typically a factor (or character or logical variable).
id.method	method for identifying points; see <a href="#">showLabels</a> .
type	to be passed to <a href="#">plot</a> .
id.n	number of points to identify; see <a href="#">showLabels</a> .
ylab	label for vertical axis; if missing, will be constructed from x; for a data frame, defaults to the column names.
legend	keyword (see <code>link[graphics]legend</code> ) giving location of the legend if groups are specified; if <code>legend=FALSE</code> , the legend is suppressed.
title	title for the legend; may normally be omitted.
col	vector of colors for the groups.
...	to be passed to <code>plot</code> .

**Value**

Returns labelled indices of identified points or (invisibly) NULL if no points are identified or if there are multiple variables with some missing data.

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**

[showLabels](#), [plot.default](#)

**Examples**

```
if (require("car")){
  with(Prestige, indexplot(income, id.n=2, labels=rownames(Prestige)))
  indexplot(Prestige[, c("income", "education", "prestige")],
            groups = Prestige$type, id.n=2)
}
```

---

KMeans

*K-Means Clustering Using Multiple Random Seeds*


---

**Description**

Finds a number of k-means clustering solutions using R's `kmeans` function, and selects as the final solution the one that has the minimum total within-cluster sum of squared distances.

**Usage**

```
KMeans(x, centers, iter.max=10, num.seeds=10)
```

**Arguments**

<code>x</code>	A numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a dataframe with all numeric columns).
<code>centers</code>	The number of clusters in the solution.
<code>iter.max</code>	The maximum number of iterations allowed.
<code>num.seeds</code>	The number of different starting random seeds to use. Each random seed results in a different k-means solution.

**Value**

A list with components:

<code>cluster</code>	A vector of integers indicating the cluster to which each point is allocated.
<code>centers</code>	A matrix of cluster centres (centroids).
<code>withinss</code>	The within-cluster sum of squares for each cluster.
<code>tot.withinss</code>	The within-cluster sum of squares summed across clusters.
<code>betweenss</code>	The between-cluster sum of squared distances.
<code>size</code>	The number of points in each cluster.

**Author(s)**

Dan Putler

**See Also**

[kmeans](#)

**Examples**

```
data(USArrests)
KMeans(USArrests, centers=3, iter.max=5, num.seeds=5)
```

---

lineplot

*Plot a one or more lines.*

---

**Description**

This function plots lines for one or more variables against another variable — typically time series against time.

**Usage**

```
lineplot(x, ..., legend)
```

**Arguments**

x	variable giving horizontal coordinates.
...	one or more variables giving vertical coordinates.
legend	plot legend? Default is TRUE if there is more than one variable to plot and FALSE if there is just one.

**Value**

Produces a plot; returns NULL invisibly.

**Author(s)**

John Fox <jfox@mcmaster.ca>

**Examples**

```
if (require("car")){
  data(Bfox)
  Bfox$time <- as.numeric(rownames(Bfox))
  with(Bfox, lineplot(time, menwage, womwage))
}
```



---

`mergeRows`*Function to Merge Rows of Two Data Frames.*

---

**Description**

This function merges two data frames by combining their rows.

**Usage**

```
mergeRows(X, Y, common.only = FALSE, ...)
```

```
## S3 method for class 'data.frame'  
mergeRows(X, Y, common.only = FALSE, ...)
```

**Arguments**

<code>X</code>	First data frame.
<code>Y</code>	Second data frame.
<code>common.only</code>	If TRUE, only variables (columns) common to the two data frame are included in the merged data set; the default is FALSE.
<code>...</code>	Not used.

**Value**

A data frame containing the rows from both input data frames.

**Author(s)**

John Fox

**See Also**

For column merges and more complex merges, see [merge](#).

**Examples**

```
if (require(car)){  
  data(Duncan)  
  D1 <- Duncan[1:20,]  
  D2 <- Duncan[21:45,]  
  D <- mergeRows(D1, D2)  
  print(D)  
  dim(D)  
}
```

---

normalityTest	<i>Normality Tests</i>
---------------	------------------------

---

### Description

Perform one of several tests of normality, either for a variable or for a variable by groups. The `normalityTest` function uses the `shapiro.test` function or one of several functions in the **nortest** package. If tests are done by groups, then adjusted p-values, computed by the Holm method, are also reported (see [p.adjust](#)).

### Usage

```
normalityTest(x, ...)

## S3 method for class 'formula'
normalityTest(formula, test, data, ...)

## Default S3 method:
normalityTest(x,
  test=c("shapiro.test", "ad.test", "cvm.test", "lillie.test",
        "pearson.test", "sf.test"),
  groups, vname, gname, ...)
```

### Arguments

<code>x</code>	numeric vector or formula.
<code>formula</code>	one-sided formula of the form $\sim x$ or two-sided formula of the form $x \sim \text{groups}$ , where $x$ is a numeric variable and <code>groups</code> is a factor.
<code>data</code>	a data frame containing the data for the test.
<code>test</code>	quoted name of the function to perform the test.
<code>groups</code>	optional factor to divide the data into groups.
<code>vname</code>	optional name for the variable; if absent, taken from <code>x</code> .
<code>gname</code>	optional name for the grouping factor; if absent, taken from <code>groups</code> .
<code>...</code>	any arguments to be passed down; the only useful such arguments are for the <a href="#">pearson.test</a> function in the <b>nortest</b> package.

### Value

If testing by groups, the function invisibly returns `NULL`; otherwise it returns an object of class `"htest"`, which normally would be printed.

### Author(s)

John Fox <jfox@mcmaster.ca>

**See Also**

[shapiro.test](#), [ad.test](#), [cvm.test](#), [lillie.test](#), [pearson.test](#), [sf.test](#).

**Examples**

```
data(Prestige, package="car")
with(Prestige, normalityTest(income))
normalityTest(income ~ type, data=Prestige, test="ad.test")
normalityTest(~income, data=Prestige, test="pearson.test", n.classes=5)
```

---

numSummary

*Summary Statistics for Numeric Variables*


---

**Description**

numSummary creates neatly formatted tables of means, standard deviations, coefficients of variation, skewness, kurtosis, and quantiles of numeric variables.

**Usage**

```
numSummary(data,
  statistics=c("mean", "sd", "se(mean)", "IQR",
    "quantiles", "cv", "skewness", "kurtosis"),
  type=c("2", "1", "3"),
  quantiles=c(0, .25, .5, .75, 1), groups)

## S3 method for class 'numSummary'
print(x, ...)
```

**Arguments**

data	a numeric vector, matrix, or data frame.
statistics	any of "mean", "sd", "se(mean)", "quantiles", "cv" (coefficient of variation — sd/mean), "skewness", or "kurtosis", defaulting to c("mean", "sd", "quantiles", "IQR").
type	definition to use in computing skewness and kurtosis; see the <a href="#">skewness</a> and <a href="#">kurtosis</a> functions in the <b>e1071</b> package. The default is "2".
quantiles	quantiles to report; default is c(0, 0.25, 0.5, 0.75, 1).
groups	optional variable, typically a factor, to be used to partition the data.
x	object of class "numSummary" to print.
...	arguments to pass down from the print method.

**Value**

numSummary returns an object of class "numSummary" containing the table of statistics to be reported along with information on missing data, if there are any.

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**

[mean](#), [sd](#), [quantile](#), [skewness](#), [kurtosis](#).

**Examples**

```
if (require("car")){
  data(Prestige)
  Prestige[1, "income"] <- NA
  print(numSummary(Prestige[,c("income", "education")],
    statistics=c("mean", "sd", "quantiles", "cv", "skewness", "kurtosis")))
  print(numSummary(Prestige[,c("income", "education")], groups=Prestige$type))
  remove(Prestige)
}
```

---

partial.cor

*Partial Correlations*

---

**Description**

Computes a matrix of partial correlations between each pair of variables controlling for the others.

**Usage**

```
partial.cor(X, tests=FALSE, use=c("complete.obs", "pairwise.complete.obs"))
```

**Arguments**

X	data matrix.
tests	show two-sided p-value and p-value adjusted for multiple testing by Holm's method for each partial correlation?
use	observations to use to compute partial correlations, default is "complete.obs".

**Value**

Returns the matrix of partial correlations, optionally with adjusted and unadjusted p-values.

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**

[cor](#)

**Examples**

```
data(DavisThin, package="car")
partial.cor(DavisThin)
partial.cor(DavisThin, tests=TRUE)
```

---

piechart

*Draw a Piechart With Percents or Counts in the Labels*

---

**Description**

piechart is a front-end to the standard R [pie](#) function, with the capability of adding percents or counts to the pie-segment labels.

**Usage**

```
piechart(x, scale = c("percent", "frequency", "none"),
         col = rainbow_hcl(nlevels(x)), ...)
```

**Arguments**

x	a factor or other discrete variable; the segments of the pie correspond to the unique values (levels) of x and are proportional to the frequency counts in the various levels.
scale	parenthetical numbers to add to the pie-segment labels; the default is "percent".
col	colors for the segments; the default is provided by the <a href="#">rainbow_hcl</a> function in the <b>colorspace</b> package.
...	further arguments to be passed to <a href="#">pie</a> .

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**

[pie](#), [rainbow\\_hcl](#)

**Examples**

```
with(Duncan, piechart(type))
```

---

`plotBoot`*Plot Bootstrap Distributions*

---

### Description

The function takes an object of class "boot" and creates an array of density estimates for the bootstrap distributions of the parameters.

### Usage

```
plotBoot(object, confint=NULL, ...)
## S3 method for class 'boot'
plotBoot(object, confint=NULL, ...)
```

### Arguments

<code>object</code>	an object of class "boot".
<code>confint</code>	an object of class "confint.boot" (or an ordinary 2-column matrix) containing confidence limits for the parameters in <code>object</code> ; if NULL (the default), these are computed from the first argument, using the defaults for "boot" objects.
<code>...</code>	not used

### Details

Creates an array of adaptive kernel density plots, using `densityPlot` in the `car` package, showing the bootstrap distribution, point estimate, and (optionally) confidence limits for each parameter.

### Value

Invisibly returns the object produced by `densityPlot`.

### Author(s)

John Fox <jfox@mcmaster.ca>

### See Also

[densityPlot](#)

### Examples

```
## Not run:
plotBoot(Boot(lm(prestige ~ income + education + type, data=Duncan)))

## End(Not run)
```

---

plotDistr	<i>Plot a probability density, mass, or distribution function.</i>
-----------	--

---

### Description

This function plots a probability density, mass, or distribution function, adapting the form of the plot as appropriate.

### Usage

```
plotDistr(x, p, discrete=FALSE, cdf=FALSE,
          regions=NULL, col="gray",
          legend=TRUE, legend.pos="topright", ...)
```

### Arguments

x	horizontal coordinates
p	vertical coordinates
discrete	is the random variable discrete?
cdf	is this a cumulative distribution (as opposed to mass) function?
regions, col	for continuous distributions only, if non-NULL, a list of regions to fill with color col; each element of the list is a pair of x values with the minimum and maximum horizontal coordinates of the corresponding region; col may be a single value or a vector.
legend	plot a legend of the regions (default TRUE).
legend.pos	position for the legend (see <a href="#">legend</a> , default "topright").
...	arguments to be passed to plot.

### Value

Produces a plot; returns NULL invisibly.

### Author(s)

John Fox <jfox@mcmaster.ca>

### Examples

```
x <- seq(-4, 4, length=100)
plotDistr(x, dnorm(x), xlab="Z", ylab="p(z)", main="Standard Normal Density")
plotDistr(x, dnorm(x), xlab="Z", ylab="p(z)", main="Standard Normal Density",
          region=list(c(1.96, Inf), c(-Inf, -1.96)), col=c("red", "blue"))
plotDistr(x, dnorm(x), xlab="Z", ylab="p(z)", main="Standard Normal Density",
          region=list(c(qnorm(0), qnorm(.025)), c(qnorm(.975), qnorm(1)))) # same

x <- 0:10
```

```
plotDistr(x, pbinom(x, 10, 0.5), xlab="successes",
          discrete=TRUE, cdf=TRUE,
          main="Binomial Distribution Function, p=0.5, n=10")
```

plotMeans

*Plot Means for One or Two-Way Layout***Description**

Plots cell means for a numeric variable in each category of a factor or in each combination of categories of two factors, optionally along with error bars based on cell standard errors or standard deviations.

**Usage**

```
plotMeans(response, factor1, factor2,
          error.bars = c("se", "sd", "conf.int", "none"),
          level=0.95, xlab=deparse(substitute(factor1)),
          ylab=paste("mean of", deparse(substitute(response))),
          legend.lab=deparse(substitute(factor2)),
          legend.pos=c("farright", "bottomright", "bottom", "bottomleft",
                     "left", "topleft", "top", "topright", "right", "center"),
          main="Plot of Means",
          pch=1:n.levs.2, lty=1:n.levs.2, col=palette(), connect=TRUE, ...)
```

**Arguments**

response	Numeric variable for which means are to be computed.
factor1	Factor defining horizontal axis of the plot.
factor2	If present, factor defining profiles of means
error.bars	If "se", the default, error bars around means give plus or minus one standard error of the mean; if "sd", error bars give plus or minus one standard deviation; if "conf.int", error bars give a confidence interval around each mean; if "none", error bars are suppressed.
level	level of confidence for confidence intervals; default is .95
xlab	Label for horizontal axis.
ylab	Label for vertical axis.
legend.lab	Label for legend.
legend.pos	Position of legend; if "farright" (the default), extra space is left at the right of the plot.
main	Label for the graph.
pch	Plotting characters for profiles of means.
lty	Line types for profiles of means.
col	Colours for profiles of means
connect	connect profiles of means, default TRUE.
...	arguments to be passed to plot.



**Value**

The function invisibly returns NULL.

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**

[interaction.plot](#)

**Examples**

```
if (require(car)){
  data(Moore)
  with(Moore, plotMeans(conformity, fcategory, partner.status, ylim=c(0, 25)))
}
```

---

rcorr.adjust

*Compute Pearson or Spearman Correlations with p-Values*

---

**Description**

This function uses the [rcorr](#) function in the **Hmisc** package to compute matrices of Pearson or Spearman correlations along with the pairwise p-values among the correlations. The p-values are corrected for multiple inference using Holm's method (see [p.adjust](#)). Observations are filtered for missing data, and only complete observations are used.

**Usage**

```
rcorr.adjust(x, type = c("pearson", "spearman"),
use=c("complete.obs", "pairwise.complete.obs"))

## S3 method for class 'rcorr.adjust'
print(x, ...)
```

**Arguments**

x	a numeric matrix or data frame, or an object of class "rcorr.adjust" to be printed.
type	"pearson" or "spearman", depending upon the type of correlations desired; the default is "pearson".
use	how to handle missing data: "complete.obs", the default, use only complete cases; "pairwise.complete.obs", use all cases with valid data for each pair.
...	not used.

**Value**

Returns an object of class "rcorr.adjust", which is normally just printed.

**Author(s)**

John Fox, adapting code from Robert A. Muenchen.

**See Also**

[rcorr](#), [p.adjust](#).

**Examples**

```
if (require(car)){
  data(Mroz)
  print(rcorr.adjust(Mroz[,c("k5", "k618", "age", "lwg", "inc")]))
  print(rcorr.adjust(Mroz[,c("k5", "k618", "age", "lwg", "inc")], type="spearman"))
}
```

---

readSAS

*Read a SAS b7dat Data Set*

---

**Description**

readSAS reads a SAS "b7dat" data set, stored in a file of type .sas7bdat, into an R data frame; it provides a front end to the [read\\_sas](#) function in the **haven** package.

**Usage**

```
readSAS(file, rownames=FALSE, stringsAsFactors=FALSE)
```

**Arguments**

**file** path to a SAS b7dat file.

**rownames** if TRUE (the default is FALSE), the first column in the data set contains row names (which must be unique—i.e., no duplicates).

**stringsAsFactors** if TRUE (the default is FALSE) then columns containing character data are converted to factors.

**Value**

a data frame

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**[read\\_sas](#)

---

`readSPSS`*Read an SPSS Data Set*

---

**Description**

`readSPSS` reads an SPSS data set, stored in a file of type `.sav` or `.por`, into an R data frame; it provides a front end to the [read\\_spss](#) function in the **haven** package and the [read.spss](#) function in the **foreign** package.

**Usage**

```
readSPSS(file, rownames=FALSE, stringsAsFactors=FALSE,  
         tolower=TRUE, use.value.labels=TRUE, use.haven=!por)
```

**Arguments**

<code>file</code>	path to an SPSS <code>.sav</code> or <code>.por</code> file.
<code>rownames</code>	if TRUE (the default is FALSE), the first column in the data set contains row names, which should be unique.
<code>stringsAsFactors</code>	if TRUE (the default is FALSE) then columns containing character data are converted to factors and factors are created from SPSS value labels.
<code>tolower</code>	change variable names to lowercase, default TRUE.
<code>use.value.labels</code>	if TRUE, the default, variables with value labels in the SPSS data set will become either factors or character variables (depending on the <code>stringsAsFactors</code> argument) with the value labels as their levels or values. As for <a href="#">read.spss</a> , this is only done if there are at least as many labels as values of the variable (and values without a matching label are returned as NA).
<code>use.haven</code>	use <a href="#">read_spss</a> from the <b>haven</b> package to read the file, in preference to <a href="#">read.spss</a> from the <b>foreign</b> package; the default is TRUE for a <code>.sav</code> file and FALSE for a <code>.por</code> file.

**Value**

a data frame

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**

[read\\_spss](#), [read.spss](#)

---

readStata	<i>Read a Stata Data Set</i>
-----------	------------------------------

---

**Description**

readStata reads a Stata data set, stored in a file of type `.dta`, into an R data frame; it provides a front end to the [read.dta13](#) function in the **readstata13** package.

**Usage**

```
readStata(file, rownames=FALSE, stringsAsFactors=FALSE, convert.dates=TRUE)
```

**Arguments**

file	path to a Stata <code>.dta</code> file.
rownames	if TRUE (the default is FALSE), the first column in the data set contains row names, which should be unique.
stringsAsFactors	if TRUE (the default is FALSE) then columns containing character data are converted to factors and factors are created from Stata value labels.
convert.dates	if TRUE (the default) then Stata dates are converted to R dates.

**Value**

a data frame

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**

[read.dta13](#)

---

readXL	<i>Read an Excel File</i>
--------	---------------------------

---

**Description**

readXL reads an Excel file, either of type `.xls` or `.xlsx` into an R data frame; it provides a front end to the [read\\_excel](#) function in the **readxl** package. [excel\\_sheets](#) is re-exported from the **readxl** package and reports the names of spreadsheets in an Excel file.

**Usage**

```
readXL(file, rownames = FALSE, header = TRUE, na = "", sheet = 1,  
stringsAsFactors = FALSE)
```

```
excel_sheets(path)
```

**Arguments**

file, path	path to an Excel file.
rownames	if TRUE (the default is FALSE), the first column in the spreadsheet contains row names (which must be unique—i.e., no duplicates).
header	if TRUE (the default), the first row in the spreadsheet contains column (variable) names.
na	character string denoting missing data; the default is the empty string, "".
sheet	number of the spreadsheet in the file containing the data to be read; the default is 1.
stringsAsFactors	if TRUE (the default is FALSE) then columns containing character data are converted to factors.

**Value**

a data frame

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**

[read\\_excel](#), [excel\\_sheets](#)

---

reliability

*Reliability of a Composite Scale*

---

**Description**

Calculates Cronbach's alpha and standardized alpha (lower bounds on reliability) for a composite (summated-rating) scale. Standardized alpha is for the sum of the standardized items. In addition, the function calculates alpha and standardized alpha for the scale with each item deleted in turn, and computes the correlation between each item and the sum of the other items.

**Usage**

```
reliability(S)
```

```
## S3 method for class 'reliability'  
print(x, digits=4, ...)
```

**Arguments**

S	the covariance matrix of the items; normally, there should be at least 3 items and certainly no fewer than 2.
x	reliability object to be printed.
digits	number of decimal places.
...	not used: for compatibility with the print generic."

**Value**

an object of class reliability, which normally would be printed.

**Author(s)**

John Fox <jfox@mcmaster.ca>

**References**

N. Cliff (1986) Psychological testing theory. Pp. 343–349 in S. Kotz and N. Johnson, eds., *Encyclopedia of Statistical Sciences*, Vol. 7. Wiley.

**See Also**

[cov](#)

**Examples**

```
if (require(car)){
  data(DavisThin)
  reliability(cov(DavisThin))
}
```

---

repeatedMeasuresPlot *Plot Means for Repeated-Measures ANOVA Designs*

---

**Description**

Creates a means plot for a repeated-measures ANOVA design with one or two within-subjects factor and zero or more between-subjects factors, for data in "wide" format.

**Usage**

```
repeatedMeasuresPlot(data, within, within.names, within.levels,
  between.names = NULL, response.name = "score", trace, xvar,
  pch=15:25, lty=1:6, col=palette()[-1],
  plot.means=TRUE, print.tables = FALSE)
```

**Arguments**

<code>data</code>	a data frame in wide format.
<code>within</code>	a character vector with the names of the data columns containing the repeated measures.
<code>within.names</code>	a character vector with one or two elements, of names of the within-subjects factor(s).
<code>within.levels</code>	a named list whose elements are character vectors of level names for the within-subjects factors, with names corresponding to the names of the within-subjects factors; the product of the numbers of levels should be equal to the number of repeated-measures columns in <code>within</code> .
<code>between.names</code>	a column vector of names of the between-subjects factors (if any).
<code>response.name</code>	optional quoted name for the response variable, defaults to "score".
<code>trace</code>	optional quoted name of the (either within- or between-subjects) factor to define profiles of means in each panel of the graph; the default is the within-subjects factor with the smaller number of levels, if there are two, or not used if there is one.
<code>xvar</code>	optional quoted name of the factor to define the horizontal axis of each panel; the default is the within-subjects factor with the larger number of levels.
<code>pch, lty</code>	vectors of symbol and line-type numbers to use for the profiles of means (i.e., levels of the <code>trace</code> factor); for the meaning of the defaults, see <a href="#">points</a> and <a href="#">par</a> .
<code>col</code>	vector of colors for the profiles of means; the default is given by <code>palette()</code> , starting at the second color.
<code>plot.means</code>	if TRUE (the default), draw a plot of means by the factors.
<code>print.tables</code>	if TRUE (the default is FALSE), print tables of means and standard deviations of the response by the factors.

**Value**

A "trellis" object, which normally is just "printed" (i.e., plotted).

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**

[Anova](#), [OBrienKaiser](#)

**Examples**

```
if (require(carData)){
repeatedMeasuresPlot(
  data=OBrienKaiser,
  within=c("pre.1", "pre.2", "pre.3", "pre.4", "pre.5",
           "post.1", "post.2", "post.3", "post.4", "post.5",
           "fup.1", "fup.2", "fup.3", "fup.4", "fup.5"),
```

```

within.names=c("phase", "hour"),
within.levels=list(phase=c("pre", "post", "fup"),
                  hour = c("1", "2", "3", "4", "5")),
between.names=c("gender", "treatment"),
response.name="improvement",
print.tables=TRUE)
}

if (require(carData)){
repeatedMeasuresPlot(data=OBrienKaiser,
  within=c("pre.1", "pre.2", "pre.3", "pre.4", "pre.5",
          "post.1", "post.2", "post.3", "post.4", "post.5",
          "fup.1", "fup.2", "fup.3", "fup.4", "fup.5"),
  within.names=c("phase", "hour"),
  within.levels=list(phase=c("pre", "post", "fup"),
                    hour = c("1", "2", "3", "4", "5")),
  between.names=c("gender", "treatment"),
  trace="gender") # note that gender is between subjects
}

if (require(carData)){
repeatedMeasuresPlot(
  data=OBrienKaiser,
  within=c("fup.1", "fup.2", "fup.3", "fup.4", "fup.5"),
  within.names="hour",
  within.levels=list(hour = c("1", "2", "3", "4", "5")),
  between.names=c("treatment", "gender"),
  response.name="improvement")
}

```

---

 reshapeL2W

*Reshape Repeated-Measures Data from Long to Wide Format*


---

### Description

A simple front-end to the standard R [reshape](#) function. The data are assumed to be in "long" format, with several rows for each subject.

### Usage

```
reshapeL2W(data, within, id, varying, ignore)
```

### Arguments

data	a data frame in long format.
within	a character vector of names of the within-subjects factors in the long form of the data; there must be at least one within-subjects factor.
id	the (character) name of the variable representing the subject identifier in the long form of the data set; that is, rows with the same id belong to the same subject.



varying	a character vector of names of the occasion-varying variables in the long form of the data; there must be at least one such variable, and typically there will be just one, an occasion-varying response variable.
ignore	an optional character vector of names of variables in the long form of the data to exclude from the wide data set.

### Details

Between-subjects variables don't vary by occasions for each subject. Variables that aren't listed explicitly in the arguments to the function are assumed to be between-subjects variables, and a warning is printed if their values aren't invariant for each subject (see the `ignore` argument).

Within-subjects factors vary by occasions for each subject, and it is assumed that the within-subjects design is regular, completely crossed, and balanced, so that the same combinations of within-subjects factors are observed for each subject.

Occasion-varying variables, as their name implies, (potentially) vary by occasions for each subject, and include one or more "response" variables, possibly along with occasion-varying covariates; these variables can be factors as well as numeric variables.

The data are reshaped so that there is one row per subject, with columns for the between-subjects variables, and each occasion-varying variable as multiple columns representing the combinations of levels of the within-subjects factors. The names of the columns for the occasion-varying variables are composed from the combinations of levels of the within-subjects factors and from the names of the occasion-varying variables (if there is more than one). If a subject in the long form of the data set lacks any combination of levels of within-subjects factors, he or she is excluded (with a warning) from the wide form of the data.

### Value

a data frame in "wide" format, with one row for each subject, columns representing the between-subjects factors, and columns for the occasion-varying variable(s) for each combination of within-subjects factors.

### Author(s)

John Fox <jfox@mcmaster.ca>

### See Also

[reshape](#), [OBrienKaiser](#), [OBrienKaiserLong](#).

### Examples

```
if (require(carData)){
  OBW <- reshapeL2W(OBrienKaiserLong,
                   within=c("phase", "hour"),
                   id="id", varying="score")
  brief(OBW)
  # should be the same as OBrienKaiser in the carData package:
  all.equal(OBrienKaiser, OBW, check.attributes=FALSE)
}
```

---

 reshapeW2L

---

*Reshape Repeated-Measures Data from Wide to Long Format*


---

### Description

The data are assumed to be in "wide" format, with a single row for each subject, and different columns for values of one or more repeated-measures variables classified by one or more within-subjects factors.

### Usage

```
reshapeW2L(data, within, levels, varying, ignore, id = "id")
```

### Arguments

data	wide version of data set.
within	a character vector of names for the crossed within-subjects factors to be created in the long form of the data.
levels	a named list of character vectors, each element giving the names of the levels for a within-subjects factor; the names of the list elements are the names of the within-subjects factor, given in the <code>within</code> argument.
varying	a named list of the names of variables in the wide data set specifying the occasion-varying variables to be created in the long data set; each element in the list is named for an occasion-varying variable and is a character vector of column names in the wide data for that occasion-varying variable.
ignore	a character vector of names of variables in the wide data to be dropped in the long form of the data.
id	the (character) name of the subject ID variable to be created in the long form of the data, default "id".

### Details

Between-subjects variables don't vary by occasions for each subject. Variables that aren't listed explicitly in the arguments to the function are assumed to be between-subjects variables. The values of these variables are duplicated in each row pertaining to a given subject.

Within-subjects factors vary by occasions for each subject, and it is assumed that the within-subjects design is regular, completely crossed, and balanced, so that the same combinations of within-subjects factors are observed for each subject. There are typically one or two within-subject factors.

Occasion-varying variables, as their name implies, (potentially) vary by occasions for each subject, and include one or more "response" variables, possibly along with occasion-varying covariates; these variables can be factors as well as numeric variables. Each occasion-varying variable is encoded in multiple columns of the wide form of the data and in a single column in the long form. There is typically one occasion-varying variable, a response variable.

There is one value of each occasion-varying variable for each combination of levels of the within-subjects factors. Thus, the number of variables in the wide data for each occasion-varying variable

must be equal to the product of levels of the within-subjects factors, with the levels of the within-subjects factors varying most quickly from right to left in the `within` argument.

### Value

a data frame in "long" format, with multiple rows for each subject (equal to the number of combinations of levels of the within-subject factors) and one column for each between-subjects and occasion-varying variable.

### Author(s)

John Fox <jfox@mcmaster.ca>

### See Also

[reshapeL2W](#), [reshape](#), [OBrienKaiser](#), [OBrienKaiserLong](#).

### Examples

```
if (require(carData)){
  OBrienKaiserL <- reshapeW2L(OBrienKaiser, within=c("phase", "hour"),
    levels=list(phase=c("pre", "post", "fup"), hour=1:5),
    varying=list(score=c("pre.1", "pre.2", "pre.3", "pre.4", "pre.5",
      "post.1", "post.2", "post.3", "post.4", "post.5",
      "fup.1", "fup.2", "fup.3", "fup.4", "fup.5")))
  brief(OBrienKaiserL, c(15, 15))
  m1 <- Tapply(score ~ phase + hour + treatment + gender, mean, data=OBrienKaiserL)
  m2 <- Tapply(score ~ phase + hour + treatment + gender, mean, data=OBrienKaiserLong)
  all.equal(m1, m2) # should be equal
}

if (require(carData)){
  OBrienKaiserL2 <- reshapeW2L(OBrienKaiser, within="phase",
    levels=list(phase=c("pre", "post", "fup")),
    ignore=c("pre.2", "pre.3", "pre.4", "pre.5",
      "post.2", "post.3", "post.4", "post.5",
      "fup.2", "fup.3", "fup.4", "fup.5"),
    varying=list(score=c("pre.1", "post.1", "fup.1")))
  brief(OBrienKaiserL2, c(6, 6))
  m1 <- Tapply(score ~ phase + treatment + gender, mean, data=OBrienKaiserL2)
  m2 <- Tapply(score ~ phase + treatment + gender, mean,
    data=subset(OBrienKaiserLong, hour==1))
  all.equal(m1, m2) # should be equal
}
```

---

stepwise

*Stepwise Model Selection*

---

### Description

This function is a front end to the [stepAIC](#) function in the **MASS** package.

### Usage

```
stepwise(mod,  
  direction = c("backward/forward", "forward/backward", "backward", "forward"),  
  criterion = c("BIC", "AIC"), ...)
```

### Arguments

mod	a model object of a class that can be handled by stepAIC.
direction	if "backward/forward" (the default), selection starts with the full model and eliminates predictors one at a time, at each step considering whether the criterion will be improved by adding back in a variable removed at a previous step; if "forward/backwards", selection starts with a model including only a constant, and adds predictors one at a time, at each step considering whether the criterion will be improved by removing a previously added variable; "backwards" and "forward" are similar without the reconsideration at each step.
criterion	for selection. Either "BIC" (the default) or "AIC". Note that stepAIC labels the criterion in the output as "AIC" regardless of which criterion is employed.
...	arguments to be passed to stepAIC.

### Value

The model selected by stepAIC.

### Author(s)

John Fox <jfox@mcmaster.ca>

### References

W. N. Venables and B. D. Ripley *Modern Applied Statistics Statistics with S, Fourth Edition* Springer, 2002.

### See Also

[stepAIC](#)

## Examples

```
# adapted from ?stepAIC in MASS
if (require(MASS)){
  data(birthwt)
  bwt <- with(birthwt, {
    race <- factor(race, labels = c("white", "black", "other"))
    ptd <- factor(ptl > 0)
    ftv <- factor(ftv)
    levels(ftv)[-1:2] <- "2+"
    data.frame(low = factor(low), age, lwt, race, smoke = (smoke > 0),
              ptd, ht = (ht > 0), ui = (ui > 0), ftv)
  })
  birthwt.glm <- glm(low ~ ., family = binomial, data = bwt)
  print(stepwise(birthwt.glm, trace = FALSE))
  print(stepwise(birthwt.glm, direction="forward/backward"))
}
```

---

summarySandwich

*Linear Model Summary with Sandwich Standard Errors*


---

## Description

summarySandwich creates a summary of a "lm" object similar to the standard one, with sandwich estimates of the coefficient standard errors in the place of the usual OLS standard errors, also modifying as a consequence the reported t-tests and p-values for the coefficients. Standard errors may be computed from a heteroscedasticity-consistent ("HC") covariance matrix for the coefficients (of several varieties), or from a heteroscedasticity-and-autocorrelation-consistent ("HAC") covariance matrix.

## Usage

```
summarySandwich(model, ...)

## S3 method for class 'lm'
summarySandwich(model,
  type=c("hc3", "hc0", "hc1", "hc2", "hc4", "hac"), ...)
```

## Arguments

model	a linear-model object.
type	type of sandwich standard errors to be computed; see <a href="#">hccm</a> in the <b>car</b> package, and <a href="#">vcovHAC</a> in the <b>sandwich</b> package, for details.
...	arguments to be passed to hccm or vcovHAC

## Value

an object of class "summary.lm", with sandwich standard errors substituted for the usual OLS standard errors; the omnibus F-test is similarly adjusted.

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**

[hccm](#), [vcovHAC](#).

**Examples**

```
mod <- lm(prestige ~ income + education + type, data=Prestige)
summary(mod)
summarySandwich(mod)
```

# Index

- \* **distribution**
  - Gumbel, [12](#)
- \* **hplot**
  - Barplot, [3](#)
  - discretePlot, [10](#)
  - Dotplot, [11](#)
  - Hist, [13](#)
  - indexplot, [14](#)
  - lineplot, [16](#)
  - piechart, [21](#)
  - plotBoot, [22](#)
  - plotDistr, [23](#)
  - plotMeans, [24](#)
  - repeatedMeasuresPlot, [30](#)
- \* **htest**
  - normalityTest, [18](#)
  - rcorr.adjust, [25](#)
- \* **manip**
  - binVariable, [6](#)
  - mergeRows, [17](#)
  - readSAS, [26](#)
  - readSPSS, [27](#)
  - readStata, [28](#)
  - readXL, [28](#)
  - reshapeL2W, [32](#)
  - reshapeW2L, [34](#)
- \* **misc**
  - assignCluster, [2](#)
  - colPercents, [7](#)
  - KMeans, [15](#)
  - numSummary, [19](#)
  - partial.cor, [20](#)
  - reliability, [29](#)
  - summarySandwich, [37](#)
- \* **models**
  - DeltaMethod, [8](#)
  - stepwise, [36](#)
- \* **univar**
  - binnedCounts, [5](#)
  - discreteCounts, [9](#)
  - ad.test, [19](#)
  - Anova, [31](#)
  - assignCluster, [2](#)
  - Barplot, [3](#)
  - barplot, [3, 4](#)
  - bin.var (binVariable), [6](#)
  - binnedCounts, [5, 9](#)
  - binVariable, [6](#)
  - colPercents, [7](#)
  - cor, [20](#)
  - cov, [30](#)
  - cut, [6](#)
  - cutree, [3](#)
  - cvm.test, [19](#)
  - DeltaMethod, [8](#)
  - deltaMethod, [8](#)
  - densityPlot, [22](#)
  - dgumbel (Gumbel), [12](#)
  - discreteCounts, [5, 9](#)
  - discretePlot, [10](#)
  - Dotplot, [11](#)
  - excel\_sheets, [28, 29](#)
  - excel\_sheets (readXL), [28](#)
  - Gumbel, [12](#)
  - hccm, [37, 38](#)
  - hclust, [3](#)
  - Hist, [10, 13](#)
  - hist, [5, 11, 13](#)
  - indexplot, [14](#)
  - interaction.plot, [25](#)
  - KMeans, [3, 15](#)

kmeans, [3](#), [6](#), [16](#)  
kurtosis, [19](#), [20](#)

legend, [4](#), [23](#)  
lillie.test, [19](#)  
lineplot, [16](#)

mean, [20](#)  
merge, [17](#)  
mergeRows, [17](#)

normalityTest, [18](#)  
numSummary, [19](#)

OBrienKaiser, [31](#), [33](#), [35](#)  
OBrienKaiserLong, [33](#), [35](#)

p.adjust, [18](#), [25](#), [26](#)  
par, [31](#)  
partial.cor, [20](#)  
pearson.test, [18](#), [19](#)  
pgumbel (Gumbel), [12](#)  
pie, [21](#)  
piechart, [21](#)  
plot, [14](#)  
plot.default, [15](#)  
plotBoot, [22](#)  
plotDistr, [23](#)  
plotMeans, [24](#)  
points, [31](#)  
print.DeltaMethod (DeltaMethod), [8](#)  
print.numSummary (numSummary), [19](#)  
print.rcorr.adjust (rcorr.adjust), [25](#)  
print.reliability (reliability), [29](#)

qgumbel (Gumbel), [12](#)  
quantile, [20](#)

rainbow\_hcl, [4](#), [21](#)  
rcorr, [25](#), [26](#)  
rcorr.adjust, [25](#)  
read.dta13, [28](#)  
read.spss, [27](#)  
read\_excel, [28](#), [29](#)  
read\_sas, [26](#), [27](#)  
read\_spss, [27](#)  
readSAS, [26](#)  
readSPSS, [27](#)  
readStata, [28](#)  
readXL, [28](#)

reliability, [29](#)  
repeatedMeasuresPlot, [30](#)  
reshape, [32](#), [33](#), [35](#)  
reshapeL2W, [32](#), [35](#)  
reshapeW2L, [34](#)  
rgumbel (Gumbel), [12](#)  
rowPercents (colPercents), [7](#)

sd, [20](#)  
sf.test, [19](#)  
shapiro.test, [18](#), [19](#)  
showLabels, [14](#), [15](#)  
skewness, [19](#), [20](#)  
stepAIC, [36](#)  
stepwise, [36](#)  
summarySandwich, [37](#)

totPercents (colPercents), [7](#)

vcovHAC, [37](#), [38](#)