

Package ‘agsemisc’

April 17, 2009

Version 1.1-3

Date 2006-05-19

Title Miscellaneous plotting and utility functions

Author Lutz Prechelt <prechelt@inf.fu-berlin.de>

Maintainer Lutz Prechelt <prechelt@inf.fu-berlin.de>

Depends R (>= 1.8.0), lattice, grid

Suggests MASS

Description High-featured panel functions for bwplot and xyplot, various plot management helpers,
some other utility functions

License GPL-2

Repository CRAN

Date/Publication 2006-05-29 07:45:03

R topics documented:

misc helpers	2
panel.bwstrip	3
panel.xy	5
plotf	7
plothelpers	9
stats helpers	10

Index	12
--------------	-----------

Description

`eqc(x, number=4, overlap=1/sqrt(length(x)))` an abbreviation for `equal.count` with useful defaults. Often useful for Lattice plots.

`grep.b(pattern, x)` performs a `grep`, but returns the result as a logical (boolean, thus the `.b`) vector along `x` rather than as an integer vector indices. Often useful for Lattice subset calls.

`grep.b(pattern, x)` performs `grep(pattern, x, value=T)` and thus returns the matched strings rather than their indices.

`orderavg(x, by)` orders the observations from vector `x` using the equal-weighted ranking criteria given in matrix `by` (which has `length(x)` rows and `n` columns), a `by` vector is coerced. The ranks computed from each matrix column are summed row-wise, the resulting vector is sorted according to these rank sums, and returned. This function can be used to pair elements from two vectors (by ranking them both by equivalent criteria).

`tableNA` abbreviation for `table` with an additional argument `exclude = NULL` so that NA and NaN values are tabulated as well.

`or.else(x, alternative = 0)` returns `x` where available and `alternative` where `is.na(x)`. `alternative` can be a scalar or vector and should have the same mode as `x`.

`printn(x, digits)` prints name and value of `x`. If the optional argument `digits` is used, `x` will be coerced to numeric and formatted with `digits` digits precision. Useful for quick-and-dirty debugging output.

`tracebck()` is like `traceback` except that it returns (rather than prints) only the first line of each frame. This is useful because the argument lists of calls often contain large data objects which clutter the output of `traceback` and make it very difficult to read.

Details

Type the name of a function to see its source code for details.

Author(s)

Lutz Prechelt (prechelt@inf.fu-berlin.de)

See Also

[equal.count](#), [grep](#), [order](#), [ifelse](#), [table](#), [traceback](#).

Examples

```
## Not run: plot(rnorm(8)~runif(8),xlim=c(10,NA))
## Not run: traceback()
## Not run: tracebck()
```

```

data(iris)
xyplot(Sepal.Width ~ Sepal.Length | eqc(Sepal.Length),
       subset=grep.b("v", Species), data=iris)

x = c(4, 9, NA, 4, 4, 27, NA, 27); table(or.else(x, 88)); table(x); tableNA(x)

printn(levels(iris$Species))

```

panel.bwstrip

Flexible panel.bwplot replacement

Description

Box plots with added stripplots, densityplots, mean/stderr marker etc.

Usage

```

panel.bwstrip(x, y, groups, subscripts, pch, col,
             box.ratio = 5, varwidth = FALSE, whiskerpos = 0.1, logbase,
             type="mean,mad,strip,N,grid",
             densityplot=expression(density(X,cut=2)), strip.limit=100,
             seplines = NULL, N.label = "N=% \n",
             extend = TRUE,
             levels.fos = NULL, ...)

```

Arguments

x	the data, as in panel.bwplot
y	the number of the box, as in panel.bwplot
groups	a factor indicating a partitioning of the data x. Relevant for the pch and col arguments.
subscripts	Argument internally used by Lattice to realize the groups functionality.
pch	A integer vector, character vector or list that indicates the value to be used as pch (plot character) for the stripplot for each level of groups. You will need a list if you want to mix plot symbols (indicated by integers) with plot characters (indicated by characters). There is a simpler method to specify any of these cases, namely a single string with entries separated by commas. One-digit numeric entries will evaluate to a plot character, two-digit numeric entries will evaluate to an integer plot symbol. For instance, "1, w, 13, 2, 02" will evaluate into <code>list("1", "w", 13, "2", 2)</code> . Defaults to all <code>trellis.par.get("superpose.symbol")</code> . Note that with <code>groups=gg</code> you can often use <code>pch=levels(gg)</code> , because Lattice will ignore all but the first character of a string.
col	Analogous to pch, but indicating color. Integers indicate color numbers, strings indicate color names, both can be mixed in the comma-separated string format. Defaults to all <code>trellis.par.get("superpose.symbol")\$col</code> .

<code>box.ratio</code>	Like in <code>panel.bwplot</code> , but the different default value withstands the call default of 1 that is imposed by Lattice (as of R 2.2).
<code>varwidth</code>	vary box thickness according to number of data values, as in <code>panel.bwplot</code>
<code>whiskerpos</code>	With large values in the range 1..Inf, <code>whiskerpos</code> is equivalent to the <code>coef</code> argument to <code>panel.bwplot</code> (or really <code>boxplot.stats</code>), i.e. each whisker is at the farthest data value that is at most <code>whiskerpos</code> times the interquartile range (or box width) away from the box. Only the values 1.5, 2 or 3 are common. In contrast, with small values <code>w</code> in the range 0..0.25, the whiskers will indicate the fixed quantiles <code>w</code> and <code>1-w</code> . This is easier to explain to non-statisticians and often more appropriate for larger samples. So <code>whiskerpos==0</code> will produce whiskers at the min and max of the data, the default will indicate the 10-percentile and the 90-percentile, and <code>whiskerpos==0.25</code> makes the whiskers disappear.
<code>logbase</code>	An argument <code>logbase=b</code> indicates that the high-level plot is using a log scale axis to base <code>b</code> and hence the data is logarithmic rather than real and needs to be converted back before computing mean, interpolating quantiles etc.
<code>type</code>	Declares which boxplot elements to include in the plot. Is either a comma-separated string of element names (as shown in the default) or a vector of such names. The elements have the following meaning: <code>mean</code> plots something like <code>-M-</code> indicating the mean and its standard error. <code>mad</code> will indicate the stderror of the median (median absolute deviation, as computed by <code>mad</code> divided by <code>sqrt(n)</code>) as a line left and right of the median dot. <code>strip</code> will produce a stripplot of the individual data points, scattered vertically to make similar values more visible. <code>density</code> will add a densityplot and a support line (extending along the range of the data). The plot can be customized via the <code>densityplot</code> argument. <code>N</code> will indicate the number of datapoints according to the <code>N.label</code> argument. <code>grid</code> will draw dotted vertical lines aligned with x-axis labels. Elements not mentioned will be left out of the plot. The box and median dot are always included (this is a boxplot, after all), the whiskers can be suppressed by <code>whiskerpos=0.25</code> .
<code>densityplot</code>	Relevant if "density" is mentioned in the <code>type</code> argument. Must be an expression describing a call to <code>density</code> that concerns the data vector <code>X</code> (an uppercase <code>x!</code>), which will be the data for the current boxplot for each evaluation of the expression. If a densityplot appears, the boxplot will not be color-filled.
<code>strip.limit</code>	If <code>T</code> , will reduce the stripplot to only the outliers, i.e., the values beyond the whiskers. If an integer, will suppress the stripplot entirely if there are more than this many values in the current boxplot.
<code>seplines</code>	A vector of vertical positions where horizontal lines will be drawn to separate the boxplots into groups. Position 1.5, 2.5 etc. is above the lowest, second-lowest boxplot etc.
<code>N.label</code>	A string such as " number of values underlying the boxplot and the resulting string is printed at the right of the plot iff <code>N</code> is mentioned in the <code>type</code> argument. Use trailing blanks and newlines to adjust positioning.
<code>extend</code>	If <code>TRUE</code> , will print to console some statistics for each sample: the quantiles (0, 0.25, 0.5, 0.75, 1), mean, quartile ratio (or at least interquartile range) and the number of data points. If <code>extend</code> is a function, it will be called with four arguments: data vector <code>x</code> , current <code>y</code> , groups, subscripts.

```
levels.fos    weird stuff, but means the same as in panel.bwplot
...          all other arguments will be ignored.
```

Details

A lattice panel function to be used with `bwplot`. Can draw a boxplot plus stripplot plus density-plot, indicate groups, mark the mean and its stderr, report N, and more. Graphical parameters are controlled by the `trellis.par.set` parameters `box.rectangle`, `box.umbrella`, `superpose.symbol` (which is non-standard), and `reference.line`. The `horizontal=F` option available in `panel.bwplot` is not supported here.

Author(s)

Lutz Prechelt (prechelt@inf.fu-berlin.de)

See Also

[a.resetplotparams](#), [plotf](#), [panel.xy](#).

Examples

```
# set grid.prompt(TRUE) to see each plot separately (click graphics window)
data(iris)
a.resetplotparams()
print(bwplot(Species~Sepal.Length, data=iris, panel=panel.bwstrip))
# A plot including a density plot:
print(bwplot(Species~Sepal.Length, data=iris, panel=panel.bwstrip,
             type="mean,strip,density"))
# A customized plot:
print(bwplot(~Sepal.Length, data=iris, panel=panel.bwstrip,
            groups=Species, pch=levels(iris$Species), strip.limit=200,
            type="mean,strip,density",
            densityplot=expression(density(X, cut=1))))
# A conventional-style plot:
print(bwplot(Species~Sepal.Length, data=iris, panel=panel.bwstrip,
            type="mean,strip,grid", strip.limit=TRUE, whiskerpos=1.5,
            densityplot=expression(density(X, cut=1))))
# A plot showing some other features:
print(bwplot(cut(Sepal.Width,4)~Sepal.Length, data=iris, panel=panel.bwstrip,
            groups=Species, varwidth=TRUE, box.ratio=20,
            which="strip,N", strip.limit=50, pch="1,2,3"))
```

Description

`panel.xy` is used like `panel.xyplot` but offers more features. In particular, it can add an `lqs` resistant regression line besides a normal one, can plot a prediction interval around the standard regression line, can add text indicating correlation and number of data points, and automatically uses different colors and line styles.

Usage

```
panel.xy(x, y, type = "p,r,r.pred,r=,N=,grid",
         r.min = 0.5, level = 0.8, unicolor = FALSE, ...)
```

Arguments

<code>x</code>	Vector of x coordinates for the plot.
<code>y</code>	Vector of y coordinates for the plot. Pairs (x,y) containing NAs will be filtered out.
<code>type</code>	A character vector (or single comma-separated string) of options to use for plotting. Entries "p", "l", "b", "o", "h", "s", "S" are handled just as <code>plot</code> or <code>panel.xyplot</code> would (namely as points, lines, both, both overplotted, histogram lines, stair steps, stair steps with vertical preference). "g" adds a coordinate grid via <code>panel.grid</code> . "r" adds an <code>lm</code> regression line. See <code>r.min</code> and <code>unicolor</code> for details. The line is suppressed if there are less than 8 points. "r.pred" adds a prediction interval around the "r" line. Implies "r". See <code>level</code> for details. "r.conf" adds a confidence interval around the "r" line. This concerns the regression line itself rather than the predictions it makes and is not often of much interest. Implies "r". <code>level</code> applies just like it does for "r.pred". "lqs" adds an <code>lqs</code> resistant regression line. Roughly speaking, this produces approximately the best regression line that is possible if it is allowed to throw the most difficult 50% of the points away. The line is suppressed if there are less than 8 points. "smooth", "loess", "lowess" are all synonymous and add a locally weighted regression via <code>panel.loess</code> , whose configuration parameters will be passed to the call if given. The line is suppressed if there are less than 8 points. "ab" adds an arbitrary line (described by parameters <code>a=</code> and <code>b=</code>) via <code>panel.ab</code> . This does currently not work due to parameter name clashes. "rug" adds a rug plot via <code>panel.rug</code> . "cor" prints the correlation coefficient computed by <code>cor</code> in the lower right corner of the plot. "N" prints the number of non-NA data points in the lower right corner of the plot. "v" (verbose) prints various statistics regarding the plot to the console.
<code>r.min</code>	Even if <code>type</code> "r" is requested, the regression line will appear only if <code>abs(cor(x,y)) >= r.min</code> .
<code>level</code>	The confidence level of the intervals plotted by <code>type</code> "r.conf" and <code>type</code> "r.pred".
<code>unicolor</code>	The colors used for the lines drawn by types "r", "r.pred", "r.conf", "loess" are normally taken from <code>trellis.par.get("regression.line")</code> (as introduced by <code>a.resetplotparams</code>). However, if you use <code>xyplot</code> with

`panel=panel.superpose`, `panel.groups=panel.xy`, you must specify `unicolor=T`. The regression lines will then be plotted in the style given by `trellis.par.get("superpose.line")` according to the respective group.

... Further parameters as needed for instance by `panel.loess` and `lqs`.

Details

The types "p", "l", "b", "o", "h", "s", "S" are in fact processed by `panel.xyplot`. Specifying `col`, `lty`, `lwd` will not work, as there are calls that include both ... and some of these.

Author(s)

Lutz Prechelt, prechelt@inf.fu-berlin.de

See Also

[a.resetplotparams](#), [trellis.par.get](#), [panel.xyplot](#), [panel.superpose](#), [panel.loess](#), [lm](#), [lqs](#).

Examples

```
# set grid.prompt(TRUE) to see each plot separately (click graphics window)
data(iris)
a.resetplotparams()
print(xyplot(Sepal.Width ~ Sepal.Length|Species, data=iris,
             panel=panel.xy, type="p,grid,v,lqs,r.pred,loess"))
print(xyplot(Sepal.Width ~ Sepal.Length, data=iris, groups=Species,
             panel=panel.superpose, panel.groups=panel.xy,
             type=c("p","grid","v","loess"), unicolor=TRUE))
```

plotf

Plot to multiple devices

Description

Sends the same plot to one or several devices in a uniform manner. Helpful for preparing a script that can be configured to produce graphics in several different formats, e.g. small bitmaps for web pages, large bitmaps for slides, and pdf, eps or wmf for inclusion in documents.

Usage

```
plotf(theplot, file="plot%03d", type=c("active"),
      size=c(6,4,10,96), prepare=a.resetplotparams, ...)
```

Arguments

<code>theplot</code>	An object representing the plot. This can either be an expression such as <code>expression(plot(x), points(pp))</code> or a Lattice object such as <code>xyplot(y~x)</code> . The object will be evaluated (if an expression) or printed (if a Lattice object) once on each device to be used. Note that this repeated processing may be inefficient if the plotting process itself (rather than just the resulting plot) is extremely complex. In such (rare) cases, you may want to consider manually using <code>dev.print</code> instead. <code>plotf</code> uses repeated processing rather than <code>dev.print</code> to ensure that the plotting function works as intended even if it discriminates between different devices.
<code>file</code>	The basename of the file to be produced. A suitable suffix of <code>.eps</code> , <code>.ps</code> , <code>.pdf</code> , <code>.png</code> , <code>.jpg</code> , or <code>.wmf</code> , respectively, will be appended for the actual files produced. The filename is ignored for the "active" device.
<code>type</code>	A character vector in which each entry represents one device to be used. Allowed values are "active", "current", "eps", "ps", "pdf", "png", "jpg", "jpeg", "wmf" and they represent the use of <code>dev.cur</code> , <code>dev.cur</code> , <code>dev.copy2eps</code> , <code>postscript</code> , <code>pdf</code> , <code>png</code> , <code>jpeg</code> , <code>jpeg</code> , and <code>win.metafile</code> , respectively. "eps" is ignored unless "active" (or "current") is also present.
<code>size</code>	A numeric vector of 1 to 4 components to which missing values will be added from the default value. Components 1 and 2 are width and height (in inches) of the plot to be produced. They will be supplied to the devices' <code>width</code> and <code>height</code> arguments, respectively. Component 3 is the base character size and will be supplied to the <code>pointsize</code> argument. For the bitmap devices, it will be corrected for resolution (which is indicated (in dpi) by component 4), because those devices always assume a resolution of 72dpi. Component 4 will also be supplied as the <code>res</code> argument to the bitmap devices.
<code>prepare</code>	A function without parameters that will be called each time after a new device has been opened and before the plot is produced.
<code>...</code>	All further arguments will be handed over to each(!) of the device opening functions

Details

The function uses `on.exit` to close a device correctly even if the plot function fails. `png` is almost always preferable to `jpeg`. Note that expressions can contain multiple calls by separating them with commas.

Value

`invisible()`

Author(s)

Lutz Prechelt (prechelt@inf.fu-berlin.de)

See Also

[a.resetplotparams](#), [dev.cur](#), [dev.copy2eps](#), [postscript](#), [pdf](#), [png](#), [jpeg](#).

Examples

```
x = runif(40)*4
y = x + runif(40)
plotf(expression(plot(x, y, type="p"))) # plots to active device only
plotf(xyplot(y~x), "myplot", type=c("active", "png", "pdf"), size=c(3,3))
```

plotheelpers

Plot helper functions

Description

`l.resetparams()` sets colors and plotting parameters suitable for document production.

`plotfit(fit, ...)` plots an `lm` object using `mfrow=c(2,3)`

`prepanel.0` Lattice `prepanel` function that forces that zero be included on both axes. Does nothing for non-numeric axes (such as the factor axis in `bwplot`).

Details

Type the name of a function to see its source code for details.

Author(s)

Lutz Prechelt (prechelt@inf.fu-berlin.de)

See Also

[trellis.par.set](#), [plot.lm](#), [Lattice](#), [prepanel.lm](#), [line](#).

Examples

```
data(iris)
a.resetplotparams()
print(bwplot(Sepal.Length, data=iris, panel=panel.bwstrip,
            prepanel=prepanel.0))
plotfit(lm(Sepal.Length~Sepal.Width, data=iris))
```

Description

`a.findcorrelations(df, vars1, vars2, min.cor)` checks pairs of variables for correlation. Returns a sorted vector of pairwise correlations whose absolute value is greater-or-equal `min.cor`. Uses all pairs of variables from dataframe `df` mentioned in character vectors (`vars1` x `vars2`), except that it returns only the lower triangle of that matrix if `vars2` is missing. Assesses the variable itself and also the corresponding rank vector (for rank correlations) Value pairs containing NAs are ignored in each `cor` call individually.

`a.iqr(x)` computes the inter-quartile range of numeric vector `x`.

`a.proportion.test(x1, x2, y1, y2)` tests the proportion `x1/x2` against `y1/y2` using both Fisher exact p and Chi-squared test, prints results of both and returns results of Chi-squared. If `totals=TRUE`, tests `x1/(x2-x1)` against `y1/(y2-y1)` instead. The arguments must be scalars, not vectors.

`a.qr(x)` computes the quartile ratio (`q75/q25`) of numeric vector `x`.

`a.rankval(x)` ranks the values from vector `x` using `rank()` but leaves NAs as NAs.

`a.showextremes(df, vars, largest, showalso)` shows the records from data frame `df` that contain the `largest` largest values (or the `largest` smallest if `largest` is negative) with respect to each of `df`'s variables whose name is mentioned in `vars` (which is a character vector). The extreme records for each of these variables are found; from each record, only these variables plus those whose name is mentioned in character vector `showalso` are shown.

Usage

```
a.findcorrelations(df, vars1=names(df), vars2=vars1, min.cor=0.5)
```

```
a.iqr(x)
```

```
a.proportion.test(x1, x2, y1, y2, totals=FALSE)
```

```
a.qr(x)
```

```
a.rankval(x)
```

```
a.showextremes(df, vars, largest=5, showalso=NULL)
```

Arguments

<code>df</code>	a dataframe, see description above.
<code>vars1</code>	a character vector, see description above.
<code>vars2</code>	a character vector, see description above.
<code>min.cor</code>	a number in range 0..1, see description above.

x	a vector, see description above.
x1	see description above.
x2	see description above.
y1	see description above.
y2	see description above.
totals	a logical value, see description above.
vars	a character vector, see description above.
largest	an integer, see description above.
showalso	a character vector, see description above.

Author(s)

Lutz Prechelt (prechelt@inf.fu-berlin.de)

Examples

```
data(iris)
summary(iris$Sepal.Width)
a.iqr(iris$Sepal.Width)
a.qr(iris$Sepal.Width)
a.findcorrelations(iris)
a.proportion.test(7,11, 17,19)
a.showextremes(iris, c("Sepal.Width","Sepal.Length"), largest = -5,
                showalso = c("Petal.Length"))
```

Index

- *Topic **aplot**
 - panel.bwstrip, 3
 - panel.xy, 5
- *Topic **color**
 - plotheelpers, 9
- *Topic **device**
 - plotf, 7
- *Topic **dplot**
 - plotheelpers, 9
- *Topic **manip**
 - misc helpers, 1
 - stats helpers, 9

- a.findcorrelations(*stats helpers*), 9
- a.iqr(*stats helpers*), 9
- a.proportion.test(*stats helpers*), 9
- a.qr(*stats helpers*), 9
- a.rankval(*stats helpers*), 9
- a.resetplotparams, 5, 7, 8
- a.resetplotparams(*plotheelpers*), 9
- a.showextremes(*stats helpers*), 9

- boxplot.stats, 3

- cor, 6

- density, 4
- dev.copy2eps, 8
- dev.cur, 8

- eqc(*misc helpers*), 1
- equal.count, 1, 2

- grep, 1, 2
- grep.b(*misc helpers*), 1
- grep.s(*misc helpers*), 1

- ifelse, 2

- jpeg, 8

- Lattice, 9
- lm, 7, 9
- lqs, 7

- mad, 4
- misc helpers, 1

- or.else(*misc helpers*), 1
- order, 2
- orderavg(*misc helpers*), 1

- panel.bwplot, 3, 4
- panel.bwstrip, 3
- panel.lminterval(*panel.xy*), 5
- panel.loess, 7
- panel.superpose, 7
- panel.xy, 5, 5
- panel.xyplot, 5-7
- pdf, 8
- plot, 6
- plot.lm, 9
- plotf, 5, 7
- plotfit(*plotheelpers*), 9
- plotheelpers, 9
- png, 8
- postscript, 8
- prepanel.0(*plotheelpers*), 9
- prepanel.lmline, 9
- printn(*misc helpers*), 1

- stats helpers, 9

- table, 2
- tableNA(*misc helpers*), 1
- traceback, 2
- tracebck(*misc helpers*), 1
- trellis.par.get, 7
- trellis.par.set, 9