Package: car (via r-universe)

October 19, 2026

Version 3.1-5

Date 2026-01-05

Title Companion to Applied Regression

Depends R (>= 3.5.0), carData (>= 3.0-0)

Imports abind, Formula, MASS, mgcv, nnet, pbkrtest (>= 0.4-4), quantreg, grDevices, utils, stats, graphics, lme4 (>= 1.1-27.1), nlme, scales

Suggests alr4, boot, coxme, effects, knitr, leaps, lmtest, Matrix, MatrixModels, ordinal, plotrix, mvtnorm, rgl (>= 0.111.3), rio, sandwich, SparseM, survival, survey

ByteCompile yes

LazyLoad yes

Description Functions to Accompany J. Fox and S. Weisberg, An R Companion to Applied Regression, Third Edition, Sage, 2019.

License GPL (>= 2)

URL https://github.com/bprice2652/car_repo, https://CRAN.R-project.org/package=car, https://z.umn.edu/carbook

VignetteBuilder knitr

NeedsCompilation no

Author John Fox [aut], Sanford Weisberg [aut], Brad Price [aut, cre], Daniel Adler [ctb], Douglas Bates [ctb], Gabriel Baud-Bovy [ctb], Ben Bolker [ctb], Steve Ellison [ctb], David Firth [ctb], Michael Friendly [ctb], Gregor Gorjanc [ctb], Spencer Graves [ctb], Richard Heiberger [ctb], Pavel Krivitsky [ctb], Rafael Laboissiere [ctb], Martin Maechler [ctb], Georges Monette [ctb], Duncan Murdoch [ctb], Henric Nilsson [ctb], Derek Ogle [ctb], Iain Proctor [ctb], Brian Ripley [ctb], Tom Short [ctb], William Venables [ctb], Steve Walker [ctb], David Winsemius [ctb], Achim Zeileis [ctb], R-Core [ctb]

Maintainer Brad Price <brad.price@mail.wvu.edu>

Config/pak/sysreqs cmake make libicu-dev

Repository https://bprice2652.r-universe.dev

Date/Publication 2026-02-03 10:10:16 UTC

RemoteUrl https://github.com/cran/car

RemoteRef HEAD

RemoteSha 6dd8b5e1e4bffb797d325df38d3f557cf0899a2f

Contents

Anova . . . 3
avPlots . . . 11
bcPower . . . 14
Boot . . . 17
boxCox . . . 19
boxCoxVariable . . . 22
Boxplot . . . 23
boxTidwell . . . 25
brief . . . 26
car-defunct . . . 29
car-deprecated . . . 31
car-internal.Rd . . . 31
carHexsticker . . . 32
carPalette . . . 32
carWeb . . . 33
ceresPlots . . . 34
compareCoefs . . . 36
Contrasts . . . 37
crPlots . . . 39
deltaMethod . . . 43
densityPlot . . . 47
dfbetaPlots . . . 50
durbinWatsonTest . . . 52
Ellipses . . . 53
Export . . . 58
hccm . . . 59
hist.boot . . . 61
Import . . . 64
infIndexPlot . . . 66
influence.mixed.models . . . 67
influencePlot . . . 70
invResPlot . . . 71
invTranPlot . . . 73
leveneTest . . . 75
leveragePlots . . . 76
linearHypothesis . . . 78
logit . . . 85

mcPlots 86
mmps 89
ncvTest 92
outlierTest 94
panel.car 95
pointLabel 96
poTest 98
powerTransform 99
Predict 103
qqPlot 104
recode 107
regLine 110
residualPlots 111
S 114
scatter3d 119
scatterplot 124
scatterplotMatrix 128
ScatterplotSmoothers 132
showLabels 135
sigmaHat 137
some 138
spreadLevelPlot 139
strings2factors 141
subsets 143
symbox 144
Tapply 146
testTransform 147
TransformationAxes 149
vif 151
wcrossprod 153
whichNames 154

Index **156**

`Anova` *Anova Tables for Various Statistical Models*

Description

Calculates type-II or type-III analysis-of-variance tables for model objects produced by `lm`, `glm`, `multinom` (in the **nnet** package), `polr` (in the **MASS** package), `coxph` (in the **survival** package), `coxme` (in the **coxme** pckage), `svyglm` and `svycoxph` (in the **survey** package), `rlm` (in the **MASS** package), `lmer` (in the **lme4** package), `lme` (in the **nlme** package), `clm` and `clmm` (in the **ordinal** package), and (by the default method) for most models with a linear predictor and asymptotically normal coefficients (see details below). For linear models, F-tests are calculated; for generalized linear models, likelihood-ratio chisquare, Wald chisquare, or F-tests are calculated; for multinomial logit and proportional-odds logit models, likelihood-ratio tests are calculated. Various test statistics are provided for multivariate linear models produced by `lm` or `manova`. Partial-likelihood-ratio tests

or Wald tests are provided for Cox models. Wald chi-square tests are provided for fixed effects in linear and generalized linear mixed-effects models. Wald chi-square or F tests are provided in the default case.

Usage

```
Anova(mod, ...)

Manova(mod, ...)

## S3 method for class 'lm'
Anova(mod, error, type=c("II","III", 2, 3),
 white.adjust=c(FALSE, TRUE, "hc3", "hc0", "hc1", "hc2", "hc4"),
 vcov.=NULL, singular.ok, ...)

## S3 method for class 'aov'
Anova(mod, ...)

## S3 method for class 'glm'
Anova(mod, type=c("II","III", 2, 3),
    test.statistic=c("LR", "Wald", "F"),
    error, error.estimate=c("pearson", "dispersion", "deviance"),
   vcov.=vcov(mod, complete=TRUE),  singular.ok, ...)

## S3 method for class 'multinom'
Anova(mod, type = c("II","III", 2, 3), ...)

## S3 method for class 'polr'
Anova(mod, type = c("II","III", 2, 3), ...)

## S3 method for class 'mlm'
Anova(mod, type=c("II","III", 2, 3), SSPE, error.df,
    idata, idesign, icontrasts=c("contr.sum", "contr.poly"), imatrix,
    test.statistic=c("Pillai", "Wilks", "Hotelling-Lawley", "Roy"),...)

## S3 method for class 'manova'
Anova(mod, ...)

## S3 method for class 'mlm'
Manova(mod, ...)

## S3 method for class 'Anova.mlm'
print(x, ...)

## S3 method for class 'Anova.mlm'
summary(object, test.statistic, univariate=object$repeated,
    multivariate=TRUE, p.adjust.method, ...)

## S3 method for class 'summary.Anova.mlm'
```

```
print(x, digits = getOption("digits"),
    SSP=TRUE, SSPE=SSP, ... )

## S3 method for class 'univaov'
print(x, digits = max(getOption("digits") - 2L, 3L),
                          style=c("wide", "long"),
                          by=c("response", "term"),
                          ...)

## S3 method for class 'univaov'
as.data.frame(x, row.names, optional, by=c("response", "term"), ...)

## S3 method for class 'coxph'
Anova(mod, type=c("II", "III", 2, 3),
 test.statistic=c("LR", "Wald"), ...)

## S3 method for class 'coxme'
Anova(mod, type=c("II", "III", 2, 3),
    test.statistic=c("Wald", "LR"), ...)

## S3 method for class 'lme'
Anova(mod, type=c("II","III", 2, 3),
  vcov.=vcov(mod, complete=FALSE), singular.ok, ...)

## S3 method for class 'mer'
Anova(mod, type=c("II", "III", 2, 3),
 test.statistic=c("Chisq", "F"), vcov.=vcov(mod, complete=FALSE), singular.ok, ...)

## S3 method for class 'merMod'
Anova(mod, type=c("II", "III", 2, 3),
   test.statistic=c("Chisq", "F"), vcov.=vcov(mod, complete=FALSE), singular.ok, ...)

## S3 method for class 'svyglm'
Anova(mod, ...)

## S3 method for class 'svycoxph'
Anova(mod, type=c("II", "III", 2, 3),
  test.statistic="Wald", ...)

## S3 method for class 'rlm'
Anova(mod, ...)

## S3 method for class 'clm'
Anova(mod, ...)

## S3 method for class 'clmm'
Anova(mod, ...)
```

```
## Default S3 method:
Anova(mod, type=c("II", "III", 2, 3),
 test.statistic=c("Chisq", "F"), vcov.=vcov(mod, complete=FALSE),
 singular.ok, error.df, ...)
```

Arguments

mod	lm, aov, glm, multinom, polr mlm, coxph, coxme, lme, mer, merMod, svyglm, svycoxph, rlm, clm, clmm, or other suitable model object.
error	for a linear model, an lm model object from which the error sum of squares and degrees of freedom are to be calculated. For F-tests for a generalized linear model, a glm object from which the dispersion is to be estimated. If not specified, mod is used.
type	type of test, "II", "III", 2, or 3. Roman numerals are equivalent to the corresponding Arabic numerals.
singular.ok	defaults to TRUE for type-II tests, and FALSE for type-III tests where the tests for models with aliased coefficients will not be straightforwardly interpretable; if FALSE, a model with aliased coefficients produces an error. This argument is available only for some Anova methods.
test.statistic	for a generalized linear model, whether to calculate "LR" (likelihood-ratio), "Wald", or "F" tests; for a Cox or Cox mixed-effects model, whether to calculate "LR" (partial-likelihood ratio) or "Wald" tests (with "LR" tests unavailable for Cox models using the tt argument); in the default case or for linear mixed models fit by lmer, whether to calculate Wald "Chisq" or Kenward-Roger "F" tests with Satterthwaite degrees of freedom (*warning:* the KR F-tests can be very time-consuming). For a multivariate linear model, the multivariate test statistic to compute — one of "Pillai", "Wilks", "Hotelling-Lawley", or "Roy", with "Pillai" as the default. The summary method for Anova.mlm objects permits the specification of more than one multivariate test statistic, and the default is to report all four.
error.estimate	for F-tests for a generalized linear model, base the dispersion estimate on the Pearson residuals ("pearson", the default); use the dispersion estimate in the model object ("dispersion"); or base the dispersion estimate on the residual deviance ("deviance"). For binomial or Poisson GLMs, where the dispersion is fixed to 1, setting error.estimate="dispersion" is changed to "pearson", with a warning.
white.adjust	if not FALSE, the default, tests use a heteroscedasticity-corrected coefficient covariance matrix; the various values of the argument specify different corrections. See the documentation for hccm for details. If white.adjust=TRUE then the "hc3" correction is selected.
SSPE	For Anova for a multivariate linear model, the error sum-of-squares-and-products matrix; if missing, will be computed from the residuals of the model; for the print method for the summary of an Anova of a multivariate linear model, whether or not to print the error SSP matrix (defaults to TRUE).
SSP	if TRUE (the default), print the sum-of-squares and cross-products matrix for the hypothesis and the response-transformation matrix.

`error.df` The degrees of freedom for error; if `error.df` missing for a multivariate linear model (object of class `"mlm"`), the error degrees of freedom will be taken from the model.

For the `default` `Anova` method, if an F-test is requested and if `error.df` is missing, the error degrees of freedom will be computed by applying the `df.residual` function to the model; if `df.residual` returns `NULL` or `NA`, then a chi-square test will be substituted for the F-test (with a message to that effect.

`idata` an optional data frame giving a factor or factors defining the intra-subject model for multivariate repeated-measures data. See *Details* for an explanation of the intra-subject design and for further explanation of the other arguments relating to intra-subject factors.

`idesign` a one-sided model formula using the "data" in `idata` and specifying the intra-subject design.

`icontrasts` names of contrast-generating functions to be applied by default to factors and ordered factors, respectively, in the within-subject "data"; the contrasts must produce an intra-subject model matrix in which different terms are orthogonal. The default is `c("contr.sum", "contr.poly")`.

`imatrix` as an alternative to specifying `idata`, `idesign`, and (optionally) `icontrasts`, the model matrix for the within-subject design can be given directly in the form of list of named elements. Each element gives the columns of the within-subject model matrix for a term to be tested, and must have as many rows as there are responses; the columns of the within-subject model matrix for different terms must be mutually orthogonal.

`x, object` object of class `"Anova.mlm"` to print or summarize.

`multivariate, univariate`
compute and print multivariate and univariate tests for a repeated-measures ANOVA or multivariate linear model; the default is `TRUE` for both for repeated measures and `TRUE` for `multivariate` for a multivariate linear model.

`p.adjust.method`
if given for a multivariate linear model when univariate tests are requested, the univariate tests are corrected for simultaneous inference by term; if specified, should be one of the methods recognized by `p.adjust` or `TRUE`, in which case the default (Holm) adjustment is used.

`digits` minimum number of significant digits to print.

`style` for printing univariate tests if requested for a multivariate linear model; one of `"wide"`, the default, or `"long"`.

`by` if univariate tests are printed in `"long"` `style`, they can be ordered by `"response"`, the default, or by `"term"`.

`row.names, optional`
not used.

`vcov.` in the `default` method, an optional coefficient-covariance matrix or function to compute a covariance matrix, computed by default by applying the generic `vcov` function to the model object. A similar argument may be supplied to the `lm` method, and the default (`NULL`) is to ignore the argument; if both `vcov.` and `white.adjust` are supplied to the `lm` method, the latter is used. In the `glm`

method, vcov. is ignored unless test="Wald"; in the mer and merMod methods, vcov. is ignored if test="F".

Note that arguments supplied to ... are *not* passed to vcov. when it's a function; in this case either use an anonymous function in which the additional arguments are set, or supply the coefficient covariance matrix directly (see the examples).

... do not use.

Details

The designations "type-II" and "type-III" are borrowed from SAS, but the definitions used here do not correspond precisely to those employed by SAS. Type-II tests are calculated according to the principle of marginality, testing each term after all others, except ignoring the term's higher-order relatives; so-called type-III tests violate marginality, testing each term in the model after all of the others. This definition of Type-II tests corresponds to the tests produced by SAS for analysis-of-variance models, where all of the predictors are factors, but not more generally (i.e., when there are quantitative predictors). Be very careful in formulating the model for type-III tests, or the hypotheses tested will not make sense.

As implemented here, type-II Wald tests are a generalization of the linear hypotheses used to generate these tests in linear models.

For tests for linear models, multivariate linear models, and Wald tests for generalized linear models, Cox models, mixed-effects models, generalized linear models fit to survey data, and in the default case, Anova finds the test statistics without refitting the model. The svyglm method simply calls the default method and therefore can take the same arguments.

The standard R anova function calculates sequential ("type-I") tests. These rarely test interesting hypotheses in unbalanced designs.

A MANOVA for a multivariate linear model (i.e., an object of class "mlm" or "manova") can optionally include an intra-subject repeated-measures design. If the intra-subject design is absent (the default), the multivariate tests concern all of the response variables. To specify a repeated-measures design, a data frame is provided defining the repeated-measures factor or factors via idata, with default contrasts given by the icontrasts argument. An intra-subject model-matrix is generated from the formula specified by the idesign argument; columns of the model matrix corresponding to different terms in the intra-subject model must be orthogonal (as is insured by the default contrasts). Note that the contrasts given in icontrasts can be overridden by assigning specific contrasts to the factors in idata. As an alternative, the within-subjects model matrix can be specified directly via the imatrix argument. Manova is essentially a synonym for Anova for multivariate linear models.

If univariate tests are requested for the summary of a multivariate linear model, the object returned contains a univaov component of "univaov"; print and as.data.frame methods are provided for the "univaov" class.

For the default method to work, the model object must contain a standard terms element, and must respond to the vcov, coef, and model.matrix functions. If any of these requirements is missing, then it may be possible to supply it reasonably simply (e.g., by writing a missing vcov method for the class of the model object).

Value

An object of class "anova", or "Anova.mlm", which usually is printed. For objects of class "Anova.mlm", there is also a summary method, which provides much more detail than the print method about the MANOVA, including traditional mixed-model univariate F-tests with Greenhouse-Geisser and Huynh-Feldt corrections.

Warning

Be careful of type-III tests: For a traditional multifactor ANOVA model with interactions, for example, these tests will normally only be sensible when using contrasts that, for different terms, are orthogonal in the row-basis of the model, such as those produced by contr.sum, contr.poly, or contr.helmert, but *not* by the default contr.treatment. In a model that contains factors, numeric covariates, and interactions, main-effect tests for factors will be for differences over the origin. In contrast (pun intended), type-II tests are invariant with respect to (full-rank) contrast coding. If you don't understand this issue, then you probably shouldn't use Anova for type-III tests.

Author(s)

John Fox <jfox@mcmaster.ca>; the code for the Mauchly test and Greenhouse-Geisser and Huynh-Feldt corrections for non-sphericity in repeated-measures ANOVA are adapted from the functions stats:::stats:::mauchly.test.SSD and stats:::sphericity by R Core; summary.Anova.mlm and print.summary.Anova.mlm incorporates code contributed by Gabriel Baud-Bovy.

References

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Hand, D. J., and Taylor, C. C. (1987) *Multivariate Analysis of Variance and Repeated Measures: A Practical Approach for Behavioural Scientists.* Chapman and Hall.

O'Brien, R. G., and Kaiser, M. K. (1985) MANOVA method for analyzing repeated measures designs: An extensive primer. *Psychological Bulletin* **97**, 316–333.

See Also

linearHypothesis, anova anova.lm, anova.glm, anova.mlm, anova.coxph, svyglm.

Examples

```
## Two-Way Anova

mod <- lm(conformity ~ fcategory*partner.status, data=Moore,
  contrasts=list(fcategory=contr.sum, partner.status=contr.sum))
Anova(mod)
Anova(mod, type=3)  # note use of contr.sum in call to lm()

## use of vcov.; the following are equivalent:

Anova(mod, white.adjust = TRUE)
Anova(mod, vcov. = hccm) # vcov. is a function, type = "hc3" is the default
```

```
Anova(mod, vcov. = hccm(mod, type = "hc3")) # vcov. is a matrix
Anova(mod, vcov. = function(m) hccm(m, type = "hc3")) # passing type as an argument

## One-Way MANOVA
## See ?Pottery for a description of the data set used in this example.

summary(Anova(lm(cbind(Al, Fe, Mg, Ca, Na) ~ Site, data=Pottery)))

## MANOVA for a randomized block design (example courtesy of Michael Friendly:
##  See ?Soils for description of the data set)

soils.mod <- lm(cbind(pH,N,Dens,P,Ca,Mg,K,Na,Conduc) ~ Block + Contour*Depth,
    data=Soils)
Manova(soils.mod)
summary(Anova(soils.mod), univariate=TRUE, multivariate=FALSE,
    p.adjust.method=TRUE)

## a multivariate linear model for repeated-measures data
## See ?OBrienKaiser for a description of the data set used in this example.

phase <- factor(rep(c("pretest", "posttest", "followup"), c(5, 5, 5)),
    levels=c("pretest", "posttest", "followup"))
hour <- ordered(rep(1:5, 3))
idata <- data.frame(phase, hour)
idata

mod.ok <- lm(cbind(pre.1, pre.2, pre.3, pre.4, pre.5,
                     post.1, post.2, post.3, post.4, post.5,
                     fup.1, fup.2, fup.3, fup.4, fup.5) ~  treatment*gender,
                data=OBrienKaiser)
(av.ok <- Anova(mod.ok, idata=idata, idesign=~phase*hour))

summary(av.ok, multivariate=FALSE)

## A "doubly multivariate" design with two  distinct repeated-measures variables
## (example courtesy of Michael Friendly)
## See ?WeightLoss for a description of the dataset.

imatrix <- matrix(c(
 1,0,-1, 1, 0, 0,
 1,0, 0,-2, 0, 0,
 1,0, 1, 1, 0, 0,
 0,1, 0, 0,-1, 1,
 0,1, 0, 0, 0,-2,
 0,1, 0, 0, 1, 1), 6, 6, byrow=TRUE)
colnames(imatrix) <- c("WL", "SE", "WL.L", "WL.Q", "SE.L", "SE.Q")
rownames(imatrix) <- colnames(WeightLoss)[-1]
(imatrix <- list(measure=imatrix[,1:2], month=imatrix[,3:6]))
contrasts(WeightLoss$group) <- matrix(c(-2,1,1, 0,-1,1), ncol=2)
(wl.mod<-lm(cbind(wl1, wl2, wl3, se1, se2, se3)~group, data=WeightLoss))
Anova(wl.mod, imatrix=imatrix, test="Roy")

## mixed-effects models examples:
```

```
## Not run:  # loads nlme package
 library(nlme)
 example(lme)
 Anova(fm2)

## End(Not run)

## Not run:  # loads lme4 package
 library(lme4)
 example(glmer)
 Anova(gm1)

## End(Not run)
```

---

avPlots *Added-Variable Plots*

Description

These functions construct added-variable, also called partial-regression, plots for linear and generalized linear models.

Usage

```
avPlots(model, ...)

## Default S3 method:
avPlots(model, terms=~., intercept=FALSE,
  layout=NULL, ask, main, ...)

avp(...)

avPlot(model, ...)

## S3 method for class 'lm'
avPlot(model, variable,
 id=TRUE, col = carPalette()[1], col.lines = carPalette()[2],
 xlab, ylab, pch = 1, lwd = 2, cex = par("cex"), pt.wts = FALSE,
 main=paste("Added-Variable Plot:", variable),
 grid=TRUE,
 ellipse=FALSE,
  marginal.scale=FALSE, ...)

## S3 method for class 'glm'
avPlot(model, variable,
 id=TRUE,
```

```
 col = carPalette()[1], col.lines = carPalette()[2],
 xlab, ylab, pch = 1, lwd = 2, cex = par("cex"), pt.wts = FALSE,
 type=c("Wang", "Weisberg"),
 main=paste("Added-Variable Plot:", variable), grid=TRUE,
 ellipse=FALSE, ...)

avPlot3d(model, coef1, coef2, id=TRUE, ...)

## S3 method for class 'lm'
avPlot3d(model, coef1, coef2, id=TRUE, fit="linear", ...)

## S3 method for class 'glm'
avPlot3d(model, coef1, coef2, id=TRUE, type=c("Wang", "Weisberg"),
  fit="linear", ...)
```

Arguments

model	model object produced by lm or glm.
terms	A one-sided formula that specifies a subset of the predictors. One added-variable plot is drawn for each term. For example, the specification terms = ~.-X3 would plot against all terms except for X3. If this argument is a quoted name of one of the terms, the added-variable plot is drawn for that term only.
coef1, coef2	the quoted names of the two coefficients for a 3D added variable plot.
intercept	Include the intercept in the plots; default is FALSE.
variable	A quoted string giving the name of a regressor in the model matrix for the horizontal axis.
layout	If set to a value like c(1, 1) or c(4, 3), the layout of the graph will have this many rows and columns. If not set, the program will select an appropriate layout. If the number of graphs exceed nine, you must select the layout yourself, or you will get a maximum of nine per page. If layout=NA, the function does not set the layout and the user can use the par function to control the layout, for example to have plots from two models in the same graphics window.
main	The title of the plot; if missing, one will be supplied.
ask	If TRUE, ask the user before drawing the next plot; if FALSE don't ask.
id	controls point identification; if FALSE, no points are identified; can be a list of named arguments to the showLabels function; TRUE, the default, is equivalent to list(method=list(abs(residuals(model, type="pearson")), "x"), n=2, cex=1, col=carPalette()[1], location="lr"), which identifies the 2 points with the largest residuals and the 2 points with the most extreme horizontal values (i.e., largest partial leverage). For avPlot3d, point identication is through Identify3d.
col	color for points; the default is the *second* entry in the current **car** palette (see carPalette and par).
col.lines	color for the fitted line.
pch	plotting character for points; default is 1 (a circle, see par).

lwd	line width; default is 2 (see par).
cex	size of plotted points; default is taken from par("cex").
pt.wts	if TRUE (the default is FALSE), for a weighted least squares fit or a generalized linear model, the areas of plotted points are made proportional to the weights, with the average size taken from the cex argument.
xlab	x-axis label. If omitted a label will be constructed.
ylab	y-axis label. If omitted a label will be constructed.
type	if "Wang" use the method of Wang (1985); if "Weisberg" use the method in the Arc software associated with Cook and Weisberg (1999).
grid	If TRUE, the default, a light-gray background grid is put on the graph.
ellipse	controls plotting data-concentration ellipses. If FALSE (the default), no ellipses are plotted. Can be a list of named values giving levels, a vector of one or more bivariate-normal probability-contour levels at which to plot the ellipses; and robust, a logical value determing whether to use the cov.trob function in the **MASS** package to calculate the center and covariance matrix for the data ellipses. TRUE is equivalent to list(levels=c(.5, .95), robust=TRUE).
marginal.scale	Consider an added-variable plot of Y versus X given Z. If this argument is FALSE then the limits on the horizontal axis are determined by the range of the residuals from the regression of X on Z and the limits on the vertical axis are determined by the range of the residuals from the regressnio of Y on Z. If the argument is TRUE, then the limits on the horizontal axis are determined by the range of X minus it mean, and on the vertical axis by the range of Y minus its means; adjustment is made if necessary to include outliers. This scaling allows visualization of the correlations between Y and Z and between X and Z. For example, if the X and Z are highly correlated, then the points will be concentrated on the middle of the plot.
fit	one or both of "linear" (linear least-squares, the default) and "robust" (robust regression) surfaces to be fit to the 3D added-variable plot; see scatter3d for details.
...	avPlots passes these arguments to avPlot and avPlot passes them to plot; for avPlot3d, additional optional arguments to be passed to scatter3d.

Details

The functions intended for direct use are avPlots (for which avp is an abbreviation) and avPlot3d.

Value

These functions are used for their side effect of producing plots, but also invisibly return the coordinates of the plotted points.

Author(s)

John Fox <jfox@mcmaster.ca>, Sanford Weisberg <sandy@umn.edu>

References

Cook, R. D. and Weisberg, S. (1999) *Applied Regression, Including Computing and Graphics.* Wiley.

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Wang, P C. (1985) Adding a variable in generalized linear models. *Technometrics* **27**, 273–276.

Weisberg, S. (2014) *Applied Linear Regression*, Fourth Edition, Wiley.

See Also

residualPlots, crPlots, ceresPlots, link{dataEllipse}, showLabels, dataEllipse.

Examples

```
avPlots(lm(prestige ~ income + education + type, data=Duncan))

avPlots(glm(partic != "not.work" ~ hincome + children,
  data=Womenlf, family=binomial), id=FALSE, pt.wts=TRUE)

m1 <- lm(partic ~ tfr + menwage + womwage + debt + parttime, Bfox)
par(mfrow=c(1,3))
# marginal plot, ignoring other predictors:
with(Bfox, dataEllipse(womwage, partic, levels=0.5))
abline(lm(partic ~ womwage, Bfox), col="red", lwd=2)
# AV plot, adjusting for others:
avPlots(m1, ~ womwage, ellipse=list(levels=0.5))
# AV plot, adjusting and scaling as in marginal plot
avPlots(m1, ~ womwage, marginal.scale=TRUE, ellipse=list(levels=0.5))

# 3D AV plot, requires the rgl package
if (interactive() && require("rgl")){
avPlot3d(lm(prestige ~ income + education + type, data=Duncan),
  "income", "education")
}
```

---

bcPower — *Box-Cox, Box-Cox with Negatives Allowed, Yeo-Johnson and Basic Power Transformations*

Description

Transform the elements of a vector or columns of a matrix using, the Box-Cox, Box-Cox with negatives allowed, Yeo-Johnson, or simple power transformations.

Usage

```
bcPower(U, lambda, jacobian.adjusted=FALSE, gamma=NULL)

bcnPower(U, lambda, jacobian.adjusted = FALSE, gamma)

bcnPowerInverse(z, lambda, gamma)

yjPower(U, lambda, jacobian.adjusted = FALSE)

basicPower(U,lambda, gamma=NULL)
```

Arguments

U	A vector, matrix or data.frame of values to be transformed
lambda	Power transformation parameter with one element for each column of U, usually in the range from -2 to 2.
jacobian.adjusted	If TRUE, the transformation is normalized to have Jacobian equal to one. The default FALSE is almost always appropriate.
gamma	For bcPower or basicPower, the transformation is of U + gamma, where gamma is a positive number called a start that must be large enough so that U + gamma is strictly positive. For the bcnPower, Box-cox power with negatives allowed, see the details below.
z	a numeric vector the result of a call to bcnPower with jacobian.adjusted=FALSE

.

Details

The Box-Cox family of *scaled power transformations* equals $(x^\lambda - 1)/\lambda$ for $\lambda \neq 0$, and $\log(x)$ if $\lambda = 0$. The bcPower function computes the scaled power transformation of $x = U + \gamma$, where γ is set by the user so $U + \gamma$ is strictly positive for these transformations to make sense.

The Box-Cox family with negatives allowed was proposed by Hawkins and Weisberg (2017). It is the Box-Cox power transformation of

$$z = .5(U + \sqrt{U^2 + \gamma^2})$$

where for this family γ is either user selected or is estimated. gamma must be positive if U includes negative values and non-negative otherwise, ensuring that z is always positive. The bcnPower transformations behave similarly to the bcPower transformations, and introduce less bias than is introduced by setting the parameter γ to be non-zero in the Box-Cox family.

The function bcnPowerInverse computes the inverse of the bcnPower function, so U = bcnPowerInverse(bcnPower(U, lambda=lam, jacobian.adjusted=FALSE, gamma=gam), lambda=lam, gamma=gam) is true for any permitted value of gam and lam.

If family="yeo.johnson" then the Yeo-Johnson transformations are used. This is the Box-Cox transformation of $U+1$ for nonnegative values, and of $|U|+1$ with parameter $2-\lambda$ for U negative.

The basic power transformation returns U^λ if λ is not 0, and $\log(\lambda)$ otherwise for U strictly positive.

If jacobian.adjusted is TRUE, then the scaled transformations are divided by the Jacobian, which is a function of the geometric mean of U for skewPower and yjPower and of $U + gamma$ for bcPower. With this adjustment, the Jacobian of the transformation is always equal to 1. Jacobian adjustment facilitates computing the Box-Cox estimates of the transformation parameters.

Missing values are permitted, and return NA where ever U is equal to NA.

Value

Returns a vector or matrix of transformed values.

Author(s)

Sanford Weisberg, <sandy@umn.edu>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Hawkins, D. and Weisberg, S. (2017) Combining the Box-Cox Power and Generalized Log Transformations to Accomodate Nonpositive Responses In Linear and Mixed-Effects Linear Models *South African Statistics Journal*, 51, 317-328.

Weisberg, S. (2014) *Applied Linear Regression*, Fourth Edition, Wiley Wiley, Chapter 7.

Yeo, In-Kwon and Johnson, Richard (2000) A new family of power transformations to improve normality or symmetry. *Biometrika*, 87, 954-959.

See Also

powerTransform, testTransform

Examples

```
U <- c(NA, (-3:3))
## Not run: bcPower(U, 0)  # produces an error as U has negative values
bcPower(U, 0, gamma=4)
bcPower(U, .5, jacobian.adjusted=TRUE, gamma=4)
bcnPower(U, 0, gamma=2)
basicPower(U, lambda = 0, gamma=4)
yjPower(U, 0)
V <- matrix(1:10, ncol=2)
bcPower(V, c(0, 2))
basicPower(V, c(0,1))
```

Boot *Bootstrapping for regression models*

Description

This function provides a simple front-end to the boot function in the **boot** package that is tailored to bootstrapping based on regression models. Whereas boot is very general and therefore has many arguments, the Boot function has very few arguments.

Usage

```
Boot(object, f=coef, labels=names(f(object)), R=999,
  method=c("case", "residual"), ncores=1, ...)

## Default S3 method:
Boot(object, f=coef, labels=names(f(object)),
  R=999, method=c("case", "residual"), ncores=1,
  start = FALSE, ...)

## S3 method for class 'lm'
Boot(object, f=coef, labels=names(f(object)),
  R=999, method=c("case", "residual"), ncores=1, ...)

## S3 method for class 'glm'
Boot(object, f=coef, labels=names(f(object)),
  R=999, method=c("case", "residual"), ncores=1, ...)

## S3 method for class 'nls'
Boot(object, f=coef, labels=names(f(object)),
  R=999, method=c("case", "residual"), ncores=1, ...)
```

Arguments

object	A regression object of class "lm", "glm" or "nls". The function may work with other regression objects that support the update method and have a subset argument. See discussion of the default method in the details below.
f	A function whose one argument is the name of a regression object that will be applied to the updated regression object to compute the statistics of interest. The default is coef, to return regression coefficient estimates. For example, f = function(obj) coef(obj)[1]/coef(obj)[2] will bootstrap the ratio of the first and second coefficient estimates.
labels	Provides labels for the statistics computed by f. Default labels are obtained from a call to f, or generic labels if f does not return names.
R	Number of bootstrap samples. The number of bootstrap samples actually computed may be smaller than this value if either the fitting method is iterative and

fails to converge for some boothstrap samples, or if the rank of a fitted model is different in a bootstrap replication than in the original data.

`method` The bootstrap method, either "case" for resampling cases or "residuals" for a residual bootstrap. See the details below. The residual bootstrap is available only for `lm` and `nls` objects and will return an error for `glm` objects.

`...` Arguments passed to the `boot` function, see `boot`.

`start` Should the estimates returned by `f` be passed as starting values for each bootstrap iteration? Alternatively, `start` can be a numeric vector of starting values. The default is to use the estimates from the last bootstrap iteration as starting values for the next iteration.

`ncores` A numeric argument that specifies the number of cores for parallel processing for unix systems. If less than or equal to 1, no parallel processing wiill be used. Note in a Windows platform will produce a warning and set this argument to 1.

Details

`Boot` uses a regression object and the choice of `method`, and creates a function that is passed as the `statistic` argument to the `boot` function in the **boot** package. The argument `R` is also passed to `boot`. If `ncores` is greater than 1, then the `parallel` and `ncpus` arguments to `boot` are set appropriately to use multiple codes, if available, on your computer. All other arguments to `boot` are kept at their default values unless you pass values for them.

The methods available for `lm` and `nls` objects are "case" and "residual". The case bootstrap resamples from the joint distribution of the terms in the model and the response. The residual bootstrap fixes the fitted values from the original data, and creates bootstraps by adding a bootstrap sample of the residuals to the fitted values to get a bootstrap response. It is an implementation of Algorithm 6.3, page 271, of Davison and Hinkley (1997). For `nls` objects ordinary residuals are used in the resampling rather than the standardized residuals used in the `lm` method. The residual bootstrap for generalized linear models has several competing approaches, but none are without problems. If you want to do a residual bootstrap for a glm, you will need to write your own call to `boot`.

For the default object to work with other types of regression models, the model must have methods for the the following generic functions: `residuals(object, type="pearson")` must return Pearson residuals; `fitted(object)` must return fitted values; `hatvalues(object)` should return the leverages, or perhaps the value 1 which will effectively ignore setting the hatvalues. In addition, the `data` argument should contain no missing values among the columns actually used in fitting the model, as the resampling may incorrectly attempt to include cases with missing values. For `lm`, `glm` and `nls`, missing values cause the return of an error message.

An attempt to fit using a bootstrap sample may fail. In a `lm` or `glm` fit, the bootstrap sample could have a different rank from the original fit. In an `nls` fit, convergence may not be obtained for some bootstraps. In either case, `NA` are returned for the value of the function `f`. The summary methods handle the `NA`s appropriately.

Fox and Weisberg (2017) cited below discusses this function and provides more examples.

Value

See `boot` for the returned value of the structure returned by this function.

Warning

C=A call like `car::Boot(object, method="residual")` will fail for the residual method if not preceded by `library(car)`. If `method="case"` the `library(car)` command is not required.

Author(s)

Sanford Weisberg, <sandy@umn.edu>. Achim Zeileis added multicore support, and also fixed the default method to work for many more regression models.

References

Davison, A, and Hinkley, D. (1997) *Bootstrap Methods and their Applications*. Oxford: Oxford University Press.

Fox, J. and Weisberg, S. (2019) *Companion to Applied Regression*, Third Edition. Thousand Oaks: Sage.

Fox, J. and Weisberg, S. (2019) *Bootstrapping Regression Models in R*, https://www.john-fox.ca/Companion/appendices/Appendix-Bootstrapping.pdf.

Weisberg, S. (2014) *Applied Linear Regression*, Fourth Edition, Wiley Wiley, Chapters 4 and 11.

See Also

Functions that work with `boot` objects from the **boot** package are `boot.array`, `boot.ci`, `plot.boot` and `empinf`. Additional functions in the **car** package are `summary.boot`, `confint.boot`, and `hist.boot`.

Examples

```
m1 <- lm(Fertility ~ ., swiss)
betahat.boot <- Boot(m1, R=199) # 199 bootstrap samples--too small to be useful
summary(betahat.boot)  # default summary
confint(betahat.boot)
hist(betahat.boot)
# Bootstrap for the estimated residual standard deviation:
sigmahat.boot <- Boot(m1, R=199, f=sigmaHat, labels="sigmaHat")
summary(sigmahat.boot)
confint(sigmahat.boot)
```

---

boxCox — *Graph the profile log-likelihood for Box-Cox transformations in 1D, or in 2D with the bcnPower family.*

Description

Computes and optionally plots profile log-likelihoods for the parameter of the Box-Cox power family, the Yeo-Johnson power family, or for either of the parameters in a bcnPower family. This is a slight generalization of the `boxcox` function in the **MASS** package that allows for families of transformations other than the Box-Cox power family. the `boxCox2d` function produces a contour plot of the two-dimensional likelihood profile for the bcnPower family.

Usage

```
boxCox(object, ...)

## Default S3 method:
boxCox(object,
        lambda = seq(-2, 2, 1/10), plotit = TRUE,
        interp = plotit, eps = 1/50,
        xlab=NULL, ylab=NULL, main= "Profile Log-likelihood",
        family="bcPower",
        param=c("lambda", "gamma"), gamma=NULL,
        grid=TRUE, ...)

## S3 method for class 'formula'
boxCox(object, lambda = seq(-2, 2, 1/10), plotit = TRUE, family = "bcPower",
    param = c("lambda", "gamma"), gamma = NULL, grid = TRUE,
    ...)

## S3 method for class 'lm'
boxCox(object, lambda = seq(-2, 2, 1/10), plotit = TRUE, ...)

boxCox2d(x, ksds = 4, levels = c(0.5, 0.95, 0.99, 0.999),
                main = "bcnPower Log-likelihood", grid=TRUE, ...)
```

Arguments

object	a formula or fitted model object of class lm or aov.
lambda	vector of values of λ, with default (-2, 2) in steps of 0.1, where the profile log-likelihood will be evaluated.
plotit	logical which controls whether the result should be plotted; default TRUE.
interp	logical which controls whether spline interpolation is used. Default to TRUE if plotting with lambda of length less than 100.
eps	Tolerance for lambda = 0; defaults to 0.02.
xlab	defaults to "lambda" or "gamma".
ylab	defaults to "log-Likelihood" or for bcnPower family to the appropriate label.
family	Defaults to "bcPower" for the Box-Cox power family of transformations. If set to "yjPower" the Yeo-Johnson family, which permits negative responses, is used. If set to bcnPower the function gives the profile log-likelihood for the parameter selected via param.
param	Relevant only to family="bcnPower", produces a profile log-likelihood for the parameter selected, maximizing over the remaining parameter.
gamma	For use when the family="bcnPower", param="gamma". If this is a vector of positive values, then the profile log-likelihood for the location (or start) parameter in the bcnPower family is evaluated at these values of gamma. If gamma is NULL, then evaulation is done at 100 equally spaced points between min(.01,

	gmax - 3*sd) and gmax + 3*sd, where gmax is the maximimum likelihood estimate of gamma, and sd is the sd of the response. See bcnPower for the definition of gamma.
grid	If TRUE, the default, a light-gray background grid is put on the graph.
...	additional arguments passed to plot, or to contour with boxCox2d.
x	An object created by a call to powerTransform using family="bcnPower".
ksds	Contour plotting of the log-likelihood surface will cover plus of minus ksds standard deviations on each axis.
levels	Contours will be drawn at the values of levels. For example, levels=c(.5, .99) would display two contours, at the 50% level and at the 99% level.
main	Title for the contour plot or the profile log-likelihood plot

Details

The boxCox function is an elaboration of the boxcox function in the **MASS** package. The first 7 arguments are the same as in boxcox, and if the argument family="bcPower" is used, the result is essentially identical to the function in **MASS**. Two additional families are the yjPower and bcnPower families that allow a few values of the response to be non-positive. The bcnPower family has two parameters: a power λ and a start or location parameter γ, and the boxCox function can be used to obtain a profile log-likelihood for either parameter with λ as the default. Alternatively, the boxCox2d function can be used to get a contour plot of the profile log-likelihood.

Value

Both functions ae designed for their side effects of drawing a graph. The boxCox function returns a list of the lambda (or possibly, gamma) vector and the computed profile log-likelihood vector, invisibly if the result is plotted. If plotit=TRUE plots log-likelihood vs lambda and indicates a 95% confidence interval about the maximum observed value of lambda. If interp=TRUE, spline interpolation is used to give a smoother plot.

Author(s)

Sanford Weisberg, <sandy@umn.edu>

References

Box, G. E. P. and Cox, D. R. (1964) An analysis of transformations. *Journal of the Royal Statisistical Society, Series B*. 26 211-46.

Cook, R. D. and Weisberg, S. (1999) *Applied Regression Including Computing and Graphics*. Wiley.

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Hawkins, D. and Weisberg, S. (2017) Combining the Box-Cox Power and Generalized Log Transformations to Accomodate Nonpositive Responses In Linear and Mixed-Effects Linear Models *South African Statistics Journal*, 51, 317-328.

Weisberg, S. (2014) *Applied Linear Regression*, Fourth Edition, Wiley.

Yeo, I. and Johnson, R. (2000) A new family of power transformations to improve normality or symmetry. *Biometrika*, 87, 954-959.

See Also

boxcox, yjPower, bcPower, bcnPower, powerTransform, contour

Examples

```
with(trees, boxCox(Volume ~ log(Height) + log(Girth), data = trees,
       lambda = seq(-0.25, 0.25, length = 10)))

data("quine", package = "MASS")
with(quine, boxCox(Days ~ Eth*Sex*Age*Lrn,
       lambda = seq(-0.05, 0.45, len = 20), family="yjPower"))
```

---

boxCoxVariable *Constructed Variable for Box-Cox Transformation*

Description

Computes a constructed variable for the Box-Cox transformation of the response variable in a linear model.

Usage

```
boxCoxVariable(y)
```

Arguments

y	response variable.

Details

The constructed variable is defined as $y[\log(y/\widetilde{y}) - 1]$, where $\widetilde{y}$ is the geometric mean of y.

The constructed variable is meant to be added to the right-hand-side of the linear model. The t-test for the coefficient of the constructed variable is an approximate score test for whether a transformation is required.

If b is the coefficient of the constructed variable, then an estimate of the normalizing power transformation based on the score statistic is $1 - b$. An added-variable plot for the constructed variable shows leverage and influence on the decision to transform y.

Value

a numeric vector of the same length as y.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Atkinson, A. C. (1985) *Plots, Transformations, and Regression*. Oxford.

Box, G. E. P. and Cox, D. R. (1964) An analysis of transformations. *JRSS B* **26** 211–246.

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

boxcox, powerTransform, bcPower

Examples

```
mod <- lm(interlocks + 1 ~ assets, data=Ornstein)
mod.aux <- update(mod, . ~ . + boxCoxVariable(interlocks + 1))
summary(mod.aux)
# avPlots(mod.aux, "boxCoxVariable(interlocks + 1)")
```

Boxplot *Boxplots With Point Identification*

Description

Boxplot is a wrapper for the standard R boxplot function, providing point identification, axis labels, and a formula interface for boxplots without a grouping variable.

Usage

```
Boxplot(y, ...)

## Default S3 method:
Boxplot(y, g, id=TRUE, xlab, ylab, ...)

## S3 method for class 'formula'
Boxplot(formula, data=NULL, subset, na.action=NULL,
    id=TRUE, xlab, ylab, ...)

## S3 method for class 'list'
Boxplot(y, xlab="", ylab="", ...)

## S3 method for class 'data.frame'
Boxplot(y, id=TRUE, ...)

## S3 method for class 'matrix'
Boxplot(y, ...)
```

Arguments

y	a numeric variable for which the boxplot is to be constructed; a list of numeric variables, each element of which will be treated as a group; a numeric data frame or a numeric matrix, each of whose columns will be treated as a group.
g	a grouping variable, usually a factor, for constructing parallel boxplots.
id	a list of named elements giving one or more specifications for labels of individual points ("outliers"): n, the maximum number of points to label (default 10); location, "lr" (left or right) of points or "avoid" to try to avoid overplotting; method, one of "y" (automatic, the default), "identify" (interactive), or "none"; col for labels (default is the first color in carPalette()); and cex size of labels (default is 1). Can be FALSE to suppress point identification or TRUE (the default) to use all defaults. This is similar to how showLabels handles point labels for other functions in the **car** package, except that the usual default is id=FALSE.
xlab, ylab	text labels for the horizontal and vertical axes; if missing, Boxplot will use the variable names, or, in the case of a list, data frame, or matrix, empty labels.
formula	a 'model' formula, of the form ~ y to produce a boxplot for the variable y, or of the form y ~ g, y ~ g1*g2*..., or y ~ g1 + g2 + ... to produce parallel boxplots for y within levels of the grouping variable(s) g, etc., usually factors.
data, subset, na.action	as for statistical modeling functions (see, e.g., lm).
...	further arguments, such as at, to be passed to boxplot.

Author(s)

John Fox <jfox@mcmaster.ca>, with a contribution from Steve Ellison to handle at argument (see boxplot).

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

boxplot

Examples

```
Boxplot(~income, data=Prestige, id=list(n=Inf)) # identify all outliers
Boxplot(income ~ type, data=Prestige)
Boxplot(income ~ type, data=Prestige, at=c(1, 3, 2))
Boxplot(k5 + k618 ~ lfp*wc, data=Mroz)
with(Prestige, Boxplot(income, id=list(labels=rownames(Prestige))))
with(Prestige, Boxplot(income, type, id=list(labels=rownames(Prestige))))
Boxplot(scale(Prestige[, 1:4]))
```

boxTidwell *Box-Tidwell Transformations*

Description

Computes the Box-Tidwell power transformations of the predictors in a linear model.

Usage

```
boxTidwell(y, ...)

## S3 method for class 'formula'
boxTidwell(formula, other.x=NULL, data=NULL, subset,
  na.action=getOption("na.action"), verbose=FALSE, tol=0.001,
  max.iter=25, ...)

## Default S3 method:
boxTidwell(y, x1, x2=NULL, max.iter=25, tol=0.001,
  verbose=FALSE, ...)

## S3 method for class 'boxTidwell'
print(x, digits=getOption("digits") - 2, ...)
```

Arguments

formula	two-sided formula, the right-hand-side of which gives the predictors to be transformed.
other.x	one-sided formula giving the predictors that are *not* candidates for transformation, including (e.g.) factors.
data	an optional data frame containing the variables in the model. By default the variables are taken from the environment from which boxTidwell is called.
subset	an optional vector specifying a subset of observations to be used.
na.action	a function that indicates what should happen when the data contain NAs. The default is set by the na.action setting of options.
verbose	if TRUE a record of iterations is printed; default is FALSE.
tol	if the maximum relative change in coefficients is less than tol then convergence is declared.
max.iter	maximum number of iterations.
y	response variable.
x1	matrix of predictors to transform.
x2	matrix of predictors that are *not* candidates for transformation.
...	not for the user.
x	boxTidwell object.
digits	number of digits for rounding.

Details

The maximum-likelihood estimates of the transformation parameters are computed by Box and Tidwell's (1962) method, which is usually more efficient than using a general nonlinear least-squares routine for this problem. Score tests for the transformations are also reported.

Value

an object of class `boxTidwell`, which is normally just printed.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Box, G. E. P. and Tidwell, P. W. (1962) Transformation of the independent variables. *Technometrics* **4**, 531-550.

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Examples

```
boxTidwell(prestige ~ income + education, ~ type + poly(women, 2), data=Prestige)
```

---

brief *Print Abbreviated Ouput*

Description

Print data objects and statistical model summaries in abbreviated form.

Usage

```
brief(object, ...)

## S3 method for class 'data.frame'
brief(object, rows = if (nr <= 10) c(nr, 0) else c(3, 2),
    cols, head=FALSE, tail=FALSE, elided = TRUE,
    classes = inherits(object, "data.frame"), ...)
## S3 method for class 'tbl'
brief(object, ...)
## S3 method for class 'matrix'
brief(object, rows = if (nr <= 10) c(nr, 0) else c(3, 2), ...)

## S3 method for class 'numeric'
brief(object, rows = c(2, 1), elided = TRUE, ...)
## S3 method for class 'integer'
```

```
brief(object, rows = c(2, 1), elided = TRUE, ...)
## S3 method for class 'character'
brief(object, rows = c(2, 1), elided = TRUE, ...)
## S3 method for class 'factor'
brief(object, rows=c(2, 1), elided=TRUE, ...)

## S3 method for class 'list'
brief(object, rows = c(2, 1), elided = TRUE, ...)

## S3 method for class 'function'
brief(object, rows = c(5, 3), elided = TRUE, ...)

## S3 method for class 'lm'
brief(object,  terms = ~ .,
    intercept=missing(terms), pvalues=FALSE,
    digits=3, horizontal=TRUE, vcov., ...)

## S3 method for class 'glm'
brief(object, terms = ~ .,
    intercept=missing(terms), pvalues=FALSE,
    digits=3, horizontal=TRUE, vcov., dispersion, exponentiate, ...)

## S3 method for class 'multinom'
brief(object, terms = ~ .,
    intercept=missing(terms), pvalues=FALSE,
    digits=3, horizontal=TRUE, exponentiate=TRUE, ...)

## S3 method for class 'polr'
brief(object, terms = ~ .,
    intercept, pvalues=FALSE,
    digits=3, horizontal=TRUE, exponentiate=TRUE, ...)

## Default S3 method:
brief(object, terms = ~ .,
    intercept=missing(terms), pvalues=FALSE,
    digits=3, horizontal=TRUE, ...)
```

Arguments

object	a data or model object to abbreviate.
rows	for a matrix or data frame, a 2-element integer vector with the number of rows to print at the beginning and end of the display; for a vector or factor, the number of lines of output to show at the beginning and end; for a list, the number of elements to show at the beginning and end; for a function, the number of lines to show at the beginning and end.
cols	for a matrix or data frame, a 2-element integer vector with the number of columns to print at the beginning (i.e., left) and end (right) of the display.
head, tail	alternatives to the rows argument; if TRUE, print the first or last 6 rows; can also

	be the number of the first or last few rows to print; only one of `heads` and `tails` should be specified; ignored if `FALSE` (the default).
`elided`	controls whether to report the number of elided elements, rows, or columns; default is `TRUE`.
`classes`	show the class of each column of a data frame at the top of the column; the classes are shown in single-character abbreviated form—e.g., `[f]` for a factor, `[i]` for an integer variable, `[n]` for a numeric variable, `[c]` for a character variable.
`terms`	a one-sided formula giving the terms to summarize; the default is `~ .`—i.e., to summarize all terms in the model.
`intercept`	whether or not to include the intercept; the default is `TRUE` unless the `terms` argument is given, in which case the default is `FALSE`; ignored for `polr` models.
`pvalues`	include the p-value for each coefficient in the table; default is `FALSE`.
`exponentiate`	for a `"glm"` or `"glmerMod"` model using the `log` or `logit` link, or a `"polr"` or `"multinom"` model, show exponentiated coefficient estimates and confidence bounds.
`digits`	significant digits for printing.
`horizontal`	if `TRUE` (the default), orient the summary produced by `brief` horizontally, which typically saves space.
`dispersion`	use an estimated covariance matrix computed as the dispersion times the unscaled covariance matrix; see `summary.glm`
`vcov.`	either a matrix giving the estimated covariance matrix of the estimates, or a function that when called with `object` as an argument returns an estimated covariance matrix of the estimates. If not set, `vcov(object, complete=FALSE)` is called to use the usual estimated covariance matrix with aliased regressors removed. Other choices include the functions documented at `hccm`, and a bootstrap estimate `vcov.=vcov(Boot(object))`; see the documentation for `Boot`. NOTES: (1) The `dispersion` and `vcov.` arguments may not *both* be specified. (2) Setting `vcov.=vcov` returns an error if the model includes aliased terms; use `vcov.=vcov(object, complete=FALSE)`. (3) The `hccm` method will generally return a matrix of full rank even if the model has aliased terms. Similarly `vcov.=vcov(Boot(object))` may return a full rank matrix.
`...`	arguments to pass down.

Value

Invisibly returns `object` for a data object, or summary for a model object.

Note

The method `brief.matrix` calls `brief.data.frame`, and `brief.tbl` (for tibbles) calls `print`.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

S

Examples

```
brief(rnorm(100))
brief(Duncan)
brief(OBrienKaiser, elided=TRUE)
brief(matrix(1:500, 10, 50))
brief(lm)

mod.prestige <- lm(prestige ~ education + income + type, Prestige)
brief(mod.prestige, pvalues=TRUE)
brief(mod.prestige, ~ type)
mod.mroz <- glm(lfp ~ ., data=Mroz, family=binomial)
brief(mod.mroz)
```

---

car-defunct *Defunct Functions in the car Package*

Description

These functions are were deprecated in 2009 and are now defunct.

Usage

```
av.plot(...)
av.plots(...)
box.cox(...)
bc(...)
box.cox.powers(...)
box.cox.var(...)
box.tidwell(...)
cookd(...)
confidence.ellipse(...)
ceres.plot(...)
ceres.plots(...)
cr.plot(...)
cr.plots(...)
data.ellipse(...)
durbin.watson(...)
levene.test(...)
leverage.plot(...)
leverage.plots(...)
```

```
linear.hypothesis(...)
ncv.test(...)
outlier.test(...)
qq.plot(...)
skewPower(...)
spread.level.plot(...)
```

Arguments

...	pass arguments down.

Details

av.plot and av.plots are replaced by avPlot and avPlots functions.

box.cox and bc are now replaced by bcPower.

box.cox.powers is replaced by powerTransform.

box.cox.var is replaced by boxCoxVariable.

box.tidwell is replaced by boxTidwell.

cookd is replaced by cooks.distance in the **stats** package.

confidence.ellipse is replaced by confidenceEllipse.

ceres.plot and ceres.plots are now replaced by the ceresPlot and ceresPlots functions.

cr.plot and cr.plots are now replaced by the crPlot and crPlots functions.

data.ellipse is replaced by dataEllipse.

durbin.watson is replaced by durbinWatsonTest.

levene.test is replaced by leveneTest function.

leverage.plot and leverage.plots are now replaced by the leveragePlot and leveragePlots functions.

linear.hypothesis is replaced by the linearHypothesis function.

ncv.test is replaced by ncvTest.

outlier.test is replaced by outlierTest.

qq.plot is replaced by qqPlot.

skewPower is replaced by bcnPower.

spread.level.plot is replaced by spreadLevelPlot.

car-deprecated *Deprecated Functions in the car Package*

Description

These functions are provided for compatibility with older versions of the **car** package only, and may be removed eventually. Commands that worked in versions of the **car** package prior to version 3.0-0 will not necessarily work in version 3.0-0 and beyond, or may not work in the same manner.

Usage

```
bootCase(...)

nextBoot(...)
```

Arguments

...	arguments to pass to methods.

Details

These functions are replaced by `Boot`.

See Also

See Also `Boot`

car-internal.Rd *Internal Objects for the* **car** *package*

Description

These objects (currently only the `.carEnv` environment) are exported for technical reasons and are not for direct use.

Author(s)

John Fox `<jfox@mcmaster.ca>`

carHexsticker *View the Official Hex Sticker for the car Package*

Description

Open the official hex sticker for the car package in your browser

Usage

```
carHexsticker()
```

Value

Used for its side effect of openning the hex sticker for the car package in your browser.

Author(s)

John Fox <jfox@mcmaster.ca>

Examples

```
## Not run:  # intended for interactive use
carHexsticker()

## End(Not run)
```

carPalette *Set or Retrieve* **car** *Package Color Palette*

Description

This function is used to set or retrieve colors to be used in **car** package graphics functions.

Usage

```
carPalette(palette)
```

Arguments

palette	if missing, returns the colors that will be used in **car** graphics; if present, the colors to be used in graphics will be set. The palette argument may also be one of "car" or "default" to use the default **car** palette (defined below), "R" to use the default R palette, or "colorblind" to use a colorblind-friendly palette (from https://jfly.uni-koeln.de/color/).

Details

This function sets or returns the value of `options(carPalette=pallete)` that will be use in **car** graphics functions to determine colors. The default is `c("black", "blue", "magenta", "cyan", "orange", "gray", "green3", "red"))`, which is nearly a permutation of the colors returned by the standard `palette` function that minimizes the use of red and green in the same graph, and that substitutes orange for the often hard to see yellow.

Value

Invisibly returns the previous value of the **car** palette.

Author(s)

Sanford Weisberg and John Fox

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

`palette`, `colors`

Examples

```
# Standard color palette
palette()
# car standard color palette
carPalette()
# set colors to all black
carPalette(rep("black", 8))
# Use a custom color palette with 12 distinct colors
carPalette(sample(colors(distinct=TRUE), 12))
# restore default
carPalette("default")
```

---

carWeb *Access to the R Companion to Applied Regression Website*

Description

This function will access the website for *An R Companion to Applied Regression*.

Usage

```
carWeb(...)
```

Arguments

...	optional arguments if present are passed to utils::browseURL

Value

Displays the web page for *An R Companion to Applied Regression*, Third Edition, Sage

---

`ceresPlots` *Ceres Plots*

Description

These functions draw Ceres plots for linear and generalized linear models.

Usage

```
ceresPlots(model, ...)

## Default S3 method:
ceresPlots(model, terms = ~., layout = NULL, ask, main,
    ...)

ceresPlot(model, ...)

## S3 method for class 'lm'
ceresPlot(model, variable, id=FALSE,
  line=TRUE,  smooth=TRUE, col=carPalette()[1], col.lines=carPalette()[-1],
  xlab, ylab, pch=1, lwd=2,  grid=TRUE, ...)

## S3 method for class 'glm'
ceresPlot(model, ...)
```

Arguments

`model`	model object produced by `lm` or `glm`.
`terms`	A one-sided formula that specifies a subset of the regressors. One component-plus-residual plot is drawn for each term. The default `~.` is to plot against all numeric predictors. For example, the specification `terms = ~ . - X3` would plot against all predictors except for `X3`. Factors and nonstandard predictors such as B-splines are skipped. If this argument is a quoted name of one of the regressors, the component-plus-residual plot is drawn for that predictor only.
`layout`	If set to a value like `c(1, 1)` or `c(4, 3)`, the layout of the graph will have this many rows and columns. If not set, the program will select an appropriate layout. If the number of graphs exceed nine, you must select the layout yourself, or you will get a maximum of nine per page. If `layout=NA`, the function does not set the layout and the user can use the `par` function to control the layout, for example to have plots from two models in the same graphics window.

ask	If TRUE, ask the user before drawing the next plot; if FALSE, the default, don't ask. This is relevant only if not all the graphs can be drawn in one window.
main	Overall title for any array of cerers plots; if missing a default is provided.
...	ceresPlots passes these arguments to ceresPlot. ceresPlot passes them to plot.
variable	A quoted string giving the name of a variable for the horizontal axis
id	controls point identification; if FALSE (the default), no points are identified; can be a list of named arguments to the showLabels function; TRUE is equivalent to list(method=list(abs(residuals(model, type="pearson")), "x"), n=2, cex=1, col=carPalette()[1], location="lr"), which identifies the 2 points with the largest residuals and the 2 points with the most extreme horizontal (X) values.
line	TRUE to plot least-squares line.
smooth	specifies the smoother to be used along with its arguments; if FALSE, no smoother is shown; can be a list giving the smoother function and its named arguments; TRUE, the default, is equivalent to list(smoother=loessLine). See ScatterplotSmoothers for the smoothers supplied by the **car** package and their arguments. Ceres plots do not support variance smooths.
col	color for points; the default is the first entry in the current **car** palette (see carPalette and par).
col.lines	a list of at least two colors. The first color is used for the ls line and the second color is used for the fitted lowess line. To use the same color for both, use, for example, col.lines=c("red", "red")
xlab, ylab	labels for the x and y axes, respectively. If not set appropriate labels are created by the function.
pch	plotting character for points; default is 1 (a circle, see par).
lwd	line width; default is 2 (see par).
grid	If TRUE, the default, a light-gray background grid is put on the graph

Details

Ceres plots are a generalization of component+residual (partial residual) plots that are less prone to leakage of nonlinearity among the predictors.

The function intended for direct use is ceresPlots.

The model cannot contain interactions, but can contain factors. Factors may be present in the model, but Ceres plots cannot be drawn for them.

Value

NULL. These functions are used for their side effect: producing plots.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Cook, R. D. and Weisberg, S. (1999) *Applied Regression, Including Computing and Graphics.* Wiley.

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

crPlots, avPlots, showLabels

Examples

```
ceresPlots(lm(prestige~income+education+type, data=Prestige), terms= ~ . - type)
```

`compareCoefs` *Print estimated coefficients and their standard errors in a table for several regression models.*

Description

This function extracts estimates of regression parameters and their standard errors from one or more models and prints them in a table.

Usage

```
compareCoefs(..., se = TRUE, zvals = FALSE, pvals = FALSE, vcov.,
    print = TRUE, digits = 3)
```

Arguments

`...`	One or more regression-model objects. These may be of class `lm`, `glm`, `nlm`, or any other regression method for which the functions `coef` and `vcov` return appropriate values, or if the object inherits from the `mer` class created by the `lme4` package or `lme` in the `nlme` package.
`se`	If `TRUE`, the default, show standard errors as well as estimates.
`zvals`	If `TRUE` (the default is `FALSE`), print Wald statistics, the ratio of each coefficient to its standard error.
`pvals`	If `TRUE` (the default is `FALSE`), print two-sided p-values from the standard normal distribution corresponding to the Wald statistics.
`vcov.`	an optional argument, specifying a function to be applied to all of the models, returning a coefficient covariance matrix for each, or a list with one element for each model, with each element either containing a function to be applied to the corresponding model or a coefficient covariance matrix for that model. If omitted, `vcov` is applied to each model. This argument can also be a list of estimated covariance matrices of the coefficient estimates.

print	If TRUE, the default, the results are printed in a nice format using printCoefmat. If FALSE, the results are returned as a matrix
digits	Passed to the printCoefmat function for printing the result.

Value

This function is mainly used for its side-effect of printing the result. It also invisibly returns a matrix of estimates, standard errors, Wald statistics, and p-values.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Examples

```
mod1 <- lm(prestige ~ income + education, data=Duncan)
mod2 <- update(mod1, subset=-c(6,16))
mod3 <- update(mod1, . ~ . + type)
mod4 <- update(mod1, . ~ . + I(income + education)) # aliased coef.
compareCoefs(mod1)
compareCoefs(mod1, mod2, mod4)
compareCoefs(mod1, mod2, mod3, zvals=TRUE, pvals=TRUE)
compareCoefs(mod1, mod2, se=FALSE)
compareCoefs(mod1, mod1, vcov.=list(vcov, hccm))
```

Contrasts *Functions to Construct Contrasts*

Description

These are substitutes for similarly named functions in the **stats** package (note the uppercase letter starting the second word in each function name). The only difference is that the contrast functions from the **car** package produce easier-to-read names for the contrasts when they are used in statistical models.

The functions and this documentation are adapted from the **stats** package.

Usage

```
contr.Treatment(n, base = 1, contrasts = TRUE)

contr.Sum(n, contrasts = TRUE)

contr.Helmert(n, contrasts = TRUE)
```

Arguments

n	a vector of levels for a factor, or the number of levels.
base	an integer specifying which level is considered the baseline level. Ignored if contrasts is FALSE.
contrasts	a logical indicating whether contrasts should be computed.

Details

These functions are used for creating contrast matrices for use in fitting analysis of variance and regression models. The columns of the resulting matrices contain contrasts which can be used for coding a factor with n levels. The returned value contains the computed contrasts. If the argument contrasts is FALSE then a square matrix is returned.

Several aspects of these contrast functions are controlled by options set via the options command:

decorate.contrasts This option should be set to a 2-element character vector containing the prefix and suffix characters to surround contrast names. If the option is not set, then c("[", "]") is used. For example, setting options(decorate.contrasts=c(".", "")) produces contrast names that are separated from factor names by a period. Setting options(decorate.contrasts=c("", "")) reproduces the behaviour of the R base contrast functions.

decorate.contr.Treatment A character string to be appended to contrast names to signify treatment contrasts; if the option is unset, then "T." is used.

decorate.contr.Sum Similar to the above, with default "S.".

decorate.contr.Helmert Similar to the above, with default "H.".

contr.Sum.show.levels Logical value: if TRUE (the default if unset), then level names are used for contrasts; if FALSE, then numbers are used, as in contr.sum in the base package.

Note that there is no replacement for contr.poly in the base package (which produces orthogonal-polynomial contrasts) since this function already constructs easy-to-read contrast names.

Value

A matrix with n rows and k columns, with k = n - 1 if contrasts is TRUE and k = n if contrasts is FALSE.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

contr.treatment, contr.sum, contr.helmert, contr.poly

Examples

```
# contr.Treatment vs. contr.treatment in the base package:

lm(prestige ~ (income + education)*type, data=Prestige,
    contrasts=list(type="contr.Treatment"))

##  Call:
##  lm(formula = prestige ~ (income + education) * type, data = Prestige,
##      contrasts = list(type = "contr.Treatment"))
##
##  Coefficients:
##          (Intercept)                  income               education
##              2.275753                0.003522                1.713275
##          type[T.prof]              type[T.wc]     income:type[T.prof]
##              15.351896              -33.536652                -0.002903
##      income:type[T.wc]  education:type[T.prof]    education:type[T.wc]
##              -0.002072                1.387809                4.290875

lm(prestige ~ (income + education)*type, data=Prestige,
    contrasts=list(type="contr.treatment"))

##  Call:
##  lm(formula = prestige ~ (income + education) * type, data = Prestige,
##      contrasts = list(type = "contr.treatment"))
##
##  Coefficients:
##      (Intercept)              income           education
##          2.275753            0.003522            1.713275
##          typeprof              typewc     income:typeprof
##          15.351896          -33.536652           -0.002903
##      income:typewc  education:typeprof    education:typewc
##          -0.002072            1.387809            4.290875
```

---

crPlots *Component+Residual (Partial Residual) Plots*

Description

These functions construct component+residual plots, also called partial-residual plots, for linear and generalized linear models.

Usage

```
crPlots(model, ...)

## Default S3 method:
crPlots(model, terms = ~., layout = NULL, ask, main,
    ...)
```

```
crp(...)

crPlot(model, ...)

## S3 method for class 'lm'
crPlot(model, variable, id=FALSE,
    order=1, line=TRUE, smooth=TRUE,
    col=carPalette()[1], col.lines=carPalette()[-1],
    xlab, ylab, pch=1, lwd=2, grid=TRUE, ...)

crPlot3d(model, var1, var2, ...)

## S3 method for class 'lm'
crPlot3d(model, var1, var2,
    xlab = var1,
    ylab = paste0("C+R(", eff$response, ")"), zlab = var2,
    axis.scales = TRUE, axis.ticks = FALSE, revolutions = 0,
    bg.col = c("white", "black"),
    axis.col = if (bg.col == "white") c("darkmagenta", "black", "darkcyan")
        else c("darkmagenta", "white", "darkcyan"),
    surface.col = carPalette()[2:3], surface.alpha = 0.5,
    point.col = "yellow", text.col = axis.col,
    grid.col = if (bg.col ==  "white") "black" else "gray",
    fogtype = c("exp2", "linear", "exp", "none"),
    fill = TRUE, grid = TRUE, grid.lines = 26,
    smoother = c("loess", "mgcv", "none"), df.mgcv = NULL, loess.args = NULL,
    sphere.size = 1, radius = 1, threshold = 0.01, speed = 1, fov = 60,
    ellipsoid = FALSE, level = 0.5, ellipsoid.alpha = 0.1,
    id = FALSE,
    mouseMode=c(none="none", left="polar", right="zoom", middle="fov",
                wheel="pull"),
    ...)
```

Arguments

`model`	model object produced by `lm` or `glm`.
`terms`	A one-sided formula that specifies a subset of the regressors. One component-plus-residual plot is drawn for each regressor. The default `~.` is to plot against all numeric regressors. For example, the specification `terms = ~ . - X3` would plot against all regressors except for `X3`, while `terms = ~ log(X4)` would give the plot for the predictor X4 that is represented in the model by log(X4). If this argument is a quoted name of one of the predictors, the component-plus-residual plot is drawn for that predictor only.
`var1, var2`	The quoted names of the two predictors in the model to use for a 3D C+R plot.
`layout`	If set to a value like `c(1, 1)` or `c(4, 3)`, the layout of the graph will have this many rows and columns. If not set, the program will select an appropriate layout. If the number of graphs exceed nine, you must select the layout yourself, or you will get a maximum of nine per page. If `layout=NA`, the function does not set the

layout and the user can use the par function to control the layout, for example to have plots from two models in the same graphics window.

ask — If TRUE, ask the user before drawing the next plot; if FALSE, the default, don't ask. This is relevant only if not all the graphs can be drawn in one window.

main — The title of the plot; if missing, one will be supplied.

... — crPlots passes these arguments to crPlot. crPlot passes them to plot.

variable — A quoted string giving the name of a variable for the horizontal axis.

id — controls point identification; if FALSE (the default), no points are identified; can be a list of named arguments to the showLabels function; TRUE is equivalent to list(method=list(abs(residuals(model, type="pearson")), "x"), n=2, cex=1, col=carPalette()[1], location="lr"), which identifies the 2 points with the largest residuals and the 2 points with the most extreme horizontal (X) values. For 3D C+R plots, see Identify3d.

order — order of polynomial regression performed for predictor to be plotted; default 1.

line — TRUE to plot least-squares line.

smooth — specifies the smoother to be used along with its arguments; if FALSE, no smoother is shown; can be a list giving the smoother function and its named arguments; TRUE, the default, is equivalent to list(smoother=loessLine). See ScatterplotSmoothers for the smoothers supplied by the **car** package and their arguments.

smoother, df.mgcv, loess.args
smoother specifies quoted name of the surface smoother to use for the partial residuals, either loess, the default, or mgcv. df.mgcv gives the degrees of freedom for the mgcv smoother; NULL, the default, causes the df to be computed by mgcv. loess.args is an optional list with named elements span, family and degree, with default span = 2/3; family = "gaussian" for a binomial or Poisson GLM and family = "symmetric" otherwise; and degree = 1 (see loess).

col — color for points; the default is the first entry in the current **car** palette (see carPalette and par).

col.lines — a list of at least two colors. The first color is used for the ls line and the second color is used for the fitted lowess line. To use the same color for both, use, for example, col.lines=c("red", "red")

xlab, ylab, zlab — labels for the x and y axes, and for the z axis of a 3D plot. If not set appropriate labels are created by the function. for the 3D C+R plot, the predictors are on the x and z axes and the response on the y (vertical) axis.

pch — plotting character for points; default is 1 (a circle, see par).

lwd — line width; default is 2 (see par).

grid — If TRUE, the default, a light-gray background grid is put on the graph. For a 3D C+R plot, see the grid argument for scatter3d.

grid.lines — number of horizontal and vertical lines to be drawn on regression surfaces for 2D C+R plots (26 by default); the square of grid.lines corresponds to the number of points at which the fitted partial regression surface is evaluated and so this argument should not be set too small.

`axis.scales`, `axis.ticks`, `revolutions`, `bg.col`, `axis.col`, `surface.col`, `surface.alpha`, `point.col`, `text.col`, `grid.col`, `fogtype`, `fill`, `sphere.size`, `radius`, `threshold`, `speed`, `fov`, `ellipsoid`, `level`, `ellipsoid.alpha`, `mouseMode`

see `scatter3d`.

Details

The functions intended for direct use are `crPlots`, for which `crp` is an abbreviation, and, for 3D C+R plots, `crPlot3d`.

For 2D plots, the model cannot contain interactions, but can contain factors. Parallel boxplots of the partial residuals are drawn for the levels of a factor. `crPlot3d` can handle models with two-way interactions.

For 2D C+R plots, the fit is represented by a broken blue line and a smooth of the partial residuals by a solid magenta line. For 3D C+R plots, the fit is represented by a blue surface and a smooth of the partial residuals by a magenta surface.

Value

These functions are used for their side effect of producing plots, but also invisibly return the coordinates of the plotted points.

Author(s)

John Fox `<jfox@mcmaster.ca>`

References

Cook, R. D. and Weisberg, S. (1999) *Applied Regression, Including Computing and Graphics.* Wiley.

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

`ceresPlots`, `avPlots`

Examples

```
crPlots(m<-lm(prestige ~ income + education, data=Prestige))

crPlots(m, terms=~ . - education) # get only one plot

crPlots(lm(prestige ~ log2(income) + education + poly(women,2), data=Prestige))

crPlots(glm(partic != "not.work" ~ hincome + children,
  data=Womenlf, family=binomial), smooth=list(span=0.75))

# 3D C+R plot, requires the rgl, effects, and mgcv packages
if (interactive() && require(rgl) && require(effects) && require(mgcv)){
```

```
crPlot3d(lm(prestige ~ income*education + women, data=Prestige),
    "income", "education")
}
```

---

deltaMethod *Estimate and Standard Error of a Nonlinear Function of Estimated Regression Coefficients*

Description

deltaMethod is a generic function that uses the delta method to get a first-order approximate standard error for a nonlinear function of a vector of random variables with known or estimated covariance matrix.

Usage

```
deltaMethod(object, ...)

## Default S3 method:
deltaMethod(object, g., vcov., func=g., constants, level=0.95,
  rhs, singular.ok=FALSE, ..., envir=parent.frame())
## S3 method for class 'lm'
 deltaMethod(object, g., vcov.=vcov(object, complete=FALSE),
           parameterNames=names(coef(object)), ..., envir=parent.frame())
## S3 method for class 'nls'
deltaMethod(object, g., vcov.=vcov(object, complete=FALSE), ..., envir=parent.frame())
## S3 method for class 'multinom'
 deltaMethod(object, g., vcov. = vcov(object, complete=FALSE),
           parameterNames = if (is.matrix(coef(object)))
       colnames(coef(object)) else names(coef(object)), ..., envir=parent.frame())
## S3 method for class 'polr'
 deltaMethod(object, g., vcov.=vcov(object, complete=FALSE), ..., envir=parent.frame())
## S3 method for class 'survreg'
 deltaMethod(object, g., vcov. = vcov(object, complete=FALSE),
           parameterNames = names(coef(object)), ..., envir=parent.frame())
## S3 method for class 'coxph'
 deltaMethod(object, g., vcov. = vcov(object, complete=FALSE),
           parameterNames = names(coef(object)), ..., envir=parent.frame())
## S3 method for class 'mer'
 deltaMethod(object, g., vcov. = vcov(object, complete=FALSE),
           parameterNames = names(fixef(object)), ..., envir=parent.frame())
## S3 method for class 'merMod'
 deltaMethod(object, g., vcov. = vcov(object, complete=FALSE),
           parameterNames = names(fixef(object)), ..., envir=parent.frame())
## S3 method for class 'lme'
 deltaMethod(object, g., vcov. = vcov(object, complete=FALSE),
           parameterNames = names(fixef(object)), ..., envir=parent.frame())
```

```
## S3 method for class 'lmList'
 deltaMethod(object, g.,  ..., envir=parent.frame())
```

Arguments

object	For the default method, object is either (1) a vector of p named elements, so names(object) returns a list of p character strings that are the names of the elements of object; or (2) a model object for which there are coef and vcov methods, and for which the named coefficient vector returned by coef is asymptotically normally distributed with asymptotic covariance matrix returned by vcov. For the other methods, object is a regression object for which coef(object) or fixef(object) returns a vector of parameter estimates.
g.	A quoted string that is the function of the parameter estimates to be evaluated; see the details below.
vcov.	The (estimated) covariance matrix of the coefficient estimates. For the default method, this argument is required. For all other methods, this argument must either provide the estimated covariance matrix or a function that when applied to object returns a covariance matrix. The default is to use the function vcov. Note that arguments supplied to ... are *not* passed to vcov. when it's a function; in this case either use an anonymous function in which the additional arguments are set, or supply the coefficient covariance matrix directly.
func	A quoted string used to annotate output. The default of func = g. is usually appropriate.
parameterNames	A character vector of length p that gives the names of the parameters in the same order as they appear in the vector of estimates. This argument will be useful if some of the names in the vector of estimates include special characters, like I(x2^2), or x1:x2 that will confuse the numerical differentiation function. See details below.
constants	This argument is a named vector whose elements are constants that are used in the f argument. It isn't generally necessary to specify this argument but it may be convenient to do so.
level	level for confidence interval, default 0.95.
rhs	hypothesized value for the specified function of parameters; if absent no hypothesis test is performed.
singular.ok	if TRUE (the default is FALSE), aliased coefficients are eliminated from the coefficient vector and covariance matrix; otherwise, aliased coefficients produce an error.
...	Used to pass arguments to the generic method.
envir	Environment in which g. is evaluated; not normally specified by the user.

Details

Suppose x is a random vector of length p that is at least approximately normally distributed with mean β and estimated covariance matrix C. Then any function $g(\beta)$ of β, is estimated by $g(x)$, which is in large samples normally distributed with mean $g(\beta)$ and estimated variance $h'Ch$, where

h is the first derivative of $g(\beta)$ with respect to β evaluated at x. This function returns both $g(x)$ and its standard error, the square root of the estimated variance.

The default method requires that you provide x in the argument `object`, C in the argument `vcov.`, and a text expression in argument `g.` that when evaluated gives the function g. The call `names(object)` must return the names of the elements of `x` that are used in the expression `g.`.

Since the delta method is often applied to functions of regression parameter estimates, the argument `object` may be the name of a regression object from which the estimates and their estimated variance matrix can be extracted. In most regression models, estimates are returned by the `coef(object)` and the variance matrix from `vcov(object)`. You can provide an alternative function for computing the sample variance matrix, for example to use a sandwich estimator.

For mixed models using `lme4` or `nlme`, the coefficient estimates are returned by the `fixef` function, while for `multinom`, `lmList` and `nlsList` coefficient estimates are returned by `coef` as a matrix. Methods for these models are provided to get the correct estimates and variance matrix.

The argument `g.` must be a quoted character string that gives the function of interest. For example, if you set `m2 <- lm(Y ~ X1 + X2 + X1:X2)`, then `deltaMethod(m2,"X1/X2")` applies the delta method to the ratio of the coefficient estimates for `X1` and `X2`. The argument `g.` can consist of constants and names associated with the elements of the vector of coefficient estimates.

In some cases the names may include characters such as the colon `:` used in interactions, or mathematical symbols like `+` or `-` signs that would confuse the function that computes numerical derivatives, and for this case you can replace the names of the estimates with the `parameterNames` argument. For example, the ratio of the `X2` main effect to the interaction term could be computed using `deltaMethod(m2, "b1/b3", parameterNames=c("b0", "b1", "b2", "b3"))`. The name "`(Intercept)`" used for the intercept in linear and generalized linear models is an exception, and it will be correctly interpreted by `deltaMethod`. Another option is to use back-ticks to quote non-standard R names, as in `deltaMethod(m2,"X1/`X1:X2`")`.

For `multinom` objects, the `coef` function returns a matrix of coefficients, with each row giving the estimates for comparisons of one category to the baseline. The `deltaMethod` function applies the delta method to each row of this matrix. Similarly, for `lmList` and `nlsList` objects, the delta method is computed for each element of the list of models fit.

For nonlinear regression objects produced by the `nls` function, the call `coef(object)` returns the estimated coefficient vectors with names corresponding to parameter names. For example, `m2 <- nls(y ~ theta/(1 + gamma * x), start = list(theta=2, gamma=3))` will have parameters named `c("theta", "gamma")`. In many other familiar regression models, such as those produced by `lm` and `glm`, the names of the coefficient estimates are the corresponding regressor names, not parameter names.

For mixed-effects models fit with `lmer` and `glmer` from the **lme4** package or `lme` and `nlme` from the **nlme** package, only fixed-effect coefficients are considered.

For regression models for which methods are not provided, you can extract the named vector of coefficient estimates and and estimate of its covariance matrix and then apply the default `deltaMethod` function.

Note: Earlier versions of `deltaMethod` included an argument `parameterPrefix` that implemented the same functionality as the `parameterNames` argument, but which caused several problems that were not easily fixed without the change in syntax.

Value

An object of class "deltaMethod", inheriting from "data.frame", for which a print method is provided. The object contains columns named Estimate for the estimate, SE for its standard error, and columns for confidence limits and possibly a hypothesis test. The value of g. is given as a row label.

Author(s)

Sanford Weisberg, <sandy@umn.edu>, John Fox <jfox@mcmaster.ca>, and Pavel Krivitsky.

References

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Weisberg, S. (2014) *Applied Linear Regression*, Fourth Edition, Wiley, Section 6.1.2.

See Also

First derivatives of g. are computed using symbolic differentiation by the function D.

Examples

```
m1 <- lm(time ~ t1 + t2, data = Transact)
deltaMethod(m1, "b1/b2", parameterNames= paste("b", 0:2, sep=""))
deltaMethod(m1, "t1/t2", rhs=1) # use names of preds. rather than coefs.
deltaMethod(m1, "t1/t2", vcov=hccm) # use hccm function to est. vars.
deltaMethod(m1, "1/(Intercept)")
# The next example invokes the default method by extracting the
# vector of estimates and covariance matrix explicitly
deltaMethod(coef(m1), "t1/t2", vcov.=vcov(m1))
# Mroz example, in carData, logistic regression using 2 predictors
mroz.mod1 <- glm(lfp ~ k5 + age, family=binomial, data=Mroz)
# The S() method in car returns exponentiated logistic regression coefficients by default
S(mroz.mod1)
# Exponentiated coefficient estimates can also be computed one at a time by deltaMethod
deltaMethod(mroz.mod1, "exp(k5)")
# The point estimate is the same as in the S() output but the confidence interval is
# different because S() simply exponentiated the ends of the confidence interval before
# exponentiation.
# A more complex case is to get the estimate and standard error for the inverse logit
# of the coefficient estimate.
deltaMethod(mroz.mod1, "exp(b1)/(1 + exp(b1))", parameterNames= paste("b", 0:2, sep=""))
# the parameterNames argument was used to change the name of the coefficient from
# k5 to b1.
```

densityPlot *Nonparametric Density Estimates*

Description

densityPlot contructs and graphs nonparametric density estimates, possibly conditioned on a factor, using the standard R density function or by default adaptiveKernel, which computes an adaptive kernel density estimate. depan provides the Epanechnikov kernel and dbiwt provides the biweight kernel.

Usage

```
densityPlot(x, ...)

## Default S3 method:
densityPlot(x, g, method=c("adaptive", "kernel"),
    bw=if (method == "adaptive") bw.nrd0 else "SJ", adjust=1,
    kernel, xlim, ylim,
    normalize=FALSE, xlab=deparse(substitute(x)), ylab="Density", main="",
    col=carPalette(), lty=seq_along(col), lwd=2, grid=TRUE,
    legend=TRUE, show.bw=FALSE, rug=TRUE, ...)

## S3 method for class 'formula'
densityPlot(formula, data=NULL, subset,
    na.action=NULL, xlab, ylab, main="", legend=TRUE, ...)

adaptiveKernel(x, kernel=dnorm, bw=bw.nrd0, adjust=1.0, n=500,
    from, to, cut=3, na.rm=TRUE)

depan(x)
dbiwt(x)
```

Arguments

x	a numeric variable, the density of which is estimated; for depan and dbiwt, the argument of the kernel function.
g	an optional factor to divide the data.
formula	an R model formula, of the form ~ variable to estimate the unconditional density of variable, or variable ~ factor to estimate the density of variable within each level of factor.
data	an optional data frame containing the data.
subset	an optional vector defining a subset of the data.
na.action	a function to handle missing values; defaults to the value of the R na.action option, initially set to na.omit.
method	either "adaptive" (the default) for an adaptive-kernel estimate or "kernel" for a fixed-bandwidth kernel estimate.

bw	the geometric mean bandwidth for the adaptive-kernel or bandwidth of the kernel density estimate(s). Must be a numerical value or a function to compute the bandwidth (default bw.nrd0) for the adaptive kernel estimate; for the kernel estimate, may either the quoted name of a rule to compute the bandwidth, or a numeric value. If plotting by groups, bw may be a vector of values, one for each group. See density and bw.SJ for details of the kernel estimator.
adjust	a multiplicative adjustment factor for the bandwidth; the default, 1, indicates no adjustment; if plotting by groups, adjust may be a vector of adjustment factors, one for each group. The default bandwidth-selection rule tends to give a value that's too large if the distribution is asymmetric or has multiple modes; try setting adjust < 1, particularly for the adaptive-kernel estimator.
kernel	for densityPlot this is the name of the kernel function for the kernel estimator (the default is "gaussian", see density); or a kernel function for the adaptive-kernel estimator (the default is dnorm, producing the Gaussian kernel). For adaptivekernel this is a kernel function, defaulting to dnorm, which is the Gaussian kernel (standard-normal density).
xlim, ylim	axis limits; if missing, determined from the range of x-values at which the densities are estimated and the estimated densities.
normalize	if TRUE (the default is FALSE), the estimated densities are rescaled to integrate approximately to 1; particularly useful if the density is estimated over a restricted domain, as when from or to are specified.
xlab	label for the horizontal-axis; defaults to the name of the variable x.
ylab	label for the vertical axis; defaults to "Density".
main	plot title; default is empty.
col	vector of colors for the density estimate(s); defaults to the color carPalette.
lty	vector of line types for the density estimate(s); defaults to the successive integers, starting at 1.
lwd	line width for the density estimate(s); defaults to 2.
grid	if TRUE (the default), grid lines are drawn on the plot.
legend	a list of up to two named elements: location, for the legend when densities are plotted for several groups, defaults to "upperright" (see legend); and title of the legend, which defaults to the name of the grouping factor. If TRUE, the default, the default values are used; if FALSE, the legend is suppressed.
n	number of equally spaced points at which the adaptive-kernel estimator is evaluated; the default is 500.
from, to, cut	the range over which the density estimate is computed; the default, if missing, is min(x) - cut*bw, max(x) + cut*bw.
na.rm	remove missing values from x in computing the adaptive-kernel estimate? The default is TRUE.
show.bw	if TRUE, show the bandwidth(s) in the horizontal-axis label or (for multiple groups) the legend; the default is FALSE.
rug	if TRUE (the default), draw a rug plot (one-dimentional scatterplot) at the bottom of the density estimate.
...	arguments to be passed down to graphics functions.

Details

If you use a different kernel function than the default dnorm that has a standard deviation different from 1 along with an automatic rule like the default function bw.nrd0, you can attach an attribute to the kernel function named "scale" that gives its standard deviation. This is true for the two supplied kernels, depan and dbiwt

Value

densityPlot invisibly returns the "density" object computed (or list of "density" objects) and draws a graph. adaptiveKernel returns an object of class "density" (see density).

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

W. N. Venables and B. D. Ripley (2002) *Modern Applied Statistics with S*. New York: Springer.

B.W. Silverman (1986) *Density Estimation for Statistics and Data Analysis*. London: Chapman and Hall.

See Also

density, bw.SJ, plot.density

Examples

```
densityPlot(~ income, show.bw=TRUE, method="kernel", data=Prestige)
densityPlot(~ income, show.bw=TRUE, data=Prestige)
densityPlot(~ income, from=0, normalize=TRUE, show.bw=TRUE, data=Prestige)

densityPlot(income ~ type, data=Prestige)
densityPlot(~ income, show.bw=TRUE, method="kernel", data=Prestige)
densityPlot(~ income, show.bw=TRUE, data=Prestige)
densityPlot(~ income, from=0, normalize=TRUE, show.bw=TRUE, data=Prestige)

densityPlot(income ~ type, kernel=depan, data=Prestige)
densityPlot(income ~ type, kernel=depan, legend=list(location="top"), data=Prestige)

plot(adaptiveKernel(UN$infantMortality, from=0, adjust=0.75), col="magenta")
lines(density(na.omit(UN$infantMortality), from=0, adjust=0.75), col="blue")
rug(UN$infantMortality, col="cyan")
legend("topright", col=c("magenta", "blue"), lty=1,
       legend=c("adaptive kernel", "kernel"), inset=0.02)


plot(adaptiveKernel(UN$infantMortality, from=0, adjust=0.75), col="magenta")
lines(density(na.omit(UN$infantMortality), from=0, adjust=0.75), col="blue")
rug(UN$infantMortality, col="cyan")
```

```
legend("topright", col=c("magenta", "blue"), lty=1,
       legend=c("adaptive kernel", "kernel"), inset=0.02)
```

---

dfbetaPlots *dfbeta and dfbetas Index Plots*

Description

These functions display index plots of dfbeta (effect on coefficients of deleting each observation in turn) and dfbetas (effect on coefficients of deleting each observation in turn, standardized by a deleted estimate of the coefficient standard error). In the plot of dfbeta, horizontal lines are drawn at 0 and +/- one standard error; in the plot of dfbetas, horizontal lines are drawn and 0 and +/- 1.

Usage

```
dfbetaPlots(model, ...)

dfbetasPlots(model, ...)

## S3 method for class 'lm'
dfbetaPlots(model, terms= ~ ., intercept=FALSE, layout=NULL, ask,
     main, xlab, ylab, labels=rownames(dfbeta),
        id.method="y",
        id.n=if(id.method[1]=="identify") Inf else 0, id.cex=1,
     id.col=carPalette()[1], id.location="lr", col=carPalette()[1], grid=TRUE, ...)

## S3 method for class 'lm'
dfbetasPlots(model, terms=~., intercept=FALSE, layout=NULL, ask,
     main, xlab, ylab,
        labels=rownames(dfbetas), id.method="y",
        id.n=if(id.method[1]=="identify") Inf else 0, id.cex=1,
     id.col=carPalette()[1], id.location="lr", col=carPalette()[1], grid=TRUE, ...)
```

Arguments

model	model object produced by lm or glm.
terms	A one-sided formula that specifies a subset of the terms in the model. One dfbeta or dfbetas plot is drawn for each regressor. The default ~. is to plot against all terms in the model with the exception of an intercept. For example, the specification terms = ~.-X3 would plot against all terms except for X3. If this argument is a quoted name of one of the terms, the index plot is drawn for that term only.
intercept	Include the intercept in the plots; default is FALSE.

layout	If set to a value like c(1, 1) or c(4, 3), the layout of the graph will have this many rows and columns. If not set, the program will select an appropriate layout. If the number of graphs exceed nine, you must select the layout yourself, or you will get a maximum of nine per page. If layout=NA, the function does not set the layout and the user can use the par function to control the layout, for example to have plots from two models in the same graphics window.
main	The title of the graph; if missing, one will be supplied.
xlab	Horizontal axis label; defaults to "Index".
ylab	Vertical axis label; defaults to coefficient name.
ask	If TRUE, ask the user before drawing the next plot; if FALSE, the default, don't ask.
...	optional additional arguments to be passed to plot, points, and showLabels
.	
id.method, labels, id.n, id.cex, id.col, id.location	Arguments for the labelling of points. The default is id.n=0 for labeling no points. See showLabels for details of these arguments.
col	color for points; defaults to the first entry in the color carPalette.
grid	If TRUE, the default, a light-gray background grid is put on the graph

Value

NULL. These functions are used for their side effect: producing plots.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

dfbeta ,dfbetas

Examples

```
dfbetaPlots(lm(prestige ~ income + education + type, data=Duncan))

dfbetasPlots(glm(partic != "not.work" ~ hincome + children,
  data=Womenlf, family=binomial))
```

durbinWatsonTest *Durbin-Watson Test for Autocorrelated Errors*

Description

Computes residual autocorrelations and generalized Durbin-Watson statistics and their bootstrapped p-values. dwt is an abbreviation for durbinWatsonTest.

Usage

```
durbinWatsonTest(model, ...)

dwt(...)

## S3 method for class 'lm'
durbinWatsonTest(model, max.lag=1, simulate=TRUE, reps=1000,
    method=c("resample","normal"),
    alternative=c("two.sided", "positive", "negative"), ...)

## Default S3 method:
durbinWatsonTest(model, max.lag=1, ...)

## S3 method for class 'durbinWatsonTest'
print(x, ...)
```

Arguments

model	a linear-model object, or a vector of residuals from a linear model.
max.lag	maximum lag to which to compute residual autocorrelations and Durbin-Watson statistics.
simulate	if TRUE p-values will be estimated by bootstrapping.
reps	number of bootstrap replications.
method	bootstrap method: "resample" to resample from the observed residuals; "normal" to sample normally distributed errors with 0 mean and standard deviation equal to the standard error of the regression.
alternative	sign of autocorrelation in alternative hypothesis; specify only if max.lag = 1; if max.lag > 1, then alternative is taken to be "two.sided".
...	arguments to be passed down.
x	durbinWatsonTest object.

Value

Returns an object of type "durbinWatsonTest".

Note

p-values are available only from the `lm` method.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Examples

```
durbinWatsonTest(lm(fconvict ~ tfr + partic + degrees + mconvict, data=Hartnagel))
```

Ellipses *Ellipses, Data Ellipses, and Confidence Ellipses*

Description

These functions draw ellipses, including data ellipses, and confidence ellipses for linear, generalized linear, and possibly other models.

Usage

```
ellipse(center, shape, radius, log="", center.pch=19, center.cex=1.5,
  segments=51, draw=TRUE, add=draw, xlab="", ylab="",
  col=carPalette()[2], lwd=2, fill=FALSE, fill.alpha=0.3, grid=TRUE, ...)

dataEllipse(x, ...)

## S3 method for class 'formula'
dataEllipse(formula, data, subset, weights, xlab, ylab, id=FALSE, ...)

## Default S3 method:
dataEllipse(x, y, groups,
  group.labels=group.levels, ellipse.label,
  weights, log="", levels=c(0.5, 0.95), center.pch=19,
  center.cex=1.5, draw=TRUE, plot.points=draw, add=!plot.points, segments=51,
  robust=FALSE,
  xlab=deparse(substitute(x)),
  ylab=deparse(substitute(y)),
 col=if (missing(groups)) carPalette()[1:2] else carPalette()[1:length(group.levels)],
  pch=if (missing(groups)) 1 else seq(group.levels),
  lwd=2, fill=FALSE, fill.alpha=0.3, grid=TRUE, id=FALSE,
  label.pos=NULL, label.cex=1.25, label.xpd=FALSE, ...)
```

```
confidenceEllipse(model, ...)

## S3 method for class 'lm'
confidenceEllipse(model, which.coef, vcov.=vcov,
  L, levels=0.95, Scheffe=FALSE, dfn,
  center.pch=19, center.cex=1.5, segments=51, xlab, ylab,
  col=carPalette()[2], lwd=2, fill=FALSE, fill.alpha=0.3, draw=TRUE, add=!draw,
  grid=TRUE, ...)

## S3 method for class 'glm'
confidenceEllipse(model, chisq, ...)

## S3 method for class 'mlm'
confidenceEllipse(model, xlab, ylab, which.coef=1:2, ...)

## Default S3 method:
confidenceEllipse(model, which.coef, vcov.=vcov,
  L, levels=0.95, Scheffe=FALSE,  dfn,
  center.pch=19, center.cex=1.5, segments=51, xlab, ylab,
  col=carPalette()[2], lwd=2, fill=FALSE, fill.alpha=0.3, draw=TRUE, add=!draw,
  grid=TRUE, ...)

confidenceEllipses(model, ...)

## Default S3 method:
confidenceEllipses(model, coefnames,  main, grid=TRUE, ...)

## S3 method for class 'mlm'
confidenceEllipses(model, coefnames, main, ...)
```

Arguments

center	2-element vector with coordinates of center of ellipse.
shape	2×2 shape (or covariance) matrix.
radius	radius of circle generating the ellipse.
log	when an ellipse is to be added to an existing plot, indicates whether computations were on logged values and to be plotted on logged axes; "x" if the x-axis is logged, "y" if the y-axis is logged, and "xy" or "yx" if both axes are logged. The default is "", indicating that neither axis is logged.
center.pch	character for plotting ellipse center; if FALSE or NULL the center point isn't plotted.
center.cex	relative size of character for plotting ellipse center.
segments	number of line-segments used to draw ellipse.
draw	if TRUE produce graphical output; if FALSE, only invisibly return coordinates of ellipse(s).
add	if TRUE add ellipse(s) to current plot.

`xlab`	label for horizontal axis.
`ylab`	label for vertical axis.
`x`	a numeric vector, or (if `y` is missing) a 2-column numeric matrix.
`y`	a numeric vector, of the same length as `x`.
`groups`	optional: a factor to divide the data into groups; a separate ellipse will be plotted for each group (level of the factor).
`formula`	for the formula method, a model formula, of the form `y ~ x` or, if plotting by groups, `y ~ x | z`, where `z` evaluates to a factor or other variable dividing the data into groups.
`data`	for the formula method, a data frame within which to evaluate the formula.
`subset`	for the formula method, expression defining a subset of observations.
`group.labels`	labels to be plotted for the groups; by default, the levels of the `groups` factor.
`ellipse.label`	a label for the ellipse(s) or a vector of labels; if several ellipses are drawn and just one label is given, then that label will be repeated. The default is not to label the ellipses.
`label.pos`	position(s) of labels for the ellipses, a vector of integers (in 0:4) or character strings (in c("center", "bottom", "left", "top", "right"), or in c("C", "S", "W", "N", "E") used in labeling ellipses, recycled as necessary. Values of 1, 2, 3 and 4, respectively indicate positions below, to the left of, above and to the right of the max/min coordinates of the ellipse; the value 0 specifies the centroid of the ellipse object. The default, `label.pos=NULL` uses the correlation of the ellipse to determine "top" (r>=0) or "bottom" (r<0)
.	
`label.cex`	character size for ellipse labels.
`label.xpd`	allow labels to be drawn outside of plotting region (default `FALSE`).
`weights`	a numeric vector of weights, of the same length as `x` and `y` to be used by `cov.wt` or `cov.trob` in computing a weighted covariance matrix; if absent, weights of 1 are used.
`plot.points`	if `FALSE` data ellipses are drawn, but points are not plotted.
`levels`	draw elliptical contours at these (normal) probability or confidence levels.
`robust`	if `TRUE` use the `cov.trob` function in the **MASS** package to calculate the center and covariance matrix for the data ellipse.
`model`	a model object produced by `lm` or `glm`.
`which.coef`	2-element vector giving indices of coefficients to plot; if missing, the first two coefficients (disregarding the regression constant) will be selected.
`vcov.`	a coefficient-covariance matrix or a function (such as `hccm`) to compute the coefficent-covariance matrix from `model`; the default is the `vcov` function. Note that arguments supplied to `...` are *not* passed to `vcov.` when it's a function; in this case either use an anonymous function in which the additional arguments are set, or supply the coefficient covariance matrix directly.

L	As an alternative to selecting coefficients to plot, a transformation matrix can be specified to compute two linear combinations of the coefficients; if the L matrix is given, it takes precedence over the which.coef argument. L should have two rows and as many columns as there are coefficients. It can be given directly as a numeric matrix, or specified by a pair of character-valued expressions, in the same manner as for the link{linearHypothesis} function, but with no right-hand side.
Scheffe	if TRUE scale the ellipse so that its projections onto the axes give Scheffe confidence intervals for the coefficients.
dfn	"numerator" degrees of freedom (or just degrees of freedom for a GLM) for drawing the confidence ellipse. Defaults to the number of coefficients in the model (disregarding the constant) if Scheffe is TRUE, or to 2 otherwise; selecting dfn = 1 will draw the "confidence-interval generating" ellipse, with projections on the axes corresponding to individual confidence intervals with the stated level of coverage.
chisq	if TRUE, the confidence ellipse for the coefficients in a generalized linear model are based on the chisquare statistic, if FALSE on the F-statistic. This corresponds to using the default and linear-model methods respectively.
col	color for lines and ellipse center; the default is the *second* entry in the current **car** palette (see carPalette and par). For dataEllipse, two colors can be given, in which case the first is for plotted points and the second for lines and the ellipse center; if ellipses are plotted for groups, then this is a vector of colors for the groups.
pch	for dataEllipse this is the plotting character (default, symbol 1, a hollow circle) to use for the points; if ellipses are plotted by groups, then this a vector of plotting characters, with consecutive symbols starting with 1 as the default.
lwd	line width; default is 2 (see par).
fill	fill the ellipse with translucent color col (default, FALSE)?
fill.alpha	transparency of fill (default = 0.3).
...	other plotting parameters to be passed to plot and line.
id	controls point identification; if FALSE (the default), no points are identified; can be a list of named arguments to the showLabels function; TRUE is equivalent to list(method="mahal", n=2, cex=1, col=carPalette()[1], location="lr") (with the default col actually dependent on the number of groups), which identifies the 2 points with the largest Mahalanobis distances from the center of the data.
grid	If TRUE, the default, a light-gray background grid is put on the graph
coefnames	character vector of coefficient names to use to label the diagonal of the pairwise confidence ellipse matrix plotted by confidenceEllipses; defaults to the names of the coefficients in the model.
main	title for matrix of pairwise confidence ellipses.

Details

The ellipse is computed by suitably transforming a unit circle.

`dataEllipse` superimposes the normal-probability contours over a scatterplot of the data.

`confidenceEllipses` plots a matrix of all pairwise confidence ellipses; each panel of the matrix is created by `confidenceEllipse`.

Value

These functions are mainly used for their side effect of producing plots. For greater flexibility (e.g., adding plot annotations), however, `ellipse` returns invisibly the (x, y) coordinates of the calculated ellipse. `dataEllipse` and `confidenceEllipse` return invisibly the coordinates of one or more ellipses, in the latter instance a list named by `levels`; `confidenceEllipses` invisibly returns `NULL`.

Author(s)

Georges Monette, John Fox <jfox@mcmaster.ca>, and Michael Friendly.

References

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Monette, G. (1990) Geometry of multiple regression and 3D graphics. In Fox, J. and Long, J. S. (Eds.) *Modern Methods of Data Analysis.* Sage.

See Also

`cov.trob`, `cov.wt`, `linearHypothesis`.

Examples

```
dataEllipse(Duncan$income, Duncan$education, levels=0.1*1:9,
    ellipse.label=0.1*1:9, lty=2, fill=TRUE, fill.alpha=0.1)

# formula method
dataEllipse(education ~ income | type, data=Duncan,
            levels = 0.68)

dataEllipse(education ~ income | type, data=Duncan,
            levels=0.68, label.xpd=TRUE)

# vary label positions
dataEllipse(education ~ income | type, data=Duncan,
            levels = 0.68,
            fill = TRUE, fill.alpha = 0.1,
            label.pos = c(3, 2, 4),
            xlim = c(0, 100),
            ylim = c(2, 120))


confidenceEllipse(lm(prestige ~ income + education, data=Duncan), Scheffe=TRUE)
```

```
confidenceEllipse(lm(prestige ~ income + education, data=Duncan), vcov.=hccm)

confidenceEllipse(lm(prestige ~ income + education, data=Duncan),
 L=c("income + education", "income - education"))

confidenceEllipses(lm(prestige ~ income + education + type, data=Duncan),
  fill=TRUE)
cov2cor(vcov(lm(prestige ~ income + education + type,
  data=Duncan))) # correlations among coefficients

wts <- rep(1, nrow(Duncan))
wts[c(6, 16)] <- 0 # delete Minister, Conductor
with(Duncan, {
 dataEllipse(income, prestige, levels=0.68)
 dataEllipse(income, prestige, levels=0.68, robust=TRUE,
     plot.points=FALSE, col="green3")
 dataEllipse(income, prestige, weights=wts, levels=0.68,
     plot.points=FALSE, col="brown")
 dataEllipse(income, prestige, weights=wts, robust=TRUE, levels=0.68,
  plot.points=FALSE, col="blue")
 })

with(Prestige, dataEllipse(income, education, type,
    id=list(n=2, labels=rownames(Prestige)), pch=15:17,
    xlim=c(0, 25000), center.pch="+",
    group.labels=c("Blue Collar", "Professional", "White Collar"),
    ylim=c(5, 20), level=.95, fill=TRUE, fill.alpha=0.1))
```

`Export` *Export a data frame to disk in one of many formats*

Description

Uses the `export` function in the **rio** package to export a file to disk. This function adds an argument for converting row.names to a column in the resulting file.

Usage

```
Export(x, file, format, ..., keep.row.names)
```

Arguments

`x`	A data frame or matrix to be written to a file.
`file`	A character string naming a file. If the file name has an extension, such as `.xlsx`, the extention is used to infer the type of file to be exported. See `export` for the file types supported.
`format`	see `export`.
`...`	Additional arguments; see `export`.

`keep.row.names` If set to `TRUE`, then the data frame's row.names are appended to the left of the data frame with the name "id". If set to quoted character string, the row.names are added using the character string as its name. If set to `FALSE` row.names are lost.

Details

This is a convenience function in the **car** package for exporting (writing) a data frame to a file in a wide variety of formats including csv, Microsoft Excel. It optionally allows converting the row.names for the data frame to a column before writing. It then calls `export` in the `rio` package. That function in turn uses many other packages and functions for writing the function to a file.

Value

The name of the output file as a character string (invisibly).

Author(s)

Sanford Weisberg <sandy@umn.edu>

References

Chung-hong Chan, Geoffrey CH Chan, Thomas J. Leeper, and Jason Becker (2017). rio: A Swiss-army knife for data file I/O. R package version 0.5.0.

See Also

`export`, `Import`

Examples

```
if(require("rio")) {

Export(Duncan, "Duncan.csv", keep.row.names="occupation")
Duncan2 <- Import("Duncan.csv") # Automatically restores row.names
identical(Duncan, Duncan2)
# cleanup
unlink("Duncan.csv")

}
```

---

hccm *Heteroscedasticity-Corrected Covariance Matrices*

Description

Calculates heteroscedasticity-corrected covariance matrices linear models fit by least squares or weighted least squares. These are also called "White-corrected" or "White-Huber" covariance matrices.

Usage

```
hccm(model, ...)

## S3 method for class 'lm'
hccm(model, type=c("hc3", "hc0", "hc1", "hc2", "hc4"),
 singular.ok=TRUE, ...)

## Default S3 method:
hccm(model, ...)
```

Arguments

model	a unweighted or weighted linear model, produced by lm.
type	one of "hc0", "hc1", "hc2", "hc3", or "hc4"; the first of these gives the classic White correction. The "hc1", "hc2", and "hc3" corrections are described in Long and Ervin (2000); "hc4" is described in Cribari-Neto (2004).
singular.ok	if FALSE (the default is TRUE), a model with aliased coefficients produces an error; otherwise, the aliased coefficients are ignored in the coefficient covariance matrix that's returned.
...	arguments to pass to hccm.lm.

Details

The original White-corrected coefficient covariance matrix ("hc0") for an unweighted model is

$$V(b) = (X'X)^{-1}X'diag(e_i^2)X(X'X)^{-1}$$

where e_i^2 are the squared residuals, and X is the model matrix. The other methods represent adjustments to this formula. If there are weights, these are incorporated in the corrected covariance matrix.

The function hccm.default simply catches non-lm objects.

See Freedman (2006) and Fox and Weisberg (2019, Sec. 5.1.2) for discussion of the use of these methods in generalized linear models or models with nonconstant variance.

Value

The heteroscedasticity-corrected covariance matrix for the model.

The function will return an error, rather than a matrix, under two circumstances. First, if any of the cases have hatvalue (leverage) equal to one, then the corresponding fitted value will always equal the observed value. For types hc2, hc3 and hc4 the hccm matrix is undefined. For hc0 and hc1 it is defined but it can be shown to be singular, and therefore not a consistent estimate of the covariance matrix of the coefficient estimates. A singular estimate of the covariance matrix may also be obstained if the matrix X is ill-conditioned. In this latter case rescaling the model matrix may give a full-rank estimate.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Cribari-Neto, F. (2004) Asymptotic inference under heteroskedasticity of unknown form. *Computational Statistics and Data Analysis* **45**, 215–233.

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Freedman, D. (2006) On the so-called "Huber sandwich estimator" and "robust standard errors", *American Statistician*, **60**, 299–302.

Long, J. S. and Ervin, L. H. (2000) Using heteroscedasity consistent standard errors in the linear regression model. *The American Statistician* **54**, 217–224.

White, H. (1980) A heteroskedastic consistent covariance matrix estimator and a direct test of heteroskedasticity. *Econometrica* **48**, 817–838.

Examples

```
mod <- lm(interlocks ~ assets + nation, data=Ornstein)
print(vcov(mod), digits=4)
##             (Intercept)     assets  nationOTH   nationUK   nationUS
## (Intercept)   1.079e+00 -1.588e-05 -1.037e+00 -1.057e+00 -1.032e+00
## assets       -1.588e-05  1.642e-09  1.155e-05  1.362e-05  1.109e-05
## nationOTH    -1.037e+00  1.155e-05  7.019e+00  1.021e+00  1.003e+00
## nationUK     -1.057e+00  1.362e-05  1.021e+00  7.405e+00  1.017e+00
## nationUS     -1.032e+00  1.109e-05  1.003e+00  1.017e+00  2.128e+00

print(hccm(mod), digits=4)
##             (Intercept)     assets  nationOTH   nationUK   nationUS
## (Intercept)   1.664e+00 -3.957e-05 -1.569e+00 -1.611e+00 -1.572e+00
## assets       -3.957e-05  6.752e-09  2.275e-05  3.051e-05  2.231e-05
## nationOTH    -1.569e+00  2.275e-05  8.209e+00  1.539e+00  1.520e+00
## nationUK     -1.611e+00  3.051e-05  1.539e+00  4.476e+00  1.543e+00
## nationUS     -1.572e+00  2.231e-05  1.520e+00  1.543e+00  1.946e+00
```

`hist.boot` *Methods Functions to Support* `boot` *Objects*

Description

The `Boot` function in the **car** package uses the `boot` function from the **boot** package to do a straightforward case or residual bootstrap for many regression objects. These are method functions for standard generics to summarize the results of the bootstrap. Other tools for this purpose are available in the `boot` package.

Usage

```
## S3 method for class 'boot'
hist(x, parm, layout = NULL, ask, main = "", freq = FALSE,
    estPoint = TRUE, point.col = carPalette()[1], point.lty = 2, point.lwd = 2,
```

```
    estDensity = !freq, den.col = carPalette()[2], den.lty = 1, den.lwd = 2,
    estNormal = !freq, nor.col = carPalette()[3], nor.lty = 2, nor.lwd = 2,
    ci = c("bca", "none", "perc", "norm"), level = 0.95,
    legend = c("top", "none", "separate"), box = TRUE, ...)

## S3 method for class 'boot'
summary(object, parm, high.moments = FALSE, extremes = FALSE, ...)

## S3 method for class 'boot'
confint(object, parm, level = 0.95, type = c("bca", "norm",
    "basic", "perc"), ...)

## S3 method for class 'boot'
Confint(object, parm, level = 0.95, type = c("bca", "norm",
    "basic", "perc"), ...)

## S3 method for class 'boot'
vcov(object, use="complete.obs", ...)
```

Arguments

x, object	An object created by a call to boot in the boot package, or to Boot in the **car** package of class "boot".
parm	A vector of numbers or coefficient names giving the coefficients for which a histogram or confidence interval is desired. If numbers are used, 1 corresponds to the intercept, if any. The default is all coefficients.
layout	If set to a value like c(1, 1) or c(4, 3), the layout of the graph will have this many rows and columns. If not set, the program will select an appropriate layout. If the number of graphs exceed nine, you must select the layout yourself, or you will get a maximum of nine per page. If layout=NA, the function does not set the layout and the user can use the par function to control the layout, for example to have plots from two models in the same graphics window.
ask	If TRUE, ask the user before drawing the next plot; if FALSE, don't ask.
main	Main title for the graphs. The default is main="" for no title.
freq	The default for the generic hist function is freq=TRUE to give a frequency histogram. The default for hist.boot is freq=FALSE to give a density histogram. A density estimate and/or a fitted normal density can be added to the graph if freq=FALSE but not if freq=TRUE.
estPoint, point.col, point.lty, point.lwd	If estPoint=TRUE, the default, a vertical line is drawn on the histgram at the value of the point estimate computed from the complete data. The remaining three optional arguments set the color, line type and line width of the line that is drawn.
estDensity, den.col, den.lty, den.lwd	If estDensity=TRUE andfreq=FALSE, the default, a kernel density estimate is drawn on the plot with a call to the density function with no additional arguments. The remaining three optional arguments set the color, line type and line width of the lines that are drawn.

estNormal, nor.col, nor.lty, nor.lwd
: If estNormal=TRUE andfreq=FALSE, the default, a normal density with mean and sd computed from the data is drawn on the plot. The remaining three optional arguments set the color, line type and line width of the lines that are drawn.

ci
: A confidence interval based on the bootstrap will be added to the histogram using the BCa method if ci="bca" the percentile method if ci="perc", or the normal method if ci="norm". No interval is drawn if ci="none". The default is "bca". The interval is indicated by a thick horizontal line at y=0. For some bootstraps the BCa method is unavailable, in which case a warning is issued and ci="perc" is substituted. If you wish to see all the options at once, see boot.ci. The normal method is computed as the (estimate from the original data) minus the bootstrap bias plus or minus the standard deviation of the bootstrap replicates times the appropriate quantile of the standard normal distribution.

legend
: A legend can be added to the (array of) histograms. The value "top" puts at the top-left of the plots. The value "separate" puts the legend in its own graph following all the histograms. The value "none" suppresses the legend.

box
: Add a box around each histogram.

...
: Additional arguments passed to hist; for other methods this is included for compatibility with the generic method. For example, the argument border=par()$bg in hist will draw the histogram transparently, leaving only the density estimates. With the vcov function, the additional arguments are passed to cov. See the Value section, below.

high.moments
: Should the skewness and kurtosis be included in the summary? Default is FALSE.

extremes
: Should the minimum, maximum and range be included in the summary? Default is FALSE.

level
: Confidence level, a number between 0 and 1. In confint, level can be a vector; for example level=c(.50, .90, .95) will return the following estimated quantiles: c(.025, .05, .25, .75, .95, .975).

type
: Selects the confidence interval type. The types implemented are the "percentile" method, which uses the function quantile to return the appropriate quantiles for the confidence limit specified, the default bca which uses the bias-corrected and accelerated method presented by Efron and Tibshirani (1993, Chapter 14). For the other types, see the documentation for boot.

use
: The default use="complete.obs" for vcov computes a bootstrap covariance matrix by deleting bootstraps that returned NAs. Setting use to anything else will result in a matrix of NAs.

Value

hist is used for the side-effect of drawing an array of historgams of each column of the first argument. summary returns a matrix of summary statistics for each of the columns in the bootstrap object. The confint method returns confidence intervals. Confint appends the estimates based on the original fitted model to the left of the confidence intervals.

The function vcov returns the sample covariance of the bootstrap sample estimates, by default skipping any bootstrap samples that returned NA.

Author(s)

Sanford Weisberg, <sandy@umn.edu>

References

Efron, B. and Tibsharini, R. (1993) *An Introduction to the Bootstrap*. New York: Chapman and Hall.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition. Thousand Oaks: Sage.

Fox, J. and Weisberg, S. (2018) *Bootstrapping Regression Models in R*, https://www.john-fox.ca/Companion/appendices/Appendix-Bootstrapping.pdf.

Weisberg, S. (2013) *Applied Linear Regression*, Fourth Edition, Wiley

See Also

See Also Boot, hist, density, Fox and Weisberg (2017), cited above

Examples

```
m1 <- lm(Fertility ~ ., swiss)
betahat.boot <- Boot(m1, R=99) # 99 bootstrap samples--too small to be useful
summary(betahat.boot)  # default summary
confint(betahat.boot)
hist(betahat.boot)
```

Import *Import data from many file formats*

Description

Uses the `import` function from the **rio** package to read a data.frame from a variety of file types. The `Import` function includes 2 additional arguments adding row names and for converting character and logical variables to factors for some file types.

Usage

```
Import(file, format, ..., row.names=TRUE,
       stringsAsFactors = FALSE)
```

Arguments

`file`	A character string naming a file, URL, or .zip or .tar archive. See the details below. If the file name has an extension like `.xlsx` or `.csv` then the type of file is inferred from the extension.
`format`	If an extension is not present in the file name or it is wrong, the file format can be set with this argument; see `import`.

`...` Additional arguments passed to `import`.

`row.names` If `TRUE`, the default, the left-most character variable that has all unique elements is removed from the data frame and set to be `row.names`. To match `import`, set `row.names=FALSE`.

`stringsAsFactors`
If `TRUE`, then character variables that do not have all unique elements are converted to factors. The default is `FALSE`. Prior to May 2020 the default was determined by `getOption("stringsAsFactors")`, which then defaulted to `TRUE`. This option is `FALSE` in R 4.0.0 and has been deprecated.

Details

This function calls the `import` function to read a data frame from a file. Many file types are supported. For files of type `"txt"`, `"csv"`, `"xlsx"`, `"xls"` or `"ods"` the arguments `row.names` and `stringsAsFactors` can be used to add row names and convert character variables to factors, respectively. Many more details are given on the man page for `import`.

Value

A data frame. See `import` for more details

Author(s)

Sanford Weisberg <sandy@umn.edu>

See Also

`import`, `Export`, `strings2factors`

Examples

```
if(require("rio")) {

head(Duncan, 3) # first three rows
Export(Duncan, "Duncan.csv", keep.row.names="occupation")
Duncan2 <- Import("Duncan.csv") # Automatically restores row.names and factors
brief(Duncan2)
identical(Duncan, Duncan2) # FALSE because type is of a different class
Duncan3 <- Import("Duncan.csv", stringsAsFactors=TRUE)
brief(Duncan3)
identical(Duncan, Duncan3) # TRUE type is of same class
# cleanup
unlink("Duncan.csv")

}
```

infIndexPlot *Influence Index Plot*

Description

Provides index plots of influence and related diagnostics for a regression model.

Usage

```
infIndexPlot(model, ...)

influenceIndexPlot(model, ...)

## S3 method for class 'lm'
infIndexPlot(model, vars=c("Cook", "Studentized", "Bonf", "hat"),
    id=TRUE, grid=TRUE, main="Diagnostic Plots", ...)

## S3 method for class 'influence.merMod'
infIndexPlot(model,
    vars = c("dfbeta", "dfbetas", "var.cov.comps",
    "cookd"), id = TRUE, grid = TRUE, main = "Diagnostic Plots", ...)
## S3 method for class 'influence.lme'
infIndexPlot(model,
    vars = c("dfbeta", "dfbetas", "var.cov.comps",
    "cookd"), id = TRUE, grid = TRUE, main = "Diagnostic Plots", ...)
```

Arguments

model	A regression object of class lm, glm, or lmerMod, or an influence object for a lmer, glmer, or lme object (see influence.mixed.models). The "lmerMod" method calls the "lm" method and can take the same arguments.
vars	All the quantities listed in this argument are plotted. Use "Cook" for Cook's distances, "Studentized" for Studentized residuals, "Bonf" for Bonferroni p-values for an outlier test, and and "hat" for hat-values (or leverages) for a linear or generalized linear model, or "dfbeta", "dfbetas", "var.cov.comps", and "cookd" for an influence object derived from a mixed model. Capitalization is optional. All but "dfbeta" and "dfbetas" may be abbreviated by the first one or more letters.
main	main title for graph
id	a list of named values controlling point labelling. The default, TRUE, is equivalent to id=list(method="y", n=2, cex=1, col=carPalette()[1], location="lr"); FALSE suppresses point labelling. See showLabels for details.
grid	If TRUE, the default, a light-gray background grid is put on the graph.
...	Arguments passed to plot

Value

Used for its side effect of producing a graph. Produces index plots of diagnostic quantities.

Author(s)

Sanford Weisberg <sandy@umn.edu> and John Fox

References

Cook, R. D. and Weisberg, S. (1999) *Applied Regression, Including Computing and Graphics.* Wiley.

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.
Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Weisberg, S. (2014) *Applied Linear Regression*, Fourth Edition, Wiley.

See Also

cooks.distance, rstudent, outlierTest, hatvalues, influence.mixed.models.

Examples

```
influenceIndexPlot(lm(prestige ~ income + education + type, Duncan))

## Not run:  # a little slow
  if (require(lme4)){
      print(fm1 <- lmer(Reaction ~ Days + (Days | Subject),
          sleepstudy)) # from ?lmer
      infIndexPlot(influence(fm1, "Subject"))
      infIndexPlot(influence(fm1))
      }

  if (require(lme4)){
      gm1 <- glmer(cbind(incidence, size - incidence) ~ period + (1 | herd),
          data = cbpp, family = binomial) # from ?glmer
      infIndexPlot(influence(gm1, "herd", maxfun=100))
      infIndexPlot(influence(gm1, maxfun=100))
      gm1.11 <- update(gm1, subset = herd != 11) # check deleting herd 11
      compareCoefs(gm1, gm1.11)
      }

## End(Not run)
```

---

influence.mixed.models

Influence Diagnostics for Mixed-Effects Models

Description

These functions compute deletion influence diagnostics for linear mixed-effects models fit by `lme` in the **nlme** package. The main function is a method for the `influence` generic function. Other functions are provided for computing `dfbeta`, `dfbetas`, `cooks.distance`, and influence on variance-covariance components based on the object computed by `influence.lme`.

Usage

```
## S3 method for class 'lme'
influence(model, groups, data, ncores=1, ...)

## S3 method for class 'influence.lme'
cooks.distance(model, ...)
## S3 method for class 'influence.lme'
dfbeta(model, which = c("fixed", "var.cov"), ...)
## S3 method for class 'influence.lme'
dfbetas(model, ...)
```

Arguments

`model`	in the case `influence`, a model of class `"lme"`; in the case of `cooks.distance`, `dfbeta`, or `dfbetas`, an object returned by `influence.lme`.
`groups`	a character vector containing the name of a grouping factor or names of grouping factors; if more than one name is supplied, then groups are defined by all combinations of levels of the grouping factors that appear in the data. If omitted, then each individual row of the data matrix is treated as a "group" to be deleted in turn.
`data`	an optional data frame with the data to which `model` was fit; `influence.lme` can access the data unless `keep.data=FALSE` was specified in the call to `lme`, so it's usually unnecessary to supply the `data` argument.
`ncores`	number of cores for parallel computation of diagnostics; if `1` (the default), the computation isn't parallelized; if `Inf`, all of the available *physical* cores (not necessarily *logical* cores — see `detectCores`) on the computer will be used.
`which`	if `"fixed.effects"` (the default), return influence on the fixed effects; if `"var.cov"`, return influence on the variance-covariance components.
`...`	ignored.

Details

`influence.lme` starts with the estimated variance-covariance components from `model` and then refits the model omitting each group in turn.

The other functions are methods for the `dfbeta`, `dfbetas`, and `cooks.distance` generics, to be applied to the `"influence.lme"` object produced by the `influence` function; the `dfbeta` methods can also return influence on the variance-covariance components.

Value

`influence.lme` returns an object of class `"influence.lme"`, which contains the following elements:

`"fixed.effects"` the estimated fixed effects for the model.

`"fixed.effects[-groups]"` a matrix with columns corresponding to the fixed-effects coefficients and rows corresponding to groups, giving the estimated fixed effects with each group deleted in turn; *groups* is formed from the name(s) of the grouping factor(s).

`"var.cov.comps"` the estimated variance-covariance parameters for the model.

`"var.cov.comps[-groups]"` a matrix with the estimated covariance parameters (in columns) with each group deleted in turn.

`"vcov"` The estimated covariance matrix of the fixed-effects coefficients.

`"vcov[-groups]"` a list each of whose elements is the estimated covariance matrix of the fixed-effects coefficients with one group deleted.

`"groups"` a character vector giving the names of the grouping factors.

`"deleted"` the possibly composite grouping factor, each of whose elements is deleted in turn.

For plotting `"influence.lme"` objects, see `infIndexPlot`.

Author(s)

J. Fox <jfox@mcmaster.ca>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

`lme`, `infIndexPlot`.

Examples

```
if (require(nlme)){
    print(fm1 <- lme(distance ~ age, data = Orthodont))
    infIndexPlot(influence(fm1, "Subject"))

  # a little slow:
    ## Not run:
    infIndexPlot(influence(fm1))

## End(Not run)

    }
```

`influencePlot` *Regression Influence Plot*

Description

This function creates a "bubble" plot of Studentized residuals versus hat values, with the areas of the circles representing the observations proportional to the value Cook's distance. Vertical reference lines are drawn at twice and three times the average hat value, horizontal reference lines at -2, 0, and 2 on the Studentized-residual scale.

Usage

```
influencePlot(model, ...)

## S3 method for class 'lm'
influencePlot(model, scale=10,
 xlab="Hat-Values", ylab="Studentized Residuals", id=TRUE,
 fill=TRUE, fill.col=carPalette()[2], fill.alpha=0.5, ...)

## S3 method for class 'lmerMod'
influencePlot(model, ...)
```

Arguments

`model`	a linear, generalized-linear, or linear mixed model; the `"lmerMod"` method calls the `"lm"` method and can take the same arguments.
`scale`	a factor to adjust the size of the circles.
`xlab, ylab`	axis labels.
`id`	settings for labelling points; see `link{showLabels}` for details. To omit point labelling, set `id=FALSE`; the default, `id=TRUE` is equivalent to `id=list(method="noteworthy", n=2, cex=1, col=carPalette()[1], location="lr")`. The default `method="noteworthy"` is used only in this function and indicates setting labels for points with large Studentized residuals, hat-values or Cook's distances. Set `id=list(method="identify")` for interactive point identification.
`fill`	if `TRUE` (the default) fill the circles, with the opacity of the filled color proportional to Cook's D, using the `alpha` function in the **scales** package to compute the opacity of the fill.
`fill.col`	color to use for the filled points, taken by default from the second element of the `carPalette` color palette.
`fill.alpha`	the maximum alpha (opacity) of the points.
`...`	arguments to pass to the `plot` and `points` functions.

Value

If points are identified, returns a data frame with the hat values, Studentized residuals and Cook's distance of the identified points. If no points are identified, nothing is returned. This function is primarily used for its side-effect of drawing a plot.

Author(s)

John Fox <jfox@mcmaster.ca>, minor changes by S. Weisberg <sandy@umn.edu> and a contribution from Michael Friendly

References

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

cooks.distance, rstudent, alpha, carPalette, hatvalues, showLabels

Examples

```
influencePlot(lm(prestige ~ income + education, data=Duncan))
## Not run:  # requires user interaction to identify points
influencePlot(lm(prestige ~ income + education, data=Duncan),
    id=list(method="identify"))

## End(Not run)
```

---

invResPlot *Inverse Response Plots to Transform the Response*

Description

For a lm model, draws an inverse.response plot with the response Y on the vertical axis and the fitted values $\hat{Y}$ on the horizontal axis. Uses nls to estimate λ in the function $\hat{Y} = b_0 + b_1 Y^{\lambda}$. Adds the fitted curve to the plot. invResPlot is an alias for inverseResponsePlot.

Usage

```
inverseResponsePlot(model, lambda=c(-1,0,1), robust=FALSE, xlab=NULL, ...)

## S3 method for class 'lm'
inverseResponsePlot(model, lambda=c(-1, 0, 1),
    robust=FALSE, xlab=NULL, id=FALSE, ...)

invResPlot(model, ...)
```

Arguments

model	A "lm" regression object.
lambda	A vector of values for lambda. A plot will be produced with curves corresponding to these lambdas and to the nonlinear least squares estimate of lambda.

robust	If TRUE, then estimation uses Huber M-estimates with the median absolute deviation to estimate scale and k= 1.345. The default is FALSE.
xlab	The horizontal axis label. If NULL, it is constructed by the function.
id	controls point identification; if FALSE (the default), no points are identified; can be a list of named arguments to the showLabels function; TRUE is equivalent to list(method=list(method="x", n=2, cex=1, col=carPalette()[1], location="lr"), which identifies the 2 points with the most extreme horizontal (X) values.
...	Other arguments passed to invTranPlot and then to plot.

Value

As a side effect, a plot is produced with the response on the horizontal axis and fitted values on the vertical axis. Several lines are added to be plot as the ols estimates of the regression of $\hat{Y}$ on Y^{λ}, interpreting $\lambda = 0$ to be natural logarithms.

Numeric output is a list with elements

lambda	Estimate of transformation parameter for the response
RSS	The residual sum of squares at the minimum if robust=FALSE. If robust = TRUE, the value of Huber objective function is returned.

Author(s)

Sanford Weisberg, sandy@umn.edu

References

Fox, J. and Weisberg, S. (2011) *An R Companion to Applied Regression*, Second Edition, Sage.

Prendergast, L. A., & Sheather, S. J. (2013) On sensitivity of inverse response plot estimation and the benefits of a robust estimation approach. *Scandinavian Journal of Statistics*, 40(2), 219-237.

Weisberg, S. (2014) *Applied Linear Regression*, Fourth Edition, Wiley, Chapter 7.

See Also

invTranPlot, powerTransform, showLabels

Examples

```
m2 <- lm(rate ~ log(len) + log(adt) + slim + shld + log(sigs1), Highway1)
invResPlot(m2)
```

`invTranPlot` *Choose a Predictor Transformation Visually or Numerically*

Description

`invTranPlot` draws a two-dimensional scatterplot of Y versus X, along with the OLS fit from the regression of Y on $(X^\lambda - 1)/\lambda$. `invTranEstimate` finds the nonlinear least squares estimate of λ and its standard error.

Usage

```
invTranPlot(x, ...)

## S3 method for class 'formula'
invTranPlot(x, data, subset, na.action, id=FALSE, ...)

## Default S3 method:
invTranPlot(x, y, lambda=c(-1, 0, 1), robust=FALSE,
        lty.lines=rep(c("solid", "dashed", "dotdash", "longdash", "twodash"),
        length=1 + length(lambda)), lwd.lines=2,
        col=carPalette()[1], col.lines=carPalette(),
        xlab=deparse(substitute(x)), ylab=deparse(substitute(y)),
        family="bcPower", optimal=TRUE, key="auto", id=FALSE,
        grid=TRUE, ...)

invTranEstimate(x, y, family="bcPower", confidence=0.95, robust=FALSE)
```

Arguments

`x`	The predictor variable, or a formula with a single response and a single predictor
`y`	The response variable
`data`	An optional data frame to get the data for the formula
`subset`	Optional, as in `lm`, select a subset of the cases
`na.action`	Optional, as in `lm`, the action for missing data
`lambda`	The powers used in the plot. The optimal power than minimizes the residual sum of squares is always added unless optimal is `FALSE`.
`robust`	If `TRUE`, then the estimated transformation is computed using Huber M-estimation with the MAD used to estimate scale and k=1.345. The default is `FALSE`.
`family`	The transformation family to use, `"bcPower"`, `"yjPower"`, or a user-defined family.
`confidence`	returns a profile likelihood confidence interval for the optimal transformation with this confidence level. If `FALSE`, or if `robust=TRUE`, no interval is returned.
`optimal`	Include the optimal value of lambda?
`lty.lines`	line types corresponding to the powers

lwd.lines	the width of the plotted lines, defaults to 2 times the standard
col	color(s) of the points in the plot. If you wish to distinguish points according to the levels of a factor, we recommend using symbols, specified with the pch argument, rather than colors.
col.lines	color of the fitted lines corresponding to the powers. The default is to use the colors returned by carPalette
key	The default is "auto", in which case a legend is added to the plot, either above the top marign or in the bottom right or top right corner. Set to NULL to suppress the legend.
xlab	Label for the horizontal axis.
ylab	Label for the vertical axis.
id	controls point identification; if FALSE (the default), no points are identified; can be a list of named arguments to the showLabels function; TRUE is equivalent to list(method=list(method="x", n=2, cex=1, col=carPalette()[1], location="lr"), which identifies the 2 points with the most extreme horizontal values — i.e., the response variable in the model.
...	Additional arguments passed to the plot method, such as pch.
grid	If TRUE, the default, a light-gray background grid is put on the graph

Value

invTranPlot plots a graph and returns a data frame with λ in the first column, and the residual sum of squares from the regression for that λ in the second column.

invTranEstimate returns a list with elements lambda for the estimate, se for its standard error, and RSS, the minimum value of the residual sum of squares.

Author(s)

Sanford Weisberg, <sandy@umn.edu>

References

Fox, J. and Weisberg, S. (2011) *An R Companion to Applied Regression*, Second Edition, Sage.

Prendergast, L. A., & Sheather, S. J. (2013) On sensitivity of inverse response plot estimation and the benefits of a robust estimation approach. *Scandinavian Journal of Statistics*, 40(2), 219-237.

Weisberg, S. (2014) *Applied Linear Regression*, Fourth Edition, Wiley, Chapter 7.

See Also

inverseResponsePlot,optimize

Examples

```
with(UN, invTranPlot(ppgdp, infantMortality))
with(UN, invTranEstimate(ppgdp, infantMortality))
```

leveneTest *Levene's Test*

Description

Computes Levene's test for homogeneity of variance across groups.

Usage

```
leveneTest(y, ...)
## S3 method for class 'formula'
leveneTest(y, data, ...)
## S3 method for class 'lm'
leveneTest(y, ...)
## Default S3 method:
leveneTest(y, group, center=median, ...)
```

Arguments

y	response variable for the default method, or a lm or formula object. If y is a linear-model object or a formula, the variables on the right-hand-side of the model must all be factors and must be completely crossed.
group	factor defining groups.
center	The name of a function to compute the center of each group; mean gives the original Levene's test; the default, median, provides a more robust test.
data	a data frame for evaluating the formula.
...	arguments to be passed down, e.g., data for the formula and lm methods; can also be used to pass arguments to the function given by center (e.g., center=mean and trim=0.1 specify the 10% trimmed mean).

Value

returns an object meant to be printed showing the results of the test.

Note

adapted from a response posted by Brian Ripley to the r-help email list.

Author(s)

John Fox <jfox@mcmaster.ca>; original generic version contributed by Derek Ogle

References

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Examples

```
with(Moore, leveneTest(conformity, fcategory))
with(Moore, leveneTest(conformity, interaction(fcategory, partner.status)))
leveneTest(conformity ~ fcategory*partner.status, data=Moore)
leveneTest(lm(conformity ~ fcategory*partner.status, data=Moore))
leveneTest(conformity ~ fcategory*partner.status, data=Moore, center=mean)
leveneTest(conformity ~ fcategory*partner.status, data=Moore, center=mean, trim=0.1)
```

---

leveragePlots *Regression Leverage Plots*

Description

These functions display a generalization, due to Sall (1990) and Cook and Weisberg (1991), of added-variable plots to multiple-df terms in a linear model. When a term has just 1 df, the leverage plot is a rescaled version of the usual added-variable (partial-regression) plot.

Usage

```
leveragePlots(model, terms = ~., layout = NULL, ask,
    main, ...)

leveragePlot(model, ...)

## S3 method for class 'lm'
leveragePlot(model, term.name,
  id=TRUE, col=carPalette()[1], col.lines=carPalette()[2], lwd=2,
  xlab, ylab, main="Leverage Plot", grid=TRUE, ...)

## S3 method for class 'glm'
leveragePlot(model, ...)
```

Arguments

`model`	model object produced by `lm`
`terms`	A one-sided formula that specifies a subset of the numeric regressors, factors and interactions. One added-variable plot is drawn for each term, either a main effect or an interactions. The default `~.` is to plot against all terms in the model. For example, the specification `terms = ~ . - X3` would plot against all predictors except for `X3`. If this argument is a quoted name of one of the predictors, the added-variable plot is drawn for that predictor only. The plots for main effects with interactions present violate the marginality principle and may not be easily interpreted.
`layout`	If set to a value like `c(1, 1)` or `c(4, 3)`, the layout of the graph will have this many rows and columns. If not set, the program will select an appropriate layout. If the number of graphs exceed nine, you must select the layout yourself, or you

	will get a maximum of nine per page. If `layout=NA`, the function does not set the layout and the user can use the `par` function to control the layout, for example to have plots from two models in the same graphics window.
`ask`	if `TRUE`, a menu is provided in the R Console for the user to select the term(s) to plot.
`xlab, ylab`	axis labels; if missing, labels will be supplied.
`main`	title for plot; if missing, a title will be supplied.
`...`	arguments passed down to method functions.
`term.name`	Quoted name of term in the model to be plotted; this argument is omitted for `leveragePlots`.
`id`	controls point identification; if `FALSE`, no points are identified; can be a list of named arguments to the `showLabels` function; `TRUE`, the default, is equivalent to `list(method=list(abs(residuals(model, type="pearson")), "x"), n=2, cex=1, col=carPalette()[1], location="lr")`, which identifies the 2 points with the largest residuals and the 2 points with the greatest partial leverage.
`col`	color(s) of points
`col.lines`	color of the fitted line
`lwd`	line width; default is 2 (see `par`).
`grid`	If TRUE, the default, a light-gray background grid is put on the graph

Details

The function intended for direct use is `leveragePlots`.

The model can contain factors and interactions. A leverage plot can be drawn for each term in the model, including the constant.

`leveragePlot.glm` is a dummy function, which generates an error message.

Value

`NULL`. These functions are used for their side effect: producing plots.

Author(s)

John Fox `<jfox@mcmaster.ca>`

References

Cook, R. D. and Weisberg, S. (1991). Added Variable Plots in Linear Regression. In Stahel, W. and Weisberg, S. (eds.), *Directions in Robust Statistics and Diagnostics*. Springer, 47-60.

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Sall, J. (1990) Leverage plots for general linear hypotheses. *American Statistician* **44**, 308–315.

See Also

`avPlots`

Examples

```
leveragePlots(lm(prestige~(income+education)*type, data=Duncan))
```

`linearHypothesis` *Test Linear Hypothesis*

Description

Generic function for testing a linear hypothesis, and methods for linear models, generalized linear models, multivariate linear models, linear and generalized linear mixed-effects models, generalized linear models fit with `svyglm` in the **survey** package, robust linear models fit with `rlm` in the **MASS** package, and other models that have methods for `coef` and `vcov`. For mixed-effects models, the tests are Wald chi-square tests for the fixed effects.

Usage

```
linearHypothesis(model, ...)

lht(model, ...)

## Default S3 method:
linearHypothesis(model, hypothesis.matrix, rhs=NULL,
  test=c("Chisq", "F"), vcov.=NULL, singular.ok=FALSE, verbose=FALSE,
    coef. = coef(model), suppress.vcov.msg=FALSE, error.df, ...)

## S3 method for class 'lm'
linearHypothesis(model, hypothesis.matrix, rhs=NULL,
    test=c("F", "Chisq"), vcov.=NULL,
 white.adjust=c(FALSE, TRUE, "hc3", "hc0", "hc1", "hc2", "hc4"),
 singular.ok=FALSE, ...)

## S3 method for class 'glm'
linearHypothesis(model,  ...)

## S3 method for class 'lmList'
linearHypothesis(model,  ..., vcov.=vcov, coef.=coef)

## S3 method for class 'nlsList'
linearHypothesis(model,  ..., vcov.=vcov, coef.=coef)

## S3 method for class 'mlm'
linearHypothesis(model, hypothesis.matrix, rhs=NULL, SSPE, V,
    test, idata, icontrasts=c("contr.sum", "contr.poly"), idesign, iterms,
   check.imatrix=TRUE, P=NULL, title="", singular.ok=FALSE, verbose=FALSE, ...)

## S3 method for class 'polr'
```

```
linearHypothesis(model, hypothesis.matrix, rhs=NULL, vcov.,
 verbose=FALSE, ...)

## S3 method for class 'linearHypothesis.mlm'
print(x, SSP=TRUE, SSPE=SSP,
    digits=getOption("digits"), ...)

## S3 method for class 'lme'
linearHypothesis(model, hypothesis.matrix, rhs=NULL,
  vcov.=NULL, singular.ok=FALSE, verbose=FALSE, ...)

## S3 method for class 'mer'
linearHypothesis(model, hypothesis.matrix, rhs=NULL,
  vcov.=NULL, test=c("Chisq", "F"), singular.ok=FALSE, verbose=FALSE, ...)

## S3 method for class 'merMod'
linearHypothesis(model, hypothesis.matrix, rhs=NULL,
     vcov.=NULL, test=c("Chisq", "F"), singular.ok=FALSE, verbose=FALSE, ...)

## S3 method for class 'svyglm'
linearHypothesis(model, ...)

## S3 method for class 'rlm'
linearHypothesis(model, ...)

## S3 method for class 'survreg'
linearHypothesis(model, hypothesis.matrix, rhs=NULL,
  test=c("Chisq", "F"), vcov., verbose=FALSE, ...)

matchCoefs(model, pattern, ...)

## Default S3 method:
matchCoefs(model, pattern, coef.=coef, ...)

## S3 method for class 'lme'
matchCoefs(model, pattern, ...)

## S3 method for class 'mer'
matchCoefs(model, pattern, ...)

## S3 method for class 'merMod'
matchCoefs(model, pattern, ...)

## S3 method for class 'mlm'
matchCoefs(model, pattern, ...)

## S3 method for class 'lmList'
matchCoefs(model, pattern, ...)
```

Arguments

model	fitted model object. The default method of linearHypothesis works for models for which the estimated parameters can be retrieved by coef and the corresponding estimated covariance matrix by vcov. See the *Details* for more information.
hypothesis.matrix	matrix (or vector) giving linear combinations of coefficients by rows, or a character vector giving the hypothesis in symbolic form (see *Details*).
rhs	right-hand-side vector for hypothesis, with as many entries as rows in the hypothesis matrix; can be omitted, in which case it defaults to a vector of zeroes. For a multivariate linear model, rhs is a matrix, defaulting to 0. This argument isn't available for F-tests for linear mixed models.
singular.ok	if FALSE (the default), a model with aliased coefficients produces an error; if TRUE, the aliased coefficients are ignored, and the hypothesis matrix should not have columns for them. For a multivariate linear model: will return the hypothesis and error SSP matrices even if the latter is singular; useful for computing univariate repeated-measures ANOVAs where there are fewer subjects than df for within-subject effects.
error.df	For the default linearHypothesis method, if an F-test is requested and if error.df is missing, the error degrees of freedom will be computed by applying the df.residual function to the model; if df.residual returns NULL or NA, then a chi-square test will be substituted for the F-test (with a message to that effect.
idata	an optional data frame giving a factor or factors defining the intra-subject model for multivariate repeated-measures data. See *Details* for an explanation of the intra-subject design and for further explanation of the other arguments relating to intra-subject factors.
icontrasts	names of contrast-generating functions to be applied by default to factors and ordered factors, respectively, in the within-subject "data"; the contrasts must produce an intra-subject model matrix in which different terms are orthogonal.
idesign	a one-sided model formula using the "data" in idata and specifying the intra-subject design.
iterms	the quoted name of a term, or a vector of quoted names of terms, in the intra-subject design to be tested.
check.imatrix	check that columns of the intra-subject model matrix for different terms are mutually orthogonal (default, TRUE). Set to FALSE only if you have *already* checked that the intra-subject model matrix is block-orthogonal.
P	transformation matrix to be applied to the repeated measures in multivariate repeated-measures data; if NULL *and* no intra-subject model is specified, no response-transformation is applied; if an intra-subject model is specified via the idata, idesign, and (optionally) icontrasts arguments, then P is generated automatically from the iterms argument.
SSPE	in linearHypothesis method for mlm objects: optional error sum-of-squares-and-products matrix; if missing, it is computed from the model. In print method for linearHypothesis.mlm objects: if TRUE, print the sum-of-squares and cross-products matrix for error.

test	character string, "F" or "Chisq", specifying whether to compute the finite-sample F statistic (with approximate F distribution) or the large-sample Chi-squared statistic (with asymptotic Chi-squared distribution). For a multivariate linear model, the multivariate test statistic to report — one or more of "Pillai", "Wilks", "Hotelling-Lawley", or "Roy", with "Pillai" as the default.
title	an optional character string to label the output.
V	inverse of sum of squares and products of the model matrix; if missing it is computed from the model.
vcov.	a function for estimating the covariance matrix of the regression coefficients, e.g., hccm, or an estimated covariance matrix for model. See also white.adjust. For the "lmList" and "nlsList" methods, vcov. must be a function (defaulting to vcov) to be applied to each model in the list. Note that arguments supplied to ... are *not* passed to vcov. when it's a function; in this case either use an anonymous function in which the additional arguments are set, or supply the coefficient covariance matrix directly (see the examples).
coef.	a vector of coefficient estimates. The default is to get the coefficient estimates from the model argument, but the user can input any vector of the correct length. For the "lmList" and "nlsList" methods, coef. must be a function (defaulting to coef) to be applied to each model in the list.
white.adjust	logical or character. Convenience interface to hccm (instead of using the argument vcov.). Can be set either to a character value specifying the type argument of hccm or TRUE, in which case "hc3" is used implicitly. The default is FALSE.
verbose	If TRUE, the hypothesis matrix, right-hand-side vector (or matrix), and estimated value of the hypothesis are printed to standard output; if FALSE (the default), the hypothesis is only printed in symbolic form and the value of the hypothesis is not printed.
x	an object produced by linearHypothesis.mlm.
SSP	if TRUE (the default), print the sum-of-squares and cross-products matrix for the hypothesis and the response-transformation matrix.
digits	minimum number of signficiant digits to print.
pattern	a regular expression to be matched against coefficient names.
suppress.vcov.msg	for internal use by methods that call the default method.
...	arguments to pass down.

Details

linearHypothesis computes either a finite-sample F statistic or asymptotic Chi-squared statistic for carrying out a Wald-test-based comparison between a model and a linearly restricted model. The default method will work with any model object for which the coefficient vector can be retrieved by coef and the coefficient-covariance matrix by vcov (otherwise the argument vcov. has to be set explicitly). For computing the F statistic (but not the Chi-squared statistic) a df.residual method needs to be available. If a formula method exists, it is used for pretty printing.

The method for "lm" objects calls the default method, but it changes the default test to "F", supports the convenience argument white.adjust (for backwards compatibility), and enhances the output by the residual sums of squares. For "glm" objects just the default method is called (bypassing the "lm" method). The "svyglm" method also calls the default method.

Multinomial logit models fit by the multinom function in the **nnet** package invoke the default method, and the coefficient names are composed from the response-level names and conventional coefficient names, separated by a period (".") : see one of the examples below.

The function lht also dispatches to linearHypothesis.

The hypothesis matrix can be supplied as a numeric matrix (or vector), the rows of which specify linear combinations of the model coefficients, which are tested equal to the corresponding entries in the right-hand-side vector, which defaults to a vector of zeroes.

Alternatively, the hypothesis can be specified symbolically as a character vector with one or more elements, each of which gives either a linear combination of coefficients, or a linear equation in the coefficients (i.e., with both a left and right side separated by an equals sign). Components of a linear expression or linear equation can consist of numeric constants, or numeric constants multiplying coefficient names (in which case the number precedes the coefficient, and may be separated from it by spaces or an asterisk); constants of 1 or -1 may be omitted. Spaces are always optional. Components are separated by plus or minus signs. Newlines or tabs in hypotheses will be treated as spaces. See the examples below.

If the user sets the arguments coef. and vcov., then the computations are done without reference to the model argument. This is like assuming that coef. is normally distibuted with estimated variance vcov. and the linearHypothesis will compute tests on the mean vector for coef., without actually using the model argument.

A linear hypothesis for a multivariate linear model (i.e., an object of class "mlm") can optionally include an intra-subject transformation matrix for a repeated-measures design. If the intra-subject transformation is absent (the default), the multivariate test concerns all of the corresponding coefficients for the response variables. There are two ways to specify the transformation matrix for the repeated measures:

1. The transformation matrix can be specified directly via the P argument.
2. A data frame can be provided defining the repeated-measures factor or factors via idata, with default contrasts given by the icontrasts argument. An intra-subject model-matrix is generated from the one-sided formula specified by the idesign argument; columns of the model matrix corresponding to different terms in the intra-subject model must be orthogonal (as is insured by the default contrasts). Note that the contrasts given in icontrasts can be overridden by assigning specific contrasts to the factors in idata. The repeated-measures transformation matrix consists of the columns of the intra-subject model matrix corresponding to the term or terms in iterms. In most instances, this will be the simpler approach, and indeed, most tests of interests can be generated automatically via the Anova function.

matchCoefs is a convenience function that can sometimes help in formulating hypotheses; for example matchCoefs(mod, ":") will return the names of all interaction coefficients in the model mod.

Value

For a univariate model, an object of class "anova" which contains the residual degrees of freedom in the model, the difference in degrees of freedom, Wald statistic (either "F" or "Chisq"), and

corresponding p value. The value of the linear hypothesis and its covariance matrix are returned respectively as "value" and "vcov" attributes of the object (but not printed).

For a multivariate linear model, an object of class "linearHypothesis.mlm", which contains sums-of-squares-and-product matrices for the hypothesis and for error, degrees of freedom for the hypothesis and error, and some other information.

The returned object normally would be printed.

Author(s)

Achim Zeileis and John Fox <jfox@mcmaster.ca>

References

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Hand, D. J., and Taylor, C. C. (1987) *Multivariate Analysis of Variance and Repeated Measures: A Practical Approach for Behavioural Scientists.* Chapman and Hall.

O'Brien, R. G., and Kaiser, M. K. (1985) MANOVA method for analyzing repeated measures designs: An extensive primer. *Psychological Bulletin* **97**, 316–333.

See Also

anova, Anova, waldtest, hccm, vcovHC, vcovHAC, coef, vcov

Examples

```
mod.davis <- lm(weight ~ repwt, data=Davis)

## the following are equivalent:
linearHypothesis(mod.davis, diag(2), c(0,1))
linearHypothesis(mod.davis, c("(Intercept) = 0", "repwt = 1"))
linearHypothesis(mod.davis, c("(Intercept)", "repwt"), c(0,1))
linearHypothesis(mod.davis, c("(Intercept)", "repwt = 1"))

## use asymptotic Chi-squared statistic
linearHypothesis(mod.davis, c("(Intercept) = 0", "repwt = 1"), test = "Chisq")


## the following are equivalent:
  ## use HC3 standard errors via white.adjust option
linearHypothesis(mod.davis, c("(Intercept) = 0", "repwt = 1"),
    white.adjust = TRUE)
  ## covariance matrix *function* (where type = "hc3" is the default)
linearHypothesis(mod.davis, c("(Intercept) = 0", "repwt = 1"), vcov. = hccm)
  ## covariance matrix *estimate*, specifying type explicitly
linearHypothesis(mod.davis, c("(Intercept) = 0", "repwt = 1"),
    vcov = hccm(mod.davis, type = "hc3"))
  ## specifying additional type argument to covariance matrix function
linearHypothesis(mod.davis, c("(Intercept) = 0", "repwt = 1"),
    vcov = function(m) hccm(m, type = "hc3"))
```

```
mod.duncan <- lm(prestige ~ income + education, data=Duncan)

## the following are all equivalent:
linearHypothesis(mod.duncan, "1*income - 1*education = 0")
linearHypothesis(mod.duncan, "income = education")
linearHypothesis(mod.duncan, "income - education")
linearHypothesis(mod.duncan, "1income - 1education = 0")
linearHypothesis(mod.duncan, "0 = 1*income - 1*education")
linearHypothesis(mod.duncan, "income-education=0")
linearHypothesis(mod.duncan, "1*income - 1*education + 1 = 1")
linearHypothesis(mod.duncan, "2income = 2*education")

mod.duncan.2 <- lm(prestige ~ type*(income + education), data=Duncan)
coefs <- names(coef(mod.duncan.2))

## test against the null model (i.e., only the intercept is not set to 0)
linearHypothesis(mod.duncan.2, coefs[-1])

## test all interaction coefficients equal to 0
linearHypothesis(mod.duncan.2, coefs[grep(":", coefs)], verbose=TRUE)
linearHypothesis(mod.duncan.2, matchCoefs(mod.duncan.2, ":"), verbose=TRUE) # equivalent
lh <- linearHypothesis(mod.duncan.2, coefs[grep(":", coefs)])
attr(lh, "value") # value of linear function
attr(lh, "vcov")  # covariance matrix of linear function

## a multivariate linear model for repeated-measures data
## see ?OBrienKaiser for a description of the data set used in this example.

mod.ok <- lm(cbind(pre.1, pre.2, pre.3, pre.4, pre.5,
                   post.1, post.2, post.3, post.4, post.5,
                   fup.1, fup.2, fup.3, fup.4, fup.5) ~  treatment*gender,
             data=OBrienKaiser)
coef(mod.ok)

## specify the model for the repeated measures:
phase <- factor(rep(c("pretest", "posttest", "followup"), c(5, 5, 5)),
    levels=c("pretest", "posttest", "followup"))
hour <- ordered(rep(1:5, 3))
idata <- data.frame(phase, hour)
idata

## test the four-way interaction among the between-subject factors
## treatment and gender, and the intra-subject factors
## phase and hour

linearHypothesis(mod.ok, c("treatment1:gender1", "treatment2:gender1"),
    title="treatment:gender:phase:hour", idata=idata, idesign=~phase*hour,
    iterms="phase:hour")

## mixed-effects models examples:

## Not run:  # loads nlme package
```

```
 library(nlme)
 example(lme)
 linearHypothesis(fm2, "age = 0")

## End(Not run)

## Not run:  # loads lme4 package
 library(lme4)
 example(glmer)
 linearHypothesis(gm1, matchCoefs(gm1, "period"))

## End(Not run)

if (require(nnet)){
  print(m <- multinom(partic ~ hincome + children, data=Womenlf))
  print(coefs <- as.vector(outer(c("not.work.", "parttime."),
                           c("hincome", "childrenpresent"),
                           paste0)))
  linearHypothesis(m, coefs) # ominbus Wald test
}
```

`logit` *Logit Transformation*

Description

Compute the logit transformation of proportions or percentages.

Usage

```
logit(p, percents, adjust)
```

Arguments

`p`	numeric vector or array of proportions or percentages.
`percents`	`TRUE` for percentages, `FALSE` for proportions. If the argument is missing and the largest value of `p` > 1, `percents` is set to `TRUE`, otherwise to `FALSE`.
`adjust`	adjustment factor to avoid proportions of 0 or 1; defaults to `0` if there are no such proportions in the data, and to `.025` if there are.

Details

Computes the logit transformation $\text{logit} = \log[p/(1-p)]$ for the proportion p.

If $p = 0$ or 1, then the logit is undefined. `logit` can remap the proportions to the interval `(adjust, 1 - adjust)` prior to the transformation. If it adjusts the data automatically, `logit` will print a warning message.

Value

a numeric vector or array of the same shape and size as p.

Author(s)

John Fox <jfox@mcmaster.ca> and Iain Proctor

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

probabilityAxis

Examples

```
save.opt <- options(digits=4)
logit(.1*0:10)
logit(.1*0:10, adjust=0)
logit(10*1:9, percents=TRUE)
options(save.opt)
```

---

mcPlots *Draw Linear Model Marginal and Conditional Plots in Parallel or Overlaid*

Description

the mcPlot function draws two graphs or overlays the two graphs. For a response Y and a regressor X, the first plot is the *m*arginal plot of Y versus X with both variables centered, visualizing the conditional distribution of Y given X ignoring all other regressors. The second plot is an added-variable for X after all other regressors, visualizing the *c*onditional distribution of Y given X after adjusting for all other predictors. The added variable plot by default is drawn using the same xlim and ylim as the centered marginal plot to emphasize that conditioning removes variation in both the regressor and the response.The plot is primarily intended as a pedagogical tool for understanding coefficients in first-order models.

Usage

```
mcPlots(model, ...)

## Default S3 method:
mcPlots(model, terms=~., layout=NULL, ask, overlaid=TRUE, ...)

mcPlot(model, ...)

## S3 method for class 'lm'
```

```
mcPlot(model, variable, id=FALSE,
    col.marginal=carPalette()[2], col.conditional=carPalette()[3],
    col.arrows="gray", pch = c(16,1), cex=par("cex"), pt.wts=FALSE,
    lwd = 2, grid=TRUE, ellipse=FALSE, overlaid=TRUE, new=TRUE,
    title=TRUE, ...)

## S3 method for class 'glm'
mcPlot(model, ...)
```

Arguments

model	model object produced by lm; the "glm" method just reports an error.
terms	A one-sided formula that specifies a subset of the predictors. One added-variable plot is drawn for each regressor and for each basis vector used to define a factor. For example, the specification terms = ~ . - X3 would plot against all terms except for X3. If this argument is a quoted name of one of the regressors or factors, the added-variable plot is drawn for that regressor or factor only. Unlike other car functions, the formula should include the names of regressors, not predictors. That is, if log(X4) is used to represent a predictor X4, the formula should specify terms = ~ log(X4).
variable	A quoted string giving the name of a numeric predictor in the model matrix for the horizontal axis. To plot against a factor, you need to specify the full name of one of the indicator variables that define the factor. For example, for a factor called type with levels A, B and C, using the usual drop-first level parameterization of the factor, the regressors for type would be typeB or typeC. Similarly, to plot against the regressor log(X4), you must specify "log((X4)", not "X4".
layout	If set to a value like c(1, 2) or c(6, 2), the layout of the graph will have this many rows and columns. If not set, behavior depends on the value of the overlaid argument; see the details
ask	If TRUE, ask the user before drawing the next plot; if FALSE don't ask.
...	mcPlots passes these arguments to mcPlot. mcPlot passes arguments to plot.
id	controls point identification; if FALSE (the default), no points are identified; can be a list of named arguments to the showLabels function; TRUE is equivalent to list(method=list(abs(residuals(model, type="pearson")), "x"), n=2, cex=1, col=carPalette()[1], location="lr"), which identifies the 2 points with the largest residuals and the 2 points with the most extreme horizontal (X) values.
overlaid	If TRUE, the default, overlay the marginal and conditional plots on the same graph; otherwise plot them side-by-side. See the details below
col.marginal, col.conditional	colors for points, lines, ellipses in the marginal and conditional plots, respectively. The defaults are determined by the carPalette function.
col.arrows	color for the arrows with overlaid=TRUE
pch	Plotting character for marginal and conditional plots, respectively.
cex	size of plotted points; default is taken from par("cex").

pt.wts	if TRUE (the default is FALSE), the areas of plotted points for a weighted least squares fit are made proportional to the weights, with the average size taken from the cex argument.
lwd	line width; default is 2 (see par).
grid	If TRUE, the default, a light-gray background grid is put on the graph.
ellipse	Arguments to pass to the dataEllipse function, in the form of a list with named elements; e.g., ellipse.args=list(robust=TRUE)) will cause the ellipse to be plotted using a robust covariance-matrix. if FALSE, the default, no ellipse is plotted. TRUE is equivalent to ellipse=list(levels=0.5), which plots a bivariate-normal 50 percent concentration ellipse.
new	if TRUE, the default, the plot window is reset when overlaid=FALSE using par{mfrow=c(1, 2)}. If FALSE, the layout of the plot window is not reset. Users will ordinarily ignore this argument.
title	If TRUE, the default, the standard main argument in plot is used to add a standard title to each plot. If FALSE no title is used.

Details

With an lm object, suppose the response is Y, X is a numeric regressor of interest, and Z is all the remaining predictors, possibly including interactions and factors. This function produces two graphs. The first graph is the marginal plot of Y versus X, with each variable centered around its mean. The second conditional plot is the added-variable plot of e(Y|Z) versus e(X|Z) where e(a|b) means the Pearson residuals from the regression of a on b. If overlaid=TRUE, these two plots are overlaid in one graph, with the points in different colors. In addition, each point in the marginal plot is joined to its value in the conditional plot by an arrow. Least squares regression lines fit to the marginal and conditional graphs are also shown; data ellipsoids can also be added. If overlaid=FALSE, then the two graphs are shown in side-by-side plots as long as the second argument to layout is equal to 2, or layout is set by the function. The arrows are omitted if the graphs are not overlaid.

These graphs are primarily for teaching, as the marginal plot shows the relationship between Y and X ignoring Z, while the conditional is the relationship between Y and X given X. By keeping the scales the same in both graphs the effect of conditioning on both X and Y can be visualized.

This function is intended for first-order models with numeric predictors only. For a factor, one (pair) of mcPlots will be produced for each of the dummy variables in the basis for the factor, and the resulting plots are not generally meaningful because they depend on parameterization. If the mean function includes interactions, then mcPlots for main effects may violate the hierarchy principle, and may also be of little interest. mcPlots for interactions of numerical predictors, however, can be useful.

These graphs are closely related to the ARES plots proposed by Cook and Weisberg (1989). This plot would benefit from animation.

Value

These functions are used for their side effect of producing plots.

Author(s)

John Fox <jfox@mcmaster.ca>, Sanford Weisberg <sandy@umn.edu>

References

Cook, R. D. and Weisberg, S. (1989) *Regression diagnostics with dynamic graphics,* Technometrics, 31, 277.

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Weisberg, S. (2014) *Applied Linear Regression*, Fourth Edition, Wiley.

See Also

avPlots, residualPlots, crPlots, ceresPlots, dataEllipse

Examples

```
m1 <- lm(partic ~ tfr + menwage + womwage + debt + parttime, data = Bfox)
mcPlot(m1, "womwage")
mcPlot(m1, "womwage", overlaid=FALSE, ellipse=TRUE)
```

---

mmps *Marginal Model Plotting*

Description

For a regression object, draw a plot of the response on the vertical axis versus a linear combination u of regressors in the mean function on the horizontal axis. Added to the plot are a smooth for the graph, along with a smooth from the plot of the fitted values on u. `mmps` is an alias for `marginalModelPlots`, and `mmp` is an alias for `marginalModelPlot`.

Usage

```
marginalModelPlots(...)

mmps(model, terms= ~ ., fitted=TRUE, layout=NULL, ask,
        main, groups, key=TRUE, ...)

marginalModelPlot(...)

mmp(model, ...)

## S3 method for class 'lm'
mmp(model, variable, sd = FALSE,
    xlab = deparse(substitute(variable)),
    smooth=TRUE, key=TRUE, pch, groups=NULL, ...)

## Default S3 method:
mmp(model, variable, sd = FALSE,
    xlab = deparse(substitute(variable)), ylab, smooth=TRUE,
```

```
    key=TRUE, pch, groups=NULL,
    col.line = carPalette()[c(2, 8)], col=carPalette()[1],
    id=FALSE, grid=TRUE, ...)

## S3 method for class 'glm'
mmp(model, variable, sd = FALSE,
    xlab = deparse(substitute(variable)), ylab,
    smooth=TRUE, key=TRUE, pch, groups=NULL,
    col.line = carPalette()[c(2, 8)], col=carPalette()[1],
    id=FALSE, grid=TRUE, ...)
```

Arguments

model	A regression object, usually of class either lm or glm, for which there is a predict method defined.
terms	A one-sided formula. A marginal model plot will be drawn for each term on the right-side of this formula that is not a factor. The default is ~ ., which specifies that all the terms in formula(object) will be used. If a conditioning argument is given, eg terms = ~. | a, then separate colors and smoothers are used for each unique non-missing value of a. See examples below.
fitted	If TRUE, the default, then a marginal model plot in the direction of the fitted values for a linear model or the linear predictor of a generalized linear model will be drawn.
layout	If set to a value like c(1, 1) or c(4, 3), the layout of the graph will have this many rows and columns. If not set, the program will select an appropriate layout. If the number of graphs exceed nine, you must select the layout yourself, or you will get a maximum of nine per page. If layout=NA, the function does not set the layout and the user can use the par function to control the layout, for example to have plots from two models in the same graphics window.
ask	If TRUE, ask before clearing the graph window to draw more plots.
main	Main title for the array of plots. Use main="" to suppress the title; if missing, a title will be supplied.
...	Additional arguments passed from mmps to mmp and then to plot. Users should generally use mmps, or equivalently marginalModelPlots.
variable	The quantity to be plotted on the horizontal axis. If this argument is missing, the horizontal variable is the linear predictor, returned by predict(object) for models of class lm, with default label "Fitted values", or returned by predict(object, type="link") for models of class glm, with default label "Linear predictor". It can be any other vector of length equal to the number of observations in the object. Thus the mmp function can be used to get a marginal model plot versus any regressor or predictor while the mmps function can be used only to get marginal model plots for the first-order regressors in the formula. In particular, terms defined by a spline basis are skipped by mmps, but you can use mmp to get the plot for the variable used to define the splines.
sd	If TRUE, display sd smooths. For a binomial regression with all sample sizes equal to one, this argument is ignored as the SD bounds don't make any sense.

xlab	label for horizontal axis.
ylab	label for vertical axis, defaults to name of response.
smooth	specifies the smoother to be used along with its arguments; if FALSE, no smoother is shown; can be a list giving the smoother function and its named arguments; TRUE, the default, is equivalent to list(smoother=loessLine, span=2/3) for linear models and list(smoother=gamLine, k=3) for generalized linear models. See ScatterplotSmoothers for the smoothers supplied by the **car** package and their arguments; the spread argument is not supported for marginal model plots.
groups	The name of a vector that specifies a grouping variable for separate colors/smoothers. This can also be specified as a conditioning argument on the terms argument.
key	If TRUE, include a key at the top of the plot, if FALSE omit the key. If grouping is present, the key is only printed for the upper-left plot.
id	controls point identification; if FALSE (the default), no points are identified; can be a list of named arguments to the showLabels function; TRUE is equivalent to list(method="y", n=2, cex=1, col=carPalette()[1], location="lr"), which identifies the 2 points with the most unusual response (Y) values.
pch	plotting character to use if no grouping is present.
col.line	colors for data and model smooth, respectively. The default is to use carPalette, carPalette()[c(2, 8)], blue and red.
col	color(s) for the plotted points.
grid	If TRUE, the default, a light-gray background grid is put on the graph

Details

mmp and marginalModelPlot draw one marginal model plot against whatever is specified as the horizontal axis. mmps and marginalModelPlots draws marginal model plots versus each of the terms in the terms argument and versus fitted values. mmps skips factors and interactions if they are specified in the terms argument. Terms based on polynomials or on splines (or potentially any term that is represented by a matrix of regressors) will be used to form a marginal model plot by returning a linear combination of the terms. For example, if you specify terms = ~ X1 + poly(X2, 3) and poly(X2, 3) was part of the original model formula, the horizontal axis of the marginal model plot for X2 will be the value of predict(model, type="terms")[, "poly(X2, 3)"]). If the predict method for the model you are using doesn't support type="terms", then the polynomial/spline term is skipped. Adding a conditioning variable, e.g., terms = ~ a + b | c, will produce marginal model plots for a and b with different colors and smoothers for each unique non-missing value of c.

For linear models, the default smoother is loess. For generalized linear models, the default smoother uses gamLine, fitting a generalized additive model with the same family, link and weights as the fit of the model. SD smooths are not computed for for generalized linear models.

For generalized linear models the default number of elements in the spline basis is k=3; this is done to allow fitting for predictors with just a few support points. If you have many support points you may wish to set k to a higher number, or k=-1 for the default used by gam.

Value

Used for its side effect of producing plots.

Author(s)

Sanford Weisberg, <sandy@umn.edu>

References

Cook, R. D., & Weisberg, S. (1997). Graphics for assessing the adequacy of regression models. *Journal of the American Statistical Association*, 92(438), 490-499.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition. Sage.

Weisberg, S. (2005) *Applied Linear Regression*, Third Edition, Wiley, Section 8.4.

See Also

ScatterplotSmoothers, plot

Examples

```
c1 <- lm(infantMortality ~ ppgdp, UN)
mmps(c1)
c2 <- update(c1, ~ log(ppgdp))
mmps(c2)
# include SD lines
p1 <- lm(prestige ~ income + education, Prestige)
mmps(p1, sd=TRUE)
# condition on type:
mmps(p1, ~. | type)
# logisitic regression example
# smoothers return warning messages.
# fit a separate smoother and color for each type of occupation.
m1 <- glm(lfp ~ ., family=binomial, data=Mroz)
mmps(m1)
```

---

ncvTest *Score Test for Non-Constant Error Variance*

Description

Computes a score test of the hypothesis of constant error variance against the alternative that the error variance changes with the level of the response (fitted values), or with a linear combination of predictors.

Usage

```
ncvTest(model, ...)

## S3 method for class 'lm'
ncvTest(model, var.formula, ...)

## S3 method for class 'glm'
ncvTest(model, ...) # to report an error
```

Arguments

model	a weighted or unweighted linear model, produced by lm.
var.formula	a one-sided formula for the error variance; if omitted, the error variance depends on the fitted values.
...	arguments passed down to methods functions; not currently used.

Details

This test is often called the Breusch-Pagan test; it was independently suggested with some extension by Cook and Weisberg (1983).

ncvTest.glm is a dummy function to generate an error when a glm model is used.

Value

The function returns a chisqTest object, which is usually just printed.

Author(s)

John Fox <jfox@mcmaster.ca>, Sandy Weisberg <sandy@umn.edu>

References

Breusch, T. S. and Pagan, A. R. (1979) A simple test for heteroscedasticity and random coefficient variation. *Econometrica* **47**, 1287–1294.

Cook, R. D. and Weisberg, S. (1983) Diagnostics for heteroscedasticity in regression. *Biometrika* **70**, 1–10.

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Weisberg, S. (2014) *Applied Linear Regression*, Fourth Edition, Wiley.

See Also

hccm, spreadLevelPlot

Examples

```
ncvTest(lm(interlocks ~ assets + sector + nation, data=Ornstein))

ncvTest(lm(interlocks ~ assets + sector + nation, data=Ornstein),
    ~ assets + sector + nation, data=Ornstein)
```

outlierTest *Bonferroni Outlier Test*

Description

Reports the Bonferroni p-values for testing each observation in turn to be a mean-shift outlier, based Studentized residuals in linear (t-tests), generalized linear models (normal tests), and linear mixed models.

Usage

```
outlierTest(model, ...)

## S3 method for class 'lm'
outlierTest(model, cutoff=0.05, n.max=10, order=TRUE,
 labels=names(rstudent), ...)

## S3 method for class 'lmerMod'
outlierTest(model, ...)

## S3 method for class 'outlierTest'
print(x, digits=5, ...)
```

Arguments

model	an lm, glm, or lmerMod model object; the "lmerMod" method calls the "lm" method and can take the same arguments.
cutoff	observations with Bonferroni p-values exceeding cutoff are not reported, unless no observations are nominated, in which case the one with the largest Studentized residual is reported.
n.max	maximum number of observations to report (default, 10).
order	report Studenized residuals in descending order of magnitude? (default, TRUE).
labels	an optional vector of observation names.
...	arguments passed down to methods functions.
x	outlierTest object.
digits	number of digits for reported p-values.

Details

For a linear model, p-values reported use the t distribution with degrees of freedom one less than the residual df for the model. For a generalized linear model, p-values are based on the standard-normal distribution. The Bonferroni adjustment multiplies the usual two-sided p-value by the number of observations. The lm method works for glm objects. To show all of the observations set cutoff=Inf and n.max=Inf.

Value

an object of class `outlierTest`, which is normally just printed.

Author(s)

John Fox <jfox@mcmaster.ca> and Sanford Weisberg

References

Cook, R. D. and Weisberg, S. (1982) *Residuals and Influence in Regression.* Chapman and Hall.

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Weisberg, S. (2014) *Applied Linear Regression*, Fourth Edition, Wiley.

Williams, D. A. (1987) Generalized linear model diagnostics using the deviance and single case deletions. *Applied Statistics* **36**, 181–191.

Examples

```
outlierTest(lm(prestige ~ income + education, data=Duncan))
```

---

panel.car *Panel Function for Coplots*

Description

a panel function for use with `coplot` that plots points, a lowess line, and a regression line.

Usage

```
panel.car(x, y, col, pch, cex=1, span=0.5, lwd=2,
  reg.line=lm, lowess.line=TRUE, ...)
```

Arguments

`x`	vector giving horizontal coordinates.
`y`	vector giving vertical coordinates.
`col`	point color.
`pch`	plotting character for points.
`cex`	character expansion factor for points.
`span`	span for lowess smoother.
`lwd`	line width, default is `2`.
`reg.line`	function to compute coefficients of regression line, or `FALSE` for no line.
`lowess.line`	if `TRUE` plot lowess smooth.
`...`	other arguments to pass to functions `lines` and `regLine`.

Value

NULL. This function is used for its side effect: producing a panel in a coplot.

Author(s)

John Fox <jfox@mcmaster.ca>

See Also

coplot, regLine

Examples

```
coplot(prestige ~ income|education, panel=panel.car,
  col="red", data=Prestige)
```

---

pointLabel *Label placement for points to avoid overlaps*

Description

Use optimization routines to find good locations for point labels without overlaps.

Usage

```
pointLabel(x, y = NULL, labels = seq(along = x), cex = 1,
           method = c("SANN", "GA"),
           allowSmallOverlap = FALSE,
           trace = FALSE,
           doPlot = TRUE,
           ...)
```

Arguments

x, y	as with plot.default, these provide the x and y coordinates for the point labels. Any reasonable way of defining the coordinates is acceptable. See the function xy.coords for details.
labels	as with text, a character vector or expression specifying the text to be written. An attempt is made to coerce other language objects (names and calls) to expressions, and vectors and other classed objects to character vectors by as.character.
cex	numeric character expansion factor as with text.
method	the optimization method, either "SANN" for simulated annealing (the default) or "GA" for a genetic algorithm.
allowSmallOverlap	logical; if TRUE, labels are allowed a small overlap. The overlap allowed is 2% of the diagonal distance of the plot area.

trace	logical; if TRUE, status updates are given as the optimization algorithms progress.
doPlot	logical; if TRUE, the labels are plotted on the existing graph with text.
...	arguments passed along to text to specify labeling parameters such as col.

Details

Eight positions are candidates for label placement, either horizontally, vertically, or diagonally offset from the points. The default position for labels is the top right diagonal relative to the point (considered the preferred label position).

With the default settings, simulating annealing solves faster than the genetic algorithm. It is an open question as to which settles into a global optimum the best (both algorithms have parameters that may be tweaked).

The label positioning problem is NP-hard (nondeterministic polynomial-time hard). Placement becomes difficult and slows considerably with large numbers of points. This function places all labels, whether overlaps occur or not. Some placement algorithms remove labels that overlap.

Note that only cex is used to calculate string width and height (using strwidth and strheight), so passing a different font may corrupt the label dimensions. You could get around this by adjusting the font parameters with par prior to running this function.

Value

An xy list giving the x and y positions of the label as would be placed by text(xy, labels).

Note

This function was moved from the **maptools** package in anticipation of the retirement of that package, and with the permission of the function author.

Author(s)

Tom Short, EPRI, <tshort@epri.com>

References

https://en.wikipedia.org/wiki/Automatic_label_placement

Old URL: https://i11www.iti.uni-karlsruhe.de/map-labeling/bibliography/

http://www.eecs.harvard.edu/~shieber/Projects/Carto/carto.html

The genetic algorithm code was adapted from the python code at

https://meta.wikimedia.org/wiki/Map_generator.

The simulated annealing code follows the algorithm and guidelines in:

Jon Christensen, Joe Marks, and Stuart Shieber. Placing text labels on maps and diagrams. In Paul Heckbert, editor, Graphics Gems IV, pages 497-504. Academic Press, Boston, MA, 1994. http://www.eecs.harvard.edu/~shieber/Biblio/Papers/jc.label.pdf

See Also

text, thigmophobe.labels in package **plotrix**

Examples

```
n <- 50
x <- rnorm(n)*10
y <- rnorm(n)*10
plot(x, y, col = "red", pch = 20)
pointLabel(x, y, as.character(round(x,5)), offset = 0, cex = .7)

plot(x, y, col = "red", pch = 20)
pointLabel(x, y, expression(over(alpha, beta[123])), offset = 0, cex = .8)
```

`poTest` *Test for Proportional Odds in the Proportional-Odds Logistic-Regression Model*

Description

The `poTest` function implements tests proposed by Brant (1990) for proportional odds for logistic models fit by the `polr` function in the MASS package.

Usage

```
poTest(model, ...)
## S3 method for class 'polr'
poTest(model, ...)
## S3 method for class 'poTest'
print(x, digits=3, ...)
```

Arguments

`model`	a proptional-odds logit model fit by `polr`.
`x`	an object produced by `poTest`.
`digits`	number of significant digits to print.
`...`	ignored.

Value

`poTest` returns an object meant to be printed showing the results of the tests.

Author(s)

John Fox `<jfox@mcmaster.ca>`

References

R. Brant, "Assessing proportionality in the proportional odds model for ordinal logistic regression." Biometrics 46: 1171–1178, 1990.

Examples

```
if (require("MASS")){
    .W <- Womenlf
    .W$partic <- factor(.W$partic, levels=c("not.work", "parttime", "fulltime"))
    poTest(polr(partic ~ hincome + children + region, data=.W))
}
```

---

powerTransform *Finding Univariate or Multivariate Power Transformations*

Description

powerTransform uses the maximum likelihood-like approach of Box and Cox (1964) to select a transformatiion of a univariate or multivariate response for normality, linearity and/or constant variance. Available families of transformations are the default Box-Cox power family and two additioal families that are modifications of the Box-Cox family that allow for (a few) negative responses. The summary method automatically computes two or three likelihood ratio type tests concerning the transformation powers.

Usage

```
powerTransform(object, ...)

## Default S3 method:
powerTransform(object, family="bcPower", ...)

## S3 method for class 'lm'
powerTransform(object, family="bcPower", ...)

## S3 method for class 'formula'
powerTransform(object, data, subset, weights, na.action,
    family="bcPower", ...)

## S3 method for class 'lmerMod'
powerTransform(object, family="bcPower", ...)
```

Arguments

object	This can either be an object of class lm or lmerMod, a formula, or a matrix or vector; see below.
family	The quoted name of a family of transformations. The available options are "bcPower" for the default for the Box-Cox power family; "bcnPower" for a two-parameter modification of the Box-Cox family that allows negative responses (Hawkins and Weisberg (2017)), and the "yjPower" family (Yeo and Johnson(2000)), another modifiation of the Box-Cox family that allows a few negative values. All three families are documented at bcPower.

data	A data frame or environment, as in 'lm'.
subset	Case indices to be used, as in 'lm'.
weights	Weights as in 'lm'.
na.action	Missing value action, as in 'lm'.
...	Additional arguments that used in the interative algorithm; defaults are generally adequate. For use with the bcnPower family, a convergence criterion can be set with conv=.0001 the default, and a minimum positive value of the location parameter can be set, with default gamma.min=.1.

Details

This function implements the Box and Cox (1964) method of selecting a power transformation of a variable toward normality, and its generalization by Velilla (1993) to a multivariate response. Cook and Weisberg (1999) and Weisberg (2014) suggest the usefulness of transforming a set of predictors z1, z2, z3 for multivariate normality. It also includes two additional families that allow for negative values.

If the object argument is of class 'lm' or 'lmerMod', the Box-Cox procedure is applied to the conditional distribution of the response given the predictors. For 'lm' objects, the respose may be multivariate, and each column will have its own transformation. With 'lmerMod' the response must be univariate.

The object argument may also be a formula. For example, z ~ x1 + x2 + x3 will estimate a transformation for the response z from a family after fitting a linear model with the given formula. cbind(y1, y2, y3) ~ 1 specifies transformations to multivariate normality with no predictors. A vector value for object, for example powerTransform(ais$LBM), is equivalent topowerTransform(LBM ~ 1, ais). Similarly, powerTransform(cbind(aisLBM, aisSSF)), where the first argument is a matrix rather than a formula is equivalent to specification of a mulitvariate linear model powerTransform(cbind(LBM, SSF) ~ 1, ais).

Three families of power transformations are available. The default Box-Cox power family (family="bcPower") of power transformations effectively replaces a vector by that vector raised to a power, generally in the range from -3 to 3. For powers close to zero, the log-transformtion is suggested. In practical situations, after estimating a power using the powerTransform function, a variable would be replaced by a simple power transformation of it, for example, if $\lambda \approx 0.5$, then the correspoding variable would be replaced by its square root; if λ is close enough to zero, the the variable would be replaced by its natural logarithm. The Box-Cox family requires the responses to be strictly positive.

The family="bcnPower", or Box-Cox with negatives, family proposed by Hawkins and Weisberg (2017) allows for (a few) non-positive values, while allowing for the transformed data to be interpreted similarly to the interpretation of Box-Cox transformed values. This family is the Box-Cox transformation of $z = .5*(y+(y^2+\gamma^2)^{1/2})$ that depends on a location parameter γ. The quantity z is positive for all values of y. If $\gamma = 0$ and y is strictly positive, then the Box-Cox and the bcnPower transformations are identical. When fitting the Box-Cox with negatives family, lambda is restricted to the range [-3, 3], and gamma is restricted to the range from gamma.min=.1 to the largest positive value of the variable, since values outside these ranges are unreasonable in practice. The value of gamma.min can be changed with an argument to powerTransform.

The final family family="yjPower" uses the Yeo-Johnson transformation, which is the Box-Cox transformation of $U + 1$ for nonnegative values, and of $|U| + 1$ with parameter $2 - \lambda$ for U negative and thus it provides a family for fitting when (a few) observations are negative. Because of the

unusual constraints on the powers for positive and negative data, this transformation is not used very often, as results are difficult to interpret. In practical problems, a variable would be replaced by its Yeo-Johnson transformation computed using the yjPower function.

The function testTransform is used to obtain likelihood ratio tests for any specified value for the transformation parameter(s).

Computations maximize the likelihood-like functions described by Box and Cox (1964) and by Velilla (1993). For univariate responses, the computations are very stable and problems are unlikely, although for 'lmer' models computations may be very slow because the model is refit many times. For multivariate responses with the bcnPower family, the computing algorithm may fail. In this case we recommend adding the argument itmax = 1 to the call to powerTransform. This will return the starting value estimates of the transformation parameters, fitting a d-dimensional response as if all the d responses were independent.

Value

An object of class powerTransform or class bcnPowerTransform if family="bcnPower" that inherits from powerTransform is returned, including the components listed below.

A summary method presents estimated values for the transformation power lambda and for the 'bcnPower' family the location parameter gamma as well. Standard errors and Wald 95% confidence intervals based on the standard errors are computed from the inverse of the sample Hessian matrix evaluted at the estimates. The interval estimates for the gamma parameters will generally be very wide, reflecting little information available about the location parameter. Likelihood ratio type tests are also provided. For the 'bcnPower' family these are based on the profile loglikelihood for lambda alone; that is, we treat gamma as a nusiance parameter and average over it.

The components of the returned object includes

lambda	Estimated transformation parameter
roundlam	Convenient rounded values for the estimates. These rounded values will usually be the desired transformations.
gamma	Estimated location parameters for bcnPower, NULL otherwise
invHess	Estimated covariance matrix of the estimated parameters
llik	Value of the log-likelihood at the estimates

The summary method for powerTransform returns an array with columns labeled "Est Power" for the value of lambda that maximizes the likelihood; "Rounded Pwr" for roundlam, and columns "Wald Lwr Bnd" and "Wald Ur Bnd" for a 95 percent Wald normal theory confidence interval for lambda computed as the estimate plus or minus 1.96 times the standard error.

Author(s)

Sanford Weisberg, <sandy@umn.edu>

References

Box, G. E. P. and Cox, D. R. (1964) An analysis of transformations. *Journal of the Royal Statisistical Society, Series B*. 26 211-46.

Cook, R. D. and Weisberg, S. (1999) *Applied Regression Including Computing and Graphics*. Wiley.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Hawkins, D. and Weisberg, S. (2017) Combining the Box-Cox Power and Generalized Log Transformations to Accomodate Nonpositive Responses In Linear and Mixed-Effects Linear Models *South African Statistics Journal*, 51, 317-328.

Velilla, S. (1993) A note on the multivariate Box-Cox transformation to normality. *Statistics and Probability Letters*, 17, 259-263.

Weisberg, S. (2014) *Applied Linear Regression*, Fourth Edition, Wiley.

Yeo, I. and Johnson, R. (2000) A new family of power transformations to improve normality or symmetry. *Biometrika*, 87, 954-959.

See Also

testTransform, bcPower, bcnPower, transform, optim, boxCox.

Examples

```
# Box Cox Method, univariate
summary(p1 <- powerTransform(cycles ~ len + amp + load, Wool))
# fit linear model with transformed response:
coef(p1, round=TRUE)
summary(m1 <- lm(bcPower(cycles, p1$roundlam) ~ len + amp + load, Wool))

# Multivariate Box Cox uses Highway1 data
summary(powerTransform(cbind(len, adt, trks, sigs1) ~ 1, Highway1))

# Multivariate transformation to normality within levels of 'htype'
summary(a3 <- powerTransform(cbind(len, adt, trks, sigs1) ~ htype, Highway1))

# test lambda = (0 0 0 -1)
testTransform(a3, c(0, 0, 0, -1))

# save the rounded transformed values, plot them with a separate
# color for each highway type
transformedY <- bcPower(with(Highway1, cbind(len, adt, trks, sigs1)),
                        coef(a3, round=TRUE))
## Not run:  # generates a smoother warning
scatterplotMatrix( ~ transformedY|htype, Highway1)

## End(Not run)

# With negative responses, use the bcnPower family
m2 <- lm(I1L1 ~ pool, LoBD)
summary(p2 <- powerTransform(m2, family="bcnPower"))
testTransform(p2, .5)
summary(powerTransform(update(m2, cbind(LoBD$I1L2, LoBD$I1L1) ~ .), family="bcnPower"))

## Not run:  # takes a few seconds:
  # multivariate bcnPower, with 8 responses
```

```
  summary(powerTransform(update(m2, as.matrix(LoBD[, -1]) ~ .), family="bcnPower"))
  # multivariate bcnPower, fit with one iteration using starting values as estimates
 summary(powerTransform(update(m2, as.matrix(LoBD[, -1]) ~ .), family="bcnPower", itmax=1))

## End(Not run)

# mixed effects model
## Not run:  # uses the lme4 package
  data <- reshape(LoBD[1:20, ], varying=names(LoBD)[-1], direction="long", v.names="y")
  names(data) <- c("pool", "assay", "y", "id")
  data$assay <- factor(data$assay)
  require(lme4)
  m2 <- lmer(y ~ pool + (1|assay), data)
  summary(l2 <- powerTransform(m2, family="bcnPower", verbose=TRUE))

## End(Not run)
```

`Predict` *Model Predictions*

Description

`Predict` is a generic function with, at present, a single method for `"lm"` objects, `Predict.lm`, which is a modification of the standard `predict.lm` method in the **stats** package, but with an additional `vcov.` argument for a user-specified covariance matrix for intreval estimation.

Usage

```
Predict(object, ...)

## S3 method for class 'lm'
Predict(object, newdata, se.fit = FALSE,
  scale = NULL, df = Inf,
  interval = c("none", "confidence", "prediction"),
  level = 0.95, type = c("response", "terms"),
  terms = NULL, na.action = na.pass,
  pred.var = res.var/weights, weights = 1, vcov., ...)
```

Arguments

`object`	a model object for which predictions are desired.
`newdata, se.fit, scale, df, interval, level, type, terms, na.action, pred.var, weights`	see `predict.lm`.
`vcov.`	optional, either a function to compute the coefficient covariance matrix of `object` (e.g., `hccm`) or a coefficient covariance matrix (as returned, e.g., by `hccm`). To use a bootstrap to estimate the covariance matrix, set `vcov. = vcov(Boot(object))`.
`...`	arguments to pass down to `Predict` or `predict` methods.

Details

If there is no appropriate method for `Predict`, then a `predict` method is invoked. If there is a *specific* `predict` method for the primary class of `object` but only an *inherited* `Predict` method, then the `predict` method is invoked. Thus an object of class `c("glm", "lm")` will invoke method `predict.glm` rather than `Predict.lm`, but an object of class `c("aov", "lm")` will invoke `Predict.lm` rather than `predict.lm`.

Value

See `predict` and `predict.lm`.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

`predict`, `predict.lm`

Examples

```
mod <- lm(interlocks ~ log(assets), data=Ornstein)
newd <- data.frame(assets=exp(4:12))
(p1 <- predict(mod, newd, interval="prediction"))
p2 <- Predict(mod, newd, interval="prediction", vcov.=vcov)
all.equal(p1, p2) # the same

(predict(mod, newd, se=TRUE))
(p3 <- Predict(mod, newd, se=TRUE, vcov.=hccm)) # larger SEs
p4 <- Predict(mod, newd, se=TRUE, vcov.=hccm(mod, type="hc3"))
all.equal(p3, p4) # the same
```

---

qqPlot *Quantile-Comparison Plot*

Description

Plots empirical quantiles of a variable, or of studentized residuals from a linear model, against theoretical quantiles of a comparison distribution. Includes options not available in the `qqnorm` function.

Usage

```
qqPlot(x, ...)

qqp(...)

## Default S3 method:
qqPlot(x, distribution="norm", groups, layout,
    ylim=range(x, na.rm=TRUE), ylab=deparse(substitute(x)),
    xlab=paste(distribution, "quantiles"), glab=deparse(substitute(groups)),
    main=NULL, las=par("las"),
    envelope=TRUE, col=carPalette()[1], col.lines=carPalette()[2],
    lwd=2, pch=1, cex=par("cex"),
    line=c("quartiles", "robust", "none"), id=TRUE, grid=TRUE, ...)

## S3 method for class 'formula'
qqPlot(formula, data, subset, id=TRUE, ylab, glab, ...)

## S3 method for class 'lm'
qqPlot(x, xlab=paste(distribution, "Quantiles"),
    ylab=paste("Studentized Residuals(",
                deparse(substitute(x)), ")", sep=""),
    main=NULL, distribution=c("t", "norm"),
    line=c("robust", "quartiles", "none"), las=par("las"),
    simulate=TRUE, envelope=TRUE,  reps=100,
   col=carPalette()[1], col.lines=carPalette()[2], lwd=2, pch=1, cex=par("cex"),
    id=TRUE, grid=TRUE, ...)
```

Arguments

x	vector of numeric values or lm object.
distribution	root name of comparison distribution – e.g., "norm" for the normal distribution; t for the t-distribution.
groups	an optional factor; if specified, a QQ plot will be drawn for x within each level of groups.
layout	a 2-vector with the number of rows and columns for plotting by groups – for example c(1, 3) for 1 row and 3 columns; if omitted, the number of rows and columns will be selected automatically; the specified number of rows and columns must be sufficient to accomodate the number of groups; ignored if there is no grouping factor.
formula	one-sided formula specifying a single variable to be plotted or a two-sided formula of the form variable ~ factor, where a QQ plot will be drawn for variable within each level of factor.
data	optional data frame within which to evaluage the formula.
subset	optional subset expression to select cases to plot.
ylim	limits for vertical axis; defaults to the range of x. If plotting by groups, a common y-axis is used for all groups.

ylab	label for vertical (empirical quantiles) axis.
xlab	label for horizontal (comparison quantiles) axis.
glab	label for the grouping variable.
main	label for plot.
envelope	TRUE (the default), FALSE, a confidence level such as 0.95, or a list specifying how to plot a point-wise confidence envelope (see Details).
las	if 0, ticks labels are drawn parallel to the axis; set to 1 for horizontal labels (see par).
col	color for points; the default is the *first* entry in the current **car** palette (see carPalette and par).
col.lines	color for lines; the default is the *second* entry in the current **car** palette.
pch	plotting character for points; default is 1 (a circle, see par).
cex	factor for expanding the size of plotted symbols; the default is 1.
id	controls point identification; if FALSE, no points are identified; can be a list of named arguments to the showLabels function; TRUE is equivalent to list(method="y", n=2, cex=1, col=carPalette()[1], location="lr"), which identifies the 2 points with the 2 points with the most extreme verical values — studentized residuals for the "lm" method. Points labels are by default taken from the names of the variable being plotted is any, else case indices are used. Unlike most graphical functions in **car**, the default is id=TRUE to include point identification.
lwd	line width; default is 2 (see par).
line	"quartiles" to pass a line through the quartile-pairs, or "robust" for a robust-regression line; the latter uses the rlm function in the MASS package. Specifying line = "none" suppresses the line.
simulate	if TRUE calculate confidence envelope by parametric bootstrap; for lm object only. The method is due to Atkinson (1985).
reps	integer; number of bootstrap replications for confidence envelope.
...	arguments such as df to be passed to the appropriate quantile function.
grid	If TRUE, the default, a light-gray background grid is put on the graph

Details

Draws theoretical quantile-comparison plots for variables and for studentized residuals from a linear model. A comparison line is drawn on the plot either through the quartiles of the two distributions, or by robust regression.

Any distribution for which quantile and density functions exist in R (with prefixes q and d, respectively) may be used. When plotting a vector, the confidence envelope is based on the SEs of the order statistics of an independent random sample from the comparison distribution (see Fox, 2016). Studentized residuals from linear models are plotted against the appropriate t-distribution with a point-wise confidence envelope computed by default by a parametric bootstrap, as described by Atkinson (1985). The function qqp is an abbreviation for qqPlot.

The envelope argument can take a list with the following named elements; if an element is missing, then the default value is used:

`level` confidence level (default `0.95`).

`style` one of `"filled"` (the default), `"lines"`, or `"none"`.

`col` color (default is the value of `col.lines`).

`alpha` transparency/opacity of a filled confidence envelope, a number between 0 and 1 (default `0.15`).

`border` controls whether a border is drawn around a filled confidence envelope (default `TRUE`).

Value

These functions return the labels of identified points, unless a grouping factor is employed, in which case `NULL` is returned invisibly.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Atkinson, A. C. (1985) *Plots, Transformations, and Regression.* Oxford.

See Also

`qqplot`, `qqnorm`, `qqline`, `showLabels`

Examples

```
x<-rchisq(100, df=2)
qqPlot(x)
qqPlot(x, dist="chisq", df=2, envelope=list(style="lines"))

qqPlot(~ income, data=Prestige, subset = type == "prof")
qqPlot(income ~ type, data=Prestige, layout=c(1, 3))

qqPlot(lm(prestige ~ income + education + type, data=Duncan),
 envelope=.99)
```

---

recode *Recode a Variable*

Description

Recodes a numeric vector, character vector, or factor according to simple recode specifications. `Recode` is an alias for `recode` that avoids name clashes with packages, such as **Hmisc**, that have a `recode` function.

Usage

```
recode(var, recodes, as.factor, as.numeric=TRUE, levels,
       to.value="=", interval=":", separator=";")

Recode(...)
```

Arguments

var	numeric vector, character vector, or factor.
recodes	character string of recode specifications: see below.
as.factor	return a factor; default is TRUE if var is a factor, FALSE otherwise.
as.numeric	if TRUE (the default), and as.factor is FALSE, then the result will be coerced to numeric if all values in the result can represent numbers (contain only numerals, minus signs, etc.).
levels	an optional argument specifying the order of the levels in the returned factor; the default is to use the sort order of the level names.
to.value	The operator to separate old from new values, "=" by default; some other possibilities: "->", "~", "~>". Cannot include the interval operator (by default :) or the separator string (by default, ;), so, e.g., by default ":=>" is not allowed. The discussion in Details assumes the default "=". Use a non-default to.value if factor levels contain =.
interval	the operator used to denote numeric intervals, by default ":". The discussion in Details assumes the default ":". Use a non-default interval if factor levels contain :.
separator	the character string used to separate recode specifications, by default ";". The discussion in Details assumes the default ";". Use a non-default separator if factor levels contain ;.
...	arguments to be passed to recode.

Details

Recode specifications appear in a character string, separated by default by semicolons (see the examples below), each of the form input=output (where = may be replaced by a non-default value of the to.value argument, e.g., input -> output). Spaces may be used for clarity. If an input value satisfies more than one specification, then the first (from left to right) applies. If no specification is satisfied, then the input value is carried over to the result. NA is allowed on input and output. Several recode specifications are supported:

single value For example, 0=NA.

vector of values For example, c(7, 8, 9) = 'high'.

range of values For example, 7:9 = 'C'. The special values lo and hi may appear in a range. For example, lo:10=1. *Note:* : is *not* the R sequence operator. In addition, you may not use : with the c function within a recode specification, so for example c(1, 3, 5:7) will cause an error. The : is the default value of the recode interval operator; a non-default value may be specified.

`else` everything that does not fit a previous specification. For example, `else = NA`. Note that `else` matches *all* otherwise unspecified values on input, including `NA`, and if present should appear last among the recode specifications.

Character data and factor levels on the left-hand side of a recode specification must be quoted. Thus, e.g., `c(a, b, c) = 'low'` is not allowed, and should be `c('a', 'b', 'c') = 'low'`. Similarly, the colon is reserved for numeric data, and, e.g., `c('a':'c') = 'low'` is not allowed. If the `var` argument is a character variable with (some) values that are character representations of numbers, or a factor with (some) levels that are numbers (e.g., `'12'` or `'-2'`), then these too must be quoted and cannot be used with colons (e.g., `'15':'19' = '15 to 19'` is not allowed, and could be specified as `c('15', '16', '17', '18', '19') = '15 to 19'`, assuming that all values are the character representation of whole numbers).

If all of the output values are numeric, and if `as.factor` is `FALSE`, then a numeric result is returned; if `var` is a factor, then by default so is the result.

Value

a recoded vector of the same length as `var`.

Warning

The factor levels may not contain the character strings in `to.value` (by default `"="`), interval (by default `":"`), or `separator` (by default `";"`).

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

`cut`, `factor`

Examples

```
x <- rep(1:3, 3)
x
recode(x, "c(1, 2) = 'A';
           else = 'B'")
Recode(x, "1~2 -> ':=1' // 3 -> ';=2'", to.value="->",
       interval="~", separator="//")
```

regLine *Plot Regression Line*

Description

Plots a regression line on a scatterplot; the line is plotted between the minimum and maximum x-values.

Usage

```
regLine(mod, col=carPalette()[2], lwd=2, lty=1,...)
```

Arguments

mod	a model, such as produced by lm, that responds to the coef function by returning a 2-element vector, whose elements are interpreted respectively as the intercept and slope of a regresison line.
col	color for points and lines; the default is the *second* entry in the current **car** palette (see carPalette and par).
lwd	line width; default is 2 (see par).
lty	line type; default is 1, a solid line (see par).
...	optional arguments to be passed to the lines plotting function.

Details

In contrast to abline, this function plots only over the range of the observed x-values. The x-values are extracted from mod as the second column of the model matrix.

Value

NULL. This function is used for its side effect: adding a line to the plot.

Author(s)

John Fox <jfox@mcmaster.ca>

See Also

abline, lines

Examples

```
plot(repwt ~ weight, pch=c(1,2)[sex], data=Davis)
regLine(lm(repwt~weight, subset=sex=="M", data=Davis))
regLine(lm(repwt~weight, subset=sex=="F", data=Davis), lty=2)
```

residualPlots *Residual Plots for Linear and Generalized Linear Models*

Description

Draws a plot or plots of residuals versus one or more term in a mean function and/or versus fitted values. For linear models curvature tests are computed for each of the plots by adding a quadratic term to the regression function and testing the quadratic to be zero. This is Tukey's test for nonadditivity when plotting against fitted values.

Usage

```
### This is a generic function with only one required argument:

residualPlots (model, ...)

## Default S3 method:
residualPlots(model, terms= ~ . ,
            layout=NULL, ask, main="",
            fitted=TRUE, AsIs=TRUE, plot=TRUE, tests=TRUE, groups, ...)

## S3 method for class 'lm'
residualPlots(model, ...)

## S3 method for class 'glm'
residualPlots(model, ...)

### residualPlots calls residualPlot, so these arguments can be
### used with either function

residualPlot(model, ...)

## Default S3 method:
residualPlot(model, variable = "fitted", type = "pearson",
     groups, plot = TRUE, linear = TRUE,
     quadratic = if(missing(groups)) TRUE else FALSE,
     smooth=FALSE, id=FALSE,
     col = carPalette()[1], col.quad = carPalette()[2], pch=1,
     xlab, ylab, lwd=1, grid=TRUE, key=!missing(groups), ...)

## S3 method for class 'lm'
residualPlot(model, ...)

## S3 method for class 'glm'
residualPlot(model, variable = "fitted", type = "pearson",
    plot = TRUE, quadratic = FALSE, smooth=TRUE, ...)
```

Arguments

model	A regression object.
terms	A one-sided formula that specifies a subset of the terms that appear in the formula that defined the model. The default ~ . is to plot against all first-order terms. Interactions are skipped. For example, the specification terms = ~ . - X3 would plot against all terms except for X3. To get a plot against fitted values only, use the arguments terms = ~ 1. If a term like log(X1) is in the formula, then the corresponding plot is obtained using terms = ~ log(X1). For polynomial terms generated using the poly function, the plot is against the first-order variable, which may be centered and scaled depending on how the arguments to the poly function. Plots against factors are boxplots. Plots against splines use the result of predict(model), type="terms")[, variable]) as the horizontal axis. A grouping variable can also be specified, so, for example terms= ~ .|type would use the factor type to set a different color and symbol for each level of type. Any fits in the plots will also be done separately for each level of group. This can be useful of finding interactions between the grouping variable and the term on the horizontal axis. The grouping variable can also be set using the argument goups="type".
layout	If set to a value like c(1, 1) or c(4, 3), the layout of the graph will have this many rows and columns per "page" with a prompt given the in console to change to the next page. If not set, the program will select an appropriate layout. If the number of graphs exceed nine, you must select the layout yourself, or you will get a maximum of nine per page. If layout=NA, the function does not set the layout and the user can use the par function to control the layout, for example to have plots from two models in the same graphics window.
ask	If TRUE, ask the user before drawing the next plot; if FALSE, don't ask.
main	Main title for the graphs. The default is main="" for no title.
fitted	If TRUE, the default, include the plot against fitted values.
AsIs	Some terms in a model formula use the as-is or identity function, for example a term like I(X^2) to include a quadratic. These terms will be skipped in plotting if AsIs is FALSE. The default is TRUE.
plot	If TRUE, draw the plot(s).
tests	If TRUE, display the curvature tests.
...	Additional arguments passed to residualPlot and then to plot.
variable	Quoted variable name for the factor or regressor to be put on the horizontal axis, or the default "fitted" to plot versus fitted values.
type	Specifies the type of residual to be plotted. Any of c("working", "response", "deviance", "pearson", "partial", "rstudent", "rstandard") may be specified. The default type = "pearson" is usually appropriate, since it is equal to ordinary residuals observed minus fit with ols, and correctly weighted residuals with wls or for a glm. The last two options use the rstudent and rstandard functions and use studentized or standardized residuals.
groups	A grouping indicator, provided as the quoted name of a grouping variable in the model, such as "type", or "ageGroup". Points in different groups will be

plotted with different colors and symbols. If missing, no grouping is used. In residualPlots, the grouping variable can also be set in the terms argument, as described above. The default is no grouping.

linear — If TRUE, adds a horizontal line at zero if no groups. With groups, within level of groups display the ols regression of the residuals as response and the horizontal axis as the regressor.

quadratic — if TRUE, fits the quadratic regression of the vertical axis on the horizontal axis and displays a lack of fit test. Default is TRUE for lm and FALSE for glm or if groups not missing.

smooth — specifies the smoother to be used along with its arguments; if FALSE, which is the default except for generalized linear models, no smoother is shown; can be a list giving the smoother function and its named arguments; TRUE is equivalent to list(smoother=loessLine, span=2/3, col=carPalette()[3]), which is the default for a GLM. See ScatterplotSmoothers for the smoothers supplied by the **car** package and their arguments.

id — controls point identification; if FALSE (the default), no points are identified; can be a list of named arguments to the showLabels function; TRUE is equivalent to list(method="r", n=2, cex=1, col=carPalette()[1], location="lr"), which identifies the 2 points with the largest absolute residuals.

col — default color for points. If groups is set, col can be a list at least as long as the number of levels for groups giving the colors for each groups.

col.quad — default color for quadratic fit if groups is missing. Ignored if groups are used.

pch — plotting character. The default is pch=1. If groups are used, pch can be set to a vector at least as long as the number of groups.

xlab — X-axis label. If not specified, a useful label is constructed by the function.

ylab — Y-axis label. If not specified, a useful label is constructed by the function.

lwd — line width for the quadratic fit, if any.

grid — If TRUE, the default, a light-gray background grid is put on the graph

key — Should a key be added to the plot? Default is !is.null(groups).

Details

residualPlots draws one or more residuals plots depending on the value of the terms and fitted arguments. If terms = ~ ., the default, then a plot is produced of residuals versus each first-order term in the formula used to create the model. A plot of residuals versus fitted values is also included unless fitted=FALSE. Setting terms = ~1 will provide only the plot against fitted values.

A table of curvature tests is displayed for linear models. For plots against a term in the model formula, say X1, the test displayed is the t-test for for I(X1^2) in the fit of update, model, ~. + I(X1^2)). Econometricians call this a specification test. For factors, the displayed plot is a boxplot, no curvature test is computed, and grouping is ignored. For fitted values in a linear model, the test is Tukey's one-degree-of-freedom test for nonadditivity. You can suppress the tests with the argument tests=FALSE. If grouping is used curvature tests are not displayed.

residualPlot, which is called by residualPlots, should be viewed as an internal function, and is included here to display its arguments, which can be used with residualPlots as well. The residualPlot function returns the curvature test as an invisible result.

residCurvTest computes the curvature test only. For any factors a boxplot will be drawn. For any polynomials, plots are against the linear term. Other non-standard predictors like B-splines are skipped.

Value

For lm objects, returns a data.frame with one row for each plot drawn, one column for the curvature test statistic, and a second column for the corresponding p-value. This function is used primarily for its side effect of drawing residual plots.

Author(s)

Sanford Weisberg, <sandy@umn.edu>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition. Sage.

Weisberg, S. (2014) *Applied Linear Regression*, Fourth Edition, Wiley, Chapter 8

See Also

See Also lm, identify, showLabels

Examples

```
m1 <- lm(prestige ~ income + type, data=Prestige)
residualPlots(m1)
residualPlots(m1, terms= ~ 1 | type) # plot vs. yhat grouping by type
```

S *Modified Functions for Summarizing Linear, Generalized Linear, and Some Other Models*

Description

car package replacements for the summary (S) and confint (Confint) functions for lm, glm, multinom, and polr objects, with additional arguments but the same defaults as the original functions. The Confint method for "polr" objects profiles the likelihood to get confidence intervals for the regression parameters but uses Wald intervals for the thresholds. Default methods that call the standard R summary and confint functions are provided for the S and Confint generics, so the **car** functions should be safe to use in general. The default method for Confint also assumes that there is an appropriate coef method. For briefer model summaries, see brief.

Usage

```
S(object, brief, ...)

## S3 method for class 'lm'
S(object, brief=FALSE,
    correlation = FALSE, symbolic.cor = FALSE,
    vcov. = vcov(object, complete=FALSE), header = TRUE,
    resid.summary = FALSE, adj.r2 = FALSE,
    ...)

## S3 method for class 'glm'
S(object, brief=FALSE,
    exponentiate, dispersion, correlation = FALSE, symbolic.cor = FALSE,
    vcov. = vcov(object, complete=FALSE), header = TRUE,
    resid.summary = FALSE, ...)

## S3 method for class 'multinom'
S(object, brief=FALSE, exponentiate=FALSE, ...)

## S3 method for class 'polr'
S(object, brief=FALSE, exponentiate=FALSE, ...)

## S3 method for class 'lme'
S(object, brief=FALSE, correlation=FALSE, ...)

## S3 method for class 'lmerMod'
S(object, brief=FALSE, KR=FALSE, correlation=FALSE, ...)

## S3 method for class 'glmerMod'
S(object, brief=FALSE, correlation=FALSE, exponentiate, ...)

## S3 method for class 'data.frame'
S(object, brief=FALSE, ...)

## Default S3 method:
S(object, brief, ...)

## S3 method for class 'S.lm'
print(x, digits = max(3, getOption("digits") - 3),
   symbolic.cor = x$symbolic.cor, signif.stars = getOption("show.signif.stars"), ...)

## S3 method for class 'S.glm'
print(x, digits = max(3L, getOption("digits") - 3L),
   symbolic.cor = x$symbolic.cor, signif.stars = getOption("show.signif.stars"), ...)

## S3 method for class 'S.multinom'
print(x, digits = max(3, getOption("digits") - 3),
    signif.stars = getOption("show.signif.stars"), ...)
```

```
## S3 method for class 'S.polr'
print(x, digits = max(3, getOption("digits") - 3),
    signif.stars = getOption("show.signif.stars"), ...)

## S3 method for class 'S.lme'
print(x, digits=max(3, getOption("digits") - 3),
    signif.stars = getOption("show.signif.stars"), ...)

## S3 method for class 'S.lmerMod'
print(x, digits=max(3, getOption("digits") - 3),
    signif.stars = getOption("show.signif.stars"), ...)

## S3 method for class 'S.glmerMod'
print(x, digits=max(3, getOption("digits") - 3),
    signif.stars = getOption("show.signif.stars"), ...)

Confint(object, ...)

## S3 method for class 'lm'
Confint(object, estimate=TRUE,
    parm, level=0.95, vcov.=vcov(object, complete=FALSE), ...)

## S3 method for class 'glm'
Confint(object, estimate=TRUE, exponentiate=FALSE,
    vcov., dispersion, type=c("LR", "Wald"), ...)

## S3 method for class 'polr'
Confint(object, estimate=TRUE, exponentiate=FALSE,
    thresholds=!exponentiate, ...)

## S3 method for class 'multinom'
Confint(object, estimate=TRUE, exponentiate=FALSE, ...)

## S3 method for class 'lme'
Confint(object, estimate=TRUE, level=0.95, ...)

## S3 method for class 'lmerMod'
Confint(object, estimate=TRUE, level=0.95, ...)

## S3 method for class 'glmerMod'
Confint(object, estimate=TRUE, level=0.95,
    exponentiate=FALSE, ...)

## Default S3 method:
Confint(object, estimate=TRUE, level=0.95, vcov., ...)
```

Arguments

object	a model or other object, e.g., of class "lm" as produced by a call to lm.
exponentiate	for a "glm" or "glmerMod" model using the log or logit link, or a "polr" or "multinom" model, show exponentiated coefficient estimates and confidence bounds.
correlation, symbolic.cor	see summary.lm
x, digits, signif.stars	see summary.lm
dispersion	see summary.glm
vcov.	either a matrix giving the estimated covariance matrix of the estimates, or a function that when called with object as an argument returns an estimated covariance matrix of the estimates. The default of vcov. = vcov uses the usual estimated covariance matrix. Other choices include the functions documented at hccm. See example below for using a bootstrap to estimate the covariance matrix. Note that arguments supplied to ... are *not* passed to vcov. when it's a function; in this case either use an anonymous function in which the additional arguments are set, or supply the coefficient covariance matrix directly. For the glm methods of Confint and S, if the vcov. or dispersion argument is specified, then Wald-based confidence limits are computed; otherwise the reported confidence limits are computed by profiling the likelihood. NOTE: The dispersion and vcov. arguments may not *both* be specified.
header	if TRUE, print the header for the summary output, default is TRUE
resid.summary	if TRUE, print the five-number summary of the residuals in the summary, defaults to FALSE
adj.r2	if TRUE, print the adjusted r-squared in the summary, default is FALSE
brief	if TRUE, set header, resid.summary and adj.r.squared to FALSE, and suppress exponeniated coefficients for GLMs with log or logit link. For a data frame, equivalent to use of brief.
KR	if TRUE (default is FALSE), compute Kenward-Roger standard errors and Satterthwaite degrees of freedom for t-tests. *Warning:* This computation can be very time-consuming.
parm, level	see confint
estimate	show the estimated coefficients in the confidence-interval table; default is TRUE.
thresholds	show confidence intervals for the estimated thresholds in the "polr" model.
type	if "LR" (the default) compute confidence intervals based on the LR statistics by profiling the likelihood; if "Wald" base confidence intervals on the Wald statistic using the coefficient standard error and the normal distribution.
...	additional arguments to be passed down, for consistency with summary and confint methods

Details

All these functions mimic functions in the **stats** and other standard R packages for summarizing aspects of linear, generalized linear, and some other statistical models. The S function also provides an alterntive to summary for data frames, treating character variables as if they were factors.

The S and Confint functions add support for the vcov. argument for linear models, which allows specifying a covariance matrix for the regression coefficients other than the usual covariance matrix returned by the function vcov. This argument may be either the name of a function, so that the call to vcov.(object) returns a covariance matrix, or else vcov. is set equal to a covariance matrix. For example, setting vcov.=hccm uses 'proposal 3' described by Long and Ervin (2000) for a sandwich coefficient-variance estimator that may be robust against nonconstant variance (see hccm). Setting vcov. = hccm(object, type = "hc2") would use the matrix returned by the hccm function using proposal 2. For use with a bootstrap, see the examples below. The overall F-test in the S.lm output uses the supplied covariance matrix in a call to the linearHypothesis function.

The supplied print method for S.lm (and for other S methods) has additional arguments to customize the standard summary.lm output. Standard output is obtained by setting resid.summary=TRUE, adj.r2=TRUE.

Using a heterscedasticy-corrected covariance matrix computed using hccm with GLMs other than Gaussian is not justified; see the article by Freedman (2006).

The Summary.glm method for models fit with the log or logit link by default prints a table of exponentiated coefficients and their confidence limits; Summary.multinom and Summary.polr optionally print tables of exponentiated coefficients.

Value

The S.lm and S.glm functions return a list with all the elements shown at summary.lm and summary.glm. The S.multinom and S.polr functions return a list with all the elements shown at summary.multinom and summary.polr plus potentially a table of exponentiated coefficients and confidence bounds.

The Confint.lm function returns either the output from confint.lm if vcov. = vcov or Wald-type confidence intervals using the supplied covariance matrix for any other choice of vcov..

Finally, Confint applied to any object that does not inherit from "lm", "multinom", or "polr" simply calls confint, along with, by default, using coef to add a column of estimates to the confidence limits.

Author(s)

Sanford Weisberg <sandy@umn.edu>

References

Freedman, David A. (2006). On the so-called Huber sandwich estimator and robust standard errors. *The American Statistician*, **60**, 299-302.

Long, J. S. and Ervin, L. H. (2000) Using heteroscedasity consistent standard errors in the linear regression model. *The American Statistician* **54**, 217–224.

White, H. (1980) A heteroskedastic consistent covariance matrix estimator and a direct test of heteroskedasticity. *Econometrica* **48**, 817–838.

See Also

brief, summary, confint, coef, summary.lm, confint, vcov.lm, hccm, Boot, linearHypothesis

Examples

```
mod.prestige <- lm(prestige ~ education + income + type, Prestige)
S(mod.prestige, vcov.=hccm)
S(mod.prestige, brief=TRUE)
Confint(mod.prestige, vcov.=hccm)

# A logit model
mod.mroz <- glm(lfp ~ ., data=Mroz, family=binomial)
S(mod.mroz)

# use for data frames vs. summary()
Duncan.1 <-Duncan
Duncan.1$type <- as.character(Duncan$type)
summary(Duncan.1)
S(Duncan.1)

## Not run:  # generates an error, which can then be corrected to run example
# Using the bootstrap for standard errors
b1 <- Boot(mod.prestige)
S(mod.prestige, vcov.= vcov(b1))
Confint(b1) # run with the boot object to get corrected confidence intervals

## End(Not run)
```

---

scatter3d *Three-Dimensional Scatterplots and Point Identification*

Description

The scatter3d function uses the rgl package to draw 3D scatterplots with various regression surfaces. The function Identify3d allows you to label points interactively with the mouse: Press the right mouse button (on a two-button mouse) or the centre button (on a three-button mouse), drag a rectangle around the points to be identified, and release the button. Repeat this procedure for each point or set of "nearby" points to be identified. To exit from point-identification mode, click the right (or centre) button in an empty region of the plot.

Usage

```
scatter3d(x, ...)

## S3 method for class 'formula'
scatter3d(formula, data, subset, radius, xlab, ylab, zlab, id=FALSE, ...)

## Default S3 method:
```

```
scatter3d(x, y, z,
   xlab=deparse(substitute(x)), ylab=deparse(substitute(y)),
   zlab=deparse(substitute(z)), axis.scales=TRUE, axis.ticks=FALSE,
   revolutions=0,
   bg.col=c("white", "black"),
   axis.col=if (bg.col == "white") c("darkmagenta", "black", "darkcyan")
              else c("darkmagenta", "white", "darkcyan"),
   surface.col=carPalette()[-1], surface.alpha=0.5,
   neg.res.col="magenta", pos.res.col="cyan",
   square.col=if (bg.col == "white") "black" else "gray",
   point.col="yellow", text.col=axis.col,
   grid.col=if (bg.col == "white") "black" else "gray",
   fogtype=c("exp2", "linear", "exp", "none"),
   residuals=(length(fit) == 1),
   surface=TRUE, fill=TRUE,
   grid=TRUE, grid.lines=26, df.smooth=NULL, df.additive=NULL,
   sphere.size=1, radius=1, threshold=0.01, speed=1, fov=60,
   fit="linear", groups=NULL, parallel=TRUE,
   ellipsoid=FALSE, level=0.5, ellipsoid.alpha=0.1, id=FALSE,
   model.summary=FALSE,
   reg.function, reg.function.col=surface.col[length(surface.col)],
   mouseMode=c(none="none", left="polar", right="zoom", middle="fov",
               wheel="pull"),
   ...)

Identify3d(x, y, z, axis.scales=TRUE, groups = NULL, labels = 1:length(x),
 col = c("blue", "green", "orange", "magenta", "cyan", "red", "yellow", "gray"),
 offset = ((100/length(x))^(1/3)) * 0.02)
```

Arguments

formula	"model" formula, of the form y ~ x + z or to plot by groups y ~ x + z | g, where g evaluates to a factor or other variable dividing the data into groups.
data	data frame within which to evaluate the formula.
subset	expression defining a subset of observations.
x	variable for horizontal axis.
y	variable for vertical axis (response).
z	variable for out-of-screen axis.
xlab, ylab, zlab	axis labels.
axis.scales	if TRUE, label the values of the ends of the axes. *Note:* For Identify3d to work properly, the value of this argument must be the same as in scatter3d.
axis.ticks	if TRUE, print interior axis-"tick" labels; the default is FALSE. (The code for this option was provided by David Winsemius.)
revolutions	number of full revolutions of the display.
bg.col	background colour; one of "white", "black".

axis.col	colours for axes; if axis.scales is FALSE, then the second colour is used for all three axes.
surface.col	vector of colours for regression planes, used in the order specified by fit; for multi-group plots, the colours are used for the regression surfaces and points in the several groups.
surface.alpha	transparency of regression surfaces, from 0.0 (fully transparent) to 1.0 (opaque); default is 0.5.
neg.res.col, pos.res.col	colours for lines representing negative and positive residuals.
square.col	colour to use to plot squared residuals.
point.col	colour of points.
text.col	colour of axis labels.
grid.col	colour of grid lines on the regression surface(s).
fogtype	type of fog effect; one of "exp2", "linear", "exp", "none".
residuals	plot residuals if TRUE; if residuals="squares", then the squared residuals are shown as squares (using code adapted from Richard Heiberger). Residuals are available only when there is one surface plotted.
surface	plot surface(s) (TRUE or FALSE).
fill	fill the plotted surface(s) with colour (TRUE or FALSE).
grid	plot grid lines on the regression surface(s) (TRUE or FALSE).
grid.lines	number of lines (default, 26) forming the grid, in each of the x and z directions.
df.smooth	degrees of freedom for the two-dimensional smooth regression surface; if NULL (the default), the gam function will select the degrees of freedom for a smoothing spline by generalized cross-validation; if a positive number, a fixed regression spline will be fit with the specified degrees of freedom.
df.additive	degrees of freedom for each explanatory variable in an additive regression; if NULL (the default), the gam function will select degrees of freedom for the smoothing splines by generalized cross-validation; if a positive number or a vector of two positive numbers, fixed regression splines will be fit with the specified degrees of freedom for each term.
sphere.size	general size of spheres representing points; the actual size is dependent on the number of observations.
radius	relative radii of the spheres representing the points. This is normally a vector of the same length as the variables giving the coordinates of the points, and for the formula method, that must be the case or the argument may be omitted, in which case spheres are the same size; for the default method, the default for the argument, 1, produces spheres all of the same size. The radii are scaled so that their median is 1.
threshold	if the actual size of the spheres is less than the threshold, points are plotted instead.
speed	relative speed of revolution of the plot.
fov	field of view (in degrees); controls degree of perspective.

`fit`	one or more of `"linear"` (linear least squares regression), `"quadratic"` (quadratic least squares regression), `"smooth"` (nonparametric regression), `"additive"` (nonparametric additive regression), `"robust"` (robust linear regression); to display fitted surface(s); partial matching is supported – e.g., `c("lin", "quad")`.
`groups`	if `NULL` (the default), no groups are defined; if a factor, a different surface or set of surfaces is plotted for each level of the factor; in this event, the colours in `surface.col` are used successively for the points, surfaces, and residuals corresponding to each level of the factor.
`parallel`	when plotting surfaces by `groups`, should the surfaces be constrained to be parallel? A logical value, with default `TRUE`.
`ellipsoid`	plot concentration ellipsoid(s) (`TRUE` or `FALSE`).
`level`	expected proportion of bivariate-normal observations included in the concentration ellipsoid(s); default is 0.5.
`ellipsoid.alpha`	transparency of ellipsoids, from `0.0` (fully transparent) to `1.0` (opaque); default is `0.1`.
`id`	`FALSE`, `TRUE`, or a list controlling point identification, similar to `showLabels` for 2D plots (see Details).
`model.summary`	print summary or summaries of the model(s) fit (`TRUE` or `FALSE`). `scatter3d` rescales the three variables internally to fit in the unit cube; this rescaling will affect regression coefficients.
`labels`	text labels for the points, one for each point; defaults to the observation indices.
`col`	colours for the point labels, given by group. There must be at least as many colours as groups; if there are no groups, the first colour is used. Normally, the colours would correspond to the `surface.col` argument to `scatter3d`.
`offset`	vertical displacement for point labels (to avoid overplotting the points).
`reg.function`	an arithmetic expression that is a function of `x` and `z` (respectively, the horizontal and out-of-screen explanatory variables), representing an arbitrary regression function to plot.
`reg.function.col`	color to use for the surface produced by `reg.function`; defaults to the *last* color in `surface.col`.
`mouseMode`	defines what the mouse buttons, etc., do; passed to `par3d` in the **rgl** package; the default in `scatter3d` is the same as in the **rgl** package, except for the left mouse button.
`...`	arguments to be passed down.

Details

The `id` argument to `scatter3d` can be `FALSE`, `TRUE` (in which case 2 points will be identified according to their Mahalanobis distances from the center of the data), or a list containing any or all of the following elements:

method if `"mahal"` (the default), relatively extreme points are identified automatically according to their Mahalanobis distances from the centroid (point of means); if `"identify"`, points

are identified interactively by right-clicking and dragging a box around them; right-click in an empty area to exit from interactive-point-identification mode; if "xz", identify extreme points in the predictor plane; if "y", identify unusual values of the response; if "xyz" identify unusual values of an variable; if "none", no point identification. See showLabels for more information.

n Number of relatively extreme points to identify automatically (default, 2, unless method="identify", in which case identification continues until the user exits).

labels text labels for the points, one for each point; in the default method defaults to the observation indices, in the formula method to the row names of the data.

offset vertical displacement for point labels (to avoid overplotting the points).

Value

scatter3d does not return a useful value; it is used for its side-effect of creating a 3D scatterplot. Identify3d returns the labels of the identified points.

Note

You have to install the rgl package to produce 3D plots. On a Macintosh (but not on Windows or Linux), you may also need to install the X11 windowing system. Go to https://www.xquartz.org/ and click on the link for XQuartz. Double-click on the downloaded disk-image file, and then double-click on XQuartz.pkg to start the installer. You may take all of the defaults in the installation. After XQuartz is installed, you should restart your Macintosh.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

rgl-package, gam

Examples

```
    if(interactive() && require(rgl) && require(mgcv)){
scatter3d(prestige ~ income + education, data=Duncan, id=list(n=3))
Sys.sleep(5) # wait 5 seconds
scatter3d(prestige ~ income + education | type, data=Duncan)
Sys.sleep(5)
scatter3d(prestige ~ income + education | type, surface=FALSE,
 ellipsoid=TRUE, revolutions=3, data=Duncan)
scatter3d(prestige ~ income + education, fit=c("linear", "additive"),
 data=Prestige)
Sys.sleep(5)
scatter3d(prestige ~ income + education | type,
    radius=(1 + women)^(1/3), data=Prestige)
```

```
Sys.sleep(5)
if (require(mvtnorm)){
  local({
    set.seed(123)
    Sigma <- matrix(c(
      1, 0.5,
      0.5, 1),
      2, 2
    )
    X <- rmvnorm(200, sigma=Sigma)
    D <- data.frame(
      x1 = X[, 1],
      x2 = X[, 2]
    )
    D$y <- with(D, 10 + 1*x1 + 2*x2 + 3*x1*x2 + rnorm(200, sd=3))
    # plot true regression function
    scatter3d(y ~ x1 + x2, D,
              reg.function=10 + 1*x + 2*z + 3*x*z,
              fit="quadratic", revolutions=2)
  })
}
 }
 ## Not run:  # requires user interaction to identify points
# drag right mouse button to identify points, click right button in open area to exit
scatter3d(prestige ~ income + education, data=Duncan, id=list(method="identify"))
scatter3d(prestige ~ income + education | type, data=Duncan, id=list(method="identify"))

## End(Not run)
```

`scatterplot` *Enhanced Scatterplots with Marginal Boxplots, Point Marking, Smoothers, and More*

Description

This function uses basic R graphics to draw a two-dimensional scatterplot, with options to allow for plot enhancements that are often helpful with regression problems. Enhancements include adding marginal boxplots, estimated mean and variance functions using either parametric or nonparametric methods, point identification, jittering, setting characteristics of points and lines like color, size and symbol, marking points and fitting lines conditional on a grouping variable, and other enhancements. sp is an abbreviation for `scatterplot`.

Usage

```
scatterplot(x, ...)

## S3 method for class 'formula'
scatterplot(formula, data, subset, xlab, ylab, id=FALSE,
    legend=TRUE, ...)
```

```
## Default S3 method:
scatterplot(x, y, boxplots=if (by.groups) "" else "xy",
        regLine=TRUE, legend=TRUE, id=FALSE, ellipse=FALSE, grid=TRUE,
        smooth=TRUE,
        groups, by.groups=!missing(groups),
        xlab=deparse(substitute(x)), ylab=deparse(substitute(y)),
        log="", jitter=list(), cex=par("cex"),
        col=carPalette()[-1], pch=1:n.groups,
        reset.par=TRUE, ...)

sp(x, ...)
```

Arguments

x	vector of horizontal coordinates (or first argument of generic function).
y	vector of vertical coordinates.
formula	a model formula, of the form y ~ x or, if plotting by groups, y ~ x | z, where z evaluates to a factor or other variable dividing the data into groups. If x is a factor, then parallel boxplots are produced using the Boxplot function.
data	data frame within which to evaluate the formula.
subset	expression defining a subset of observations.
boxplots	if "x" a marginal boxplot for the horizontal x-axis is drawn below the plot; if "y" a marginal boxplot for vertical y-axis is drawn to the left of the plot; if "xy" both marginal boxplots are drawn; set to "" or FALSE to suppress both boxplots.
regLine	controls adding a fitted regression line to the plot. if regLine=FALSE, no line is drawn. If TRUE, the default, an OLS line is fit. This argument can also be a list. The default of TRUE is equivalent to regLine=list(method=lm, lty=1, lwd=2, col=col), which specifies using the lm function to estimate the fitted line, to draw a solid line (lty=1) of width 2 times the nominal width (lwd=2) in the color given by the first element of the col argument described below.
legend	when the plot is drawn by groups and legend=TRUE, controls placement and properties of a legend; if FALSE, the legend is suppressed. Can be a list of named arguments, as follows: title for the legend; inset, giving space as a proportion of the axes to offset the legend from the axes; coords specifying the position of the legend in any form acceptable to the legend function or, if not given, the legend is placed *above* the plot in the upper margin; columns for the legend, determined automatically to prefer a horizontal layout if not given explicitly; cex giving the relative size of the legend symbols and text. TRUE (the default) is equivalent to list(title=deparse(substitute(groups)), inset=0.02, cex=1).
id	controls point identification; if FALSE (the default), no points are identified; can be a list of named arguments to the showLabels function; TRUE is equivalent to list(method="mahal", n=2, cex=1, col=carPalette()[-1], location="lr"), which identifies the 2 points (in each group) with the largest Mahalanobis distances from the center of the data. See showLabels for a description of the other

arguments. The default behavior of id is not the same in all graphics functions in **car**, as the method used depends on the type of plot.

ellipse — controls plotting data-concentration ellipses. If FALSE (the default), no ellipses are plotted. Can be a list of named values giving levels, a vector of one or more bivariate-normal probability-contour levels at which to plot the ellipses; robust, a logical value determing whether to use the cov.trob function in the **MASS** package to calculate the center and covariance matrix for the data ellipses; and fill and fill.alpha, which control whether the ellipse is filled and the transparency of the fill. TRUE is equivalent to list(levels=c(.5, .95), robust=TRUE, fill=TRUE, fill.alpha=0.2).

grid — If TRUE, the default, a light-gray background grid is put on the graph

smooth — specifies a nonparametric estimate of the mean or median function of the vertical axis variable given the horizontal axis variable and optionally a nonparametric estimate of the conditional variance. If smooth=FALSE neither function is drawn. If smooth=TRUE, then both the mean function and variance funtions are drawn for ungrouped data, and the mean function only is drawn for grouped data. The default smoother is loessLine, which uses the loess function from the **stats** package. This smoother is fast and reliable. See the details below for changing the smoother, line type, width and color, of the added lines, and adding arguments for the smoother.

groups — a factor or other variable dividing the data into groups; groups are plotted with different colors, plotting characters, fits, and smooths. Using this argument is equivalent to specifying the grouping variable in the formula.

by.groups — if TRUE (the default when there are groups), regression lines are fit by groups.

xlab — label for horizontal axis.

ylab — label for vertical axis.

log — same as the log argument to plot, to produce log axes.

jitter — a list with elements x or y or both, specifying jitter factors for the horizontal and vertical coordinates of the points in the scatterplot. The jitter function is used to randomly perturb the points; specifying a factor of 1 produces the default jitter. Fitted lines are unaffected by the jitter.

col — with no grouping, this specifies a color for plotted points; with grouping, this argument should be a vector of colors of length at least equal to the number of groups. The default is value returned by carPalette[-1].

pch — plotting characters for points; default is the plotting characters in order (see par).

cex — sets the size of plotting characters, with cex=1 the standard size. You can use cex = 0 to suppress the plotting of points if all you want to show are other graphical features, such as data ellipses, regression lines, smooths, etc. You can also set the sizes of other elements with the arguments cex.axis, cex.lab, cex.main, and cex.sub. See par.

reset.par — if TRUE (the default) then plotting parameters are reset to their previous values when scatterplot exits; if FALSE then the mar and mfcol parameters are altered for the current plotting device. Set to FALSE if you want to add graphical elements (such as lines) to the plot.

... other arguments passed down and to plot. For example, the argument las sets the style of the axis labels, and xlim and ylim set the limits on the horizontal and vertical axes, respectively; see par.

Details

Many arguments to scatterplot were changed in version 3 of **car** to simplify use of this function.

The smooth argument is used to control adding smooth curves to the plot to estimate the conditional center of the vertical axis variable given the horizontal axis variable, and also the conditional variability. Setting smooth=FALSE omits all smoothers, while smooth=TRUE, the default, includes default smoothers. Alternatively smooth can be set to a list of subarguments that provide finer control over the smoothing.

The default behavior of smooth=TRUE is equivalent to smooth=list(smoother=loessLine, var=!by.groups, lty.var=2, lty.var=4, style="filled", alpha=0.15, border=TRUE, vertical=TRUE), specifying the default loessLine smoother for the conditional mean smooth and variance smooth. The color of the smooths is the same of the color of the points, but this can be changed with the arguments col.smooth and col.var.

Additional available smoothers are gamLine which uses the gam function and quantregLine which uses quantile regression to estimate the median and quartile functions using rqss. All of these smoothers have one or more arguments described on their help pages, and these arguments can be added to the smooth argument; for example, smooth = list(span=1/2) would use the default loessLine smoother, include the variance smooth, and change the value of the smoothing parameter to 1/2.

For loessLine and gamLine the variance smooth is estimated by separately smoothing the squared positive and negative residuals from the mean smooth, using the same type of smoother. The displayed curves are equal to the mean smooth plus the square root of the fit to the positive squared residuals, and the mean fit minus the square root of the smooth of the negative squared residuals. The lines therefore represent the comnditional variabiliity at each value on the horizontal axis. Because smoothing is done separately for positive and negative residuals, the variation shown will generally not be symmetric about the fitted mean function. For the quantregLine method, the center estimates the conditional median, and the variability estimates the lower and upper quartiles of the estimated conditional distribution.

The default depection of the variance functions is via a shaded envelope between the upper and lower estimate of variability. setting the subarguement style="lines" will display only the boundaries of this region, and style="none" suppresses variance smoothing.

For style="filled" several subarguments modify the appearance of the region: alpha is a number between 0 and 1 that specifies opacity with defualt 0.15, border, TRUE or FALSE specifies a border for the envelope, and vertical either TRUE or FALSE, modifies the behavior of the envelope at the edges of the graph.

The sub-arguments spread, lty.spread and col.spread of the smooth argument are equivalent to the newer var, col.var and lty.var, respectively, recognizing that the spread is a measuure of conditional variability.

Value

If points are identified, their labels are returned; otherwise NULL is returned invisibly.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

boxplot, jitter, legend, scatterplotMatrix, dataEllipse, Boxplot, cov.trob, showLabels, ScatterplotSmoothers.

Examples

```
scatterplot(prestige ~ income, data=Prestige, ellipse=TRUE,
  smooth=list(style="lines"))

scatterplot(prestige ~ income, data=Prestige,
  smooth=list(smoother=quantregLine))

scatterplot(prestige ~ income, data=Prestige,
  smooth=list(smoother=quantregLine, border="FALSE"))

# use quantile regression for median and quartile fits
scatterplot(prestige ~ income | type, data=Prestige,
  smooth=list(smoother=quantregLine, var=TRUE, span=1, lwd=4, lwd.var=2))

scatterplot(prestige ~ income | type, data=Prestige,
  legend=list(coords="topleft"))

scatterplot(vocabulary ~ education, jitter=list(x=1, y=1),
            data=Vocab, smooth=FALSE, lwd=3)

scatterplot(infantMortality ~ ppgdp, log="xy", data=UN, id=list(n=5))

scatterplot(income ~ type, data=Prestige)

## Not run:  # interactive point identification
    # remember to exit from point-identification mode
    scatterplot(infantMortality ~ ppgdp, id=list(method="identify"), data=UN)

## End(Not run)
```

---

scatterplotMatrix *Scatterplot Matrices*

Description

This function provides a convenient interface to the pairs function to produce enhanced scatterplot matrices, including univariate displays on the diagonal and a variety of fitted lines, smoothers, variance functions, and concentration ellipsoids. spm is an abbreviation for scatterplotMatrix.

Usage

```
scatterplotMatrix(x, ...)

## S3 method for class 'formula'
scatterplotMatrix(formula, data=NULL, subset, ...)

## Default S3 method:
scatterplotMatrix(x, smooth = TRUE,
    id = FALSE, legend = TRUE, regLine = TRUE,
    ellipse = FALSE, var.labels = colnames(x), diagonal = TRUE,
    plot.points = TRUE, groups = NULL, by.groups = TRUE,
    use = c("complete.obs", "pairwise.complete.obs"), col =
    carPalette()[-1], pch = 1:n.groups, cex = par("cex"),
    cex.axis = par("cex.axis"), cex.labels = NULL,
    cex.main = par("cex.main"), row1attop = TRUE, ...)

spm(x, ...)
```

Arguments

x	a data matrix or a numeric data frame.
formula	a one-sided "model" formula, of the form ~ x1 + x2 + ... + xk or ~ x1 + x2 + ... + xk | z where z evaluates to a factor or other variable to divide the data into groups.
data	for scatterplotMatrix.formula, a data frame within which to evaluate the formula.
subset	expression defining a subset of observations.
smooth	specifies a nonparametric estimate of the mean or median function of the vertical axis variable given the horizontal axis variable and optionally a nonparametric estimate of the spread or variance function. If smooth=FALSE neither function is drawn. If smooth=TRUE, then both the mean function and variance funtions are drawn for ungrouped data, and the mean function only is drawn for grouped data. The default smoother is loessLine, which uses the loess function from the stats package. This smoother is fast and reliable. See the details below for changing the smoother, line type, width and color, of the added lines, and adding arguments for the smoother.
id	controls point identification; if FALSE (the default), no points are identified; can be a list of named arguments to the showLabels function; TRUE is equivalent to list(method="mahal", n=2, cex=1, location="lr"), which identifies the 2 points (in each group, if by.groups=TRUE) with the largest Mahalanobis distances from the center of the data; list(method="identify") for interactive point identification is not allowed.
legend	controls placement, point size, and text size of a legend if the plot is drawn by groups; if FALSE, the legend is suppressed. Can be a list with the named element coords specifying the position of the legend in any form acceptable to the legend function, and elements pt.cex and cex corresponding respectively

to the pt.cex and cex arguments of the legend function; TRUE (the default) is equivalent to list(coords=NULL, pt.cex=cex, cex=cex), for which placement will vary by the the value of the diagonal argument—e.g., "topright" for diagonal=TRUE.

regLine — controls adding a fitted regression line to each plot, or to each group of points if by.groups=TRUE. If regLine=FALSE, no line is drawn. This argument can also be a list with named list, with default regLine=TRUE equivalent to regLine = list(method=lm, lty=1, lwd=2, col=col[1]) specifying the name of the function that computes the line, with line type 1 (solid) of relative line width 2 and the color equal to the first value in the argument col. Setting method=MASS::rlm would fit using a robust regression.

ellipse — controls plotting data-concentration ellipses. If FALSE (the default), no ellipses are plotted. Can be a list of named values giving levels, a vector of one or more bivariate-normal probability-contour levels at which to plot the ellipses; robust, a logical value determing whether to use the cov.trob function in the **MASS** package to calculate the center and covariance matrix for the data ellipses; and fill and fill.alpha, which control whether the ellipse is filled and the transparency of the fill. TRUE is equivalent to list(levels=c(.5, .95), robust=TRUE, fill=TRUE, fill.alpha=0.2).

var.labels — variable labels (for the diagonal of the plot).

diagonal — contents of the diagonal panels of the plot. If diagonal=TRUE adaptive kernel density estimates are plotted, separately for each group if grouping is present. diagonal=FALSE suppresses the diagonal entries. See details below for other choices for the diagonal.

plot.points — if TRUE the points are plotted in each off-diagonal panel.

groups — a factor or other variable dividing the data into groups; groups are plotted with different colors and plotting characters.

by.groups — if TRUE, the default, regression lines and smooths are fit by groups.

use — if "complete.obs" (the default), cases with missing data are omitted; if "pairwise.complete.obs"), all valid cases are used in each panel of the plot.

pch — plotting characters for points; default is the plotting characters in order (see par).

col — colors for points; the default is carPalette starting at the second color. The color of the regLine and smooth are the same as for points but can be changed using the the regLine and smooth arguments.

cex — relative size of plotted points. You can use cex = 0.001 to suppress the plotting of points if all you want to show are other graphical features, such as data ellipses, regression lines, smooths, etc.

cex.axis — relative size of axis labels

cex.labels — relative size of labels on the diagonal

cex.main — relative size of the main title, if any

row1attop — If TRUE (the default) the first row is at the top, as in a matrix, as opposed to at the bottom, as in graph (argument suggested by Richard Heiberger).

... — arguments to pass down.

Details

Many arguments to scatterplotMatrix were changed in version 3 of **car**, to simplify use of this function.

The smooth argument is usually either set to TRUE or FALSE to draw, or omit, the smoother. Alternatively smooth can be set to a list of arguments. The default behavior of smooth=TRUE is equivalent to smooth=list(smoother=loessLine, spread=TRUE, lty.smooth=1, lwd.smooth=1.5, lty.spread=3, lwd.spread=1), specifying the smoother to be used, including the spread or variance smooth, and the line widths and types for the curves. You can also specify the colors you want to use for the mean and variance smooths with the arguments col.smooth and col.spread. Alternative smoothers are gamline which uses the gam function from the **mgcv** package, and quantregLine which uses quantile regression to estimate the median and quartile functions using rqss from the **quantreg** package. All of these smoothers have one or more arguments described on their help pages, and these arguments can be added to the smooth argument; for example, smooth = list(span=1/2) would use the default loessLine smoother, include the variance smooth, and change the value of the smoothing parameter to 1/2. For loessLine and gamLine the variance smooth is estimated by separately smoothing the squared positive and negative residuals from the mean smooth, using the same type of smoother. The displayed curves are equal to the mean smooth plus the square root of the fit to the positive squared residuals, and the mean fit minus the square root of the smooth of the negative squared residuals. The lines therefore represent the comnditional variabiliity at each value on the horizontal axis. Because smoothing is done separately for positive and negative residuals, the variation shown will generally not be symmetric about the fitted mean function. For the quantregLine method, the center estimates the median for each value on the horizontal axis, and the spread estimates the lower and upper quartiles of the estimated conditional distribution for each value of the horizontal axis.

The sub-arguments spread, lty.spread and col.spread of the smooth argument are equivalent to the newer var, col.var and lty.var, respectively, recognizing that the spread is a measuure of conditional variability.

By default the diagonal argument is used to draw kernel density estimates of the variables by setting diagonal=TRUE, which is equivalent to setting diagonal = list(method="adaptiveDensity", bw=bw.nrd0, adjust=1, kernel=dnorm, na.rm=TRUE). The additional arguments shown are described at adaptiveKernel. The other methods avaliable, with their default arguments, are diagonal=list(method="densit
bw="nrd0", adjust=1, kernel="gaussian", na.rm=TRUE) which uses density for nonadaptive kernel density estimation; diagonal=list(method ="histogram", breaks="FD") which uses hist for drawing a histogram that ignores grouping, if present; diagonal=list(method="boxplot") with no additional arguments which draws (parallel) boxplots; diagonal=list(method="qqplot") with no additional arguments which draws a normal QQ plot; and diagonal=list(method="oned") with no additional arguments which draws a rug plot tilted to the diagonal, as suggested by Richard Heiberger.

Earlier versions of scatterplotMatrix included arguments transform and family to estimate power transformations using the powerTransform function before drawing the plot. The same functionality can be achieved by calling powerTransform directly to estimate a transformation, saving the transformed variables, and then plotting.

Value

NULL, returned invisibly. This function is used for its side effect: producing a plot. If point identification is used, a vector of identified points is returned.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

pairs, scatterplot, dataEllipse, powerTransform, bcPower, yjPower, cov.trob, showLabels, ScatterplotSmoothers.

Examples

```
scatterplotMatrix(~ income + education + prestige | type, data=Duncan)
scatterplotMatrix(~ income + education + prestige | type, data=Duncan,
    regLine=FALSE, smooth=list(span=1))
scatterplotMatrix(~ income + education + prestige,
    data=Duncan, id=TRUE, smooth=list(method=gamLine))
```

ScatterplotSmoothers *Smoothers to Draw Lines on Scatterplots*

Description

These smoothers are used to draw nonparametric-regression lines on scatterplots produced by the scatterplot, scatterplotMatrix, and several other **car** functions. The functions are not meant to be called directly by the user, although the user can supply options via the smoother.args argument, the contents of which vary by the smoother (see *Details* below). The gamLine smoother uses the gam function in the **mgcv** package, the loessLine smoother uses the loess function in the **stats** package, and the quantregLine smoother uses the rqss function in the **quantreg** package.

Usage

```
gamLine(x, y, col=carPalette()[1], log.x=FALSE, log.y=FALSE, var=FALSE, spread=var,
    smoother.args=NULL, draw=TRUE, offset=0)

loessLine(x, y, col=carPalette()[1], log.x=FALSE, log.y=FALSE, var=FALSE, spread=var,
    smoother.args=NULL, draw=TRUE, offset=0)

quantregLine(x, y, col=carPalette()[1], log.x=FALSE, log.y=FALSE, var=FALSE, spread=var,
    smoother.args=NULL, draw=TRUE, offset=0)
```

Arguments

x	horizontal coordinates of points.
y	vertical coordinates of points.
col	line color.
log.x	should be set to TRUE (default is FALSE) if the horizontal axis is logged.
log.y	should be set to TRUE (default is FALSE) if the vertical axis is logged.
spread, var	the default is to plot only an estimated mean or median function. If either of these arguments is TRUE, then a measure of variability is also plotted.
smoother.args	additional options accepted by the smoother, in the form of a list of named values (see *Details* below).
draw	if TRUE, the default, draw the smoother on the currently active graph. If FALSE, return a list with coordinates x and y for the points that make up the smooth and if requested x.pos, y.pos, x.neg, y.neg for the spread smooths.
offset	For use when spread=TRUE, the vertical axis is sqrt(offset^2 + variance smooth).

Details

The loessLine function is a re-implementation of the loess smoother that was used in **car** prior to September 2012. The main enhancement is the ability to set more options through the smoother.args argument.

The gamLine function is more general than loessLine because it supports fitting a generalized spline regression model, with user-specified error distribution and link function.

The quantregLine function fits a model using splines with estimation based on L1 regression for the median and quantile regression the (optional) spread. It is likely to be more robust than the other smoothers.

The smoother.args argument is a list of named elements (or sub-arguments) used to pass additional arguments to the smoother. As of November, 2016, the smoother is evaluated by default at an equally spaced grid of 50 points in the range of the horizontal variable. With any of the smoothers, you can change to, say, 100 evaluation points via the argument smoother.args=list(evaluation=100). As of version 3.0-1, the smoother.args elements col.var, lty.var, and lwd.var are equivalent to col.spread, lty.spread, and lwd.spread, respectively. The style sub-argument controls how spread/variance envelopes are displayed, with choices "filled" (the default), "lines", and "none" (which is equivalent to var=FALSE). The alpha subargument controls the transparency/opacity of filled spread envelopes with allowable values between 0 and 1 (default 0.15). The border subargument controls whether a border line is drawn around the filled region (the default is TRUE). The vertical subargument controls whether the left and right ends of the filled region are forced to be vertical (the default is TRUE).

For loessLine, the default is smoother.args=list(lty.smooth=1, lwd.smooth=2, lty.spread=4, lwd.spread=2, style="filled", alpha=0.15, span=2/3,degree=1, family="symmetric", iterations=4). (Prior to November 2016, the default span was 1/2.) The elements lty.smooth, lwd.smooth, and col.spread are the line type, line width, and line color, respectively of the mean or median smooth; lty.spread, lwd.spread, and col.spread are the line type, width, and color of the spread smooths, if requested. The elements span, degree, and family are passed as arguments to the loess function, along with iterations robustness iterations.

For gamLine, the default is smoother.args=list(lty.smooth=1, lwd.smooth=2, lty.spread=4, lwd.spread=2, style="filled", alpha=0.15,k=-1, bs="tp", family="gaussian", link=NULL, weights=NULL). The first six elements are as for loessLine. The next two elements are passed to the gam function to control smoothing: k=-1 allows gam to choose the number of splines in the basis function; bs="tp" provides the type of spline basis to be used, with "tp" for the default thin-plate splines. The last three arguments specify a distributional family, link function, and weights as in generalized linear models. See the examples below. The spread element is ignored unless family="gaussian" and link=NULL.

For quantregLine, the default is smoother.args=list(lty.smooth=1, lwd.smooth=2, lty.spread=4, lwd.spread=2, style="filled", alpha=0.15,lambda=IQR(x)). The first six elements are as for loessLine. The last element is passed to the qss function in **quantreg**. It is a smoothing parameter, by default a robust estimate of the scale of the horizontal axis variable. This is an arbitrary choice, and may not work well in all circumstances.

Author(s)

John Fox <jfox@mcmaster.ca> and Sanford Weisberg <sandy@umn.edu>.

See Also

scatterplot, scatterplotMatrix, gam, loess, and rqss.

Examples

```
scatterplot(prestige ~ income, data=Prestige)
scatterplot(prestige ~ income, data=Prestige, smooth=list(smoother=gamLine))
scatterplot(prestige ~ income, data=Prestige,
    smooth=list(smoother=quantregLine))

scatterplot(prestige ~ income | type, data=Prestige)
scatterplot(prestige ~ income | type, data=Prestige,
    smooth=list(smoother=gamLine))
scatterplot(prestige ~ income | type, data=Prestige,
    smooth=list(smoother=quantregLine))
scatterplot(prestige ~ income | type, data=Prestige, smooth=FALSE)

scatterplot(prestige ~ income | type, data=Prestige,
    smooth=list(spread=TRUE))
scatterplot(prestige ~ income | type, data=Prestige,
    smooth=list(smoother=gamLine, spread=TRUE))
scatterplot(prestige ~ income | type, data=Prestige,
    smooth=list(smoother=quantregLine, spread=TRUE))

scatterplot(weight ~ repwt | sex, data=Davis,
    smooth=list(smoother=loessLine, spread=TRUE, style="lines"))
scatterplot(weight ~ repwt | sex, data=Davis,
    smooth=list(smoother=gamLine, spread=TRUE, style="lines")) # messes up
scatterplot(weight ~ repwt | sex, data=Davis,
    smooth=list(smoother=quantregLine, spread=TRUE, style="lines")) #  robust

set.seed(12345)
```

```
w <- 1 + rpois(100, 5)
x <- rnorm(100)
p <- 1/(1 + exp(-(x + 0.5*x^2)))
y <- rbinom(100, w, p)
scatterplot(y/w ~ x, smooth=list(smoother=gamLine, family="binomial",
    weights=w))
scatterplot(y/w ~ x, smooth=list(smoother=gamLine, family=binomial,
    link="probit", weights=w))
scatterplot(y/w ~ x, smooth=list(smoother=loessLine), reg=FALSE)

y <- rbinom(100, 1, p)
scatterplot(y ~ x, smooth=list(smoother=gamLine, family=binomial))
```

---

showLabels *Functions to Identify and Mark Extreme Points in a 2D Plot.*

Description

This function is called by several graphical functions in the **car** package to mark extreme points in a 2D plot. Although the user is unlikely to call this function directly, the documentation below applies to all these other functions.

Usage

```
showLabels(x, y, labels=NULL, method="identify",
  n = length(x), cex=1, col=carPalette()[1], location=c("lr", "ab", "avoid"), ...)
```

Arguments

x	Plotted horizontal coordinates.
y	Plotted vertical coordinates.
labels	Plotting labels. When called from within a **car** plotting function, the labels are automatically obtained from the row names in the data frame used to create the modeling object. If `labels=NULL`, case numbers will be used. If labels are long, the `substr` or `abbreviate` functions can be used to shorten them. Users may supply their own labels as a character vector of length equal to the number of plotted points. For use with **car** plotting functions, the number of plotted points is equal to the number of rows of data that have neither missing values nor are excluded using the 'subset' argument. When called directly, the length of labels shoud equal the length of x.
method	How points are to be identified. See Details below.
n	Number of points to be identified. If set to 0, no points are identified.
cex	Controls the size of the plotted labels. The default is 1.
col	Controls the color of the plotted labels. The default is the first element returned by `carPalette()`.

location	Where should the label be drawn? The default is "lr" to draw the label to the left of the point for points in the right-half of the graph and to the right for points in the left-half. The other option is "ab" for above the point for points below the middle of the graph and above the point below the middle. Finally, "avoid" tries to avoid over-plotting labels.
...	not used.

Details

The argument method determine how the points to be identified are selected. For the default value of method="identify", the identify function is used to identify points interactively using the mouse. Up to n points can be identified, so if n=0, which is the default in many functions in the **car** package, then no point identification is done.

Automatic point identification can be done depending on the value of the argument method.

- method = "x" select points according to their value of abs(x - mean(x))
- method = "y" select points according to their value of abs(y - mean(y))
- method = "r" select points according to their value of abs(y), as may be appropriate in residual plots, or others with a meaningful origin at 0
- method = "mahal" Treat (x, y) as if it were a bivariate sample, and select cases according to their Mahalanobis distance from (mean(x), mean(y))
- method can be a vector of the same length as x consisting of values to determine the points to be labeled. For example, for a linear model m, setting method=cooks.distance(m) will label the points corresponding to the largest values of Cook's distance, or method = which(abs(residuals(m, type="pearson")) > 2 would label all observations with Pearson residuals greater than 2 in absolute value. Warning: If missing data are present, points may be incorrectly labelled.
- method can be a vector of case numbers or case-labels, in which case those cases will be labeled. Warning: If missing data are present, a list of case numbers may identify the wrong points. A list of case labels, however, will work correctly with missing values.
- method = "none" causes no point labels to be shown.

With showLabels, the method argument can be a list, so, for example method=list("x", "y") would label according to the horizontal and vertical axes variables.

Finally, if the axes in the graph are logged, the function uses logged-variables where appropriate.

Value

A function primarily used for its side-effect of drawing point labels on a plot. Returns invisibly the labels of the selected points, or NULL if no points are selected. Although intended for use with other functions in the **car** package, this function can be used directly.

Author(s)

John Fox <jfox@mcmaster.ca>, Sanford Weisberg <sandy@umn.edu>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

avPlots, residualPlots, crPlots, leveragePlots

Examples

```
plot(income ~ education, Prestige)
with(Prestige, showLabels(education, income,
     labels = rownames(Prestige), method=list("x", "y"), n=3))
m <- lm(income ~ education, Prestige)
plot(income ~ education, Prestige)
abline(m)
with(Prestige, showLabels(education, income,
     labels=rownames(Prestige), method=abs(residuals(m)), n=4))
```

---

sigmaHat *Return the scale estimate for a regression model*

Description

This function provides a consistent method to return the estimated scale from a linear, generalized linear, nonlinear, or other model.

Usage

```
sigmaHat(object)
```

Arguments

object	A regression object of type lm, glm or nls

Details

For an lm or nls object, the returned quantity is the square root of the estimate of σ^2. For a glm object, the returned quantity is the square root of the estimated dispersion parameter.

Value

A nonnegative number

Author(s)

Sanford Weisberg, <sandy@umn.edu>

Examples

```
m1 <- lm(prestige ~ income + education, data=Duncan)
sigmaHat(m1)
```

some *Sample a Few Elements of an Object*

Description

Randomly select a few elements of an object, typically a data frame, matrix, vector, or list. If the object is a data frame or a matrix, then rows are sampled.

Usage

```
some(x, ...)

## S3 method for class 'data.frame'
some(x, n=10, cols=NULL, ...)

## S3 method for class 'matrix'
some(x, n=10, cols=NULL, ...)

## Default S3 method:
some(x, n=10, ...)
```

Arguments

x	the object to be sampled.
n	number of elements to sample.
cols	if NULL, use all columns, if a vector of column names or numbers, use only the columns indicated
...	arguments passed down.

Value

Sampled elements or rows.

Note

These functions are adapted from head and tail in the utils package.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

head, tail, brief.

Examples

```
some(Duncan)
some(Duncan, cols=names(Duncan)[1:3])
```

---

spreadLevelPlot *Spread-Level Plots*

Description

Creates plots for examining the possible dependence of spread on level, or an extension of these plots to the studentized residuals from linear models.

Usage

```
spreadLevelPlot(x, ...)

slp(...)

## S3 method for class 'formula'
spreadLevelPlot(x, data=NULL, subset, na.action,
    main=paste("Spread-Level Plot for", varnames[response],
    "by", varnames[-response]), ...)

## Default S3 method:
spreadLevelPlot(x, by, robust.line=TRUE,
 start=0, xlab="Median", ylab="Hinge-Spread",
 point.labels=TRUE, las=par("las"),
 main=paste("Spread-Level Plot for", deparse(substitute(x)),
  "by", deparse(substitute(by))),
 col=carPalette()[1], col.lines=carPalette()[2],
    pch=1, lwd=2, grid=TRUE, ...)

## S3 method for class 'lm'
spreadLevelPlot(x, robust.line=TRUE,
 xlab="Fitted Values", ylab="Absolute Studentized Residuals", las=par("las"),
 main=paste("Spread-Level Plot for\n", deparse(substitute(x))),
 pch=1, col=carPalette()[1], col.lines=carPalette()[2:3], lwd=2, grid=TRUE,
    id=FALSE, smooth=TRUE, ...)

## S3 method for class 'spreadLevelPlot'
print(x, ...)
```

Arguments

x	a formula of the form y ~ x, where y is a numeric vector and x is a factor, or an lm object to be plotted; alternatively a numeric vector.

data	an optional data frame containing the variables to be plotted. By default the variables are taken from the environment from which spreadLevelPlot is called.
subset	an optional vector specifying a subset of observations to be used.
na.action	a function that indicates what should happen when the data contain NAs. The default is set by the na.action setting of options.
by	a factor, numeric vector, or character vector defining groups.
robust.line	if TRUE a robust line is fit using the rlm function in the MASS package; if FALSE a line is fit using lm.
start	add the constant start to each data value.
main	title for the plot.
xlab	label for horizontal axis.
ylab	label for vertical axis.
point.labels	if TRUE label the points in the plot with group names.
las	if 0, ticks labels are drawn parallel to the axis; set to 1 for horizontal labels (see par).
col	color for points; the default is the first entry in the current **car** palette (see carPalette and par).
col.lines	for the default method, the line color, defaulting to the second entry in the **car** color palette; for the "lm" method, a vector of two colors for, respectively, the fitted straight line and a nonparametric regression smooth line, default to the second and third entries in the **car** color palette.
pch	plotting character for points; default is 1 (a circle, see par).
lwd	line width; default is 2 (see par).
grid	If TRUE, the default, a light-gray background grid is put on the graph
id	controls point identification; if FALSE (the default), no points are identified; can be a list of named arguments to the showLabels function; TRUE is equivalent to list(method=list("x", "y"), n=2, cex=1, col=carPalette()[1], location="lr"), which identifies the 2 points the most extreme horizontal ("X", absolute studentized residual) values and the 2 points with the most extreme horizontal ("Y", fitted values) values.
smooth	specifies the smoother to be used along with its arguments; if FALSE, no smoother is shown; can be a list giving the smoother function and its named arguments; TRUE, the default, is equivalent to list(smoother=loessLine). See ScatterplotSmoothers for the smoothers supplied by the **car** package and their arguments.
...	arguments passed to plotting functions.

Details

Except for linear models, computes the statistics for, and plots, a Tukey spread-level plot of log(hinge-spread) vs. log(median) for the groups; fits a line to the plot; and calculates a spread-stabilizing transformation from the slope of the line.

For linear models, plots log(abs(studentized residuals) vs. log(fitted values). Point labeling was added in November, 2016.

The function slp is an abbreviation for spreadLevelPlot.

Value

An object of class `spreadLevelPlot` containing:

`Statistics` a matrix with the lower-hinge, median, upper-hinge, and hinge-spread for each group. (Not for an `lm` object.)

`PowerTransformation`
spread-stabilizing power transformation, calculated as $1 - slope$ of the line fit to the plot.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Hoaglin, D. C., Mosteller, F. and Tukey, J. W. (Eds.) (1983) *Understanding Robust and Exploratory Data Analysis.* Wiley.

See Also

`hccm`, `ncvTest`

Examples

```
spreadLevelPlot(interlocks + 1 ~ nation, data=Ornstein)
slp(lm(interlocks + 1 ~ assets + sector + nation, data=Ornstein))
```

---

strings2factors *Convert Character-String Variables in a Data Frame to Factors*

Description

Converts the character variables (or a subset of these variables) in a data frame to factors, with optional control of the order of the resulting factor levels.

Usage

```
strings2factors(object, which, not, exclude.unique, levels, verbose, ...)
## S3 method for class 'data.frame'
strings2factors(object, which, not,
    exclude.unique=TRUE, levels=list(), verbose=TRUE, ...)
```

Arguments

object	a data frame or an object inheriting from the "data.frame" class.
which	an optional character vector of names or column numbers of the character variables to be converted to factors; if absent, *all* character variables will be converted, except as excluded by the not and exclude.unique arguments (see below).
not	an optional character vector of names or column numbers of character variables *not* to be converted to factors.
exclude.unique	if TRUE (the default), character variables all of whose values are unique (i.e., all different from each other) are not converted to factors. Such variables, which would have as many levels as there are cases, are typically case identifiers and not categorical variables. If FALSE, character variables all of whose values are unique are converted to factors with a warning.
levels	an optional named list, each element of which is a character vector of levels of the corresponding factor. This argument allows you to control the order of levels of the factor; if omitted, or if a particular factor is omitted from the list, the levels will be in the default alphabetic order.
verbose	if TRUE (the default), the names of the character variables that were converted to factors are printed on the console.
...	not used.

Value

a data frame with (some) character variables replaced by corresponding factors.

Author(s)

John Fox <jfox@mcmaster.ca>

See Also

factor, data.frame

Examples

```
M <- Moore # from the carData package
M$partner <- as.character(Moore$partner.status)
M$fcat <- as.character(Moore$fcategory)
M$names <- rownames(M) # values are unique
str(M)
str(strings2factors(M))
str(strings2factors(M,
  levels=list(partner=c("low", "high"), fcat=c("low", "medium", "high"))))
str(strings2factors(M, which="partner", levels=list(partner=c("low", "high"))))
str(strings2factors(M, not="partner", exclude.unique=FALSE))
```

subsets *Plot Output from regsubsets Function in leaps package*

Description

The regsubsets function in the **leaps** package finds optimal subsets of predictors based on some criterion statistic. This function plots a measure of fit against subset size.

Usage

```
subsets(object, ...)

## S3 method for class 'regsubsets'
subsets(object,
    names=abbreviate(object$xnames, minlength = abbrev),
    abbrev=1, min.size=1, max.size=length(names),
    legend="interactive",
    statistic=c("bic", "cp", "adjr2", "rsq", "rss"),
    las=par('las'), cex.subsets=1, ...)
```

Arguments

object	a regsubsets object produced by the regsubsets function in the **leaps** package.
names	a vector of (short) names for the predictors, excluding the regression intercept, if one is present; if missing, these are derived from the predictor names in object.
abbrev	minimum number of characters to use in abbreviating predictor names.
min.size	minimum size subset to plot; default is 1.
max.size	maximum size subset to plot; default is number of predictors.
legend	If not FALSE, in which case the legend is suppressed, the coordinates at which to place a legend of the abbreviated predictor names on the plot, in a form recognized by the legend function. If "interactive", the legend is placed on the plot interactively with the mouse. By expanding the left or right plot margin, you can place the legend in the margin, if you wish (see par).
statistic	statistic to plot for each predictor subset; one of: "bic", Bayes Information Criterion; "cp", Mallows's C_p; "adjr2", R^2 adjusted for degrees of freedom; "rsq", unadjusted R^2; "rss", residual sum of squares.
las	if 0, ticks labels are drawn parallel to the axis; set to 1 for horizontal labels (see par).
cex.subsets	can be used to change the relative size of the characters used to plot the regression subsets; default is 1.
...	arguments to be passed down to subsets.regsubsets and plot.

Value

NULL if the legend is TRUE; otherwise a data frame with the legend.

Author(s)

John Fox

References

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

regsubsets

Examples

```
if (require(leaps)){
    subsets(regsubsets(undercount ~ ., data=Ericksen),
            legend=c(3.5, -37))
}
```

---

symbox *Boxplots for transformations to symmetry*

Description

symbox first transforms x to each of a series of selected powers, with each transformation standardized to mean 0 and standard deviation 1. The results are then displayed side-by-side in boxplots, permiting a visual assessment of which power makes the distribution reasonably symmetric. For the "lm" method, the response variable in the model is successively transformed.

Usage

```
symbox(x, ...)
## S3 method for class 'formula'
symbox(formula, data=NULL, subset, na.action=NULL, ylab,  ...)
## Default S3 method:
symbox(x, powers = c(-1, -0.5, 0, 0.5, 1), start,
 trans=bcPower, xlab="Powers", ylab, ...)
## S3 method for class 'lm'
symbox(x, powers = c(-1, -0.5, 0, 0.5, 1), start, trans=bcPower,
                      xlab, ylab="Studentized residuals", ...)
```

Arguments

x	a numeric vector.
formula	a one-sided formula specifying a single numeric variable.
data, subset, na.action	as for statistical modeling functions (see, e.g., lm).
xlab, ylab	axis labels; if ylab is missing, a label will be supplied. For the "lm" method, if xlab is missing, a label will also be supplied.
powers	a vector of selected powers to which x is to be raised. For meaningful comparison of powers, 1 should be included in the vector of powers.
start	a constant to be added to x. If start is missing and trans is bcPower (the default) or bcnPower, then a start will be automatically generated if there are zero or negative values in x, and a warning will be printed; the auto-generated start is the absolute value of the minimum x plus 1 percent of the range of x.
trans	a transformation function whose first argument is a numeric vector and whose second argument is a transformation parameter, given by the powers argument; the default is bcPower, and another possibility is yjPower. bcnPower may also be used, in which case the gamma parameter is set to the value of start.
...	arguments to be passed down.

Value

as returned by boxplot.

Author(s)

Gregor Gorjanc, John Fox <jfox@mcmaster.ca>.

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition. Sage.

See Also

boxplot, boxcox, bcPower, yjPower

Examples

```
symbox(~ income, data=Prestige)
symbox(lm(wages ~ education + poly(age, 2) + sex, data=SLID))
```

Tapply *Apply a Function to a Variable Within Factor Levels*

Description

Applies a function, typically to compute a single statistic, like a mean, median, or standard deviation, within levels of a factor or within combinations of levels of two or more factors to produce a table of statistics. This function provides a formula interface to the standard R `tapply` function.

Usage

```
Tapply(formula, fun, data, na.action = na.pass, ..., targs = list())
```

Arguments

`formula`	a two-sided formula of the form `variable ~ factor.1 + factor.2 + ... + factor.n` or, equivalently, `variable ~ factor.1*factor.2* ... *factor.n`. The variables on the right-hand side of the formula are normally factors or are otherwise coerced to factors.
`fun`	a function, like `mean`, `median`, or `sd`, that takes a vector first argument and typically returns a single number as its value.
`data`	an optional data frame within which to find the variable and factor(s).
`na.action`	a function to handle missing values, as in statistical modeling functions like `lm`; the default is `na.pass`.
`...`	arguments to pass to the function given in the `fun` argument, such as (commonly) `na.rm=TRUE`.
`targs`	a list of optional arguments to pass to `tapply`.

Details

The function given by `fun` is applied to the values of the left-hand-side variable in `formula` within (combination of) levels of the factor(s) given in the right-hand side of `formula`, producing a table of statistics.

Value

The object returned by `tapply`, typically simply printed.

Author(s)

John Fox `<jfox@mcmaster.ca>`

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition. Sage.

See Also

tapply.

Examples

```
Tapply(conformity ~ partner.status + fcategory, mean, data=Moore)
Tapply(conformity ~ partner.status + fcategory, mean, data=Moore,
    trim=0.2)

Moore[1, 2] <- NA
Tapply(conformity ~ partner.status + fcategory, mean, data=Moore)
Tapply(conformity ~ partner.status + fcategory, mean, data=Moore,
  na.rm=TRUE)
Tapply(conformity ~ partner.status + fcategory, mean, data=Moore,
  na.action=na.omit)  # equivalent
remove("Moore")
```

---

testTransform *Likelihood-Ratio Tests for Univariate or Multivariate Power Transformations to Normality*

Description

testTransform computes likelihood ratio tests for particular values of the power parameter based on powerTransform objects.

Usage

```
testTransform(object, lambda)

## S3 method for class 'powerTransform'
testTransform(object, lambda=rep(1, dim(object$y)[2]))

## S3 method for class 'lmerModpowerTransform'
testTransform(object, lambda=1)

## S3 method for class 'bcnPowerTransformlmer'
testTransform(object, lambda=1)
```

Arguments

object	An object created by a call to powerTransform.
lambda	A vector of powers of length equal to the number of variables transformed.

Details

The function powerTransform is used to estimate a power transformation for a univariate or multivariate sample or multiple linear regression problem, using the method of Box and Cox (1964). It is usual to round the estimates to nearby convenient values, and this function is use to compulte a likelihood ratio test for values of the transformation parameter other than the ml-type estimate.

For one-parameter families of transformations, namely the Box-Cox power family bcPower and the Yeo-Johnson power family yjPower, this function computes a test based on twice the difference in the log-likelihood between the maximum likelihood-like estimate and the log-likelihood evaluated at the value of lambda specified.

For the bcnPower Box-Cox power with negatives allowed, the test is based on the profile loglikelihood maximizing over the location (or gamma) parameter(s). Thus, gamma is treated as a nusiance parameter.

Value

A data frame with one row giving the value of the test statistic, its degrees of freedom, and a p-value. The test is the likelihood ratio test, comparing the value of the log-likelihood at the hypothesized value to the value of the log-likelihood at the maximum likelihood estimate.

Author(s)

Sanford Weisberg, <sandy@umn.edu>

References

Box, G. E. P. and Cox, D. R. (1964) An analysis of transformations. *Journal of the Royal Statisistical Society, Series B*. 26 211-46.

Cook, R. D. and Weisberg, S. (1999) *Applied Regression Including Computing and Graphics*. Wiley.

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Weisberg, S. (2014) *Applied Linear Regression*, Fourth Edition, Wiley.

See Also

powerTransform and bcnPower for examples of the use of this function and other tests that might be of interest in some circumstances.

Examples

```
summary(a3 <- powerTransform(cbind(len, adt, trks, sigs1) ~ htype, Highway1))
# test lambda = (0 0 0 -1)
testTransform(a3, c(0, 0, 0, -1))
summary(q1 <- powerTransform(lm(cbind(LoBD$I1L2, LoBD$I1L1) ~ pool, LoBD), family="bcnPower"))
testTransform(q1, c(.3, .8))
```

TransformationAxes *Axes for Transformed Variables*

Description

These functions produce axes for the original scale of transformed variables. Typically these would appear as additional axes to the right or at the top of the plot, but if the plot is produced with axes=FALSE, then these functions could be used for axes below or to the left of the plot as well.

Usage

```
basicPowerAxis(power, base=exp(1),
    side=c("right", "above", "left", "below"),
    at, start=0, lead.digits=1, n.ticks, grid=FALSE, grid.col=gray(0.50),
    grid.lty=2,
    axis.title="Untransformed Data", cex=1, las=par("las"))

bcPowerAxis(power, side=c("right", "above", "left", "below"),
    at, start=0, lead.digits=1, n.ticks, grid=FALSE, grid.col=gray(0.50),
    grid.lty=2,
    axis.title="Untransformed Data", cex=1, las=par("las"))

bcnPowerAxis(power, shift, side=c("right", "above", "left", "below"),
    at, start=0, lead.digits=1, n.ticks, grid=FALSE, grid.col=gray(0.50),
    grid.lty=2,
    axis.title="Untransformed Data", cex=1, las=par("las"))

yjPowerAxis(power, side=c("right", "above", "left", "below"),
 at, lead.digits=1, n.ticks, grid=FALSE, grid.col=gray(0.50),
  grid.lty=2,
 axis.title="Untransformed Data", cex=1, las=par("las"))

probabilityAxis(scale=c("logit", "probit"),
 side=c("right", "above", "left", "below"),
 at, lead.digits=1, grid=FALSE, grid.lty=2, grid.col=gray(0.50),
    axis.title = "Probability", interval = 0.1, cex = 1, las=par("las"))
```

Arguments

power	power for Box-Cox, Box-Cox with negatives, Yeo-Johnson, or simple power transformation.
shift	the shift (gamma) parameter for the Box-Cox with negatives family.
scale	transformation used for probabilities, "logit" (the default) or "probit".
side	side at which the axis is to be drawn; numeric codes are also permitted: side = 1 for the bottom of the plot, side=2 for the left side, side = 3 for the top, side = 4 for the right side.

at	numeric vector giving location of tick marks on original scale; if missing, the function will try to pick nice locations for the ticks.
start	if a *start* was added to a variable (e.g., to make all data values positive), it can now be subtracted from the tick labels.
lead.digits	number of leading digits for determining 'nice' numbers for tick labels (default is 1.
n.ticks	number of tick marks; if missing, same as corresponding transformed axis.
grid	if TRUE grid lines for the axis will be drawn.
grid.col	color of grid lines.
grid.lty	line type for grid lines.
axis.title	title for axis.
cex	relative character expansion for axis label.
las	if 0, ticks labels are drawn parallel to the axis; set to 1 for horizontal labels (see par).
base	base of log transformation for power.axis when power = 0.
interval	desired interval between tick marks on the probability scale.

Details

The transformations corresponding to the three functions are as follows:

basicPowerAxis: Simple power transformation, $x' = x^p$ for $p \neq 0$ and $x' = \log x$ for $p = 0$.

bcPowerAxis: Box-Cox power transformation, $x' = (x^\lambda - 1)/\lambda$ for $\lambda \neq 0$ and $x' = \log x$ for $\lambda = 0$.

bcnPowerAxis: Box-Cox with negatives power transformation, the Box-Cox power transformation of $z = .5 * (y + (y^2 + \gamma^2)^{1/2})$, where γ is strictly positive if y includes negative values and non-negative otherwise. The value of z is always positive.

yjPowerAxis: Yeo-Johnson power transformation, for non-negative x, the Box-Cox transformation of $x + 1$; for negative x, the Box-Cox transformation of $|x| + 1$ with power $2 - p$.

probabilityAxis: logit or probit transformation, logit $= \log[p/(1-p)]$, or probit $= \Phi^{-1}(p)$, where Φ^{-1} is the standard-normal quantile function.

These functions will try to place tick marks at reasonable locations, but producing a good-looking graph sometimes requires some fiddling with the at argument.

Value

These functions are used for their side effects: to draw axes.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

See Also

basicPower, bcPower, yjPower, logit.

Examples

```
UN <- na.omit(UN)
par(mar=c(5, 4, 4, 4) + 0.1) # leave space on right

with(UN, plot(log(ppgdp, 10), log(infantMortality, 10)))
basicPowerAxis(0, base=10, side="above",
  at=c(50, 200, 500, 2000, 5000, 20000), grid=TRUE,
  axis.title="GDP per capita")
basicPowerAxis(0, base=10, side="right",
  at=c(5, 10, 20, 50, 100), grid=TRUE,
  axis.title="infant mortality rate per 1000")

with(UN, plot(bcPower(ppgdp, 0), bcPower(infantMortality, 0)))
bcPowerAxis(0, side="above",
  grid=TRUE, axis.title="GDP per capita")
bcPowerAxis(0, side="right",
  grid=TRUE, axis.title="infant mortality rate per 1000")

with(UN, qqPlot(logit(infantMortality/1000)))
probabilityAxis()

with(UN, qqPlot(qnorm(infantMortality/1000)))
probabilityAxis(at=c(.005, .01, .02, .04, .08, .16), scale="probit")

qqPlot(bcnPower(Ornstein$interlocks, lambda=1/3, gamma=0.1))
bcnPowerAxis(1/3, 0.1, at=c(o=0, 5, 10, 20, 40, 80))
```

---

vif *Variance Inflation Factors*

Description

Calculates variance-inflation and generalized variance-inflation factors (VIFs and GVIFs) for linear, generalized linear, and other regression models.

Usage

```
vif(mod, ...)

## Default S3 method:
vif(mod, ...)

## S3 method for class 'lm'
vif(mod, type=c("terms", "predictor"), ...)
```

```
## S3 method for class 'merMod'
vif(mod, ...)

## S3 method for class 'polr'
vif(mod, ...)

## S3 method for class 'svyolr'
vif(mod, ...)
```

Arguments

mod	for the default method, an object that responds to coef, vcov, and model.matrix, such as a glm object.
type	for unweighted lm objects only, how to handle models that contain interactions: see Details below.
...	not used.

Details

If all terms in an unweighted linear model have 1 df, then the usual variance-inflation factors are calculated.

If any terms in an unweighted linear model have more than 1 df, then generalized variance-inflation factors (Fox and Monette, 1992) are calculated. These are interpretable as the inflation in size of the confidence ellipse or ellipsoid for the coefficients of the term in comparison with what would be obtained for orthogonal data.

The generalized VIFs are invariant with respect to the coding of the terms in the model (as long as the subspace of the columns of the model matrix pertaining to each term is invariant). To adjust for the dimension of the confidence ellipsoid, the function also prints $GVIF^{1/(2\times df)}$ where df is the degrees of freedom associated with the term.

Through a further generalization, the implementation here is applicable as well to other sorts of models, in particular weighted linear models, generalized linear models, and mixed-effects models: Rather than computing the GVIFs from the correlation matrix of the columns of the model matrix (disregarding the regression constant), vif() uses the correlation matrix of the coefficients. Using either correlation matrix produces the same result for a linear model fit by least-squares (as pointed out by Henric Nilsson; see Fox, 2016, Exercise 13.5), but using the correlation matrix of the coefficients works with other regression models, such as GLMs, while using the correlations among the columns of the model matrix does not.

Two methods of computing GVIFs are provided for unweighted linear models:

- Setting type="terms" (the default) behaves like the default method, and computes the GVIF for each term in the model, ignoring relations of marginality among the terms in models with interactions. GVIFs computed in this manner aren't generally sensible.
- Setting type="predictor" focuses in turn on each predictor in the model, combining the main effect for that predictor with the main effects of the predictors with which the focal predictor interacts and the interactions; e.g., in the model with formula y ~ a*b + b*c, the GVIF for the predictor a also includes the b main effect and the a:b interaction regressors; the GVIF for the predictor c includes the b main effect and the b:c interaction; and the GVIF

for the predictor b includes the a and c main effects and the a:b and a:c interactions (i.e., the whole model), and is thus necessarily 1. These predictor GVIFs should be regarded as experimental.

Specific methods are provided for ordinal regression model objects produced by polr in the **MASS** package and svyolr in the **survey** package, which are "intercept-less"; VIFs or GVIFs for linear and similar regression models without intercepts are generally not sensible.

Value

A vector of VIFs, or a matrix containing one row for each term, and columns for the GVIF, df, and $GVIF^{1/(2\times df)}$, the last of which is intended to be comparable across terms of different dimension.

Author(s)

John Fox <jfox@mcmaster.ca> and Henric Nilsson

References

Fox, J. and Monette, G. (1992) Generalized collinearity diagnostics. *JASA*, **87**, 178–183.

Fox, J. (2016) *Applied Regression Analysis and Generalized Linear Models*, Third Edition. Sage.

Fox, J. and Weisberg, S. (2018) *An R Companion to Applied Regression*, Third Edition, Sage.

Examples

```
vif(lm(prestige ~ income + education, data=Duncan))
vif(lm(prestige ~ income + education + type, data=Duncan))
vif(lm(prestige ~ (income + education)*type, data=Duncan),
    type="terms") # not recommended
vif(lm(prestige ~ (income + education)*type, data=Duncan),
    type="predictor")
```

---

wcrossprod *Weighted Matrix Crossproduct*

Description

Given matrices x and y as arguments and an optional matrix or vector of weights, w, return a weighted matrix cross-product, t(x) w y. If no weights are supplied, or the weights are constant, the function uses crossprod for speed.

Usage

```
wcrossprod(x, y, w)
```

Arguments

x, y	x, y numeric matrices; missing(y) is taken to be the same matrix as x. Vectors are promoted to single-column or single-row matrices, depending on the context.
w	A numeric vector or matrix of weights, conformable with x and y.

Value

A numeric matrix, with appropriate dimnames taken from x and y.

Author(s)

Michael Friendly, John Fox <jfox@mcmaster.ca>

See Also

crossprod

Examples

```
set.seed(12345)
n <- 24
drop <- 4
sex <- sample(c("M", "F"), n, replace=TRUE)
x1 <- 1:n
x2 <- sample(1:n)
extra <- c( rep(0, n - drop), floor(15 + 10 * rnorm(drop)) )
y1 <- x1 + 3*x2 + 6*(sex=="M") + floor(10 * rnorm(n)) + extra
y2 <- x1 - 2*x2 - 8*(sex=="M") + floor(10 * rnorm(n)) + extra
# assign non-zero weights to 'dropped' obs
wt <- c(rep(1, n-drop), rep(.2,drop))

X <- cbind(x1, x2)
Y <- cbind(y1, y2)
wcrossprod(X)
wcrossprod(X, w=wt)

wcrossprod(X, Y)
wcrossprod(X, Y, w=wt)

wcrossprod(x1, y1)
wcrossprod(x1, y1, w=wt)
```

---

whichNames *Position of Row Names*

Description

These functions return the indices of the supplied row names of a data frame or names of another object, such as a vector or list. whichNames is just an alias for which.names.

Usage

```
whichNames(names, object, ...)

which.names(names, object, ...)

## S3 method for class 'data.frame'
whichNames(names, object, ...)

## Default S3 method:
whichNames(names, object, ...)
```

Arguments

names	a name or character vector of names.
object	a data frame or an object with a names attribute.
...	not used.

Value

Returns the index or indices of names in row names of the data frame or names of an object of another class.

Author(s)

John Fox <jfox@mcmaster.ca>

References

Fox, J. and Weisberg, S. (2019) *An R Companion to Applied Regression*, Third Edition, Sage.

Examples

```
whichNames(c('minister', 'conductor'), Duncan)
```

Index

∗ **algebra**
wcrossprod, 153
∗ **aplot**
Ellipses, 53
panel.car, 95
pointLabel, 96
regLine, 110
TransformationAxes, 149
∗ **array**
wcrossprod, 153
∗ **color**
carPalette, 32
∗ **connections**
Export, 58
Import, 64
∗ **distribution**
qqPlot, 104
∗ **hplot**
avPlots, 11
Boxplot, 23
ceresPlots, 34
crPlots, 39
densityPlot, 47
dfbetaPlots, 50
Ellipses, 53
infIndexPlot, 66
invResPlot, 71
invTranPlot, 73
leveragePlots, 76
mcPlots, 86
mmps, 89
residualPlots, 111
scatter3d, 119
scatterplot, 124
scatterplotMatrix, 128
ScatterplotSmoothers, 132
spreadLevelPlot, 139
subsets, 143
symbox, 144
∗ **htest**
Anova, 3
leveneTest, 75
linearHypothesis, 78
ncvTest, 92
outlierTest, 94
poTest, 98
∗ **interface**
carWeb, 33
∗ **manip**
boxCoxVariable, 22
brief, 26
logit, 85
recode, 107
strings2factors, 141
Tapply, 146
∗ **misc**
car-internal.Rd, 31
carHexsticker, 32
Tapply, 146
∗ **models**
Anova, 3
Contrasts, 37
deltaMethod, 43
influence.mixed.models, 67
linearHypothesis, 78
poTest, 98
Predict, 103
S, 114
∗ **print**
compareCoefs, 36
∗ **regression**
Anova, 3
avPlots, 11
bcPower, 14
Boot, 17
boxCox, 19
boxCoxVariable, 22
boxTidwell, 25

car-deprecated, 31
ceresPlots, 34
Contrasts, 37
crPlots, 39
deltaMethod, 43
dfbetaPlots, 50
durbinWatsonTest, 52
hccm, 59
hist.boot, 61
infIndexPlot, 66
influencePlot, 70
invResPlot, 71
invTranPlot, 73
leveragePlots, 76
linearHypothesis, 78
mcPlots, 86
mmps, 89
ncvTest, 92
outlierTest, 94
powerTransform, 99
qqPlot, 104
residualPlots, 111
S, 114
sigmaHat, 137
spreadLevelPlot, 139
subsets, 143
testTransform, 147
vif, 151
∗ **ts**
durbinWatsonTest, 52
∗ **univar**
qqPlot, 104
∗ **utilities**
Export, 58
Import, 64
showLabels, 135
some, 138
whichNames, 154
.carEnv *(*car-internal.Rd*)*, 31

abbreviate, *135*
abline, *110*
adaptiveKernel, *131*
adaptiveKernel *(*densityPlot*)*, 47
alpha, *70, 71*
Anova, 3, *82, 83*
anova, *9*, *83*
anova.coxph, *9*
anova.glm, *9*
anova.lm, *9*
anova.mlm, *9*
as.data.frame.univaov *(*Anova*)*, 3
av.plot *(*car-defunct*)*, 29
av.plots *(*car-defunct*)*, 29
avp *(*avPlots*)*, 11
avPlot, *30*
avPlot *(*avPlots*)*, 11
avPlot3d *(*avPlots*)*, 11
avPlots, 11, *30*, *36*, *42*, *77*, *89*, *137*

basicPower, *151*
basicPower *(*bcPower*)*, 14
basicPowerAxis *(*TransformationAxes*)*, 149
bc *(*car-defunct*)*, 29
bcnPower, *21, 22*, *30*, *102*, *145*, *148*
bcnPower *(*bcPower*)*, 14
bcnPowerAxis *(*TransformationAxes*)*, 149
bcnPowerInverse *(*bcPower*)*, 14
bcPower, 14, *22, 23*, *30*, *99*, *102*, *132*, *145*, *148*, *151*
bcPowerAxis *(*TransformationAxes*)*, 149
Boot, 17, *28*, *31*, *64*, *119*
boot, *18*, *61*, *63*
boot.array, *19*
boot.ci, *19*, *63*
bootCase *(*car-deprecated*)*, 31
box.cox *(*car-defunct*)*, 29
box.tidwell *(*car-defunct*)*, 29
boxCox, 19, *102*
boxcox, *21–23*, *145*
boxCox2d *(*boxCox*)*, 19
boxCoxVariable, 22, *30*
Boxplot, 23, *125*, *128*
boxplot, *23, 24*, *128*, *145*
boxTidwell, 25, *30*
brief, 26, *114*, *117*, *119*, *138*
bw.nrd0, *48*
bw.SJ, *48, 49*

car-defunct, 29
car-deprecated, 31
car-internal.Rd, 31
carHexsticker, 32
carPalette, *12*, 32, *35*, *41*, *48*, *51*, *56*, *70, 71*, *74*, *87*, *91*, *106*, *110*, *126*, *130*, *140*
carWeb, 33
ceres.plot *(*car-defunct*)*, 29
ceres.plots *(*car-defunct*)*, 29

ceresPlot, *30*
ceresPlot *(*ceresPlots*)*, 34
ceresPlots, *14*, *30*, 34, *42*, *89*
coef, *44*, *83*, *114*, *118*, *119*, *152*
colors, *33*
compareCoefs, 36
confidence.ellipse *(*car-defunct*)*, 29
confidenceEllipse, *30*
confidenceEllipse *(*Ellipses*)*, 53
confidenceEllipses *(*Ellipses*)*, 53
Confint *(*S*)*, 114
confint, *114*, *117*, *119*
Confint.boot *(*hist.boot*)*, 61
confint.boot, *19*
confint.boot *(*hist.boot*)*, 61
confint.lm, *118*
contour, *22*
contr.Helmert *(*Contrasts*)*, 37
contr.helmert, *9*, *38*
contr.poly, *9*, *38*
contr.Sum *(*Contrasts*)*, 37
contr.sum, *9*, *38*
contr.Treatment *(*Contrasts*)*, 37
contr.treatment, *9*, *38*
Contrasts, 37
cookd *(*car-defunct*)*, 29
cooks.distance, *30*, *67*, *68*, *71*
cooks.distance.influence.lme *(*influence.mixed.models*)*, 67
coplot, *96*
cov, *63*
cov.trob, *13*, *55*, *57*, *126*, *128*, *130*, *132*
cov.wt, *55*, *57*
cr.plot *(*car-defunct*)*, 29
cr.plots *(*car-defunct*)*, 29
crossprod, *153*, *154*
crp *(*crPlots*)*, 39
crPlot, *30*
crPlot *(*crPlots*)*, 39
crPlot3d *(*crPlots*)*, 39
crPlots, *14*, *30*, *36*, 39, *89*, *137*
cut, *109*

D, *46*
data.ellipse *(*car-defunct*)*, 29
data.frame, *142*
dataEllipse, *14*, *30*, *88*, *89*, *128*, *132*
dataEllipse *(*Ellipses*)*, 53
dbiwt *(*densityPlot*)*, 47
deltaMethod, 43
density, *47–49*, *64*, *131*
densityPlot, 47
depan *(*densityPlot*)*, 47
detectCores, *68*
dfbeta, *51*, *68*
dfbeta.influence.lme *(*influence.mixed.models*)*, 67
dfbetaPlots, 50
dfbetas, *51*, *68*
dfbetas.influence.lme *(*influence.mixed.models*)*, 67
dfbetasPlots *(*dfbetaPlots*)*, 50
durbin.watson *(*car-defunct*)*, 29
durbinWatsonTest, *30*, 52
dwt *(*durbinWatsonTest*)*, 52

ellipse *(*Ellipses*)*, 53
Ellipses, 53
empinf, *19*
Export, 58, *65*
export, *58*, *59*

factor, *109*, *142*

gam, *91*, *121*, *123*, *127*, *131*, *132*, *134*
gamLine, *127*
gamLine *(*ScatterplotSmoothers*)*, 132
glm, *114*

hatvalues, *67*, *71*
hccm, *6*, *28*, *55*, 59, *81*, *83*, *93*, *103*, *117–119*, *141*
head, *138*
hist, *64*, *131*
hist.boot, *19*, 61

identify, *114*, *136*
Identify3d, *12*, *41*
Identify3d *(*scatter3d*)*, 119
Import, *59*, 64
import, *64*, *65*
infIndexPlot, 66, *69*
influence, *68*
influence.lme *(*influence.mixed.models*)*, 67
influence.mixed.models, *66*, 67, *67*
influence.plot *(*influencePlot*)*, 70
influenceIndexPlot *(*infIndexPlot*)*, 66

influencePlot, 70
inverseResponsePlot, *74*
inverseResponsePlot *(*invResPlot*)*, 71
invResPlot, 71
invTranEstimate *(*invTranPlot*)*, 73
invTranPlot, *72*, 73

jitter, *126*, *128*

legend, *48*, *125*, *128–130*, *143*
levene.test *(*car-defunct*)*, 29
leveneTest, *30*, 75
leverage.plot *(*car-defunct*)*, 29
leverage.plots *(*car-defunct*)*, 29
leveragePlot, *30*
leveragePlot *(*leveragePlots*)*, 76
leveragePlots, *30*, 76, *137*
lht *(*linearHypothesis*)*, 78
linear.hypothesis *(*car-defunct*)*, 29
linearHypothesis, *9*, *30*, *57*, 78, *118, 119*
lines, *110*
lm, *24*, *73*, *100*, *114*, *117*, *145, 146*
lme, *68, 69*
loess, *41*, *126*, *129*, *132–134*
loessLine, *126, 127*, *129*
loessLine *(*ScatterplotSmoothers*)*, 132
logit, 85, *151*

makeHypothesis *(*linearHypothesis*)*, 78
Manova *(*Anova*)*, 3
marginalModelPlot *(*mmps*)*, 89
marginalModelPlots *(*mmps*)*, 89
matchCoefs *(*linearHypothesis*)*, 78
mcPlot *(*mcPlots*)*, 86
mcPlots, 86
mgcv, *41*
mmp *(*mmps*)*, 89
mmps, 89
model.matrix, *152*
multinom, *82*, *114*

na.omit, *47*
na.pass, *146*
ncv.test *(*car-defunct*)*, 29
ncvTest, *30*, 92, *141*
nextBoot *(*car-deprecated*)*, 31

optim, *102*
optimize, *74*
outlier.test *(*car-defunct*)*, 29
outlierTest, *30*, *67*, 94

p.adjust, *7*
pairs, *132*
palette, *33*
panel.car, 95
par, *12, 13*, *35*, *41*, *56*, *77*, *88*, *106*, *110*, *126, 127*, *130*, *140*, *143*, *150*
par3d, *122*
plot, *51*, *92*, *126*
plot.boot, *19*
plot.density, *49*
pointLabel, 96
points, *51*
polr, *98*, *114*, *153*
poTest, 98
powerTransform, *16*, *21–23*, *30*, *72*, 99, *131, 132*, *148*
Predict, 103
predict, *104*
predict.lm, *103, 104*
print.Anova.mlm *(*Anova*)*, 3
print.boxTidwell *(*boxTidwell*)*, 25
print.durbinWatsonTest *(*durbinWatsonTest*)*, 52
print.linearHypothesis.mlm *(*linearHypothesis*)*, 78
print.outlierTest *(*outlierTest*)*, 94
print.poTest *(*poTest*)*, 98
print.S.glm *(*S*)*, 114
print.S.glmerMod *(*S*)*, 114
print.S.lm *(*S*)*, 114
print.S.lme *(*S*)*, 114
print.S.lmerMod *(*S*)*, 114
print.S.multinom *(*S*)*, 114
print.S.polr *(*S*)*, 114
print.spreadLevelPlot *(*spreadLevelPlot*)*, 139
print.summary.Anova.mlm *(*Anova*)*, 3
print.univaov *(*Anova*)*, 3
printCoefmat, *37*
printHypothesis *(*linearHypothesis*)*, 78
probabilityAxis, *86*
probabilityAxis *(*TransformationAxes*)*, 149

qq.plot *(*car-defunct*)*, 29
qqline, *107*

qqnorm, *104*, *107*
qqp *(qqPlot)*, 104
qqPlot, *30*, 104
qqplot, *107*
qss, *134*
quantregLine, *127*
quantregLine *(ScatterplotSmoothers)*, 132

Recode *(recode)*, 107
recode, 107
regLine, *96*, 110
regsubsets, *143, 144*
regular expression, *81*
residCurvTest *(residualPlots)*, 111
residualPlot *(residualPlots)*, 111
residualPlots, *14*, *89*, 111, *137*
rqss, *127*, *131, 132*, *134*
rstandard, *112*
rstudent, *67*, *71*, *112*

S, *29*, 114
scatter3d, *13*, *41, 42*, 119
scatterplot, 124, *132*, *134*
scatterplotMatrix, 128, *128*, *132*, *134*
ScatterplotSmoothers, *35*, *41*, *91, 92*, *113*, *128*, 132, *132*, *140*
showLabels, *12*, *14*, *24*, *35, 36*, *41*, *51*, *56*, *66*, *71, 72*, *74*, *77*, *87*, *91*, *106, 107*, *113, 114*, *122, 123*, *125*, *128, 129*, *132*, 135, *140*
sigmaHat, 137
skewPower *(car-defunct)*, 29
slp *(spreadLevelPlot)*, 139
some, 138
sp *(scatterplot)*, 124
spm *(scatterplotMatrix)*, 128
spread.level.plot *(car-defunct)*, 29
spreadLevelPlot, *30*, *93*, 139
strings2factors, *65*, 141
subsets, 143
substr, *135*
summary, *114*, *119*
summary.Anova.mlm *(Anova)*, 3
summary.boot, *19*
summary.boot *(hist.boot)*, 61
summary.glm, *28*, *117, 118*
summary.lm, *117–119*
summary.multinom, *118*
svyglm, *9*
svyolr, *153*
symbox, 144

tail, *138*
Tapply, 146
tapply, *146, 147*
testTransform, *16*, *101, 102*, 147
text, *97*
thigmophobe.labels, *97*
transform, *102*
TransformationAxes, 149
tukeyNonaddTest *(residualPlots)*, 111

vcov, *44*, *83*, *118*, *152*
vcov.boot *(hist.boot)*, 61
vcov.lm, *119*
vcovHAC, *83*
vcovHC, *83*
vif, 151

waldtest, *83*
wcrossprod, 153
which.names *(whichNames)*, 154
whichNames, 154

yjPower, *22*, *101*, *132*, *145*, *148*, *151*
yjPower *(bcPower)*, 14
yjPowerAxis *(TransformationAxes)*, 149