Package 'plgraphics'

July 23, 2025

Type Package

Title User Oriented Plotting Functions

Version 1.2

Imports stats, utils, graphics, grDevices, chron, survival, MASS, lme4

Suggests knitr

Depends R (>= 3.5.0)

Description Plots with high flexibility and easy handling, including informative regression diagnostics for many models.

License GPL-2

LazyLoad yes

VignetteBuilder knitr

NeedsCompilation no

Author Werner A. Stahel [aut, cre], Martin Maechler [ctb] (ORCID: <https://orcid.org/0000-0002-8685-9910>)

Maintainer Werner A. Stahel <stahel@stat.math.ethz.ch>

Repository CRAN

Date/Publication 2023-09-29 13:42:38 UTC

Contents

`asinp` *arc sine Transformation*

Description

Calculates the sqrt arc sine of x/100, rescaled to be in the unit interval.
This transformation is useful for analyzing percentages or proportions of any kind.

Usage

```
asinp(x)
```

Arguments

`x`	vector of data values

Value

vector of transformed values

Note

This very simple function is provided in order to simplify formulas. It has an attribute `"inverse"` that contains the inverse function, see example.

Author(s)

Werner A. Stahel, ETH Zurich

Examples

```
asinp(seq(0,100,10))
( y <- asinp(c(1,50,90,95,99)) )
attr(asinp, "inverse")(y)
```

`charSize` *Adjust character size to number of observations*

Description

Adjusts the character size `cex` to number of observations

Usage

```
charSize(n)
```

Arguments

`n`	number of observations

Details

The function simply applies `min(1.5/log10(n), 2)`

Value

A scalar, defining `cex`

Author(s)

Werner A. Stahel

Examples

```
charSize(20)
for (n in c(10,20,50,100,1000))  print(c(n,charSize(n)))
```

clipat *Clip Data Outside a Range*

Description

Drop values outside a given range

Usage

```
clipat(x, range=NULL, clipped=NULL)
```

Arguments

x	vector of data to be clipped at range
range	range, a numerical vector of 2 elements
clipped	if NULL, the clipped data will be dropped. Otherwise, they will be replaced by clipped, which is typically set to NA. If clipped is numerical of length 2, the elements of x clipped below are set to clipped[1], those clipped by range[2], by clipped[2]. Therefore, if clipped equals range, x will be "Winsorized".

Value

As the input x, with pertinent elements dropped or replaced

Author(s)

Werner A, Stahel

Examples

```
clipat(rnorm(10,8,2), c(10,20), clipped=NA)
```

colorpale *determine more pale colors for given colors*

Description

Finds colors that are 'equivalent' to the colors given as the first argument, but more pale or less pale

Usage

```
colorpale(col = NA, pale = NULL, rgb = FALSE, ...)
```

Arguments

col	a color or a vector of colors for which the pale version should be found
pale	number between -1 and 1 determining how much paler the result should be. If =0, the original color, col will be returned unchanged (but in the 'rgb' or 'hexadecimal' form). If =1 or -1, the result is white (#FFFFFF) or black, respectively.
rgb	should result be expressed in 'rgb' form? If FALSE, it will be in hexadecimal form.
...	further arguments passed on to rgb if rgb is FALSE

Details

The function increases rgb coordinates of colors 'proportionally': crgb <- t(col2rgb(col)/255); rgb(1 - pale * (1 - crgb))

Value

character vector: names of colors to be used as color argument for graphical functions.

Author(s)

Werner A. Stahel, ETH Zurich

See Also

rgb

Examples

```
( t.col <- colorpale(c("red","blue")) )
plot(0:6, type="h", col=c("black","red","blue",t.col, colorpale(t.col)), lwd=5)
```

---

colors *colors used by plgraphics*

Description

Vectors of color names to be used, mainly for distinguishing groups

Usage

```
c.colors
```

Arguments

none

Value

vector of color names

Author(s)

Werner A. Stahel, ETH Zurich

Examples

```
c.colors
```

---

condquant *Quantiles of a Conditional Distribution*

Description

Calculates quantiles of a conditional distribution, as well as corresponding random numbers. The condtion is simply to restrict the distribution (given by `dist`) to a range (given by `x`)

Usage

```
condquant(x, dist = "normal", mu = 0, sigma = 1, randomrange = 0.9)
```

Arguments

`x`	matrix with 2 columns or vector of length 2 giving the limits for the conditional distribution
`dist`	(unconditional) distribution. Currently, only `normal` (or `gaussian`), `logistic` and `revgumbel` (reverse-Gumbel, distribution of the logarithm of a Weibull variable) are implemented.
`mu`, `sigma`	locarion and scale parameter of the distribution
`randomrange`	random numbers from the conditional distribution are drawn for the inner `100*randomrange` percent of the suitable p-range. This avoids random extreme outliers and points close to the limit of the intervals on which they are conditioned.

Value

Matrix consisting of a row for each row of `x` for which x[,1] differs from x[,2] and the following columns:

`median`	Median
`lowq`, `uppq`	lower and upper quartiles
`random`	random number according to the conditional distribution (one for each row)
`prob`	probability of the condition being true
`index`	(row) index of the corresponding entry in the input 'x'

Attribute `distribution` comprises the arguments `dist`, `mu`, `sigma`.

Author(s)

Werner A. Stahel, Seminar for Statistics, ETH Zurich

Examples

```
condquant(cbind(seq(-2,1),c(0,1,Inf,1)))
```

---

d.babysurvival *Survival of Premature Infants*

Description

Survival of Premature Infants to be modeled using 5 potential explanatory variables.

Usage

```
data("d.babysurvival")
data("d.babysurvGr")
```

Format

`d.babysurvival`: A data frame with 246 observations on the following 6 variables.

`Survival` binary, 1 means the infant survived

`Weight` birth weight [g]

`Age` pregnancy in weeks

`Apgar1` A score indication the fitness of the infant at birth, scores 0 to 9

`Apgar5` alternative score

`pH` blood pH

`d.babysurvGr`: Grouped data: Number of Infants that died and survived for each class of birth weight.

`n` Number of infants in the weight class

`Survival.0`, `Survivl.1` Number of infants that died and survived, respectively

`Weight` birth weight

Source

Hibbard (1986)

Examples

```
data(d.babysurvival)
summary(d.babysurvival)
rr <- glm(Survival~Weight+Age+Apgar1, data=d.babysurvival, family="binomial")
plregr(rr, xvar= ~Age+Apgar1)
```

d.birthrates *Birthrates in Swiss Districts*

Description

Standardized fertility measure and socio-economic indicators for each of 182 districts of Switzerland at about 1888. This is an extended version of the swiss dataset of standard R.

Usage

```
data("d.birthrates")
data("d.birthratesVars")
```

Format

d.birthrates: A data frame with 182 observations on the following 25 variables.

fertility Common standardizedfertility measure, see details

fertTotal Alternative fertility measure

infantMort Infant mortality

catholic percentage of members of the catholic church

single24 percentage of women aged 20-24 who are single

single49 percentage of women aged 45-49 who are single

eAgric Proportion male labor force in agriculture

eIndustry Proportion male labor force in industry

eCommerce Proportion male labor force in trade

eTransport Proportion male labor force in transportation

eAdmin Proportion male labor force in public service

german percentage of German

french percentage of French

italian percentage of Italian

romansh percentage of Romansh

gradeHigh Prop. high grade in draftees exam

gradeLow Propr. low grade in draftees exma

educHigh Prop. draftees with > primary educ.

bornLocal Proportion living in commune of birth

bornForeign Proportion born in foreign country

sexratio Sex ratio (M/F)

canton Canton Name

district District Name

altitude altitude in three categories: low, medium, high

language dominating language: german, french, italian, romansh

d.birthratesVars: Data.frame that contains the descriptions of the variables just read.

Details

?swiss says:
(paraphrasing Mosteller and Tukey):
Switzerland, in 1888, was entering a period known as the 'demographic transition'; i.e., its fertility was beginning to fall from the high level typical of underdeveloped countries.

The exact definition of fertility is as follows.
fertility = 100 * B_l/ sum m_i f_i, where
B_l = annual legitimate births, m_i = the number of married women in age interval i, and f_i = the fertility Hutterite women in the same age interval.
"Hutterite women" are women in a population that is known to be extremely fertile.
Stillbirths are included.

Source

https://opr.princeton.edu/archive/pefp/switz.aspx

References

see source

Examples

```
data(d.birthrates)
## maybe str(d.birthrates) ; plot(d.birthrates) ...
```

---

d.blast *Blasting for a tunnel*

Description

Blasting causes tremor in buildings, which can lead to damages. This dataset shows the relation between tremor and distance and charge of blasting.

Usage

```
data("d.blast")
```

Format

A data frame with 388 observations on the following 7 variables.

`no` Identification of the date and time

`date` Date in Date format. (The day and month are correct, the year is a wild guess.)

`datetime` Date and time in the format '%d.%m. %H:%M'

`device` Number of measuring device, 1 to 4

`charge` Charge of blast

`distance` Distance between blasting and location of measurement

`tremor` Tremor energy (target variable)

`location` Code for location of the building, `loc1` to `loc8`

Details

The charge of the blasting should be controled in order to avoid tremors that exceed a threshold. This dataset can be used to establish the suitable rule: For a given `distance`, how large can `charge` be in order to avoid exceedance of the threshold?

Source

Basler and Hoffmann AG, Zurich

Examples

```
data(d.blast)
showd(d.blast)

plyx(tremor~distance, psize=charge, data=d.blast)

rr <-  lm(logst(tremor)~location+log10(distance)+log10(charge), data=d.blast)
plregr(rr)

t.date <- as.POSIXlt(paste("1999",d.blast$datetime,sep="."),
                     format='%Y.%d.%m. %H:%M')
```

---

d.fossileShapes *Coccolith Abundance and Environmental Variables*

Description

The abundance of cocolith shells can be used to infer environmental conditions in epochs corresponding to earlier epochs. This data set contains the core location, the relative abundance of Gephyrocapsa morphotypes and the sea surface temperatures from all deep see cores used in this study.

Usage

```
data("d.fossileShapes")
data("d.fossileSamples")
```

Format

d.fossilShapes: A data frame with 5864 observations on the following 15 variables:
Identification and location of the sample:

Sample Identification number of the sample

Sname Identification code

Magnification (technical)

Shape features and recommended transformations:

Angle bridge angle

Length, Width lengtha and width of the shell

CLength, CWidth length and width of the 'central area'

Cratio ratio between width and length of the central area

sAngle sqrt of Angle

lLength log10(Length)

rWidth, rCLength, rCWidth relative measures, percentage of Length

Cratio CWidth/Clength

ShapeClass shape class as defined in the cited paper, classes ar CM < CC < CT < CO < CE < CL

d.fossilSamples: A data frame with 108 observations on the following 32 variables:
Identification and location:

Sample Identification number of the sample (as above)

Sname Identification code

Latitude, Longitude Coordinates of the location

Region Ocean: Pacific, Atlantic, Indian.Ocean

SDepth sample depth below soil surface [cm]

WDepth Water depth [m]

N number of specimen measured

Shape features as above, averaged. (This is the reason for introducing transformed variables above: The transformed values are averaged.)

CM, CC, CT, CO, CE, CL percentages of shape classes in the sample

Environment:

SST Sea Surface Temperature, mean, [deg C]

SST.Spring, SST.Summer, SST.Fall, SST.Winter ... in each season

Chlorophyll, lChlorophyll Chlorophyll content [microgram/L] and log10 of it

Salinity Salinity of the sea water

Details

The paradigm of research associated with this dataset is the following: Datasets of this kind are used to establish the relationship between the shell shapes of cocoliths (species Gephyrocapsa) from the most recent sediment layer with actual environmental conditions. This relationship is then used to infer environmental conditions of earlier epochs from the shell shapes from the corresponding layers.

The analysis presented in the paper cited below consisted of first introducing classes of shells based on the shapes and then use the relative abundance of the classes to predict the environmental conditions.

Source

J\"org Bollmann, Jorijntje Henderiks and Bernhard Brabec (2002). Global calibration of Gephyrocapsa coccolith abundance in Holocene sediments for paleotemperature assessment. Paleoceanography, 17(3), 1035

References

J\"org Bollmann (1997). Morphology and biogeography of Gephyrocapsa coccoliths in Holocene sediments. Marine Micropaleontology, 29, 319-350

Examples

```
data(d.fossileShapes)
names(d.fossileShapes)

data(d.fossileSamples)
plyx(sqrt(Angle) ~ SST, data=d.fossileSamples)
```

---

d.pollZH16 *Air Pollution Monitoring in Zurich*

Description

Hourly air pollution measurements from a station in the city center of Zurich, in a courtyard, for the whole year 2016, resulting in 8784 measurements of the two pollution variables ozone and nitrogen dioxyde, the three weather variables temperature, radiation and precipitation, and 8 variables characterizing the date.

`pollZH16d` is the subset of measurements for `hour=15`.

Usage

```
data("d.pollZH16")
```

Format

A data frame with 8784 observations on the following 13 variables.

`date` date of the measurement

`hour` hour of the measurement

`O3` Ozone

`NO2` Nitroge dioxyde

`temp` temperature

`rad` solar radiation

`prec` precipitation

`dateshort` two letter identification of the day. A-L encodes the month; 1-9, a-x encodes the day within month.

`weekday` day of the week

`month` month

`sumhalf` indicator for summer half year (April to Sept)

`sunday` logical: indicator for Sunday

`daytype` a factor with levels `work` for working day, `Sat` and `Sun`

Note

Legal threshold for NO2 in the EU: The threshold of 200 micrograms/m3 must not be exceeded by more than 18 hourly measurements per year.
Source: Umweltbundesamt, Germany http://www.umweltbundesamt.de/daten/luftbelastung/stickstoffdioxid-belastung#textpart-2

Source

Bundesamt fur Umwelt (BAFU), Schw. Eidgenossenschaft https://www.bafu.admin.ch/bafu/de/home/themen/luft/zustand/da
nabel.html
The data set has been generated by downloading the files for the individual variables, converting the entries with `hour==24` to `hour==0` of the following day and restricting the data to year 2016.

Examples

```
data(d.pollZH16)
dp <- d.pollZH16
names(dp)

dp$date <- gendateaxis(date=dp$date, hour=dp$hour)

plyx(O3+NO2~date, data=dp, subset= month=="May", type="l")

dp$summer <- dp$month %in% c("Jun","Jul","Aug")
dp$daylight <- dp$hour>8 & dp$hour<17
plmatrix(O3~temp+logst(rad)+logst(prec), data=dp,
         subset = summer & daylight)
```

d.river *Chemical Compounds in a Swiss River, Time Series*

Description

This time series of chemical concentrations can be used to research the activities of photosynthesis and respiration in a river.

Usage

```
data("d.river")
```

Format

A time series with 9792 observations (10 minutes interval) on the following 12 variables.

`date` Date of the observation, class `Date`

`hour` Hour

`pH` pH

`O2` concentration of Oxygen

`O2S` Oxygen saturation value

`T` Temperature [deg C]

`H2CO3` Carbon dioxide concentration in the water

`CO2atm` Carbon dioxide concentration in the atmosphere

`Q` flow

`su` sunshine

`pr` precipitation

`ra` radiation

Note

This is not a time series in the sense of `ts` of R. The date-time information is contained in the variables `date` and `hour`.

Source

The measurements have been collected in the river Glatt near Zurich.

Examples

```
data(d.river)
range(d.river$date)
t.i <- d.river$date < as.Date("2010-03-31")

plyx(~date, ~O2, data=d.river, subset=t.i & hour==14, smooth=FALSE)

d.river$Date <- gendateaxis(d.river$date, hour=d.river$hour)
plyx(O2~Date, data=d.river, subset=t.i, type="l")

plyx(O2+T+ra~Date, data=d.river, subset=t.i & hour==14,
  smooth.par=0.5, smooth.xtrim=0.03, ycol=c(O2="blue",ra="red"))
```

---

deparseCond *Analyze formula with conditional variables*

Description

Check if `formula` is valid and, if it contains a `|` character, idenitfy regressors and conditional variables

Usage

```
deparseCond(formula)
```

Arguments

`formula`	A model formula, possibly containing a `|` character that introduces terms describing conditions

Value

Returns the formula with the following attributes:

`y`	"vertical" (response) variable(s)
`x`	"horizontal" (regressor) variable(s)
`a`	(first) conditional variable, if any
`b`	second conditional variable, if any

Note

This function is typically used for conditional plots and mixed models

Author(s)

Werner A. Stahel

Examples

```
deparseCond(yy ~ xx)
deparseCond(yy ~ xx | aa + bb)
deparseCond(y1 + y2 ~ x1 + log(x2) | sqrt(quantity))

plyx(Sepal.Width~Sepal.Length | Species, data=iris)
```

---

doc *Define and obtain the doc or tit attribute*

Description

The attributes doc and tit describe an object, typically a data frame or a model. tit should be a short description (title), doc should contain all documentation useful to identify the origin and the changes made to the object.
The doc and tit functions set them and extract these attributes.

Usage

```
doc(x)
tit(x)
doc(x) <- value
tit(x) <- value
```

Arguments

x	object to which the doc or tit attribute should be attached or from which it is obtained
value	character vector (doc) or string (tit) to be stored

Details

Plotting and printing functions may search for the tit attribute or even for the doc attribute, depending on c.env$docout.

doc(x) <- text will append the existing doc(x) text to the new text unless its first element equals (the first element of) text. (This avoids piling up the same line by unintended multiple call to doc(x) <- value with the same value.) If the first element of text equals "^", the first element of doc(x) is dropped. tit(x) <- string replaces tit(x) with string.

Value

doc and tit return the respective attributes of object x

Author(s)

Werner A. Stahel, ETH Zurich

Examples

```
data(d.blast)
doc(d.blast)
doc(d.blast) <- "I will use this dataset in class soon."
doc(d.blast)
```

`dropdata` *Drop Observations from a Data.frame*

Description

Allows for dropping observations (rows) determined by row names or factor levels from a data.frame or matrix.

Usage

```
dropdata(data, rowid = NULL, incol = "row.names", colid = NULL)
```

Arguments

`data`	a data.frame of matrix
`rowid`	vector of character strings identifying the rows to be dropped
`incol`	name or index of the column used to identify the observations (rows)
`colid`	vector of character strings identifying the columns to be dropped

Value

The data.frame or matrix without the dropped observations and/or variables. Attributes are passed on.

Note

Ordinary subsetting by `[...,...]` drops attributes like `doc` or `tit`. Furthermore, the convenient way to drop rows or columns by giving negative indices to `[...,...]` cannot be used with names of rows or columns.

Author(s)

Werner A. Stahel, ETH Zurich

See Also

`subset`

Examples

```
dd <- data.frame(rbind(a=1:3,b=4:6,c=7:9,d=10:12))
dropdata(dd,"b")
dropdata(dd, col="X3")

d1 <- dropdata(dd,"d")
d2 <- dropdata(d1,"b")
naresid(attr(d2,"na.action"),as.matrix(d2))

dropdata(letters, 3:5)
```

---

dropNA *drop or replace NA values*

Description

dropNA returns the vector 'x', without elements that are NA or NaN or, if 'inf' is TRUE, equal to Inf or -Inf. replaceNA replaces these values by values from the second argument

Usage

```
dropNA(x, inf = TRUE)
replaceNA(x, na, inf = TRUE)
```

Arguments

x	vector from which the non-real values should be dropped or replaced
na	replacement or vector from which the replacing values are taken.
inf	logical: should 'Inf' and '-Inf' be considered "non-real"?

Value

For dropNA: Vector containing the 'real' values of 'x' only
For replaceNA: Vector with 'non-real' values replaced by the respective elements of na.

Note

The differences to 'na.omit(x)' are: 'Inf' and '-Inf' are also dropped, unless 'inf==FALSE'.\ no attribute 'na.action' is appended.

Author(s)

Werner A. Stahel

See Also

na.omit, sumNA, ifelse

Examples

```
dd <- c(1, NA, 0/0, 4, -1/0, 6)
dropNA(dd)
na.omit(dd)

replaceNA(dd, 99)
replaceNA(dd, 100+1:6)
```

`fitcomp` *Component Effects for a Model Fit*

Description

Determines effects of varying each of the given variables while all others are held constant. This function is mainly used to produce plots of residuals versus explanatory variables, also showing component effects. It can handle a multivariate response fitted by `lm`.

Usage

```
fitcomp(object, data = NULL, vars=NULL, transformed=FALSE, se = FALSE,
  xm = NULL, xfromdata = FALSE, noexpand=NULL, nxcomp = 51)
```

Arguments

`object`	a model fit, result of a fitting function
`data`	data frame in which the variables are found. If not provided, it is obtained from `object`.
`vars`	character vector of names of variables for which components are required. Only variables that appear in `data` will be used. If `NULL` (the default), all variables in `data` are used.
`transformed`	logical: should components be calculated for transformed explanatory variables? If `TRUE`, the variables are transformed as implied by the model.
`se`	if TRUE, standard errors will be returned
`xm`	named vector of values of the fixed (central) point from which the individual variables are varied in turn. Defaults to the componentwise median of quantitative variables and the modes of factors.
`xfromdata`	if TRUE, the components effects will be evaluated for the data values in `data`. Otherwise, the range of each numerical variable is filled with `nxcomp` equidistant points, whereas for factors, all levels are used. This is useful for residual plots with component effects.
`noexpand`	vector determining which variables should not be "filled in", probably because they are used like factors. Either a character vector of variable names or a vector of logical or numerical values with names, in which case the names corresponding to positive values will be identified.
`nxcomp`	number of points used for each (quantitative) variable if `xfromdata` is `FALSE`

Details

The component effect is defined as the curve of fitted values obtained by varying the explanatory variable or term, keeping all the other variables (terms) at their "central value" `xm` (the mean of continuous variables and the mode of factors).

Value

A list consisting of

`comp`	component effects. A matrix, unless the response is multivariate, in which case it will be a 3-dimensional array.
`x`	the values of the x variables for which the effects have been calculated
`xm`	the values at which the x variables are held fixed while one of them is varied
`se`	standard errors of the component effects, if required by the argument `se`. Same structure as `comp`

Author(s)

Werner A. Stahel, ETH Zurich

See Also

`predict`

Examples

```
data(d.blast)
t.r <- lm(log10(tremor)~location+log10(distance)+log10(charge), data=d.blast)
t.fc <- fitcomp(t.r,se=TRUE)
t.fc$comp[1:10,]
```

---

gendateaxis *Generate a variable expressing time with its attributes for plotting*

Description

`gendateaxis` generates suitable attributes for plotting a date or time variable.
`gendate` generates a date variable and is an extension of `as.POSIXct`.

Usage

```
gendate(date = NULL, year = 2000, month = 1, day = 1, hour = 0,
  min = 0, sec = 0, data = NULL, format = "y-m-d", origin = NULL)

gendateaxis(date = NULL, year = 2000, month = 1, day = 1, hour = 0,
  min = 0, sec = 0, data = NULL, format = "y-m-d", origin = NULL,
  ploptions=NULL)
```

Arguments

date	vector of class dates or chron or vector to be converted into such an object. May also be the name of such a variable contained in data. If date has class dates or chron, the arguments year, month and day are ignored.
year, month, day, hour, min, sec	numeric vectors giving the year, month, day of month, hour, minute, second – or the name of such a variable contained in data. day, hour, min, sec can be fractional, see Details. If these arguments are used, the supersede the respective parts in date.
data	data.frame, where variables can be found
format	format for date in case that the latter is a character vector
origin	year of origin for dates, defaults to ploptions("date.origin")
ploptions	list pl options, generated by pl.control

Details

If hour is fractional, e.g., 6.2, the fraction is respected, that is, it will be the same as time 06:12. If min is also given, the fraction of hour is ignored. Similar for day and min.
If hour is >=24, the day is augmented by hour%/%24 and the hour is set to hour%%24. Similar for min and sec.

Value

For gendate, a vector of times in POSIXct format.
For gendateaxis, this is augmented by the attribute

numvalues	numerical values used for plotting. If years, months or days vary in the data, the units are days. Otherwise, they are hours, minutes, or seconds, depending on the highest category that varies.

Unless the dates only cover one of the categories (only years differ, or only months, ...), the following plotting attributes are added:

ticksat	vector where tickmarks are shown. It contains its own attribute small if secondary ticks are suitable.
ticklabels	May be years, quarters, month names, days, ...
ticklabelsat	vecor of coordinates to place the ticklabels
label	equals "", since the time scale makes it clear enough that the axis represents time.

Author(s)

Werner A. Stahel

See Also

genvarattributes, axis.Date

Examples

```
## call gendateaxis without 'real' data
tt <- gendate(year=rep(2010:2012, each=12), month=rep(1:12, 3))
ta <- gendateaxis(tt)

## ... derived from data
data(d.river)
d.river$dt <- gendateaxis(date="date", hour="hour", data=d.river)
plyx(O2~dt, data=d.river, subset=months(date)!="Sep")
plyx(O2~dt, data=d.river[months(d.river$date)!="Sep",])
plyx(O2~dt, data=d.river, subset=1:1000)
```

---

gensmooth *Smooth: wrapper function*

Description

Generate fits of a smoothing function for multiple y's. Smooths can be calculated within given groups.

Usage

```
gensmooth(x, y, band = FALSE, power = 1, resid = "difference",
  weight = NULL, plargs=NULL, ploptions=NULL, ...)
```

Arguments

x	vector of x values.
y	vector or matrix of y values.
band	logical: Should a band consisting of low and high smooth be calculated? It will only be calculated for the first column of y.
power	y will be raised to power before smoothing. Results will be back-transformed. (Useful for smoothing absolute values for a 'scale plot', for which power=0.5 is recommended.)
resid	Which residuals be calculated? resid=1 or ="difference" means usual residuals; resid=2 or ="ratio" means $y_i/\hat y_i$, which is useful to get scaled y's (regression residuals) according to a smooth fit in the scale plot.
weight	weights of observations, may also be passed by a variable .smoothWeights. in the data set plargs$pldata
plargs, ploptions	result of calling pl.control. The component plargs$pdata may contain smooth.weight and smooth.group, and ploptions specifies smoothPar and smoothIter. All of these may be used by the smoothing function.
...	Further arguments, passed to the smoothing function.

Details

This function is useful for generating the smooths enhancing residual plots. It generates a smooth for a single x variable and multiple y's. It is also used to draw smooths from simulated residuals.

NA's in either x or any column of y cause dropping the observation (equivalent to `na.omit`).

The smoothing function used to produce the smooth is `smoothRegr`, which relies `loess`, by default. This may be changed via `ploptions(smooth.function = func)` where `func` is a smoothing function with the same arguments as `smoothRegr`.

The result of the smoothing function may carry an attribute `xtrim`. This regulates if the fitted values corresponding to extreme x values will be suppressed when plotting: The number of extreme x values corresponding to `ploptions("smooth.xtrim")` will be multiplied by this attribute to obtain the number of extreme points suppressed at each end. If the smoothing function is `smoothLm`, which fits a straight line, then trimming is suppressed since this function returns 0 as the `xtrim` attribute.

If `band` is `TRUE`, a vector of "low" and a vector of "high" smooth values will be calculated for the first column of y in the following way: Residuals are calculated as the diference between the observations and the respective smoothed values hat.s_i. Then a smooth is calculated for the square roots of the positive residuals, and the squared fitted values are added to the hat.s_i. (The transformation by square roots makes the distribution of the residuals more symmetric.) This defines the "high" smooth values. The construction of the "low" one is analogous. The resulting values of the two are stored in the list component `yband`, and `ybandindex` contains the information to which group ("low" or "high") the value belongs.

Value

A list with components:

`x`	vector of x values, sorted, within levels of `group` if grouping is actif.
`y`	matrix with 1 or more columns of corresponding fitted values of the smoothing.
`group`	grouping factor, sorted, if actif. `NULL` otherwise.
`index`	vector of indices of the argument `x` used for sorting. This is useful to relate the results to the input. Use `ysmoothed[value$index,] <- value$y` to get values corresponding to input `y`.
`xorig`	original x values
`ysmorig`	corresponding fitted values
`residuals`	if required by the argument `resid`, residuals from the smooth fit are provided in the original order, i.e. `value$resid[i,j]` corresponds to the input `value$y[i,j]`.

If `band==TRUE`,

`yband`	vector of low and high smoothed values (for the first column of `y`)
`ybandindex`	Indicator if `yband` is a high value

Note

This function is called by `plyx` and `plmatrix` when `smooth=T` is set, as well as by `plregr` applied to model objects. It is rarely needed to call it directly.
A band is generated only for the first columnn of `y` since the others are supposed to be simulated versions of the first one and do not need a band.

Author(s)

Werner A. Stahel, ETH Zurich

See Also

smoothRegr, plsmooth, plsmoothline

Examples

```
data(d.blast)
r.blast <-
  lm(log10(tremor)~location+log10(distance)+log10(charge), data=d.blast,
    na.action=na.exclude)
r.smooth <- gensmooth( fitted(r.blast), residuals(r.blast))
showd(r.smooth$y)
plot(fitted(r.blast), resid(r.blast), main="Tukey-Anscombe Plot")
abline(h=0)
lines(r.smooth$x,r.smooth$y, col="red")

## grouped data
t.plargs <- list(pdata=data.frame(".smooth.group."=d.blast$location))

r.smx <- gensmooth( d.blast$dist, residuals(r.blast), plargs=t.plargs)

plot(d.blast$dist, residuals(r.blast), main="Residuals against Regressor")
abline(h=0)
plsmoothline(r.smx, d.blast$dist, resid(r.blast), plargs=t.plargs)
## or, without using plsmoothlines:
## for (lg in 1:length(levels(r.smx$group))) {
##   li <- as.numeric(r.smx$group)==lg
##   lines(r.smx$x[li],r.smx$y[li], col=lg+1, lwd=3)
## }
```

---

genvarattributes *Generate or Set Variable Attributes for Plotting*

Description

genvarattributes generates attributes of variables that are useful for the plgraphics functions. It is called by pl.control.
setvarattributes modifies or sets such attributes.

Usage

```
genvarattributes(data, vnames = NULL, vcol = NULL, vlty = NULL, vpch = NULL,
  varlabel = NULL, innerrange = NULL, plscale = NULL, zeroline = NULL,
  replace=FALSE, ploptions = NULL, ...)

setvarattributes(data, attributes = NULL, list = NULL, ...)
```

Arguments

data	data.frame consisting of the variables (columns) to be characterized by their attributes
vnames	names of variables to be treated as y variables
vcol, vlty, vpch	color, line type and plotting character to be used when multiple y-s are plotted (in the sense of matplot)
varlabel	labels of the variables, in the case that the names of data are not appropriate.
innerrange	logical indicating whether inner plotting ranges should be determined and/or used. May also be the limits of the inner plotting range, if predetermined, see Details
plscale	plot scale: name of the function to be used for generating a plotting scale, like "log". A named character vector can be given, where the names correspond to variable names in data.
zeroline	value(s) for which a horizontal or vertical line will be drawn (in addition to the gridlines). The default is given by ploptions("zeroline").
ploptions	list containing the plotting elements needed to set the attributes
replace	logical: should existing attributes be replaced?
attributes	(for setvarattributes) is a list of lists. Its names identify the variables for which the attributes are set or modified. Each component is a list which is added to the existing attributes of the respective variable or replaces them if they already exist.
list	a list of attributes to be set. Each component must have a name giving the name of the variable attribute to be set, and be itself a list (or a vector). This list must have names that identify the variables in data for which the attributes are set. See examples to understand this.
...	further arguments, which will be collected and used as or added to list

Details

If the attribute innerrange is replaced, then plcoord is also replaced.

innerrange may be a named list of ranges with names corresponding to variables (not necessarily all of them), or a scalar vector of length 2 to be used as range for all the variables. It can also be a logical vector superseding the argument innerrange, either named (as just mentioned) or unnamed, to be repeated the appropriate number of times.

Value

Data.frame, returning the original values, but the variables are supplemented by the following attributes, where available:

nvalues	number of distinct values
innerrange	inner plotting range
plcoord	plotting coordinates
ticksat	tick marks for axis

varlabel	label to be used as axis label
zeroline	value(s) for which a horizontal or vertical line will be drawn (in addition to the gridlines)

Author(s)

Werner A. Stahel

See Also

par

Examples

```
data(d.blast)
dd <- genvarattributes(d.blast)
str(attributes(dd$tremor))

ddd <- setvarattributes(dd, list( tremor=list(ticksat=seq(0,24,2),
  ticklabelsat = seq(0,24,10), ticklabels=c("low","medium","high")) ) )
str(attributes(ddd$tremor))

data(d.river)
plyx(O2+H2CO3+T ~ date, data=d.river, subset=as.Date(date)<as.Date("2010-02-28"))
dd <- setvarattributes(d.river,
  list=list(vcol=c(O2="blue", T="red")), vpch=c(O2=1, T="T", H2CO3=5) )
attributes(dd$O2)
plyx(O2+H2CO3+T ~ date, data=d.river, subset=as.Date(date)<as.Date("2010-02-28"),
  plscale = c(O2="log", H2CO3="log") )
```

---

getmeth *get S3 method of a generic function*

Description

identical to getS3method

Usage

```
getmeth(fn, mt)
```

Arguments

fn	name of generic function, quoted or unquoted
mt	name of method, quoted or unquoted

Value

Source code of the method

Author(s)

Werner A. Stahel, ETH Zurich

See Also

getS3method

Examples

```
getmeth(simresiduals, glm)
```

---

getvariables *Extract Variables or Variable Names*

Description

getvarnames extracts the variables' names occurring in a formula, in raw form (as get_all_vars) or in transformed form (as model.frame does it).

getvariables collects variables from a data.frame

Usage

```
getvariables(formula, data = NULL, transformed = TRUE,
  envir = parent.frame(), ...)
getvarnames(formula, data = NULL, transformed = FALSE)
```

Arguments

formula	a model 'formula' or 'terms' object or an R object, or a character vector of variable names
data	a data.frame, list or environment (or object coercible by 'as.data.frame' to a data.frame), containing the variables in 'formula'. Neither a matrix nor an array will be accepted.
transformed	logical. If TRUE, variables will be extracted as transformed in formula, otherwise, untransformed variables are returned.
envir	environment in which the formula will be evaluated
...	further arguments such as data, weight, subset, offset used to create extra columns in the resulting data.frame, with names between dots such as '".offset."'

Value

For `getvarnames`: names of all variables (`transformed=FALSE`) or simple terms (`transformed=TRUE`), including the attributes

`xvar`	those from the right hand side of the formula
`yvar`	left hand side, if present
`yvar`	conditioning part, denoted after a `|` symbol in formula, if applicable

For `getvariables`: data.frame containing the extracted variables or simple terms, with the attributes of `getvarnames`

Author(s)

Werner A. Stahel

See Also

`model.frame`, `get_all_vars`

Examples

```
data(d.blast)
getvarnames(log10(tremor)~log10(distance)*log10(charge), data=d.blast)

dd <- getvariables(log10(tremor)~log10(distance)*log10(charge),
                   data=d.blast, by=location)
str(dd)
```

---

legendr *Add a Legend to a Plot*

Description

Adds a legend to a plot as does `legend`. This function just expresses the position relative to the range of the coordinates

Usage

```
legendr(x = 0.05, y = 0.95, legend, ...)
```

Arguments

`x`	position in horizontal direction, between 0 for left margin and 1 for right margin
`y`	position in vertical direction, between 0 for bottom margin and 1 for top margin
`legend`	text of the legend
`...`	arguments passed to `legend`

Value

See legend

Author(s)

Werner A. Stahel, ETH Zurich

See Also

legend

Examples

```
ts.plot(ldeaths, mdeaths, fdeaths,xlab="year", ylab="deaths", lty=c(1:3))
legendr(0.7,0.95, c("total","female","male"), lty=1:3)
```

---

leverage *Get leverage values*

Description

Extracts the leverage component of a fit object using the na.action component if available

Usage

```
leverage(object)
```

Arguments

object	an object containing a component fit$leverage and possibly a component fit$na.action

Details

The difference to hatvalues is that leverage does not call influence and therefore does not require residuals. It is therefore simpler and more widely applicable.

The function uses the qr decomposition of object. If necessary, it generate it.

The leverage is the squared Mahalanobis distance of the observation from the center of the design X (model.matrix) with "covariance" X^T X. If there are weights (object$weights), the weighted center and "covariance" are used, and the distances are multiplied by the weights. To obtain the distances in the latter case, "de-weight" the leverages by dividing them by the weights.

Value

The vector fit$leverage, possibly expanded by missing values if fit$na.action has class na.exclude

Author(s)

Werner A. Stahel, ETH Zurich

See Also

hat; hatvalues; influence

Examples

```
data(d.blast)
r.blast <-
  lm(log10(tremor)~location+log10(distance)+log10(charge), data=d.blast)
showd(leverage(r.blast))
```

---

linear.predictors *linear predictors from a (generalized) linear model*

Description

extracts the linear.predictor component of a model object, taking 'na.resid' into account, in analogy to 'residuals' or 'fitted.values'

Usage

```
linear.predictors(object)
```

Arguments

object	model fit

Value

vector (or, for models inheriting from 'multinom', matrix) of linear predictor values

Author(s)

Werner A. Stahel

See Also

fitted.values

Examples

```
## example from 'glm'
clotting <- data.frame(
u = c(5,10,15,20,30,40,60,80,100),
lot1 = c(118,58,NA,35,27,25,21,19,18), ## NA inserted instead of 42
lot2 = c(69,35,26,21,18,16,13,12,12))

r.gam  <- glm(lot1 ~ log(u), data = clotting, family = Gamma)
linear.predictors(r.gam)
## 8 elements; 3rd missing.
r.gex  <- glm(lot1 ~ log(u), data = clotting, family = Gamma,
              na.action=na.exclude)
linear.predictors(r.gex)
## 9 elements, third is NA
```

---

logst *Started Logarithmic Transformation*

Description

Transforms the data by a log10 transformation, modifying small and zero observations such that the transformation yields finite values.

Usage

```
logst(data, calib=data, threshold=NULL, mult = 1)
```

Arguments

`data`	a vector or matrix of data, which is to be transformed
`calib`	a vector or matrix of data used to calibrate the transformation(s), i.e., to determine the constant c needed
`threshold`	constant c that determines the transformation, possibly a vector with a value for each variable.
`mult`	a tuning constant affecting the transformation of small values, see Details

Details

Small values are determined by the threshold c. If not given by the argument `threshold`, then it is determined by the quartiles q_1 and q_3 of the non-zero data as those smaller than $c = q_1/(q_3/q_1)^{mult}$. The rationale is that for lognormal data, this constant identifies 2 percent of the data as small. Beyond this limit, the transformation continues linear with the derivative of the log curve at this point. See code for the formula.

The function chooses log10 rather than natural logs because they can be backtransformed relatively easily in the mind.

Value

the transformed data. The value c needed for the transformation is returned as `attr(.,"threshold")`.

Note

The names of the function alludes to Tudey's idea of "started logs".

Author(s)

Werner A. Stahel, ETH Zurich

Examples

```
dd <- c(seq(0,1,0.1),5*10^rnorm(100,0,0.2))
dd <- sort(dd)
r.dl <- logst(dd)
plot(dd, r.dl, type="l")
abline(v=attr(r.dl,"threshold"),lty=2)
```

---

markextremes *Adjust the default proportion of extreme points to be labeled to the number of observations*

Description

Adjusts the proportion of extreme points to be labeled to the number of observations. It is the default of the `ploption` `markextremes`.

Usage

```
markextremes(n)
```

Arguments

`n`	number of observations

Details

The function simply applies `ceiling(sqrt(n)/2)/n`.

Value

A scalar between 0 and 0.5

Author(s)

Werner A. Stahel

Examples

```
markextremes(20)
for (n in c(10,20,50,100,1000))  print(c(n,markextremes(n)))
```

`modarg` *Modify default arguments according to a named vector or list*

Description

Makes it easy to modify one or a few elements of a vector or list of default settings. This function is to be used within functions that contain vectors of control arguments such as colors for different elements of a plot

Usage

```
modarg(arg = NULL, default)
```

Arguments

`arg`	named vector or list of the elements that should override the settings in 'default'
`default`	named vector or list of default settings

Value

Same as the argument 'default' with elements replaced according to 'arg'. See the source code of `plmboxes.default` for a typical application.

Author(s)

Werner A. Stahel

Examples

```
modarg(c(b="B", c=0), list(a=4, b="bb", c=NA))

df <- ploptions("linewidth")
cbind(df, modarg(c(dot=1.4, dashLongDot=1.3), df))

## These statements lead to a warning:
modarg(c(b=2, d=6), c(a="4", b="bb", c=NA))
modarg(1:6, c(a="4", b="bb", c=NA))
```

months *strings for month and weekday names*

Description

Vectors of month and weekday names

Usage

```
c.months
c.mon
c.weekdays
c.wkd
```

Arguments

none

Value

character vector.
`c.months` contains the 12 month names.
`c.mon` same, abbreviated to 3 characters,
`c.weekdays` names of the 7 weekdays
`c.wkd` same, abbreviated to 3 characters,

Author(s)

Werner A. Stahel, ETH Zurich

Examples

```
c.weekdays[1:5]
```

nainf.exclude *Drop Rows Containing NA or Inf*

Description

Drops the rows of a data frame that contain an NA, an NaN, or an Inf value

Usage

```
nainf.exclude(object, ...)
```

Arguments

`object`	an R object, typically a data frame
`...`	further arguments special methods could require.

Details

This is a simple modification of `na.omit` and `na.exclude`

Value

The value is of the same type as the argument `object`, with possibly less elements.

Author(s)

Werner A. Stahel, ETH Zurich

See Also

`na.omit`

Examples

```
t.d <- data.frame(V1=c(1,2,NA,4), V2=c(11,12,13,Inf))
nainf.exclude(t.d)
```

`notice` *Generate a Notice*

Description

Generate a notice to be sent to output

Usage

```
notice(..., printnotices = NULL)
```

Arguments

`...`	contents of the notice, will be pasted together
`printnotices`	logical: Should the notice be printed? Default is the respective pl option.

Details

This function is very similar to 'message'

Value

None.

Author(s)

Werner A. Stahel

See Also

message

Examples

```
ff <- function(x) {
  if (length(x)==0)  {
    notice("ff: argument 'x' is NULL. I return 0")
    return(0)
  }
  1/x
}
ff(3)
ff(NULL)
oo <- ploptions(printnotices=FALSE)
ff(NULL)
```

---

pl.control *Arguments for plotting functions*

Description

Arguments that can be specified calling plyx and other 'pl' functions are checked and data is prepared for plotting.

Usage

```
pl.control(x=NULL, y=NULL, condvar = NULL, data = NULL, subset = NULL,
  transformed = TRUE, distinguishy = TRUE, gensequence = NULL,
  csize = NULL, csize.pch = NULL,
  psize = NULL, plab = FALSE, pch = NULL, pcol = NULL,
  smooth.weights = NULL, smooth.weight = NULL,
  markextremes = NULL, smooth = NULL,
  xlab = NULL, ylab = NULL, varlabel = NULL,
  vcol = NULL, vlty = NULL, vpch = NULL, plscale = NULL, log = NULL,
  main = NULL, sub = NULL, .subdefault = NULL, mar = NULL,
  gencoord = TRUE,
  plargs = pl.envir, ploptions = NULL, .environment. = parent.frame(),
  assign = TRUE, ... )
```

Arguments

x, y, data	as in plyx
condvar	conditioning variables for plcond
subset	subset of data.frame 'data' to be used for plotting. See details.
transformed	logical: should transformed variables be used?
distinguishy	logical: should multiple y's be distinguished? This is TRUE if pl.control is called from plyx.
gensequence	logical: if only x or only y is set, should the other of these be specified as the sequence 1:nobs (where nobs is the number of observations)?
csize	character expansion, applied to both labels and plotting characters.
csize.pch	expansion of plotting symbol relative to par("pch"). By default, it adjusts to the number of observations.
psize, plab, pch, pcol	Plotting characteristics of points, specified as a (unquoted) variable name found in data or as a vector. They set the size of the plotting symbols, labels (character strings), plotting character, and color, respectively. plabs = TRUE asks for using the row names of data.
smooth.weights, smooth.weight	weights to be used in calculating smooth lines. Both are equivalent.
markextremes	scalar: proportion of extreme points to be labelled
smooth	logical: should a smooth line be added?
xlab, ylab	axis labels
varlabel	labels for variables replacing their names in the x and y arguments, either a simple vector of strings with an element for each variable, or a named vector, where names correspond to such variables.
vcol, vlty, vpch	color, line type and plotting character to be used when multiple y-s are plotted (in the sense of matplot)
plscale	plot scale: name of the function to be used for generating a plotting scale, like "log". A named character vector can be given, where the names correspond to variable names in data.
log	requires log scale as in R's basic plot function, e.g., equals either "x", "y" or "xy"
main, sub	string. Main title of the plot(s). If sub starts by ":" (the default), pl.control tries to generate an informative subtitle, determined by the data or a model formula.
.subdefault	for internal use: default of subtitle
mar	plot margins
gencoord	logical: should plotting coordinates be generated? This is avoided for low level pl graphics.
plargs	pl arguments, a list with components ploptions, see the following argument; pldata, the data used for plotting; pmarpar, graphical parameters defining margins.

ploptions	Plotting attributes, e.g., plotting character, line types, colors and the like, for different aspects of plots. Result of ploptions. Defaults to pl.envir$ploptions.
.environment.	used by the calling function to provide the environment for evaluating x and y
assign	logical: should the result of pl.control be assigned to the pl.envir environment? This will be done for high level pl functions, but avoided for low level ones. It allows for reusing the settings and helps debug unexpected behavior.
...	further arguments. These may include: psize, plab, pch, pcol, group, smooth.group, smooth.weights: these specify graphical elements for each observation (row of data). the respective columns are added to the pldata data.frame. ...: further ... arguments will be passed on to ploptions. The respective settings will be used in the calling pl function, but not permanently stored in ploptions in the pl.envir environment.

Details

The function selects the data according to the arguments x, y, data and subset (the latter by calling plsubset). The argument subset should be used instead of data[subset,] if the dataset data contains variable attributes like varlabel, ticksat, ... The argument is evaluated in the dataset defined by data, i.e., variable names may be used to define the subset.

Value

A list containing all the arguments, possibly in modified form. Specifically, the evaluations of the variables contained in x and y along with psize, plab, pch, pcol, smoothGroup, smoothWeights are collected in the component pldata. The component, ploptions, collects the ploptions, and plfeatures contains a list of additional features, both to be used in the calling high level pl function

Author(s)

Werner A. Stahel

See Also

plyx, plmatrix, ploptions

Examples

```
plyx(Sepal.Width~Sepal.Length, data=iris, axp=7, plab=TRUE, csize.plab=0.6)
## same as
plargs <- pl.control(Sepal.Width~Sepal.Length, data=iris)
plargs$pdata$plab <- row.names(iris)
plargs$csize.lab <- 0.6
plargs$axp <- 7
plyx(Sepal.Width~Sepal.Length, plargs=plargs)
```

plbars *Add bars to a pl plot*

Description

Adds horizontal or vertical bars to a plot

Usage

```
plbars(x = NULL, y = NULL, midpointwidth = NULL,
  plargs = NULL, ploptions = NULL, marpar = NULL, ...)
```

Arguments

x, y	coordinates for the horizontal and veritical axis, respectively. Either of them must have 3 columns. If y has 3 columns, x must have one only or be a vector. Then y[,1] contains the midpoints, and the other two columns determine the endpoints of the bars, which will be vertical. Analogously if x has 3 columns.
midpointwidth	for plbars: determines the length of the segments that mark the midpoints. See Details.
plargs, ploptions	result of pl.control, see Details
marpar	margin parameters, if already available. By default, they will be retieved from ploptions.
...	absorbs extra arguments

Details

For plbars, the argument midpointwidth determines the length of the segments that mark the midpoint relative to the default, which is proportional to the range of the plotting area and inversely proportional to the number of (finite) observations.

plargs and ploptions may be specified explicitly. Otherwise, they are taken from pl.envir.

Value

None.

Author(s)

Werner A. Stahel

Examples

```
data(d.river)
dd <- plsubset(d.river, 1:2000)
da <- aggregate(dd[,3:7], dd[,"date",drop=FALSE], mean, na.rm=TRUE)
ds <- aggregate(dd[,3:7], dd[,"date",drop=FALSE], sd, na.rm=TRUE)
plyx(O2~date, data=da, type="n")
td <- da$O2 + outer(ds$O2, c(0,-1,1))
plbars(y = td, midpointwidth=0.1, bar.lwd=2)
```

---

plcond *Plot Two Variables Conditional on Two Others*

Description

A scatterplot matrix is generated that shows, in each panel, the relationship between two primary variables, with the dataset restricted by appropriate subranges of two 'conditioning' variables. This corresponds to `link{coplot}`. The points that are near to the the 'window' defining the panel's restriction are also shown, in a distinct style.

Usage

```
plcond(x, y = NULL, condvar = NULL, data = NULL,
  panel = NULL, nrow = NULL, ncol = NULL,
  xaxmar = NULL, yaxmar = NULL, xlab = NULL, ylab = NULL,
  oma = NULL, plargs = NULL, ploptions = NULL, assign = TRUE, ...)
```

Arguments

`x, y`	the two variables used to generate each panel. They may be specified as vectors, as column names of `data` or by formulas as in `plyx`.
`condvar`	two (or one) variables that define the restrictions of the data for the different panels. A numerical variable is cut into intervals, see Details. A factor defines the 'ranges' as its levels. For each combination of intervals or levels of the two variables, a panel is generated.
`data`	data.frame in which the variables are found if needed
`panel`	function that generates each panel. If set by the user, it must accept the arguments `x, y, ckeyx, ckeyy, pcol, pale, cex, smooth, smooth.minobs, ploptions`. The default is `ploptions("plcond.panel")`, which in turn is initiated as the function `plpanelCond`.
`nrow, ncol`	number of maximum rows and columns on a page
`xaxmar, yaxmar`	margin in which the axis (tick marks and corresponding labels) should be shown: either 1 or 3 for `xaxmar` and 2 or 4 for `yaxmar`.
`xlab, ylab`	labels of the variables `x` and `y`
`oma`	width of outer margins, see `par`. Note that a minimum of 2.1 is generally needed for showing tick and axis labels.

plargs	result of calling pl.control. If NULL, pl.control will be called to generate it. If not null, arguments given in ... will be ignored.
ploptions	list of pl options.
assign	logical: Should the plargs be stored in pl.envir?
...	further arguments passed to the panel function and possibly further to functions called by the panel function.

Details

A numerical conditioning variable (condvar) will be split by default into classes by splitting its robust range (robrange) into ploptions("plcond.nintervals") equally long intervals. Alternatively, the variable may contain an attribute cutpoints which then defines the intervals.

For numerical conditioning variables, each panel also shows neighboring points with a different color and diminished size. The size of the neighborhood is defined by the proportion of extension ploptions("plcond.ext"). The point size of the respective 'exterior' points is given by ploptions("plcond.cex") The color are given by the 4 elements of ploptions("plcond.col"): The first element is used to paint the neighboring points to the left of the current range of the conditioning x variable, the second element paints those to the right, and the third and fourth are used in the same way for the conditioning y variable. The neighboring points that are outside both ranges get a color mixing the two applicable colors according to this rule. Finally, paling is applied to these colors with a degree that is linear in the distance from the interval, determined by the range given by ploptions("plcond.pale").

Value

None.

Author(s)

Werner A. Stahel

See Also

coplot

Examples

```
plcond(Sepal.Width~Sepal.Length, data=iris, condvar=~Species+Petal.Length)
```

---

plcoord *Determines Values for Plotting with Limited "Inner" Plot Range*

Description

For plots with an "inner plot range" (see Details) this function converts the data values to the coordinates in the plot

Usage

```
plcoord(x, range = NULL, innerrange.factor = NULL,
  innerrange.ext = NULL, plext = NULL, ploptions = NULL)
```

Arguments

x	data to be represented
range	vector of 2 elements giving the inner plot range. Data beyond the given interval will be non-linearly transformed to fit within the (outer) plot margins. Defaults to robrange(x, fac=fac).
innerrange.factor	factor used to determine the default of range
innerrange.ext	factor for extending the range to determine the outer plot range
plext	vector of 1 or 2 elements setting the extension factor for the plotting range
ploptions	plotting options

Details

When plotting data that contain outliers, the non-outlying data is represented poorly. Rather than simply clipping outliers, one can split the plotting area into an inner region, where the (non-outlying) data is plotted as usual, and a plot area margin, in which outliers are represented on a highly non-linear scale that allows to display them all.

This function converts the data to the coordinates used in the graphical display, and also returns the inner and outer ranges for plotting.

Value

vector of coordinates used for plotting, that is, unchanged x values for those within the range and transformed values for those outside.

Attributes:

attr(, "plrange")	the range to be used when plotting
attr(, "range")	the "inner" plot range, either the argument range or the values determined by default.
attr(, "nouter")	the number of modified observations

Author(s)

Werner A. Stahel

See Also

robrange

Examples

```
  set.seed(0)
  x <- c(rnorm(20),rnorm(3,5,10))
  ( xmod <- plcoord(x) )

  plot(x,xmod)
## This shows what high level pl functions do by default
  plot(xmod)
  abline(h=attr(xmod,"innerrange"),lty=3, lwd=2)
## plgraphics
  plyx(x)
```

---

plframe *Low level plotting functions for the 'pl' system*

Description

These functions set up the frame of a plot based on the 'pl' paradigm

Usage

```
plframe(x = NULL, y = NULL, xlab = NULL, ylab = NULL,
  xlim = NULL, ylim = NULL, mar = NULL, showlabels = TRUE,
  plext = NULL, axcol = rep(1, 4),
  plargs = NULL, ploptions = NULL, marpar = NULL, xy = NULL, ...)

pltitle(main=NULL, sub=NULL, csize=NULL, csizemin=NULL,
  side=3, line=NULL, adj=NULL, outer.margin=NULL, col="black",
  doc=NULL, show=NA, plargs=NULL, ploptions = NULL, marpar = NULL, ...)

plaxis(side, x=NULL, showlabels=TRUE, range=NULL, varlabel=NULL, col=1,
  tickintervals=NULL,
  plargs = NULL, ploptions = NULL, marpar = NULL, ...)
```

Arguments

x	coordinates for the horizontal axis
y	coordinates for the vertical axis
xlab, ylab	axis labels
xlim, ylim	plot ranges
mar	plot margins
showlabels	logical: should labels for tickmarks and the variable label be displayed? If ==1, they are shown if there is enough space in the margin (including outer margin), if ==2, it is shown anyway (by setting xpd=TRUE).
plext	extension of the plotting area beyond the range of the data.

axcol	colors for drawing axes scales
main, sub	main title and subtitle
varlabel	variable name
side	For pltitle: in which margin should the text be shown? For plaxis: integer indicating which axis is to be drawn
csize	character size. May be vector of length 3, giving size for main title, subtitle, and tit attribute of title, respectively. The default is given by ploptions("title.cex").
csizemin	minimal character size, to be used to adjust the character size to the length of the text (if cex is NULL)
line	line in margin on which the main title is placed – or the subtitle if main is NULL
adj	text adjustment, scalar between 0 and 1
outer.margin	logical: should title text be placed in outer margin?
col	color for the title text or axis line and tickmarks
range	range in which tickmarks are set
doc	logical: should the tit attribute of main be displayed if available?
show	logical: if FALSE, nothing will be done if there are multiple frames and the current one is not the first. If it is negative, no title will be shown, but the value will be returned.
tickintervals	number of intervals used by pretty to determine the axis ticks.
plargs, ploptions	result of pl.control, see Details
marpar	margin parameters, if already available. By default, they will be retieved from ploptions.
xy	logical: should the coordinates be obtained as in high level graphics? This is set to FALSE to save time and avoid complications, in case the user is sure that x and y are vectors rather than formulas or variable names.
...	absorbs extra arguments

Details

If the arguments x and y are not given, they are obtained from pl.envir$pldata.

plframe draws axes according to argument axes, by calling plaxis. It looks for attributes of x and y, such as innerrange and ticksat. Tick labels are shown at the values of the ticklabelsat attribute if available, otherwise at the values of ticksat. The labels can be given by the attribute ticklabels. This facilitates setting more tick marks than labels, see the example.
It also draws a grid. The positions of gridlines at ticksat by default.
Finally, it draws "zero" lines as determined by the pl option zeroline. The latter can be a numeric vector giving the positions of such threshold lines, or a list of two such vectors, the first for horizontal axis, the second for the vertical axis.

plaxis only shows the variable label, tick labels and tickmarks if there is enough space or showlabels > 1. If it is called when there are multiple panels, this is decided according to the actual mar setting if it is an inner panel; if it is a panel adjacent to an outer margin, then the oma setting is also used.

plargs and ploptions may be specified explicitly, but they are usually generated by calling pl.control.

Value

plframe and plaxis invisibly return the former par(c("cex", "mar", "mgp")) if setpar is TRUE, otherwise NULL.

pltitle invisibly return a list consisting of the main and sub title.

Author(s)

Werner A. Stahel

See Also

gendateaxis; pl.control

Examples

```
plyx(Sepal.Width ~ Sepal.Length, data=iris)

## again, each step separately
t.dt <- pl.envir$pldata
oldpar <- plframe() ## or plframe(t.dt$Sepal.Length, t.dt$Sepal.Width, plargs=pl.envir)
plsmooth() ## or plsmooth(t.dt$Sepal.Length, t.dt$Sepal.Width, plargs=pl.envir)
t.plab <- plmark(markextremes=0.03)
  ## or plmark(t.dt$Sepal.Length, t.dt$Sepal.Width, markextremes=0.03, plargs=pl.envir)
plpoints(plab=t.plab) ## or plpoints(t.dt$Sepal.Length, t.dt$Sepal.Width,
                      ##      plargs=pl.envir, plab=t.plab)
plaxis(4)

par(oldpar)   ## reset the changed graphical parameters
```

---

plinnerrange *Inner Plotting Limits*

Description

Calculates inner limits for plotting, based on a robust estimate of the range.

Usage

```
plinnerrange(innerrange, data, factor = 4, FUNC = robrange)
```

Arguments

innerrange	logical: Should range be calculated? If FALSE, the result will contain only the values FALSE. If it is a list or matrix of the approriate size, it will be returned as is.
data	vector or data.frame for which the range(s) will be calculated
factor	expansion of the calculated robust range to yield the plotting range
FUNC	function used to calculate the robust range. The factor will be handed over to FNC as the argument fac.

Value

Matrix of 2 rows giving the ranges to be used as inner plotting ranges for the variables. If `innerrange` is such a matrix or data.frame, it will be returned as is.

Author(s)

Werner A. Stahel

See Also

`robrange`, `plcoord`

Examples

```
data(d.blast)
dd <- d.blast[,c("charge","distance","tremor")]
( t.ipl <- plinnerrange(TRUE, dd) )
plot(dd[,"tremor"], plcoord(dd[,"tremor"], t.ipl[,"tremor"]))
abline(h=t.ipl[,"tremor"])
```

---

pllimits *Determine Inner Plot Range*

Description

The inner plotting range is the range in which plotting functions of the regr0 package show unmodified coordinates. This function determines the range for one or more variables.

Usage

```
pllimits(pllim, data, limfac = NULL, FUNC=NULL)
```

Arguments

`pllim`	either a logical: shall an inner plotting range be determined? – or a matrix with 2 rows and `NCOL(data)` rows, in which case the suitability will be checked.
`data`	vector or matrix or data.frame of data for which the inner plotting range is to be determined
`limfac`	scalar factor by which the range determined by `FUNC` is expanded
`FUNC`	function that determines the range of the data

Value

A matrix with 2 rows containing the minimum and the maximum of the inner plotting range. The columns correspond to those in `data`.

Author(s)

Werner A. Stahel

See Also

plcoord

Examples

```
set.seed(0)
xx <- rt(50, df=3)
( pll <- pllimits(TRUE, xx) )
sum(xx<pll[1,] | xx>pll[2,])  ## 3
```

---

plmarginpar *Set Graphical Parameters According to Those used in the Pl Function Called Last*

Description

plmarginpar calls par to set the margin widths mar and mgp equal to those used in the last call of a high level pl function

Usage

```
plmarginpar(plargs = pl.envir, csize = NULL)
```

Arguments

plargs	list from which the margin parameters are obtained. If NULL, the default, pl.envir is used.
csize	size of plot symbols and text, changes par("cex") to csize*par("cex")

Value

The old settings of par(c("mar","mgp")) are returned invisibly.

Note

plmarginpar is used to complement a plot with low level ordinary R functions like mtext or segments, see Example.

The same effect can be achieved by setting the pl option keeppar to TRUE, either by calling ploptions or by setting keeppar=TRUE in the call to the high level pl function.

Author(s)

Werner A. Stahel

Examples

```
par(mar=c(2,2,5,2))
plyx(Sepal.Width~Sepal.Length, data=iris) ## margins according to ploptions
par("mar") ## paramteres have been recovered
mtext("wrong place for text",3,1, col="red")  ## margins not appropriate for active plot
plmarginpar()
par("mar") ## margins used inside the call to  plyx . These are now active
mtext("here is the right place",3,1, col="blue")
```

---

plmark *Labels for Extreme Points*

Description

Determine extreme points and get labels for them.

Usage

```
plmark(x, y = NULL, markextremes = NULL, plabel = NULL, plargs = NULL, ploptions = NULL)
```

Arguments

x, y	coordinates of points. If x is of length 0, it is retrieved from `plargs$pldata[,1]`.
markextremes	proportion of extreme points to be 'marked'. This may be a list of proportions with names indicating the variables for which the proportion is to be applied. If a vector (of length 2), the elements define the proportions for the lower and upper end, respectively. In the default case (NULL), the proportion is obtained from `ploptions`, which in turn leads to calling the function `markextremes` with the argument equal to the number of (finite) observations.
plabel	character vector of labels to be used for extreme points. If NULL, they are obtained from `plargs$plabel`.
plargs, ploptions	result of `pl.control`, cf `plpoints`

Value

A character vector in which the 'marked' observations contain the respective label and the others equal `""`.

Author(s)

Werner A. Stahel

See Also

`plyx`

Examples

```
plyx(Sepal.Width ~ Sepal.Length, data=iris)
( t.plab <-
  plmark(iris$Sepal.Length, iris$Sepal.Width, markextremes=0.03) )
```

---

plmatrix *Scatterplot Matrix*

Description

Plots a scatterplot matrix, for which the variables shown horizontally do not necessarily coincide with those shown vertically. If desired, the matrix is divided into several blocks such that it fills more than 1 plot page.

Usage

```
plmatrix(x, y = NULL, data = NULL, panel = NULL,
  nrow = NULL, ncol = nrow, reduce = TRUE,
  xaxmar=NULL, yaxmar=NULL, xlabmar=NULL, ylabmar=NULL,
  xlab=NULL, ylab=NULL, mar=NULL, oma=NULL, diaglabel.csize = NULL,
  plargs = NULL, ploptions = NULL, assign = TRUE, ...)
```

Arguments

x	data for columns (x axis), or formula defining column variables. If it is a formula containing a left hand side, the left side variables will be used last.
y	data or formula for rows (y axis). Defaults to x
data	data.frame containing the variables in case x or y is a formula
panel	a function that generates the marks of the individual panels, see Details.
nrow, ncol	maximum number of rows and columns of panels on a page
reduce	if y is not provided and reduce==TRUE, the first row and the last column are suppressed.
xaxmar, yaxmar	margin in which the axis (tick marks and corresponding labels) should be shown: either 1 or 3 for xaxmar and 2 or 4 for yaxmar.
xlabmar, ylabmar	in which margin should the x- [y-] axis be labelled?
xlab, ylab	not used (introduced to avoid confusion with xlabmar, ylabmar)
mar, oma	width of margins, see par
diaglabel.csize	Character expansion for labels appearing in the "diagonal" of the scatterplot matrix (if present)
plargs	result of calling pl.control. If NULL, pl.control will be called to generate it. If not null, arguments given in ... will be ignored.

ploptions	list of pl options.
assign	logical: Should the plargs be stored in the pl.envir environment?
...	further arguments passed to the panel function and possibly further to functions called by the panel function

Details

The panel function can be user written. It needs $>= 5$ arguments which must correspond to the arguments of plpanel: x, y, indx, indy, plargs. If some arguments are not used, just introduce them as arguments to the function anyway in order to avoid (unnecessary) error messages and stops. Since large scatterplot matrices lead to tiny panels, plmatrix splits the matrix into blocks of at most nrow rows and ncol columns. If these numbers are missing, they default to nrow=5 and ncol=6 for landscape pages, and to nrow=8 and ncol=5 for portrait pages.

The panel argument defaults to plpanel, which results essentially in points or text depending on the argument pch, including a smooth line, to plmboxes if 'x' is a factor and 'y' is not or vice versa, or to a modification of sunflowers if both are factors.
The function must have the arguments x and y to take the coordinates of the points and may have the arguments indx and indy to transfer the variables\' index. If there is an argument plargs, the current value of plargs will be passed on. It is a list and can be extended to pass any additional items to the function.

Value

none

Note

There are many more arguments, obtained from pl.control, see ?pl.control. These can be passed to plmatrix by an argument plargs that is hidden in the ... argument list.

Author(s)

Werner A. Stahel, ETH Zurich

See Also

pairs, plyx

Examples

```
plmatrix(iris, pch=as.numeric(Species))
plmatrix(~Sepal.Length+Sepal.Width, ~Petal.Length+Petal.Width,
    data=iris, smooth=TRUE, plab=substr(Species,1,2))
```

plmboxes *Multibox plots*

Description

Draw multibox plot(s) for given (grouped) values, possibly asymmetric. 'plbox' draws a single multibox plot (low level graphical function). 'plboxes' is a high level graphics function that draws multiboxes for grouped data. A secondary, binary grouping factor can be given to produce asymmetric multiboxes.

Usage

```
plmboxes(x, ...)
## S3 method for class 'formula'
plmboxes(x, y=NULL, data, ...)

## Default S3 method:
plmboxes(x=NULL, y=NULL, data=NULL, width=1, at=NULL,
    horizontal=FALSE,
    probs=NULL, outliers=TRUE, na=FALSE, backback=NULL, refline=NULL,
    add=FALSE, xlim=NULL, ylim=NULL, axes=TRUE, xlab=NULL, ylab=NULL,
    labelsperp=FALSE, xmar=NULL, mar=NULL,
    widthfac=NULL, minheight=NULL, colors=NULL, lwd=NULL,
    .subdefault=NULL, plargs = NULL, ploptions = NULL, marpar = NULL, ...)

plmbox(x, at=0, probs=NULL, outliers=TRUE, na.pos=NULL, horizontal=FALSE,
    width=1, wfac=NULL, minheight=NULL, adj=0.5, extquant=FALSE,
    widthfac=c(max=2, med=1.3, medmin=0.3, outl=NA),
    colors=c(box="lightblue2",med="blue",na="gray90"),
    lwd=c(med=3, range=2), warn=options("warn"))
```

Arguments

x	For 'plmboxes.formula': a formula, such as 'y ~ grp' or 'y~grp+grp2', where 'y' is a numeric vector of data values to be split into groups according to the grouping variable 'grp' (usually a factor) and, if given, according to the binary variable 'grp2'. 'y~1+grp2' produces a single asymmetric mbox. For 'plmboxes.default': factor to be used as the grouping variable or matrix or data.frame with 2 columns (for asymmetric mbox plot), where the second column is binary.
y	a numeric vector of data values
data	a data.frame from which the variables in 'formula' should be taken.
width	a vector giving the widths of the multibox plot for each group 'grp1'.
at	horizontal position of the multiboxes. Must have length equal to the number of (present) levels of the factor 'grp'. if an element of 'at' is 'NA', the group will be skipped. Defaults to 1, 2, ...

horizontal	logical. If TRUE, boxes will be drawn horizontally. Note that 'y' is then the horizontal coordinate, i.e., still the quantitative variable defining the boxes, and 'x' is still the grouping.
probs	probability values for selecting the quantiles. If all 'probs' are <=0.5, they will be mirrored at 0.5 for 'plmboxes'. The default is c(0.05,0.25,0.5) if the average number of data per group (for 'plmboxes', or the number of data, for 'plmbox') is less than 20, c(0.025,0.05,0.125,0.25,0.375,0.5), otherwise.
outliers	logical: should outliers be marked?
na, na.pos	if 'na' is not NULL, NA values will be represented by a box. If 'na' is TRUE, the position of the box will be generated to be below the minimum of the data. If 'na' (for 'plmboxes') or 'na.pos' (for 'plmbox') is a scalar or a vector of length 2, the position of the box is at that value (with a generated width) or between the 2 values, respectively.
backback	logical: Should two back-to-back multiboxes be displayed if the (single) x factor is binary?
refline	vertical positions of any horizontal reference lines
add	logical. If TRUE, the mboxes will be added to an existing plot without calling 'plot'.
xlim	plotting limits for the horizontal axis.
ylim	plotting limits for the vertical axis.
axes	logical. If FALSE, no axes are drawn.
xlab	label for the x axis. Defaults to the "x factor" – the first name on the right hand side of 'formula'
ylab	label for the x axis. Defaults to the left hand side of 'formula'.
labelsperp	logical: Should the labels for the levels of the "x factor" be shown in perpendicular to the axis? If it is numeric, it determines the maximum label length, with a maximum of 20.
xmar	plot margin for the "x factor" axis. Default tries to be suitable, i.e. expand the margin if labelsperp is TRUE according to the length of the levels' labels. If xmar has two more elements, they determine the margin lines where the variable label and the levels' labels are shown.
mar	margin widths
widthfac	named vector used to modify the following settings: max=2: determes the maximal width of the boxes. Boxes that should be wider are censored and marked as such. med=1.3, medmin=0.3: determine the width of the mark for the data median. The width is 'med' times the maximal width of the boxes, but at least 'medmin'. outl=NA: length of the marks for outliers. sep=0.003: width of the gap between the "half" mbox plots in case of asymmetric mboxes (only needed for 'plmboxes'). For 'plmboxes', the argument needs to contain only the elements that should be different from the default values.

colors	named vector or list selecting the colors to be used, with named elements: box="lightblue2": color with which the central box(es) (those corresponding to probabilieties between 0.25 and 0.75) will be filled. med="blue": color of the mark for the median. na="grey90": color with which the box for NA values will be filled. For 'plmboxes', the argument needs to contain only the elements that should be different from the default values.
lwd	named vector or list selecting the line width to be used, with named elements: med=3: line width for the mark showing the median range=2: line width for the line along the range of the data For 'plmboxes', the argument needs to contain only the elements that should be different from the default values.
plargs	result of calling pl.control. If NULL, pl.control will be called to generate it. If not null, arguments given in ... will be ignored.
ploptions	list of pl options.
marpar	margin parameters, if already available. By default, they will be retieved from ploptions.
...	additional arguments passed to 'plot'
.subdefault	text for the subtitle in case that it is not specified

Specific arguments for 'plmbox':

wfac	factor by which the widths of the boxes must be multiplied. If given, it overrides 'width'
minheight	minimal class width ("height") for the boxes (in case two quantiled are [almost] identical). The default is 0.02 times the (median of the within group) IQR.
adj	adjustment of the boxes. 'adj=0' leads to boxes aligned on the left, 'adj=1', on the right, 'adj=0.5', centered. Other values of 'adj' make little sense.
extquant	logical, passed to quinterpol: Should the quantiles be extrapolated beyond the range of the data? This may make sense if the sample is small or the data is rounded or grouped or a score.
warn	level of warning for the case when there is no non-missing data

Details

A multibox plot is a generalization (and modification) of the ordinary box plot that draws more details of the distribution in the form of a histogram with variable class widths. The classes are selected such that preselected quantiles form the class breaks. By default, these quantiled include the median and the quartiles, thereby recovering the box of the traditional box plot.

Value

plmboxes invisibly returns the 'at' values that are finally used.\ plmbox returns a scalar by which the width of the boxes are multiplied for plotting, and, as attributes, the quantiles and widths used to draw the boxes

Author(s)

Werner A. Stahel

See Also

boxplot

Examples

```
plmboxes(Sepal.Length~Species, data=iris)

plmboxes(Sepal.Length~Species, data=iris,
  widthfac=c(med=2), colors=c(med="red"), horizontal=TRUE)

plmboxes(Sepal.Length~factor(Species)+I(Sepal.Width<=3), data=iris[1:100,],
         labelsperp=TRUE, horizontal=TRUE)
```

---

plmframes *Multiple Frames for Plotting*

Description

This is a short-cut to set some graphical parameters

Usage

```
plmframes(mfrow = NULL, mfcol = NULL, mft = NULL, byrow = TRUE, reduce = FALSE,
  oma = NULL, mar = NULL, mgp = NULL, plargs = NULL, ploptions = NULL, ...)
```

Arguments

mfrow, mfcol	number of rows and columns of panels. The default is 1 for both, which will reset the subdivision of the plotting page.
mft	total number of panels, to be split into mfrow and mfcol by the function. The result depends on the current aspect ratio (ratio of height to width) of the plotting area.
byrow	if TRUE, the panels will be generated by rows, otherwise, by columns
reduce	logical: If the number of rows or columns asked for by mfrow or mfcol exceeds the maximum numbers determined from ploptions("mframesmax"), suitable numbers for multiple pages are calculated. If reduce is TRUE, these suggested numbers are applied.
mar	plot margins. Any NAs in mar will be replaced by appropriate values.
oma	outer plot margins. Any NAs will be replaced by appropriate values.
mgp	margin-pars passed to par(...). If NULL, it will be generated.
plargs, ploptions	result of calling pl.control, used for generating appropriate values for the margin parameters. If NULL, pl.envir will be used.
...	further graphical parameters passed to par(...).

Details

The function calls par. Its purpose is to simplify a call like par(mfrow=c(3,4)) to plmframes(3,4) and to set some defaults differently from par.

Value

A named list containing the old values of the parameters, as for par.

Author(s)

Werner A. Stahel, ETH Zurich

See Also

par

Examples

```
plmframes(2,3)
plmframes(mft=15)  ## will split the plotting area into >= 15 panels,
plmframes()  ## reset to 1 panel

t.plo <- ploptions(mframesmax=9, assign=FALSE)
t.mf <- plmframes(4,4, reduce=TRUE, ploptions=t.plo)
par("mfg")
t.mf[c("mfigsug","npages")]
##  $mfigsug
##  [1] 2 4
##  $npages
##  [1] 2 1
##  if the device area was higher than wide,
##  the result is the other way 'round
t.mft <- plmframes(mft=12, reduce=TRUE, ploptions=t.plo)
```

---

ploptions *Set and Get User "Session" Options that Influence "plgraphics"s Behavior*

Description

The user can set (and get) 'pl' options – mostly graphical "parameters" – which influence the behavior **plgraphics** functions.

Usage

```
ploptions(x = NULL, ploptions = NULL, list = NULL, default = NULL,
          assign = TRUE, ...)

default.ploptions
```

Arguments

x	character (vector) of name(s) of ploptions to query. If x is set, all further arguments will be ignored.
ploptions	the list of options that should be inspected or modified. Defaults to usr.ploptions from the pl.envir environment. ploptions>1 is equivalent to ploptions=pl.envir$ploptions, the last (or current) list used by a high level pl function.
list	a named list of options to be set, see Details
default	character vector of option names. These ploptions will be set according to default.ploptions. default="all" or =TRUE will reset all options. If default is set, all further arguments will be ignored.
assign	logical: should the list be assigned to pl.envir$usr.ploptions? It is then permanant until changed again by calling ploptions again or the session is closed. If >1, the resulting options are stored as ploptions in pl.envir, which is changed by the high level pl functions.
...	any ploptions can be defined or modified, using name = value, as in options of basic R.

Details

If the argument list is set, it must be a named list, and each component, with name name and value value is used as described for arguments in ..., see above (in addition to such arguments).

There is an object ploptions in the pl.envir environment, which contains the ploptions that have been used (usually after modification) by the high level pl function last called. This list is used by subsequent calls of lower level pl functions. Advanced uses may want to modify this list by assigning to pl.envir$ploptions$pch, for example.

Here is an incomplete list of the components of default.ploptions, describing the suitable alternative values to be set by calling ploptions. For the full set, see ?ploptions.list.

keeppar: logical. If TRUE, the graphical parameter settings "mar", "oma", "cex", "mgp", and "mfg" will be maintained when leaving high level pl functions, otherwise, the old values will be restored (default).

colors: The palette to be used by pl functions

csize: General character size, relative to par("cex")

pale: default argument for colorpale

tickintervals: vector of length 2. The first element is the desired number of tick intervals for axes, to be used as argument n in pretty. The second determines how many tick labels are shown in the same way, and should therefore be smaller than (or equal to) the first.

pch: plotting symbols or characters

csize.pch: size of plotting symbols, relative to default. This may be a function with an argument that will be the number of observations at the time it is used.

csize.plab: size of point labels, relative to csize.pch

psize.max: maximum value of size of plotting symbols

lty, lwd: line type(s) and width(s)

col, pcol, lcol: colors to be used generally and specifically for points (symbols or text) and lines, respectively, given as index of `ploptions("colors")`. This are often (and by default) vectors to be used for showing groups. The first element is usually black.

colors: the palette to be used

censored.pch, censored.size, censored.pale: ...

gridlines: can be
– a logical indicating if gridlines should be drawn. If `TRUE`, gridlines will be drawn at the values given in `attr(.,"ticksat")`; – a vector of values at which the gridlines should appear;
– a list of length 2 of such values;
– a named list. If a name equals the attribute `varname` of either the x or y variable, the respective component will be used.

`smooth.lty, smooth.col`**:** line type and color. Note that if there is a `smooth.group` factor, `group.lty` and `group.col` are used.

`smooth.lwd`**:** line width. If of length 2 (or more), the second element is the factor by which the line width is reduced for simulated smooths (that is, for the second to the last column of `smoothline$y`). It defaults to 0.7.

smooth.xtrim: proportion of fitted values to be trimmed off on both sides when drawing a smooth line, either a number or a function that takes the number of points as its argument. The default is the simple function `2^log10(n)/n`. The smoothing function may produce an attribute `xtrim` that is used as an additional factor to `smooth.xtrim`. This is applied, e.g., to suppress trimming if a straight line is fitted instead of a smooth by requiring `smoothLm` as the smoothing function.

smooth.minobs: minimal number of observations needed for calculating a smooth.

smooth.band: Indicator (logical) determining whether "low" and "high" smooth lines should be drawn. See above for their definition.

condquant...: Conditional quantiles for censored residuals.

- **condquant:** logical: should bars be drawn for censored residuals? If `FALSE`, censored observations will be set to the median of the conditional distribution and shown by a different plotting character, see argument `censored` of `ploptions`. If `NULL`, the standard plotting character will be used.
- **condquant.probrange:** range for probabilities. If the probability corresponding to the censored part of the distribution is outside the range, bars will not be drawn.
- **condquant.pale:** factor by which the `pcol` color will be paled to show the points (`condquant.pale[1]`) and the bars (`...[2]`).

plcond...: features of `plcond`.

- **plcond/panel:** panel function to be used
- **plcond.nintervals:** number of intervals into which numerical variables will be cut
- **plcond.extend:** proportion of neighboring intervals for which points are shown. 0 means no overlap.
- **plcond.col:** 4 colors to be used to mark the points of the neighboing intervals: The first and second ones color the points lower or higher than the interval of the horizontal conditioning variable, and the other two regulating the same features for the vertical variable. The points which are outside the intervals of both conditioning variables will get a mixed color.

plond.pale: minimum and maximum paling, to be applied for distance 0 and maximal distance from the interval.

plcond.cex: symbol size, relative to cex, used to show the points outside the interval

Value

For `ploptions(x)`, where x is the name of a pl option, the current value of the option, or NULL if it is not such a name. If x contains several (valid) names, the respective list.

For `ploptions()`, the list of all plptions sorted by name.

For uses setting one or more options, the important effect is a changed list `usr.ploptions` in the `pl.envir` environment that is used by the package's functions (if `assign` is TRUE). The (invisibly) returned value is the same list, complemented by an attribute `"old"` containing the previous values of those options that have been changed. This list is useful for undoing the changes to restore the previous status.

Author(s)

Werner A. Stahel

See Also

`stamp`; `ploptions.list`; `pl.envir`; R's own predefined `options()`.

Examples

```
## get options
ploptions(c("jitter.factor", "gridlines"))
ploptions("stamp")  ## see example(stamp)
ploptions()  ## all pl options, see '?ploptions.list'

## set options
ploptions(stamp=FALSE, pch=0, col=c.colors[-1], anything="do what you want")
ploptions(c("stamp", "anything"))
ploptions(default=TRUE)  ## reset all pl options, see '?ploptions.list'

## assign to transient options
t.plopt <- ploptions(smooth.col="purple", assign=2)
t.plopt$smooth.col
attr(t.plopt, "old")
ploptions("smooth.col") ## unchanged
ploptions("smooth.col", ploptions=2) ## transient options
pl.envir$ploptions["smooth.col"] ## the same

## switching 'margin parameters' between those used
## outside and inside high level pl functions
par(mar=c(2,2,5,2))
plyx(Sepal.Width~Sepal.Length, data=iris, title="The famous iris data set")
par("mar")
mtext("wrong place for text",3,1, col="red")
t.plo <- plmarginpar()
par("mar")
```

```
mtext("here is the right place",3,1)

par(attr(t.plo, "oldpar"))  ## resets the 'margin parameters'
par("mar")
plyx(Sepal.Width~Sepal.Length, data=iris, keeppar=TRUE)
par("mar")

## manipulating 'pl.envir$ploptions'
plyx(Sepal.Width~Sepal.Length, data=iris)
pl.envir$ploptions$pch
plpoints(7,4, csize=4)
pl.envir$ploptions$pch <- 4
plpoints(7.5,4, csize=4)
```

`ploptions.list` *The List of pl Options*

Description

The user can set (and get) 'pl' options – mostly graphical "parameters" – which influence the behavior of **plgraphics** functions.

Usage

```
## not used, this gives the complete list of 'pl' options
```

Value

keeppar: logical. If TRUE, the graphical parameter settings "mar", "oma", "cex", "mgp", and "mfg" will be maintained when leaving high level pl functions, otherwise, the old values will be restored (default).

colors: The palette to be used by pl functions

pale: default argument for `colorpale`

linewidth: vector of `lwd` values to be used for the different line types (`lty`). The package sets `lwd` to a value `ploptions("linewidth")[lty]*lwd` intending to balance the visual impact of the different line types, e.g., to allow a dotted line to make a similar impression as a solid line.

csize: General character size, relative to `par("cex")`

ticklength: vector of 4 scalars: tickmark length, corresponding to `par("tcl")`. The first 2 elements define the length of the regular tickmarks, the other two, of the "small" tichmarks given by `attr(ticksat, "small")` (`ticksat` is a possible attribute of each variable). There are two elements each in order to define tickmarks that cross the axis.

tickintervals: vector of length 2. The first element is the desired number of tick intervals for axes, to be used as argument `n` in `pretty`. The second determines how many tick labels are shown in the same way, and should therefore be smaller than (or equal to) the first.

pch: plotting symbols or characters

csize.pch: size of plotting symbols, relative to default. This may be a function with an argument that will be the number of observations at the time it is used.

csize.plab: size of point labels, relative to `csize.pch`

psize.max: maximum value of size of plotting symbols

lty, lwd, col, pcol, lcol: line type, line width, color to be used

pcol, lcol: color to be used for plotting symbols and labels, respectively

*** innerrange

innerrange logical: should an innerrange be used in plots if needed?

innerrange.factor factor needed to determined the inner range

innerrange.ext extension of the inner range

innerrange.function function used to calculate the inner range

plext extension of the data range to the plotting range

markextremes proportion of observations to be marked by their labels on the lower and upper extremes

variables.pch, variables.col, variables.lty, variables.lcol: vectors of symbols, color, line type, line color to be used for showing different y variables

censored.pch, censored.size, censored.pale: plotting symbol and size, and pale value to be applied to censored observations. Different symbols are used for distinguishing right and left censoring in vertical and horizontal direction and there combination.

group.pch, group.col, group.lty, group.lcol: vector of symbols and colors used for observations and types and colors used for lines in the different groups

*** title parameters.

title.line line in margin[3] on which the title appears

title.adj adjustment of the title

title.csize character size of the title, relative to `ploptions("csize")*ploptions("margin.csize")[1]`

title.csizemin minimum csize

title.maxchars maximum number of characters in title

sub logical: should subtitle be shown?

xlab, ylab labels of x and y axes

mframesmax maximum number of panels to be shown on one page

panel panel function to be used in high level pl functions

axes: axes to be shown

*** margin parameters.

mar, oma ...

mar.default, oma.default their default values

margin.csize character size for variable labels and tick labels

margin.line lines in margin where variable labels and tick labels are shown

margin.exp expansion of margins beyond needed lines, for inner and outer margins

panelsep space between panels

*** date parameters.

date.origin The year which serves as origin of the internal (julian) date scale

date.format format for showing dates

date.ticks data.frame ruling how many small and large ticks and tick labels will be shown. The first column determines the row that will be used

gridlines: can be
– a logical indicating if gridlines should be drawn. If `TRUE`, gridlines will be drawn at the values given in `attr(.,"ticksat")`; – a vector of values at which the gridlines should appear;
– a list of length 2 of such values;
– a named list. If a name equals the attribute `varname` of either the x or y variable, the respective component will be used.

zeroline: logical: should zero (0) be shown be a special grid line? Can be numerical, then gives coordinates of such lines, generalizing the zero line.

zeroline.lty, zeroline.lwd, zeroline.col line type, width and color of the zero line

refline reference line, any line to be added to the current plot using the following properties. See `plrefline` for possible types of values

refline.lty, refline.lwd, refline.col line type, width and color of the ref line

*** smooth.

smooth logical: should a smoothing line be shown?

smooth.function: function for calculating the smoother

smooth.par, smooth.iter parameters for the function

smooth.minobs: minimal number of observations needed for calculating a smooth.

smooth.band: Indicator (logical) determining whether "low" and "high" smooth lines should be drawn. See above for their definition.

`smooth.lty, smooth.col`**:** line type and color. Note that if there is a `smooth.group` factor, `group.lty` and `group.col` are used.

`smooth.lwd`**:** line width. If of length 2 (or more), the second element is the factor by which the line width is reduced for simulated smooths (that is, for the second to the last column of `smoothline$y`). It defaults to 0.7.

smooth.pale paling factor to be applied for secondary smooth lines

smooth.xtrim: proportion of fitted values to be trimmed off on both sides when drawing a smooth line, either a number or a function that takes the number of points as its argument. The default is the simple function `2^log10(n)/n`. The smoothing function may produce an attribute `xtrim` that is used as an additional factor to `smooth.xtrim`. This is applied, e.g., to suppress trimming if a straight line is fitted instead of a smooth by requiring `smoothLm` as the smoothing function.

bar.midpointwidth width of the line shown at the central point of a bar

bar.lty, bar.lwd, bar.col line type, width (for bar and midpoint line), color of bars

*** factors, multibox plots:

factor.show: how should factors be plotted. Options are `"mbox"`, `"jitter"` or `"asis"`

mbox.minobs minimal number of observations shown as a multibox plot

mbox.minheigth see `?plmboxes`

mbox.colors colors to be used for multibox plots

jitter amount of jitter, or logical: should jittering be applied?

jitter.minobs minimal number of observations to which jittering should be applied

jitter.factor what proportion of the gap between different values will be filled by the jittering?

*** condquant: Conditional quantiles for censored residuals.

condquant: logical: should bars be drawn for censored residuals? If FALSE, censored observations will be set to the median of the conditional distribution and shown by a different plotting character, see argument censored of ploptions. If NULL, the standard plotting character will be used.

condquant.probrange: range for probabilities. If the probability corresponding to the censored part of the distribution is outside the range, bars will not be drawn.

condquant.pale: factor by which the pcol color will be paled to show the points (condquant.pale[1]) and the bars (...[2]).

*** plcond: features of plcond.

plcond/panel: panel function to be used

plcond.nintervals: number of intervals into which numerical variables will be cut

plcond.extend: proportion of neighboring intervals for which points are shown. 0 means no overlap.

plcond.col: 4 colors to be used to mark the points of the neighboing intervals: The first and second ones color the points lower or higher than the interval of the horizontal conditioning variable, and the other two regulating the same features for the vertical variable. The points which are outside the intervals of both conditioning variables will get a mixed color.

plond.pale: minimum and maximum paling, to be applied for distance 0 and maximal distance from the interval.

plcond.cex: symbol size, relative to cex, used to show the points outside the interval

subset.rgratio adjust plot range for a subset if the range is smaller than subset.rgratio times the plot range for the full data set

functionxvalues if a function is to be shown, the number of argument values for which the function is evaluated

*** options for the function plregr

regr.plotselect selection of diagnostic plots that are produced, see ...

regr.addcomp should residuals be shown as they are or component effects added to them?

leveragelimit ...

cookdistancelines values of Cook's distance for which contours will be shown on the leverage plot

stamp logical: should stamps be shown in the bottom right concern documenting the date and any project and step titles?

doc logical: should any documentations of the data set be shown as subtitles, i.e., at in the top margin of the plot?

printnotices: logical: should notices produced by the functions be shown?

debug: Some functions that produce nice-to-have features are prevented from aborting the process if they fail (by using the try function) and produce a warning instead – unless debug is TRUE

Author(s)

Werner A. Stahel

See Also

ploptions

Examples

```
names(default.ploptions)
```

---

plpanel *Panel function for multiple plots*

Description

Draw a scatterplot or multibox plot, usuallly after `pl.control` and `plframe` have been called. May also be used to augment an existing plot.

Usage

```
plpanel(x = NULL, y = NULL, indx = NULL, indy = NULL, type = "p",
  frame = FALSE, title = FALSE,
  plargs = NULL, ploptions = NULL, marpar = NULL, ...)

panelSmooth(x, y, indx, indy, plargs = NULL, ...)

plpanelCond(x, y, ckeyx, ckeyy, pch = 1, pcol = 1, psize = 1,
  pale = c(0.2, 0.6), csize = 0.8,
  smooth = NULL, smooth.minobs = NULL, plargs = NULL, ploptions = NULL, ...)
```

Arguments

x	values of the horizontal variable
y	values of the vertical variable
indx	index of the variable shown horizontally, among the y variables
indy	index of the variable shown horizontally, among the y variables
type	type of plot as usual in R: "p" for points, ...
frame	logical: should `plframe` be called?
title	logical: should `pltitle` be called?
ckeyx, ckeyy	vectors of 'keys' to calculate paling values and weights for smoothing. NA means that points should not be shown in this panel. 0 means no paling and weight 1. Other values are between -1 and 1, `cpl=(1-abs(ckeyx))*(1-abs(ckeyy))` is used for paling and weights.
pch, pcol, psize	vector of plotting symbols, colors and sizes for plotting points
pale	vector of length 2 indicating the range of paling values obtained from `cpl` values from 1 to 0.
csize	factor applied to the character expansion of the points with `cpl<1`

smooth	should a smooth line be drawn?
smooth.minobs	minimum number of points required for calculating and showing a smooth line
plargs, ploptions	result of calling pl.control. If plargs is NULL, pl.control will be called to generate it. The components are often needed to generate the panel.
marpar	margin parameters, if already available. By default, they will be retieved from ploptions.
...	further arguments passed to plpoints, plmboxes, plsmooth

Details

The panel function plpanel draws a scatterplot if both x and y are numerical, and a multibox plot if one of them is a factor and ploptions$factor.show == "mbox".
Grouping, reference and smooth lines and properties of the points are determined by the component of plargs in plpanel.

This function is usually called by the high level pl functions plyx and plmatrix. A different suitable function can be used by setting their argument panel.

The first arguments, x and y, can be formulas, and an argument data can be given. These arguments then have the same meaning as in plyx, with the restriction that only one variable should result for the x and y coordinates in the plot. When frame is true, plpanel can be used instead of plyx for generating a single plot. Note that plpanel does not modify pl.envir, in contrast to plyx.

plpanelCond shows selected points only and may show some of them with reduced size and paled color. It is appropriate for the high level function plcond.

Value

none

Note

These functions are rarely called by the user. The intention is to modify ond of them and then call the modified version when using plyx, plmatrix or plcond by setting panel=mypanel.

Author(s)

Werner A. Stahel, ETH Zurich

See Also

plyx is essentially a wrapper function of plpanel which calls pl.control and provides additional features. plmatrix also uses plpanel, whereas plcond uses plpanelCond.

Examples

```
t.plargs <-
  pl.control(~Species+Petal.Length, ~Sepal.Width+Sepal.Length,
             data=iris, smooth.group=Species, pcol=Species)
t.plargs$ploptions$group.col <- c("magenta","orange","cyan")
```

```
plpanel(iris$Petal.Length, iris$Petal.Width, plargs=t.plargs,
        frame=TRUE)
```

---

plpoints *Plot Points and Lines in the 'pl' system*

Description

Low level functions for plotting point and lines based on the 'pl' paradigm.

Usage

```
plpoints(x=NULL, y=NULL, type="p", plab=NULL, pch=NULL,
  pcol=NULL, col=NULL, lcol=NULL, lty=NULL, lwd=NULL, psize=NULL,
  csize = NULL, group = NULL, plargs = NULL, ploptions = NULL,
  marpar = NULL, xy = TRUE, ...)

pllines(x, y, type="l", ...)
```

Arguments

x, y	coordinates for the horizontal and veritical axis, respectively. If NULL, they will be retrieved from plargs$pldata.
type	type of displaying points. See ?points.
plab	labels for displaying points. Overrides labels provided by plargs$pdata[["plab"]].
pcol, col	color for points. col is used if pcol is NULL
lcol	color for lines
pch, psize, csize, lty, lwd	... and col in plpoints: plotting character(s), relative size, median character expansion, and color for plotting points, and line type. Overrides other settings, defined in plargs.
group	grouping of observations, used to determine pch and col
plargs, ploptions	result of pl.control, see Details
marpar	margin parameters, if already available. By default, they will be retieved from ploptions.
xy	logical: should the coordinates be obtained as in high level graphics? This is set to FALSE to save time and avoid complications, in case the user is sure that x and y are vectors rather than formulas or variable names.
...	absorbs extra arguments

Details

For plpoints, the first arguments, x and y can be formulas, and an argument data can be given. These arguments then have the same meaning as in plyx.

plargs and ploptions may be specified explicitly, but they are usually generated by calling pl.control.

Value

`plsmooth` invisibly returns the data.frame needed for drawing the smooth line. The other functions return `NULL`

Author(s)

Werner A. Stahel

See Also

pl.control

Examples

```
plyx(Sepal.Width ~ Sepal.Length, data=iris, pcol=Species)

da <- aggregate(iris[,1:4], list(Species=iris$Species), mean)
plpoints(Sepal.Width ~ Sepal.Length, plargs=list(pldata=da),
  plab=da$Species, csize.pch=1, pcol=as.numeric(da$Species))
```

---

plregr *Diagnostic Plots for Regr Objects*

Description

Diagnostic plots for fitted regression models: Residuals versus fit (Tukey-Anscombe plot) and/or target variable versus fit; Absolute residuals versus fit to assess equality of error variances; Normal Q-Q plot (for ordinary regression models); Residuals versus leverages to identify influential observations; Residuals versus sequence (if requested); and residuals versus explanatory variables. These plots are adjusted to the type of regression model.

Usage

```
plregr(x, data = NULL, plotselect = NULL, xvar = TRUE,
  transformed = NULL, sequence = FALSE, weights = NULL,
  addcomp = NULL, smooth = 2, smooth.legend = FALSE, markextremes = NA,
  plargs = NULL, ploptions = NULL, assign = TRUE, ...)

plresx(x, data = NULL, xvar = TRUE, transformed = NULL,
  sequence = FALSE, weights = NULL,
  addcomp = NULL, smooth = 2, smooth.legend = FALSE, markextremes = NA,
  plargs = NULL, ploptions = NULL, assign = TRUE, ...)
```

Arguments

x	"regr" (or also lm or glm) object, result of a call to regr() from package **regr**. This is the only argument needed. All others have useful defaults.
data	data set where explanatory variables and the following possible arguments are found: weights, plweights, pch, plabs
plotselect	which plots should be shown? See Details
xvar	if TRUE, residuals will be plotted versus all explanatory variables (or terms, according to argument 'transformed') in the model (plregr will call plresx). If it is a character vector, it contains the variables to be used. If it is a formula, its right hand side contains these variables. The model formula is updated by such a formula. Whence, the use of \~{}.+ adds variables to those in the model. If any variables are not be contained in the model, the argument data is needed.
transformed	logical: should residuals be shown against transformed explanatory variables? If TRUE, the variables are transformed as implied by the model.
sequence	if TRUE, residuals will be plotted versus the sequence as they appear in the data. If another explanatory variable is monotone increasing or decreasing, the plot is not shown, but a warning is given.
weights	if TRUE, residuals will be plotted versus x$weights. Alternatively, a vector of weights can be specified
addcomp	logical: should component effects be added to residuals for residuals versus input variables plots?
smooth	logical: should a smooth line be added?
smooth.legend	When a grouping factor is used (argument smooth.group, see below), this argument determines whether and where the legend for identifying the groups should be shown, see Details
markextremes	proportion of extreme residuals to be labeled. If all points should be labeled, let markextremes=1.
plargs	result of calling pl.control. If NULL, pl.control will be called to generate it. If not null, arguments given in ... will be ignored.
ploptions	list of pl options.
assign	logical: Should the plargs be stored in the pl.envir environment?
...	Many further arguments are available to customize the plots, see below for some of the most useful ones, and plregr.control for a complete list.

Details

Argument plotselect is used to determine which plots will be shown. It should be a named vector of numbers indicating

0 do not show

1 show without smooth

2 show with smooth (not for qq nor leverage)

3 show with smooth and smooth band (only for `resfit` in `plregr` and in `plresx`)

The default is `c( yfit=0, resfit=smdef, absresfit = NA, absresweights = NA, qq = NA, leverage = 2, resmatrix = 1, qqmult = 3)`, where `smdef` is 3 (actually argument `smooth` of `plregr.control` plus 1) for normal random deviations and one less (no band) for others.

Modify this vector to change the selection and the sequence in which the plots appear. Alternatively, provide a named vector defining all plots that should be shown on a different level than the default indicates, like `plotselect = c(resfit = 2, leverage = 1)`. Adding an element `default = 0` suppresses all plots not mentioned. This is useful to select single plots, like `plotselect = c(resfit = 3, default = 0)`

The names of `plotselect` refer to:

yfit response versus fitted values

resfit residuals versus fitted values (Tukey-Anscombe plot)

absresfit residuals versus fitted values, defaults to TRUE for ordinary regression, FALSE for glm and others

absresweights residuals versus weights

qq normal Q-Q plot, defaults to TRUE for ordinary regression, FALSE for glm and others

leverage residuals versus leverage (hat diabgonal)

resmatrix scatterplot matrix of residuals for multivariate regression

qqmult qq plot for Mahlanobis lengths versus sqrt of chisquare quantiles.

In the 'resfit' (Tukey-Anscombe) plot, the reference line indicates a "contour" line with constant values of the response variable, $Y = \widehat{y} + r =$ constant. It has slope `-1`. It is useful to judge whether any curvature shown by the smooth might disappear after a nonlinear, monotone transformation of the response.

If `smresid` is true, the 'absresfit' plot uses modified residuals: differences between the ordinary residuals and the smooth appearing in the 'resfit' plot. Analogously, the 'qq' plot is then based on yet another modification of these modified residuals: they are scaled by the smoothed scale shown in the 'absresfit' plot, after these scales have been standardized to have a median of 0.674 (`=qnorm(0.75)`).

The smoothing function used by default is `smoothRegr`, which calls `loess`. This can be changed by setting `ploptions(smooth.function=<func>)`, which must have the same arguments as `smoothRegr`.

The arguments `lty, lwd, colors` characterize how the graphical elements in the plot are shown. They should be three vectors of length 9 each, defining the line types, line widths, and colors to be used for ...

[1] observations;

[2] reference lines;

[3] smooth;

[4] simulated smooths;

[5] component effects in plresx;

[6] confidence bands of component effects. In the case of `glm.restype="cond.quant"`

[7] (random) observations;

[8] conditional medians;

[9] bars showing conditional quantiles.

If smooths are shown according to groups (given in `smooth.group`), then a legend can be required and positioned in the respecive plots by using the argument `smooth.legend`. If it is `TRUE`, then the legend will be placed in the `"bottomright"` corner. Alternatively, the corner can be specified as "bottomright", "bottomleft", "topleft", or "topright". A coordinate pair may also be given. These possibilities can be used individually for each plot by giving a named vector or a named list, where the names are one of "yfit", "resfit", "absresfit", "absresweight", ".xvar." or names of x variables provided by the `xvar` argument. A component ".xvar." selects the first x variable.

There is an hidden argument `innerrange.fit` that allows for fixing an inner range for plotting the fitted values.

Value

The list of the evaluations of all arguments and some more useful items is returned invisibly.

Note

This is a function under development. Future versions may behave differently and may not be compatible with this version.

Author(s)

Werner A. Stahel, ETH Zurich

See Also

`plregr.control`, `plot.lm`

Examples

```
data(LifeCycleSavings, package="datasets")
r.savings <- lm(sr ~ pop15 + pop75 + dpi + ddpi, data = LifeCycleSavings)
plregr(r.savings)

## --- *transformed* linear model
data(d.blast)
r.blast <-
     lm(log10(tremor) ~ location+log10(distance)+log10(charge),
          data=d.blast)
plregr(r.blast, sequence=TRUE, transformed=TRUE)
plregr(r.blast, xvar=FALSE, innerrange.fit=c(0.3,1.2))


## --- multivariate regression
data(d.fossileSamples)
r.foss <-
  lm(cbind(sAngle,lLength,rWidth) ~ SST+Salinity+lChlorophyll+Region+N,
  data=d.fossileSamples)
plregr(r.foss, plotselect=c(resfit=3, resmatrix=1, qqmult=1))
```

```
## --- logistic regression
data(d.babysurvival)
rr <- glm(Survival ~ Weight+Age+Apgar1, data=d.babysurvival, family=binomial)
plregr(rr, xvar= ~Weight, cex.plab=0.7, ylim=c(-5,5))
plregr(rr, condquant=FALSE)

## --- ordinal regression
if(requireNamespace("MASS")) {
data(housing, package="MASS")
rr <- MASS::polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
plregr(rr, factor.show="jitter")
}
```

`plregr.control` *Further Arguments to* `plregr`

Description

Specify some arguments of minor importance for the function `plregr`

Usage

```
plregr.control(x, data = NULL, xvar = TRUE, transformed = FALSE,
  weights = NULL, stdresid = TRUE, mar = NULL,
  glm.restype = "working", condquant = TRUE, smresid = TRUE,
  partial.resid = NULL, addcomp = NULL, cookdistlines = NULL,
  leveragelimit = NULL, condprob.range = NULL,
  testlevel = 0.05,
  refline = TRUE,
  smooth = 2,
  smooth.sim = NULL,
  xlabs = NULL, reslabs = NULL, markextremes = NULL,
  mf = TRUE, mfcol = FALSE, multnrow = 0, multncol = 0, marmult = NULL,
  oma = NULL, assign = TRUE, ...)
```

Arguments

`x`	an object (result of a call to a model fitting function such as `lm`, `glm`, ... This is the only argument that is needed. All others have useful defaults.
`data`	see ?`plregr`
`xvar`	variables for which residuals shall be plotted. Either a formula like `~ x1 + x2` or a character vector of names. Defaults to all variables (or terms, see `transformed`) in the model.
`transformed`	see ?`plregr`
`weights`	logical: should residuals be plotted against weights? Used in `plresx`.

stdresid	logical: should leverages and standardized residuals be calculated? This is avoided for plresx
mar	plot margins
glm.restype	type of residuals to be used for glm models. In addition to those allowed in residuals() for glm objects, type condquant is possible for (ungrouped) binary regression. See ?residuals.regrpolr for an explanation. Warning: type "deviance" will not work with simulated smooths since NAs will emerge.
condquant	logical: should conditional quantiles be shown for censored observations, binary and ordered responses?
smresid	logical: Should residuals from smooth be used for 'tascale' and 'qq' plots?
partial.resid, addcomp	logical, synonyms: Should component effects be added to the residuals? This leads to what some authors call "partial residual plot".
cookdistlines	levels of Cook distance for which contours are plotted in the leverage plot
leveragelimit	bound for leverages to be used in standardizing residuals and in calculation of standardized residuals from smooth (if smresid is TRUE).
condprob.range	numeric vector of length 2. In the case of residuals of class condquant, quartile bars are only drawn for residuals with probability between condprob.range[1] and condprob.range[1]. Default is c(0.05,0.8) for less than 50 observations, and c(0,0), suppressing the bars, otherwise.
testlevel	level for statistical tests
refline	logical: should reference line be shown? If refline==2, a confidence band be drawn for the component effects
smooth	if TRUE (or 1), smooths are added to the plots where appropriate. If ==2, smmooths to positive and negative residuals-from-smooth are also shown.
smooth.sim	number of simulated smooths added to each plot. If NULL (the default) 19 simulated smooths will be generated if possible and sensible (i.e., none if smooth.group is set).
xlabs	labels for x variables. Defaults to vars
reslabs	labels for vertical axes
markextremes	proportion of extreme residuals to be labeled. If all points should be labeled, let markextremes=1.
mf	vector of 2 elements, indicating the number of rows and columns of panels on each plot page. Defaults to c(2,2), except for multivariate models, where it adjusts to the number of target variables. mf=c(1,1) or mf=1 asks for a single frame per page. mf=NA or mf=0 leaves the framing (and oma) unchanged.
mfcol	if TRUE, the panel will be filled columnwise
multnrow, multncol	number of rows and columns of panels on one page, for residuals of multivariate regression only
marmult	plot margins for scatterplot matrices in the case of multivariate regression
oma	vector of length 4 giving the number of lines in the outer margin. If it is of length 2, they refer to top an right margins.

assign	logical: should the result of pl.control be assigned to the pl.envir environment? This will be done for high level pl functions, but avoided for low level ones. It allows for reusing the settings and helps debug unexpected behavior.
...	further arguments in the call, to be ignored by 'plotregr.control'

Value

A list containing all the items needed to specify plotting in plregr and plresx

Note

This function is not explicitly called by the user, but by plregr and plresx. All the arguments specified here can and should be given as arguments to these functions.

Author(s)

Werner A. Stahel, Seminar for Statistics, ETH Zurich

See Also

plregr and plresx

Examples

```
data(d.blast)
( r.blast <-
  lm(log10(tremor)~location+log10(distance)+log10(charge), data=d.blast) )

plargs <- plregr.control(r.blast, formula = ~.+distance, transformed=TRUE,
smooth.group = location )
showd(plargs$pdata)
names(plargs)
```

---

plres2x *Plot Residuals vs. Two Explanatory Variables*

Description

Plot 2 variables, showing a third one with line symbols. Most suitable for showing residuals of a model as this third variable.

Usage

```
plres2x(formula = NULL, reg = NULL, data = NULL, restrict = NULL,
  size = 1, xlab = NULL, ylab = NULL, pale = 0.2,
  plargs = NULL, ploptions = NULL, assign = TRUE, ...)
```

Arguments

formula	a formula of the form ~x+y, where x, y are the 2 variables shown by the coordinates of points, and residuals are shown by line symbols: their orientation corresponds to the sign of the residual, and their length, to the absolute value.
reg	the result of the model fit, from which the residuals are extracted
data	the data.frame where the variables are found. Only needed if the variable 'x' or 'y' is not available from the fitting results.
restrict	absolute value which truncates the size. if TRUE, the inner plotting limits of the residuals is used if available. Truncation is shown by stars at the end of the line symbols.
size	the symbols are scaled so that size*par("cin")[1] is the length of the largest symbol, as a percentage of the length of the horizontal axis.
xlab, ylab	labels for horizontal and vertical axes. Default to the variable names (or labels)
.	
pale	scalar between 0 and 1: The points are shown in a more pale color than the segments as determined by colorpale with argument pale=pale.
plargs	result of calling pl.control. If NULL, pl.control will be called to generate it. If not null, arguments given in ... will be ignored.
ploptions	list of pl options.
assign	logical: Should the plargs be stored in the pl.envir environment?
...	further arguments, passed to plotregr.control

Value

none.

Author(s)

Werner A. Stahel and Andreas Ruckstuhl

Examples

```
data(d.blast)
t.r <- lm(log10(tremor)~location+log10(distance)+log10(charge),
          data=d.blast)
plres2x(~distance+charge, t.r)
```

plscale *Generate Plscaled Plotting Scale*

Description

Generates plscaled values and appropriate tick mark positions and labels for expressing a variable on a plscaled scale, e.g., on log scale

Usage

```
plscale(x, plscale = "log10", ticksat = NULL, logscale = NULL,
  valuesonly = FALSE, ploptions = NULL)
```

Arguments

x	data to be used in plotting
plscale	name of the function defining the plscaled scale
ticksat	tick locations, If NULL, these locations will be generated by the function. An attribute attr(..., "ticklabels") may also be given.
logscale	if NULL, R's function axTicks will be called if the plscale is a log function.
valuesonly	logical: should only the transformed values be returned? Otherwise, axis ranges and tick information is also calculated.
ploptions	See ploptions

Value

The x data is returned, augmented by the following attributes:

numvalues the plscaled values to be used for plotting

ticksat the location of tick marks (plscaled values)

ticklabels the labels for the tick marks showing the original scale

plscale the name of the function used for the plscaleation

Note

Besides the logarithmic plscale that is supported by core R graphics, any other plscaleation may be used, notably the so-called "first aid plscaleations".

Author(s)

Werner A. Stahel

See Also

axTicks, prettyscale

Examples

```
x <- 10^seq(-1,3,0.5)
plscale(x)
xx <- plscale(x, plscale="sqrt")
plyx(xx)
x <- seq(0,100,2)
plyx(plscale(x, plscale="asinp"), type="l")
```

---

plsmooth *Smooth and Reference Line Plotting*

Description

These functions add smooths or reference lines to an existing pl plot.

Usage

```
plsmooth(x = NULL, y = NULL, ysec = NULL, band=NULL, power = NULL,
  group = NULL, weight = NULL, smooth = TRUE,
  plargs = NULL, ploptions = NULL, xy = TRUE, ...)

plsmoothline(smoothline = NULL, x = NULL, y = NULL, ysec = NULL,
  smooth.col = NULL, smooth.lty = NULL, smooth.lwd = NULL,
  plargs = NULL, ploptions = NULL, marpar = NULL, ...)

plrefline(refline, x=NULL, innerrange=NULL, y=NULL,
  cutrange = c(x = TRUE, y = FALSE), plargs=NULL, ploptions=NULL, ...)
```

Arguments

x, y	coordinates for the horizontal and veritical axis, respectively. If NULL, they will be retrieved from plargs$pldata.
ysec	for plsmooth, plsmoothline: matrix of secondary y values. The smooths generated or given by its columns will be drawn thinner and with paled color.
band	logical: should a band (e.g., a confidence band) be drawn together with the smooth?
power	for plsmooth: smooth will be calcutated for y^power and the back-transformed. Usually, power=0.
group	for plsmooth: grouping variable. If NULL, the variable .smooth.group. column in plargs$pldata will be used if available. If group is of length 1, there will be no grouping
weight	weights of observations used for generating the smooth
smooth	logical: should smooth be done? Will almost always be TRUE
smoothline	for plsmoothline: result of a smooth fitting

`smooth.col`, `smooth.lty`, `smooth.lwd`
: for `plsmoothline`: color, line type and line width for the smooth line(s). By default, they will be taken from ploptions.

`refline`
: for `plrefline`: A two element vector giving intercept and slope of a straight line, or a function that returns these as the first 2 elemnts of the result's `coef` component, such as `lm`. For more possibilities, see Details.

`innerrange`
: for `plrefline`: inner range in `x` direction - only needed if the refline should be clipped at a range different from the `innerrange` attribute of the horizontal variable

`cutrange`
: for `plrefline`: logical vector of length 2: should the reference line(s) be cut at the inner plotting ranges in x- and y-direction? Otherwise, it will be continued outside it with the appropriate transformation.

`plargs`, `ploptions`
: result of `pl.control`, see Details

`marpar`
: margin parameters, if already available. By default, they will be retieved from `ploptions`.

`xy`
: logical: should the coordinates be obtained as in high level graphics? This is set to `FALSE` to save time and avoid complications, in case the user is sure that `x` and `y` are vectors rather than formulas or variable names.

`...`
: absorbs extra arguments

Details

The argument `refline` accepts different types of values. If it is a function, it must either accept a formula (which will be `y~x`) as its first argument or `x` and `y` as the first two arguments.
Alternatively, `refline` can be a list with components `x` and `y` and possibly a component `band` that contains the coordinates of the line (or lines, if `y` is a matrix) and the width of a band around it (that is, additional lines, to be drawn with `ploptions("refline.col")[2]`).
In order to obtain more than one reference line, a list of such items may be given. It should not have compontents named `coef`, `coefficients`, `x` or `y`, since it would otherwise be mistaken for an argument of the types just described. The components may carry attributes `lty`, `lwd` and `lcol` to specify the properties of the lines individually. See Examples.

`plsmooth` and `plrefline` are very similar. They are both called by high level pl functions. `plsmooth` gets its smoothing function from `ploptions("smooth.function")`. Their properties (line type, width, color) come from different sets of pl options. `plsmooth` can also respect a group structure in the data.

If `x` or `y` has an attribute `"numvalues"`, these are used as the values to calculate the smooth or the refline.

`plargs` and `ploptions` may be specified explicitly, but they are usually generated by calling `pl.control`.

The argument `getpar` is used for setting the graphical parameters `mar`, `mgp` according to `ploptions`? This is needed if the high level pl function has changed `mar`, since this change has been reversed when the function was left. By default, these graphical parameters will be retieved from `pl.envir$ploptions`.

Value

plsmooth invisibly returns the data.frame needed for drawing the smooth line. The other functions return NULL

Author(s)

Werner A. Stahel

See Also

pl.control

Examples

```
plyx(Sepal.Width ~ Sepal.Length, data=iris, smooth=TRUE,
  smooth.group=Species, pch=Species)
plsmooth(smooth.group=FALSE)

## plrefline  called from  plyx
plyx(Sepal.Width ~ Sepal.Length, data=iris, smooth=TRUE, pch=Species,
     smooth.group=iris$Species, refline=lm)
## more reference lines
plrefline(list(c(-2,1), structure(c(-2.3,1), lcol="purple", lty=1)))
```

---

plsubset *Subsetting a Data.Frame with pl Attributes*

Description

Select rows of data.frames keeping the variable attributes that drive pl graphics

Usage

```
plsubset(x, subset = NULL, omit = NULL, select = NULL, drop = FALSE,
         keeprange = FALSE)
```

Arguments

x	data.frame from which the subset is to be generated
subset, omit	logical vector or vector of indices of rows or or rownames of x. If subset is used, omit is ignored.
select	vector of indices or names of variables to be selected
drop	logical: if only one variable remains, should the data.frame be converted into a vector?
keeprange	logical: should ranges (inner.range and plrange) be maintained?

Details

plsubset maintains the 'pl' attributes of the variables of the data.frame (if there are), such as 'col', 'lty', ..., and subsets the two attributes 'numvalues' and 'plcoord'. This is useful if the way of displaying the axis is to be kept when a new plot is drawn.

Value

Data.frame with the selected rows (or without the omitted rows, respectively) and all attributes as described above.

Author(s)

Werner A. Stahel

See Also

Argument subset of the high level 'pl' functions plyx, plmatrix

Examples

```
data(d.river)
dd <- d.river[seq(1,1000,4),]
dd$date <- gendateaxis("date",hour="hour", data=dd)
attr(dd$date, "ticksat")

dsubs <- plsubset(dd, subset=1:50)
attr(dsubs$date, "ticksat")

plyx(O2~date, data=dsubs)
## same as
## plyx(O2~date, data=dd, subset=1:50)
```

---

plticks *Ticks for plotting*

Description

Find ticks locations and labels

Usage

```
plticks(range, plscale = NULL, transformed = FALSE, nouter = 0,
  tickintervals = NULL, ploptions = NULL)
```

Arguments

range	range of values that the ticks should cover
plscale	function defining the scale of the axis. Either the name of the function or a function, see Details.
transformed	logical: Is range scaled according to plscale rather than in original scale?
nouter	number of outer ..
tickintervals	approximate number of tick intervals desired. Default is taken from ploptions('tickintervals').
ploptions	pl options

Details

plticks calls pretty for getting tick locations if plscale is not specified and prettyscale if it is. It generates another set for locations of tick labels if tickintervals has 2 elements, such that not all ticks are labelled.

The scaling function plscale can be given by its name if that name is one of log, log10, logst, sqrt, asinp, logit, qnorm. Otherwise, it must be a function with an attribute inverse that defines the inverse function. It should also have an attribute range and an attribute range.transformed if the possible range for its argument or its values are restricted, like asinp that is defined for values between 0 and 100 and has values in the interval from 0 to 1.

Value

A list with components

ticksat	locations of ticks
ticklabelsat	locations of tick labels
ticklabels	tick labels, if plscale is given

Author(s)

Werner A. Stahel

See Also

pretty, prettyscale, plaxis

Examples

```
plticks(c(23,87))
plticks(c(23,91), plscale="asinp", transformed=FALSE,
  tickintervals=c(10,2))

asinp ## shows the attributes 'inverse', 'range' and 'range.transformed'
```

plyx *Scatterplot, enhanced*

Description

A scatterplot or a bunch of them is produced according to the concept of the plplot package

Usage

```
plyx(x = NULL, y = NULL, by=NULL, group = NULL, data = NULL, type = "p",
  panel = NULL, xlab = NULL, ylab = NULL, xlim = NULL, ylim = NULL,
  markextremes = 0, rescale = TRUE, mar = NULL, mf = FALSE,
  plargs = NULL, ploptions = NULL, assign = TRUE, ...)
```

Arguments

x	either a formula or the data to be used for the horizontal axis. If a formula of the type 'y~x', the variable 'y' in 'data' will be plotted against the variable(s) 'x'. If a data.frame with more than one column is given, each column will be used in turn to produce a plot.
y	data to be used as the y axis.
by	grouping factor: for each by group, a plot will be shown for the respective subset of the data
group	grouping that determines plotting symbols, colors, and line types
data	data.frame containing the variables if 'x' is a formula
xlab, ylab	axis labels
xlim, ylim	plot ranges
type	type of plot, see ?plot.default
panel	panel function to do the actual drawing. See Details.
markextremes	proportion of extreme residuals to be labeled. If all points should be labeled, let markextremes=1.
rescale	logical. Only applies if there are multiple y variables. If TRUE, the vertical axis will be adjusted for each of these variables.
mar	plot margins, see par
mf	number of multiple frames. If more than one plot will be generated because of a grouping or multiple x variables, multiple frames will be produced by calling plmframes unless mf is FALSE. If mf is TRUE, the function will determine the number of rows and columns suitably. If mf is a vector of length 2, these numbers will be used for the number of panels in rows and columns (unless they are too large for the restriction in ploptions("mframesmax")). If it has lenngth 1, this is used as the total number of panels on a page.
plargs	result of calling pl.control. If NULL, pl.control will be called to generate it. If not null, arguments given in ... will be ignored.

ploptions	list of pl options.
assign	logical: Should the plargs be stored in the pl.envir environment?
...	more arguments, to be passed to pl.control

Details

panel defaults to plpanel, which results essentially in points or text depending on the argument pch including a smooth line, to plmboxes if 'x' is a factor and 'y' is not or vice versa, or to a modification of sunflowers if both are factors.
The function must have the arguments x and y to take the coordinates of the points and may have the arguments indx and indy to transfer the two variables' indexes and panelargs for any additional objects to be passed on.

Value

None.

Note

There are many more arguments, obtained from pl.control, see ?pl.control. These can be passed to plmatrix by an argument plargs that is hidden in the ... argument list.

Author(s)

Werner A. Stahel, ETH Zurich

See Also

plmatrix, plcond; pl.control, ploptions

Examples

```
plyx(Petal.Width ~ Sepal.Length, data=iris)
plyx(Petal.Width ~ Sepal.Length+Sepal.Width, data=iris, smooth=TRUE,
     group=Species)
plyx(Petal.Length + Petal.Width ~ Sepal.Length+Sepal.Width,
     by  = Species, data=iris, smooth=TRUE)
```

---

predict.regrpolr *Predict and Fitted for polr Models*

Description

Methods of predict and fitted

Usage

```
## S3 method for class 'regrpolr'
predict(object, newdata = NULL,
  type =  c("class", "probs", "link"), ...)
## S3 method for class 'regrpolr'
fitted(object, type = c("class", "probs", "link"), ...)
```

Arguments

object	result of polr
newdata	data frame in which to look for variables with which to predict. If NULL, fitted values are produced.
type	type of prediction: "link" asks for the linear predictor values. Other types are available according to the standard methods of the predict function.
...	arguments passed to standard methods of predict or fitted

Value

Vector of predicted or linear predictor values

Author(s)

Werner A. Stahel, ETH Zurich

See Also

predict, fitted, residuals.regrpolr

Examples

```
if(requireNamespace("MASS")) {
data(housing, package="MASS")
rr <- MASS::polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
aa <- fitted(rr)
bb <- predict(rr)
cc <- predict.regrpolr(rr)
}
```

prettyscale *Pretty Tickmark Locations for Transformed Scales*

Description

Compute about n 'round' values that are about equally spaced in a transformed (plotting) scale and cover the range of the values in x.

Usage

```
prettyscale(x, transformed = FALSE, plscale = "log10", inverse = NULL,
    range = NULL, range.transformed = NULL, n = NULL, logscale = NULL)
```

Arguments

x	numeric vector of data (original scale)
transformed	logical: Is x scaled according to plscale rather than in original scale?
plscale	name of the transformation defining the plotting scale
inverse	back (or inverse) back transformation
range, range.transformed	admissible range of original and transformed values, respectively. Usually not needed, cf. Details
n	approximate number of tickmark locations. If of length >=2, n[2] can be varied to obtain more adequate locations. See Details.
logscale	if NULL, R's function axTicks will be called if the plscale is a log function.

Details

prettyscale generates n+2 "anchor" values in the transformed scale which cover the range of the transformed x values and are equidistant within the range. It then back-transforms these anchor values. For each one of them, say c, it seeks a pretty value near to it by the following construction: it calls the R function pretty on the range given by the back-transformed neighboring anchor values, asking for n[2] pretty values. From these, it chooses the one for which the transformed value is closest to the transformed c.

Therefore, if n[2] is large, the pretty values may be less pretty, whereas small n[2] may lead to equal pretty values for neighboring anchors and thus to too few resulting pretty values. The default value for n[2] is 3.

The ranges are needed to get the limits as pretty values when appropriate (and to avoid warning messages). They are generated in the function for the commonly used plscales and may be given as attributes of the plscale function, see Examples.

Value

Numeric vector of tick mark locations in transformed scale, with an attribute ticklabels containing the appropriate tick marks and labels (in original scale)

Note

The function does not always lead to consistent results. Increasing n sometimes leads to fewer resulting values.

Author(s)

W. A. Stahel

See Also

axTicks, plticks

Examples

```
prettyscale(10^rnorm(10))
prettyscale(c(0.5, 2, 10, 90), plscale="sqrt")
prettyscale(c(50,90,95,99), plscale="asinp", n=10)
## asinp has the useful attributes:
asinp
```

---

prevgumbel *"Reverse" Gumbel Distribution Functions*

Description

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

Usage

```
drevgumbel (x, location = 0, scale = 1)
prevgumbel (q, location = 0, scale = 1)
qrevgumbel (p, location = 0, scale = 1)
rrevgumbel (n, location = 0, scale = 1)
```

Arguments

x, q	numeric vector of abscissa (or quantile) values at which to evaluate the density or distribution function.
p	numeric vector of probabilities at which to evaluate the quantile function.
location	location of the distribution
scale	scale (> 0) of the distribution.
n	number of random variates, i.e., length of resulting vector of rrevgumbel(..).

Value

a numeric vector, of the same length as x, q, or p for the first three functions, and of length n for rrevgumbel().

Author(s)

Werner A. Stahel; partly inspired by package **VGAM**. Martin Maechler for numeric cosmetic.

See Also

the `Weibull` distribution functions in R's **stats** package.

Examples

```
curve(prevgumbel(x, scale= 1/2), -3,2, n=1001, col=1, lwd=2,
      main = "revgumbel(x, scale = 1/2)")
abline(h=0:1, v = 0, lty=3, col = "gray30")
curve(drevgumbel(x, scale= 1/2),       n=1001, add=TRUE,
      col = (col.d <- adjustcolor(2, 0.5)), lwd=3)
legend("left", c("cdf","pdf"), col=c("black", col.d), lwd=2:3, bty="n")

med <- qrevgumbel(0.5, scale=1/2)
cat("The median is:",  format(med),"\n")
```

---

quantilew *Quantiles for weighted observations*

Description

Quantiles for weighted observations

Usage

```
quantilew(x, probs = c(0.25, 0.5, 0.75), weights = 1, na.rm=FALSE)
```

Arguments

x	numeric vector whose sample quantiles are wanted 'NA' and 'NaN' values are not allowed unless 'na.rm' is 'TRUE'.
probs	numeric vector of probabilities with values in [0,1].
weights	numeric vector of weights. They will be standardized to sum to 1.
na.rm	remove NAs from 'x'? If FALSE and 'x' contains NAs, the value will be NA.

Value

Empirical quantiles corresponding to the given probabilities and weights. If a quantile is not unique since the cumulated weights hit the probability value exactly (the case of the median of a sample of even size), the mean of the corresponding values is returned.

Author(s)

Werner A. Stahel

See Also

quantile

Examples

```
x <- c(1,3,4,8,12,13,18,20)
quantile(x, c(0.25, 0.5))
quantilew(x, c(0.25, 0.5), weights=1:8)  ##  8  13
## relative weights  (1+2+3)/36  sum to  <0.25 , with the forth, they
##   are over 0.25, therefore, the quantile is the 4th value
```

---

quinterpol *Interpolated Quantiles*

Description

This function implements a version of empirical quantiles based on interpolation

Usage

```
quinterpol(x, probs = c(0.25, 0.5, 0.75), extend = FALSE)
```

Arguments

x	vector of data determining the quantiles
probs	vector of probabilities defining which quantiles should be produced
extend	logical: Should quantiled be calculated outside the range of the data by linear extrapolation? This may make sense if the sample is small or the data is rounded or grouped or a score.

Details

The empirical quantile function jumps at the data values according to the usual definition. The version of quantiles calculated by 'quinterpol' avoids jumps. It is based on linear interpolation of the step version of the empirical cumulative distribution function, using as the given points the midpoints of both vertical and horizontal pieces of the latter. See 'examples' for a visualization.

Value

vector of quantiles

Author(s)

Werner A. Stahel

See Also

quantile

Examples

```
## This example illustrates the definition of the "interpolated quantiles"

set.seed(2)
t.x <- sort(round(2*rchisq(20,2)))
table(t.x)
t.p <- ppoints(100)
plot(quinterpol(t.x,t.p),t.p, type="l")
```

`residuals.regrpolr` *Residuals of a Binary, Ordered, or Censored Regression*

Description

Methods of `residuals` for classes `polr`, `survreg` and `coxph`, calculating quartiles and random numbers according to the conditional distribution of residuals for the latent variable of a binary or ordinal regression or a regression with censored response, given the observed response value. See Details for an explanation.

Usage

```
## S3 method for class 'polr'
residuals(object, type="condquant", ...)
## S3 method for class 'regrpolr'
residuals(object, type="condquant", ...)
## S3 method for class 'regrsurvreg'
residuals(object, type="condquant", ...)
## S3 method for class 'regrcoxph'
residuals(object, type="CoxSnellMod", ...)
```

Arguments

`object`	the result of `polr`, of `glm(,family=binomial)` with binary data for the `regrpolr` method, or of `survreg` or `coxph` for the respective methods.
`type`	type of residuals: `"condquant"` requires conditional quantiles (and more) of the residuals of the model, see Details. For `residuals.regrsurvreg`, type `CoxSnellMod` yields a modified version of Cox-Snell residuals, also including a `condquant` attribute, see Details. Other `type`s are available according to the standard methods of the `residuals` function.
`...`	arguments passed to standard methods of `residuals`

Details

For binary and ordinal regression, the regression models can be described by introducing a latent response variable Z of which the observed response Y is a classified version, and for which a linear regression applies. The errors of this "latent regression" have a logistic distribution. Given the linearly predicted value eta[i], which is the fitted value for the latent variable, the residual for Z[i] can therefore be assumed to have a logistic distribution.

This function calculates quantiles and random numbers according to the conditional distribution of residuals for Z[i], given the observed y[i].

Modified Cox-Snell residuals: Cox-Snell residuals are defined in a way that they always follow an exponential distribution. Since this is an unusual law for residuals, it is convenient to transform them such that they then obey a standard normal distribution. See the vignette for more detail.

Value

Vector of residual values. If conditional quantiles are requested, the residuals for censored observations are replaced by conditional medians, and an attribute `"condquant"` is attached, which is a data.frame with the variables

`median`	median of the conditional distributions
`lowq`	lower quartile
`uppq`	upper quartile
`random`	random number, drawn according to the conditional distribution
`prob`	probability of the condition being true
`limlow`, `limup`	lower and upper limits of the intervals
`index`	index of the observation in the sequence of the result (residuals)
`fit`	linear predictor value
`y`	observed response value

Note

`residuals.polr` and `residuals.regrpolr` are identical for the time being. Only `type="condquant"` is available now.

Author(s)

Werner A. Stahel, ETH Zurich

References

See http://stat.ethz.ch/~stahel/regression

See Also

`condquant`, `plregr`

Examples

```
require(MASS)
data(housing, package="MASS")
rr <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
t.res <- residuals.regrpolr(rr)
head   (t.res)
summary(t.res)
```

`robrange` *Robust Range of Data*

Description

Determines a robust range of the data on the basis of the trimmed mean and mean absolute deviation

Usage

```
robrange(data, trim = 0.2, fac = 5.0, na.rm=TRUE)
```

Arguments

`data`	a vector of data. Missing values are dropped
`trim`	trimming proportion
`fac`	factor used for expanding the range, see Details
`na.rm`	logical: should NAs be removed? If FALSE, result will be NA if there are NAs in 'data'.

Details

The function determines the trimmed mean `m` and then the "upper trimmed mean" `s` of absolute deviations from m, multiplied by `fac`. The robust minimum is then defined as `m-fac*s` or `min(data)`, whichever is larger, and similarly for the maximum.

Value

The robust range.

Author(s)

Werner A. Stahel

See Also

`plcoord`

Examples

```
x <- c(rnorm(20),rnorm(3,5,20))
robrange(x)
```

shortenstring *Shorten Strings*

Description

Strings are shortened if they are longer than n

Usage

```
shortenstring(x, n = 50, endstring = "..", endchars = NULL)
```

Arguments

x	a string or a vector of strings
n	maximal character length
endstring	string(s) to be appended to the shortened strings
endchars	number of last characters to be shown at the end of the abbreviated string. By default, it adjusts to n.

Value

Abbreviated string(s)

Author(s)

Werner A. Stahel

See Also

substring, abbreviate

Examples

```
shortenstring("abcdefghiklmnop", 8)

shortenstring(c("aaaaaaaaaaaaaaaaaaaaa","bbbbc",
  "This text is certainly too long, don't you think?"),c(8,3,20))
```

showd *Show a Part of a Data.frame*

Description

Shows a part of the data.frame which allows for grasping the nature of the data. The function is typically used to make sure that the data is what was desired and to grasp the nature of the variables in the phase of getting acquainted with the data.

Usage

```
showd(data, first = 3, nrow. = 4, ncol. = NULL, digits=getOption("digits"))
```

Arguments

`data`	a data.frame, a matrix, or a vector
`first`	the first `first` rows will be shown and ...
`nrow.`	a selection of `nrow.` rows will be shown in addition. They will be selected with equal row number differences. The last row is always included.
`ncol.`	number of columns (variables) to be shown. The first and last columns will also be included. If `ncol.` has more than one element, it is used to identify the columns directly.
`digits`	number of significant digits used in formatting numbers

Details

The `tit` attribute of `data` will be printed if available and `getUserOption("doc") > 0`, and any `doc` attribute, if `getUserOption("doc") >= 2` (see `tit`).

Value

returns invisibly the character vector containing the formatted data

Author(s)

Werner A. Stahel, ETH Zurich

See Also

`head` and `tail`.

Examples

```
showd(iris)

data(d.birthrates)
names(d.birthrates)
## only show 7 columns, including the first and last
showd(d.birthrates, ncol=7)

showd(cbind(1:100))
```

`simresiduals` *Simulate Residuals*

Description

Simulates residuals for a given regression model

Usage

```
simresiduals(object, ...)
## Default S3 method:
simresiduals(object, nrep=19, simfunction=NULL,
  stdresiduals = NULL, sigma = object$sigma, ...)
## S3 method for class 'glm'
simresiduals(object, nrep=19, simfunction=NULL,
        glm.restype="working", ...)
```

Arguments

`object`	result of fitting a regression
`nrep`	number of replicates
`simfunction`	if a function, it is used to generate random values for the target variable, with three arguments, which will be fed by the number of observations, the fitted values, and `object$sigma` in the case of `simresiduals.default`, respectively. If TRUE, the appropriate random number generator will be used. If NULL (default) the standardized residuals of `object` will be randomly permuted in the case of `simresiduals.default`. For `simresiduals.glm`, this is the same as TRUE.
`stdresiduals`	logical: should standardized residuals be produced?
`sigma`	scale parameter to be used, defaults to `object$sigma`
`glm.restype`	type of residuals to be generated (for glm) Warning: type "deviance" may produce NAs.
`...`	further arguments passed to forthcoming methods.

Details

The simulated residuals are obtained for the default method by replacing the response variable by permuted standardized residuals of the fitted regression, multiplied by the scale `object\$sigma`, then fitting the model to these residuals and getting the reseiduals from this new fit. This is repeated `nrep` times. If standarized residuals are not available, ordinary residuals are used.

For the `glm` method, the values of the response variable are obtained from simulating according to the model (binomial or Poisson), and the model is re-fitted to these generated values.

Value

A matrix of which each column contains an set of simulated residuals. If standardized residuals are available, attribute `"stdresisduals"` is the matrix containing corresponding standardized residuals.

Author(s)

Werner A. Stahel, ETH Zurich

Examples

```
data(d.blast)
r.blast <-
  lm(log10(tremor)~location+log10(distance)+log10(charge),
  data=d.blast)
r.simblast <- simresiduals(r.blast, nrep=5)
showd(r.simblast)
## --------------------------
data(d.babysurvival)
r.babysurv <- lm( Survival~Weight+Age+Apgar1, data=d.babysurvival)
r.simbs <- simresiduals(r.babysurv, nrep=5)
showd(r.simbs)
```

---

smoothpar *Adjust the smoothing parameter to number of observations*

Description

Adjust the smoothing parameter to number of observations

Usage

```
smoothpar(n)
```

Arguments

`n`	number of observations

Value

smoothing parameter

Author(s)

Werner A. Stahel

Examples

```
smoothpar(50)
t.n <- c(5,10,20,100,1000)
smoothpar(t.n)
```

`smoothRegr` *Smoothing function used as a default in plgraphics / straight line "smoother"*

Description

These functions wrap the `loess` smoothing function or the `lm.fit` function in order to meet the argument conventions used in the `plgraphics` package.

Usage

```
smoothRegr(x, y, weights = NULL, par = NULL, iterations = 50, minobs=NULL, ...)
smoothLm(x, y, weights = NULL, ...)
```

Arguments

`x`	vector of x values
`y`	vector of y values to be smoothed
`weights`	vector of weigths used for fitting the smooth
`par`	value for the span argument of `loess`.
`iterations`	number of iterations for the `loess` algorithm. If `==1`, the non-robust, least squares version is applied.
`minobs`	minimal number of observations. If less valid observations are provided, the result is `NULL`.
`...`	Further arguments, passed to `loess`.

Value

vector of smoothed values, with an attribute `xtrim`, which is 1 for `smoothRegr` and 0 for `smoothLm`. If `loess` fails, NAs will be returned without issuing a warning.

Author(s)

Werner A. Stahel, ETH Zurich

See Also

loess, gensmooth

Examples

```
t.x <- (1:50)^1.5
t.y <- log10(t.x) + rnorm(length(t.x),0,0.3)
t.y[40] <- 5
r.sm <- smoothRegr(t.x, t.y, par=0.5)
r.sm1 <- smoothRegr(t.x, t.y, iterations=1, par=0.5)

plot(t.x,t.y)
lines(t.x,r.sm, col=2)
lines(t.x,r.sm1, col=3)
```

---

smoothxtrim *Adjust range for smooth lines to number of observations*

Description

The range in which smooth lines are drawn should be restricted in order to avoid the ill determined parts at both ends. The proportion of suppressed values is determined as a function of the number of observations.

Usage

```
smoothxtrim(n, c=2)
```

Arguments

n	number of observations
c	tuning parameter: how rapidly should the result decrease with n?

Value

proportion of x values for which the smoothline will not be shown on both ends. Equals \ 1.6^(log10(n)*c) / n

Author(s)

W. Stahel

Examples

```
smoothxtrim(50)
t.n <- c(5,10,20,100,1000)
t.n * smoothxtrim(t.n)
```

---

stamp *Add a "Stamp", i.e., an Identification Line to a Plot*

Description

A line is added to the current plot in the lower right corner that contains project information and date.

Usage

```
stamp(sure = TRUE, outer.margin = NULL,
      project = getOption("project"), step = getOption("step"),
      stamp = NULL, line = NULL, ploptions = NULL, ...)
```

Arguments

sure	if FALSE, the stamp will only be added if options("stamp")>0
outer.margin	if TRUE, the stamp is put to the outer margin of the plot. This is the default if the plot is currently split into panels.
project, step	character string describing the project and the step of analysis.
stamp	controls default action, see details
line	line in the (outer) margin on which the stamp should be shown.
ploptions	pl options
...	arguments passed to mtext

Details

The function is used to document plots produced during a data analysis. It is called by all plotting functions of this package. For getting final presentation versions of the plots, the stamp can be suppressed by changing the default by calling options(stamp=0).

In more detail: If stamp==0 (or options("stamp")==0) the function will only do its thing if sure==TRUE.

If stamp==2, it will certainly do it.

If stamp==1 and sure==FALSE, the stamp is added when a plot page is complete.

Value

invisibly returns the string that is added to the plot – consisting of project title, step title and current date (including time).

Author(s)

Werner A. Stahel, ETH Zurich

Examples

```
options(project="Example A",  step="regression analysis")
plot(1:10)
stamp() ##-> "stamp" at bottom of right border
```

`stdresiduals` *Get Standardized Residuals*

Description

Calculates standardized residuals and leverage values.

Usage

```
stdresiduals(x, residuals=NULL, sigma=x$sigma, weights=NULL,
             leveragelimit = NULL)
```

Arguments

`x`	a fitted model object
`residuals`	unstandardized residuals. If missing, they are obtained from `x`
`sigma`	error standard deviation or other scale
`weights`	weights
`leveragelimit`	scalar a little smaller than 1: limit on leverage values to avoid unduely large or infinite standardized residuals

Details

The difference to `stdres()` from package **MASS** is that `stdresiduals` also applies to multivariate regression and can be used with regression model fits not inheriting from `lm`.

The function uses the `qr` decomposition of `object`. If necessary, it generates it.

Value

vector or matrix of standardized residuals, with attributes
`attr(.,"stdresratio")`: ratio of standardized / unstandardized residuals,
`attr(.,"leverage")`: leverage (hat) values,
`attr(.,"weighted")`: weights used in the standardization,
`attr(.,"stddev")`: error standard deviation or scale parameter.

Author(s)

Werner A. Stahel, ETH Zurich

See Also

stdres; hat; hatvalues; influence

Examples

```
data(d.blast)
r.blast <-
  lm(log10(tremor)~location+log10(distance)+log10(charge), data=d.blast)
t.stdr <- stdresiduals(r.blast)
showd(t.stdr)
showd(attr(t.stdr, "leverage"))
```

---

sumNA *Count NAs*

Description

Count the missing or non-finite values for each column of a matrix or data.frame

Usage

```
sumNA(object, inf = TRUE)
```

Arguments

object	a vector, matrix, or data.frame
inf	if TRUE, Inf and NaN values are counted along with NAs

Value

numerical vector containing the missing value counts for each column

Note

This is a simple shortcut for `apply(is.na(object),2,sum)` or `apply(!is.finite(object),2,sum)`

Author(s)

Werner A. Stahel, ETH Zurich

See Also

is.na, is.finite, dropNA

Examples

```
t.d <- data.frame(V1=c(1,2,NA,4), V2=c(11,12,13,Inf), V3=c(21,NA,23,Inf))
sumNA(t.d)
```

Tobit *Prepare a Response for a Tobit Model*

Description

Returns a Surv object that allows for setting up a Tobit regression model by calling survreg

Usage

```
Tobit(data, limit = 0, limhigh = NULL, transform = NULL, log = FALSE, ...)
```

Arguments

data	the variable to be used as the response in the Tobit regression
limit	Lower limit which censors the observations. If log is TRUE, then the default is the minimal value of logst(data), and if limit>0, it refers to the untransformed data.
limhigh	Upper limit which censors the observations (for untransformed data).
transform	if data should be transformed, specify the function to be used.
log	logical. If TRUE, data will be log transformed by calling logst. Argument transform will be ignored.
...	any additional arguments to the transform function

Details

Tobit regression is a special case of regression with left censored response data. The function survreg is suitable for fitting. In regr, this is done automatically.

Value

A Surv object.

Author(s)

Werner A. Stahel

See Also

Surv() from package **survival**.

Examples

```
if(requireNamespace("survival")) {
data("tobin", package="survival")

Tobit(tobin$durable)
(t.r <- survival::survreg(Tobit(durable) ~ age + quant,
                          data = tobin, dist="gaussian"))
if(interactive())
    plregr(t.r)
}
```

`transferAttributes` *Transfer Attributes*

Description

Attach the attributes of an object to another object

Usage

```
transferAttributes(x, xbefore, except = NULL)
```

Arguments

`x`	the object to which the attributes should be transferred
`xbefore`	the object which delvers the attributes
`except`	names of attributes that will not be transferred

Value

Object `x` with attributes from `xbefore` (and possibly some that it already had)

Note

This function would not be needed if `structure` allowed for a list of attributes.

Author(s)

W. A. Stahel

See Also

`structure`

Examples

```
a <- structure(1:10, title="sequence")
transferAttributes(31:40, a)
```

warn *List Warnings*

Description

Gives a List of Warnings

Usage

```
warn()
```

Details

This function simplyfies the output of `warnings` if there are several identical warnings, by counting their occurence

Value

the table of warnings

Author(s)

Werner A. Stahel, ETH Zurich

See Also

`warnings`

Examples

```
for (i in 3:6) m <- matrix(1:7, 3, i)

suppressWarnings(  ## or set  options(warn=-1)
for (i in 3:6) m <- matrix(1:7, 3, i))
warn()
```

weekday *Get Day of Week or Year, Month, Day*

Description

From Dates, obtain the day of the week or the year, month and day

Usage

```
weekday(date, month = NULL, day = NULL, out = NULL, factor = FALSE)
ymd(date)
```

Arguments

date	date(s), given as a Date object, a character object that can be converted into a Date, or as julian, or the year, in which case month and day must be given.
month, day	If the first argument is the year, these arguments must also be given.
factor	logical: Should the result be a (ordered) factor?
out	selection of output: either "numeric" for numeric output (0 for Sunday, ... 6 for Saturday), "full" or "long" for full weekday names, or "short" for abbreviated (3 character) names. Defaults to "full" if factor is TRUE, to "numeric" otherwise.

Value

For weekdays, the output is as described above, depending on factor and out.

Note

The functions call functions from the chron package

Author(s)

Werner A. Stahel

See Also

day.of.week, month.day.year

Examples

```
weekday(c("2020-05-01", "2020-05-02"), factor=TRUE)
## [1] Thursday Sunday
## Levels: Sunday < Monday < Tuesday < Wednesday < Thursday < Friday < Saturday

dt <- ymd(18100+1:5)
weekday(dt)
## [1] 3 4 5 6 0
```

---

xdistResdiff *Residual Differences for Near Replicates: Tabulate and Test*

Description

A test for the completeness of a linear regression model can be performed based on comparing the differences of residuals for pairs of observations that are close to each other to the estimated standard deviation of the model.

Usage

```
xdistResdiff(object, perc = c(3, 10, 80), trim = 0.1,
  nmax = 100, out = "aggregate")
xdistResscale(x, perc = c(3, 10, 90), trim = 1/6)
```

Arguments

object	an object containing the result of fitting a linear model by regr
x	an object produced by xdistResdiff
perc	Percentage points to define distance classes
trim	Trimming proportion for calculating means of absolute residual differences
nmax	maximal number of observations to form pairs
out	determines the value of xdistResdiff: if =="aggregate" (the default), the value will be produced by calling xdistResscale, otherwise, all x distances and respective residual differences will be returned.

Details

See package vignette.

Value

For xdistResdiff with out="aggregate" and xdistResscale, a matrix is returned with a row for each class of x distances and the columns

xdist	mean x distance
rdiff.mean	absolute differences of residuals for pairs of observations in the distance class, averaged over the class
rdiff.simmean	mean of (trimmed) means for simulated data
rdiff.se	standard error of (trimmed) means as obtained from simulation

The matrix carries along the following attributes:

perc	given argument perc
xd.classlim	the actual class limits corresponding to perc
trim	given argument trim
rdiff.grandmean	overall mean of absolute residual differences
p-values	p values for the classes as obtained from simulation, and p-value for the sum of squares statistic
class	The value has S3 class xdistResscale and matrix

.

If xdistResdiff with out different from "aggregate", then a data.frame is returned containing a row for each pair of observations and the columns

id1, id2	the labels of the two observations
xdist	the x distance between the two observations
resdiff	the difference of residuals for the two observations

The value has S3 class xdistResdiff and data.frame.

Author(s)

Werner A. Stahel, ETH Zurich

References

See package vignette.

See Also

plot.xdistResscale

Examples

```
data(d.blast)
rr <- lm(tremor~distance+charge, data=d.blast)
## an inadequate model!
xdrs <- xdistResdiff(rr)
xdrd <- xdistResdiff(rr, out="all")
showd(xdrd)
xdrs <- xdistResscale(xdrd)
## same as first call of  xdiffResdiff
plot(xdrs)
```

Index