Package 'ENmisc'

February 19, 2015

Type Package

Title Neuwirth miscellaneous

Version 1.2-7

Date 2012-04-01

Maintainer Erich Neuwirth <erich.neuwirth@univie.ac.at>

Description The ENmisc package contains utility function for different purposes: mtapply and mlapply (multivariate version of tapply and lapply), wtd.boxplot (a boxplot with weights), and a visual interface to restructuring mosaic plots.

License GPL-2

Depends Hmisc, vcd (>= 1.2-11), RColorBrewer

Suggests Rcmdr, gWidgets (>= 0.0-45), gWidgetstcltk (>= 0.0-44)

LazyLoad yes

Author Erich Neuwirth [aut, cre]

NeedsCompilation no

Repository CRAN

Date/Publication 2013-04-02 12:37:29

R topics documented:

`ENmisc-package` *Neuwirth Miscellaneous*

Description

Utility functions: mtapply and mlapply (multivarate tapply and lapply) wtd.boxplot (weighted boxplot), visual interface to generate mosaic plots

Details

Package:	ENmisc
Type:	Package
License:	GPL 2.0
LazyLoad:	yes

Note

Special thanks go to Rich Heiberger and John Verzani.

Extended discussions with Rich Heiberger helped bringing the mosaic part of the package in its final shape. He invested quite some time in testing uncharted territory and made valuable suggestions for improving the interface to mosaic.

Without John Verzani's package `gWidgets` and his always helpful comments the graphical interface for mosaic plots would not exist.

Author(s)

Erich Neuwirth (mosaic tools developed after discussions with Rich Heiberger)

Maintainer: Erich Neuwirth <erich.neuwirth@univie.ac.at>

`mlapply` *Apply a Function of Multiple Arguments*

Description

`mlapply` returns a list of the same length as each of the lists in `lol`. Each element of the resulting list is the result of applying `FUN` to all the first elements of the lists in lol, all the second elements of the lists in lol ...
It is the multivariate version of `lapply`

Usage

```
mlapply(lol,FUN,...)
```

Arguments

`lol`	a list of lists, the elements of each list will be used as arguments to calls to `FUN`
`FUN`	the function to be applied. In the case of functions like `+`, `%*%`, etc., the function name must be backquoted or quoted.
`...`	Any additional arguments passed to each call to `FUN`

Value

`FUN` is a function with the number of arguments being equal to the number of lists contained in `lol`. `mlapply` makes a function call to `FUN` for all the first elements of all the lists in `lol`, then a function call to all the second elements of all the lists in `lol`, and retunrs all the results as a list.
If the first list in `lol` has named elements, the names will also be used for the elements of the resulting list.

See Also

`link{lapply}`

Examples

```
mlapply(list(list(1,2,3),list(4,5,6)),function(x,y)x^2+y^2)
mlapply(list(list(a=1,b=2,c=3),list(4,5,6)),function(x,y)x^2+y^2)
mlapply(list(list(a=1,b=2,c=3),list(4,5,6)),function(x,y,e)x^e+y^e,3)
mlapply(list(list(1,2,3),list(4,5,6)),function(x,y,const=0)x^2+y^2+const)
```

`mosaicPermDialog` *Visual interface to create an restructure mosaic and assoc plots*

Description

Apply a function of multiple arguments to each cell of a ragged array, that is to each (non-empty) group of values given by a unique combination of the levels of certain factors. It is a hybrid of `tapply` and `mapply`.

Usage

```
mosaicSelectDialog()

mosaicPermDialog(tablename,allow.collapsing=TRUE, start.active=NULL,
displayPermCommand=TRUE, extendedOptions=TRUE)

margin.table.structable(x,margin)
```

```
setMosaicPalette(palettename,reverse=FALSE)

brewer.pal.ext(n,name,reverse=FALSE)
```

Arguments

`tablename`	An object for which a mosaic plot or an assoc plot can be displayed: table or ftable or structable
`allow.collapsing`	Allows to omit dimensions from the table
`start.active`	boolean vector controlling active variables (variables to be used in plot)
`displayPermCommand`	controls if command producing plot is displayed in the dialog window
`extendedOptions`	allows finer control of result of using the dialog
`palettename`	name of one of the palettes defined in package `RColorBrewer`
`x`	structable to collapse
`margin`	dimensions to keep
`n`	number of colors in the ColorBrewer palette
`name`	name of ColorBrewer palette
`reverse`	reverse color order in palette

Value

`mosaicPermDialog` either returns the function call producing the last displayed plot as string or the command producing the permuted table underlying th plot or the permuted table itself.

`mosaicSelectDialog` does not return usable values. It just creates a dialog box to select the object to be plotted as mosaic or assoc plot.

`margin.table.structable` collapes a structable to a lower dimensional structable bh keeping the indicated dimensions and summing the values over all dimensions not indicated. For a twodimensional table this would compute row sums or column sums.

`brewer.pal.ext` is a modified version of `brewer.pal` in package `RColorBrewer`. It allows palettes with one or 2 colors (which RColorBrewer does not). For documentatation see `RColorBrewer`.

Note

Special thanks go to Richard Heiberger who invested quite some time in testing uncharted territory and made valuable suggestions for improving this function

See Also

the functions `link{mosaic}` and `assoc`

Examples

```
## Not run:
data(Titanic)
myTitanic <- structable(Titanic)
mosaicPermDialog(myTitanic)

## End(Not run)
```

`mtapply` *Apply a Function of Multiple Arguments Over a Ragged Array*

Description

Apply a function of multiple arguments to cells of a identically structures ragged arrays, that is to each set of (non-empty) groups of values given by a unique combination of the levels of certain factors. It is a hybrid of `tapply` and `mapply`.

Usage

```
mtapply(X, INDEX, FUN = NULL, simplify = TRUE)
```

Arguments

`X`	a list of atomic objects, typically vectors, all of the same length
`INDEX`	list of factors, each of same length as `X`. The elements are coerced to factors by `as.factor`.
`FUN`	the function to be applied, or `NULL`. In the case of functions like `+`, `%*%`, etc., the function name must be backquoted or quoted. If `FUN` is `NULL`, tapply returns a vector which can be used to subscript the multi-way array `mtapply` normally produces.
`simplify`	If `FALSE`, `tapply` always returns an array of mode `"list"`. If `TRUE` (the default), then if `FUN` always returns a scalar, `tapply` returns an array with the mode of the scalar.

Value

If `FUN` is not `NULL`, it is passed to `match.fun`, and hence it can be a function or a symbol or character string naming a function.

When `FUN` is present, `mtapply` calls `FUN` for each set of cells that has any data in it. If `FUN` returns a single atomic value for each such cell (e.g., functions `mean` or `var`) and when `simplify` is `TRUE`, `tapply` returns a multi-way `array` containing the values, and `NA` for the empty cells. The array has the same number of dimensions as `INDEX` has components; the number of levels in a dimension is the number of levels (`nlevels()`) in the corresponding component of `INDEX`. Note that if the return value has a class (e.g. an object of class `"Date"`) the class is discarded.

If `FUN` does not return a single atomic value, `tapply` returns an array of mode `list` whose components are the values of the individual calls to `FUN`, i.e., the result is a list with a `dim` attribute.

When there is an array answer, its dimnames are named by the names of INDEX and are based on the levels of the grouping factors (possibly after coercion).

For a list result, the elements corresponding to empty cells are NULL.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

the functions link{tapply}, mapply, by and aggregate (using tapply); apply, lapply with its versions sapply and mapply.

Examples

```
require(Hmisc)
x<-1:10
fc<-rep(c("a","b"),each=5)
wt<-1:10
mtapply(list(x,wt),fc,wtd.mean)
mtapply(list(x,rep(1/10,10)),fc,wtd.mean)
```

---

wtd.boxplot *Box Plots with weighted cases*

Description

Produce box-and-whisker plot(s) of the given (grouped, weighted) values.

Usage

```
wtd.boxplot(x, ...)

## S3 method for class 'formula'
wtd.boxplot(formula, weights = NULL, data = NULL, ..., subset, na.action = NULL)

## Default S3 method:
wtd.boxplot(x, weights = NULL, ..., range = 1.5, width = NULL, varwidth = FALSE,
        notch = FALSE, outline = TRUE, names, plot = TRUE,
        border = par("fg"), col = NULL, log = "",
        pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),
        horizontal = FALSE, add = FALSE, at = NULL)
```

Arguments

`formula`	a formula, such as `y ~ grp`, where `y` is a numeric vector of data values to be split into groups according to the grouping variable `grp` (usually a factor).
`weights`	case weights, vector of the same length as the dependent variable in formula or the variable given as arguments)
`data`	a data.frame (or list) from which the variables in `formula` should be taken.
`subset`	an optional vector specifying a subset of observations to be used for plotting.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is to ignore missing values in either the response or the group.
`x`	for specifying data from which the boxplots are to be produced. Either a numeric vector, or a single list containing such vectors. Additional unnamed arguments specify further data as separate vectors (each corresponding to a component boxplot). `NA`s are allowed in the data.
`...`	For the `formula` method, named arguments to be passed to the default method. For the default method, unnamed arguments are additional data vectors (unless `x` is a list when they are ignored), and named arguments are arguments and graphical parameters to be passed to `bxp` in addition to the ones given by argument `pars` (and override those in `pars`).
`range`	this determines how far the plot whiskers extend out from the box. If `range` is positive, the whiskers extend to the most extreme data point which is no more than `range` times the interquartile range from the box. A value of zero causes the whiskers to extend to the data extremes.
`width`	a vector giving the relative widths of the boxes making up the plot.
`varwidth`	if `varwidth` is `TRUE`, the boxes are drawn with widths proportional to the square-roots of the number of observations in the groups.
`notch`	if `notch` is `TRUE`, a notch is drawn in each side of the boxes. If the notches of two plots do not overlap this is 'strong evidence' that the two medians differ (Chambers *et al.*, 1983, p. 62). See `boxplot.stats` for the calculations used.
`outline`	if `outline` is not true, the outliers are not drawn (as points whereas S+ uses lines).
`names`	group labels which will be printed under each boxplot. Can be a character vector or an expression (see plotmath).
`boxwex`	a scale factor to be applied to all boxes. When there are only a few groups, the appearance of the plot can be improved by making the boxes narrower.
`staplewex`	staple line width expansion, proportional to box width.
`outwex`	outlier line width expansion, proportional to box width.
`plot`	if `TRUE` (the default) then a boxplot is produced. If not, the summaries which the boxplots are based on are returned.
`border`	an optional vector of colors for the outlines of the boxplots. The values in `border` are recycled if the length of `border` is less than the number of plots.
`col`	if `col` is non-null it is assumed to contain colors to be used to colour the bodies of the box plots. By default they are in the background colour.

`log`	character indicating if x or y or both coordinates should be plotted in log scale.
`pars`	a list of (potentially many) more graphical parameters, e.g., `boxwex` or `outpch`; these are passed to `bxp` (if `plot` is true); for details, see there.
`horizontal`	logical indicating if the boxplots should be horizontal; default `FALSE` means vertical boxes.
`add`	logical, if true *add* boxplot to current plot.
`at`	numeric vector giving the locations where the boxplots should be drawn, particularly when `add = TRUE`; defaults to `1:n` where `n` is the number of boxes.

Details

The generic function `wtd.boxplot` currently has a default method (`wtd.boxplot.default`) and a formula interface (`wtd.boxplot.formula`).

If multiple groups are supplied either as multiple arguments or via a formula, parallel boxplots will be plotted, in the order of the arguments or the order of the levels of the factor (see `factor`).

Missing values are ignored when forming boxplots.

Value

List with the following components:

`stats`	a matrix, each column contains the extreme of the lower whisker, the lower hinge, the median, the upper hinge and the extreme of the upper whisker for one group/plot. If all the inputs have the same class attribute, so will this component.
`n`	a vector with the number of observations in each group.
`conf`	a matrix where each column contains the lower and upper extremes of the notch.
`out`	the values of any data points which lie beyond the extremes of the whiskers.
`group`	a vector of the same length as `out` whose elements indicate to which group the outlier belongs.
`names`	a vector of names for the groups.

See Also

`boxplot`

Examples

```
x<-1:10
fc<-rep(c("a","b"),each=5)
wt<-c(6:10,10:6)
wtd.boxplot(x~fc,weights=wt)
```

Index