Package ‘RcmdrMisc’

January 14, 2020

Version 2.7-0
Date 2019-12-24
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
Suggests boot, datasets
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)

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 973
Repository/R-Forge/DateTimeStamp 2019-12-24 18:40:27
Date/Publication 2020-01-14 15:10:02 UTC

NeedsCompilation no

R topics documented:

assignCluster ................................................................. 2
Barplot ................................................................. 3
binnedCounts ............................................................. 5
binVariable ............................................................... 6
colPercents ............................................................... 7
assignCluster

Append a Cluster Membership Variable to a Dataframe

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

clusterData The data matrix used in the clustering solution. The data matrix may have have only a subset of the observations contained in the original dataframe.

origData The original dataframe from which the data used in the clustering solution were taken.
Barplot

clusterVec

An integer variable containing the cluster membership assignments for the observations used in creating the clustering solution. This vector can be created using `cutree` for clustering solutions generated by `hclust` or the cluster component of a list object created by `kmeans` or `KMeans`.

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

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

---

Barplot

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

```r
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.
- `by` optionally, a second factor.
- `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 rainbow_hcl in the colorspace 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 legend 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 barplot 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")
})
### binnedCounts

**Binned Frequency Distributions of Numeric Variables**

**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**

```r
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**

```r
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
- x: numeric variable to be binned.
- bins: number of bins.
- method: 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.
- labels: 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.
- ...: arguments to be passed to binVariable.

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]))
**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:
")
  print(colPercents(xtabs(~ lfp + wc, data=Mroz)))
  cat("\n\n  row percents:
")
  print(rowPercents(xtabs(~ hc + lfp, data=Mroz)))
  cat("\n\n  total percents:
")
  print(totPercents(xtabs(~ hc + wc, data=Mroz)))
  cat("\n\n  three-way table, column percents:
")
  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

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

Arguments

- `model` a regression model; see the `deltaMethod` 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

```r
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

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

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 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 hist 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

```r
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**

```r
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 `length(n) > 1`, 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**

Hist

Examples

```r
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

```r
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 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 `hist`.
- `...`: arguments to be passed to `hist`.

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

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

### indexplot

**Index Plots**

Index plots with point identification.

#### Usage

```r
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.
- **id.method**: method for identifying points; see `showLabels`.
- **type**: to be passed to `plot`.
- **id.n**: number of points to identify; see `showLabels`.
- **ylab**: label for vertical axis; if missing, will be constructed from x; for a data frame, defaults to the column names.
- **legend**: keyword (see link[graphics]legend) giving location of the legend if groups are specified; if `legend=FALSE`, the legend is suppressed.
- **title**: title for the legend; may normally be omitted.
- **col**: vector of colors for the groups.
- **...**: to be passed to `plot`.

#### 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>


**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**

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

**Arguments**

- `x` 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).
- `centers` The number of clusters in the solution.
- `iter.max` The maximum number of iterations allowed.
- `num.seeds` The number of different starting random seeds to use. Each random seed results in a different k-means solution.

**Value**

A list with components:

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

**Author(s)**

Dan Putler

**Examples**

```r
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)
}
```
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 is 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

X First data frame.
Y Second data frame.
common.only If TRUE, only variables (columns) common to the two data frame are included in the merged data set; the default is FALSE.
... 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)
}
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

```r
normalityTest(x, ...)
```

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

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

Arguments

- `x` numeric vector or formula.
- `formula` one-sided formula of the form ~x or two-sided formula of the form x ~ groups, where x is a numeric variable and groups is a factor.
- `data` a data frame containing the data for the test.
- `test` quoted name of the function to perform the test.
- `groups` optional factor to divide the data into groups.
- `vname` optional name for the variable; if absent, taken from x.
- `gname` optional name for the grouping factor; if absent, taken from groups.
- `...` any arguments to be passed down; the only useful such arguments are for the `pearson.test` function in the `nortest` 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>
numSummary

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)

---

table

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, ...)
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
piechart

Examples

```r
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

```r
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 `rainbow_hcl` function in the `colorspace` package.
- **...**: further arguments to be passed to `pie`.

Author(s)

John Fox <jfox@mcmaster.ca>

See Also

`pie`, `rainbow_hcl`

Examples

```r
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

object  
an object of class "boot".

confint  
an object of class "confint.boot" (or an ordinary 2-column matrix) containing confidence limits for the parameters in object; if NULL (the default), these are computed from the first argument, using the defaults for "boot" objects.

...  
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

Plot a probability density, mass, or distribution function.

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 legend, 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
**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**

```r
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`.

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

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=default.stringsAsFactors())

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 then columns containing character data are converted to factors; the default is taken from default.stringsAsFactors().

Value

a data frame

Author(s)

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

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

**Arguments**

- **file** path to an SPSS *.sav* or *.por* 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 then columns containing character data are converted to factors and factors are created from SPSS value labels; the default is taken from `default.stringsAsFactors()`.
- **tolower** change variable names to lowercase, default TRUE.
- **use.value.labels** if TRUE, the default, variables with value labels in the SPSS data set will become either factors or character variables (depending on the `stringsAsFactors` argument) with the value labels as their levels or values. As for `read.spss`, 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).
- **use.haven** use `read_spss` from the `haven` package to read the file, in preference to `read.spss` from the `foreign` package; the default is TRUE for a *.sav* file and FALSE for a *.por* file.

**Value**

a data frame

**Author(s)**

John Fox <jfox@mcmaster.ca>

**See Also**

`read_sas`, `read_spss`
readStata  
Read a Stata Data Set

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=default.stringsAsFactors(), convert.dates=TRUE)

Arguments

file    path to a Stata .dta 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 then columns containing character data are converted to factors and factors are created from Stata value labels; the default is taken from default.stringsAsFactors().
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  
Read an Excel File

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 = default.stringsAsFactors())
```

```
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 then columns containing character data are converted to factors; the default is taken from default.stringsAsFactors().

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 (summed-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


See Also

cov

Examples

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

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"), ...)
stepwise

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


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)[-c(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"))
}

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

```r
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 `hccm` in the `car` package, and `vcovHAC` in the `sandwich` 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

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

<table>
<thead>
<tr>
<th>Topic</th>
<th>Functions/arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>distribution</td>
<td>Gumbel, 12</td>
</tr>
<tr>
<td>hplot</td>
<td>Barplot, 3, discretePlot, 10, Dotplot, 11, Hist, 13, indexplot, 14, lineplot, 16, piechart, 21, plotBoot, 22, plotDistr, 23, plotMeans, 24</td>
</tr>
<tr>
<td>htest</td>
<td>normalityTest, 18, rcorr.adjust, 25</td>
</tr>
<tr>
<td>manip</td>
<td>binVariable, 6, mergeRows, 17, readSAS, 26, readSPSS, 27, readStata, 28, readXL, 28</td>
</tr>
<tr>
<td>misc</td>
<td>assignCluster, 2, colPercents, 7, KMeans, 15, numSummary, 19, partial.cor, 20, reliability, 29, summarySandwich, 32</td>
</tr>
<tr>
<td>models</td>
<td>DeltaMethod, 8, stepwise, 30</td>
</tr>
<tr>
<td>univar</td>
<td>binnedCounts, 5, discreteCounts, 9</td>
</tr>
</tbody>
</table>

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, 32, 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

p.adjust, 18, 25, 26
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
print.DeltaMethod (DeltaMethod), 8
print.numSummary (numSummary), 19
print.rcorr.adjust (rcorr.adjust), 25
print.reliability (reliability), 29

pgumbel (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
rgumbel (Gumbel), 12
rowPercents (colPercents), 7

sd, 20
sf.test, 19
shapiro.test, 18, 19
showLabels, 14, 15
skewness, 19, 20
stepAIC, 30, 31
stepwise, 30

summarySandwich, 32
totPercents (colPercents), 7
vcovHAC, 32