Package 'RcmdrMisc'

August 4, 2015

Version 1.0-3

Date 2015-07-13

Title R Commander Miscellaneous Functions

Depends R (>= 3.0.0), utils, car, sandwich

Imports abind, colorspace, Hmisc, MASS, e1071, readxl, graphics, grDevices, stats

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 http://www.r-project.org, http://socserv.socsci.mcmaster.ca/jfox/

NeedsCompilation no

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

Maintainer John Fox <jfox@mcmaster.ca>

Repository CRAN

Date/Publication 2015-08-04 18:23:40

R topics documented:

assignCluster 2
Barplot 3
bin.var 4
colPercents 5
Dotplot 6
Hist 7
indexplot 8
KMeans 9
lineplot 10
mergeRows 10

numSummary 11
partial.cor 13
plotDistr 14
plotMeans 15
rcorr.adjust 16
readXL 17
reliability 18
stepwise 19
summarySandwich 20

Index **22**

`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.
`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

```
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 precentages. 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"),
  style = c("divided", "parallel"), col = rainbow_hcl(length(levels(by))),
  xlab = deparse(substitute(x)), legend.title = deparse(substitute(by)),
  ylab = scale, legend.pos = "topright")
```

Arguments

`x`	a factor.
`by`	optionally, a second factor.
`scale`	either `"frequency"` (the default) or `"percent"`.
`style`	for two-factor plots, either `"divided"` (the default) or `"parallel"`.
`col`	colors for the by factor in two-factor plots; defaults 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.
`legend.pos`	position of the legend, in a form acceptable to the `legend` function.

Value

Returns `NULL` invisibly.

Author(s)

John Fox `<jfox@mcmaster.ca>`

See Also

`barplot`, `legend`, `rainbow_hcl`

Examples

```
if (require(car)){
    data(Mroz)
    with(Mroz, {
      Barplot(wc)
      Barplot(wc, by=hc)
      Barplot(wc, by=hc, style="parallel", scale="percent")
    })
}
```

---

bin.var *Bin a Numeric Varisible*

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.

Usage

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

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.

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

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 distrbution 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	optinally 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

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

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 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 returns NULL, and is called for its side effect — plotting a histogram.

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 plot with point identification.

Usage

```
indexplot(x, labels = seq_along(x), id.method = "y", type = "h",
    id.n = 0, ylab, ...)
```

Arguments

`x`	numeric variable.
`labels`	point labels.
`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`.
`...`	to be passed to `plot`.

Value

Returns labelled indices of identified points or (invisibly) `NULL` if no points are identified.

Author(s)

John Fox <jfox@mcmaster.ca>

See Also

`showLabels`, `plot.default`

Examples

```
if (require("car")){
    data(Prestige)
    with(Prestige, indexplot(income, id.n=2, labels=rownames(Prestige)))
}
```

KMeans *K-Means Clustering Using Multiple Random Seeds*

Description

Finds a number of k-means clusting 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

`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

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)
}
```

---

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 skewness and kurtosis functions in the **e1071** 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)
```

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, ...)
```

Arguments

x	horizontal coordinates
p	vertical coordinates
discrete	is the random variable discrete?
cdf	is this a cumulative distribution (as opposed to mass) function?
...	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")

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)), main = "Plot of Means",
    pch = 1:n.levs.2, lty = 1:n.levs.2, col = palette(), ...)
```

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.
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
...	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)
    rcorr.adjust(Mroz[,c("k5", "k618", "age", "lwg", "inc")])
    rcorr.adjust(Mroz[,c("k5", "k618", "age", "lwg", "inc")], type="spearman")
    }
```

---

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.
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 (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))
}
```

---

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 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

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

Index

∗Topic **hplot**
Barplot, 3
Dotplot, 6
Hist, 7
indexplot, 8
lineplot, 10
plotDistr, 14
plotMeans, 15
∗Topic **htest**
rcorr.adjust, 16
∗Topic **manip**
bin.var, 4
mergeRows, 10
readXL, 17
∗Topic **misc**
assignCluster, 2
colPercents, 5
KMeans, 9
numSummary, 11
partial.cor, 13
reliability, 18
summarySandwich, 20
∗Topic **models**
stepwise, 19

assignCluster, 2

Barplot, 3
barplot, *3*
bin.var, 4

colPercents, 5
cor, *13*
cov, *19*
cut, *4*
cutree, *2*

Dotplot, 6

excel_sheets, *17, 18*
excel_sheets *(*readXL*)*, 17

hccm, *21*
hclust, *2*
Hist, 7
hist, *6, 7*

indexplot, 8
interaction.plot, *16*

KMeans, *2*, 9
kmeans, *2*, *4*, *9*
kurtosis, *12*

legend, *3*
lineplot, 10

mean, *12*
merge, *11*
mergeRows, 10

numSummary, 11

p.adjust, *16, 17*
partial.cor, 13
plot, *8*
plot.default, *8*
plotDistr, 14
plotMeans, 15
print.numSummary *(*numSummary*)*, 11
print.rcorr.adjust *(*rcorr.adjust*)*, 16
print.reliability *(*reliability*)*, 18

quantile, *12*

rainbow_hcl, *3*
rcorr, *16, 17*
rcorr.adjust, 16
read_excel, *17, 18*
readXL, 17
reliability, 18
rowPercents *(*colPercents*)*, 5

sd, *12*

showLabels, *8*
skewness, *12*
stepAIC, *19*, *20*
stepwise, 19
summarySandwich, 20

totPercents *(*colPercents*)*, 5

vcovHAC, *21*