Package ‘arm’

July 27, 2020

Version 1.11-2
Date 2020-7-27
Title Data Analysis Using Regression and Multilevel/Hierarchical Models
Author Andrew Gelman [aut],
   Yu-Sung Su [aut, cre],
   Masanao Yajima [ctb],
   Jennifer Hill [ctb],
   Maria Grazia Pittau [ctb],
   Jouni Kerman [ctb],
   Tian Zheng [ctb],
   Vincent Dorie [ctb]
Maintainer Yu-Sung Su <suyusung@tsinghua.edu.cn>
BugReports https://github.com/suyusung/arm/issues/
Depends R (>= 3.1.0), MASS, Matrix (>= 1.0), stats, lme4 (>= 1.0)
Imports abind, coda, graphics, grDevices, Hmisc, methods, nlme, utils
License GPL (> 2)
URL https://CRAN.R-project.org/package=arm
NeedsCompilation no
Repository CRAN
Date/Publication 2020-07-27 05:10:29 UTC

R topics documented:

balance ................................................................. 2
bayesglm .............................................................. 4
bayespolr ............................................................. 10
binnedplot ........................................................... 13
coefplot .............................................................. 15
balance

Functions to compute the balance statistics

Description

This function computes the balance statistics before and after matching.

Usage

balance(rawdata, treat, matched, estimand="ATT")

## S3 method for class 'balance'
print(x, ..., combined = FALSE, digits = 2)

## S3 method for class 'balance'
plot(x, longcovnames=NULL, which.covs="mixed",
    v.axis=TRUE, cex.main=1, cex.vars=1, cex.pts=1,
    mar=c(4, 3, 5.1, 2), plot=TRUE, x.max = NULL, ...)
Arguments

rawdata    The full covariate dataset
treat     the vector of treatment assignments for the full dataset
matched   vector of weights to apply to the full dataset to create the restructured data: for matching without replacement these will all be 0’s and 1’s; for one-to-one matching with replacement these will all be non-negative integers; for IPTW or more complicated matching methods these could be any non-negative numbers
estimand  can either be ATT, ATC, or ATE, default is ATT
x        an object return by the balance function.
combined default is FALSE
digits    minimal number of significant digits, default is 2.
longcovnames long covariate names. If not provided, plot will use covariate variable name by default
which.covs mixed then it plots all as std diffs; binary it only plots binary and as abs unstd diffs; cont it only plots non-binary and as abs std diffs
v.axis    default is TRUE, which shows the top axis–axis(3).
cex.main   font size of main title
cex.vars   font size of variable names
cex.pts    point size of the estimates
mar        A numerical vector of the form c(bottom,left,top,right) which gives the number of lines of margin to be specified on the four sides of the plot. The default is c(0,3,5.1,2).
plot      default is TRUE, which will plot the plot.
x.max     set the max of the xlim, default is NULL
...       other plot options may be passed to this function

Details

This function plots the balance statistics before and after matching. The open circle dots represent the unmatched balance statistics. The solid dots represent the matched balance statistics. The closer the value of the estimates to the zero, the better the treated and control groups are balanced after matching.

Note

The function does not work with predictors that contain factor(x), log(x) or all other data transformation. Create new objects for these variables. Attach them into the original dataset before doing the matching procedure.

Author(s)

Jennifer Hill <jennifer.hill@nyu.edu>; Yu-Sung Su <suyusung@tsinghua.edu.cn>
References


See Also

matching, par

Examples

```r
# matching first
old.par <- par(no.readonly = TRUE)
data(lalonde)
attach(lalonde)
fit <- glm(treat ~ re74 + re75 + age + factor(educ) +
   black + hisp + married + nodegr + u74 + u75,
   family=binomial(link="logit"))
pscores <- predict(fit, type="link")
matches <- matching(z=lalonde$treat, score=pscores)
mixed <- matches$cnts

# balance check
b.stats <- balance(lalonde, treat, matched, estimand = "ATT")
print(b.stats)
plot(b.stats)
par(old.par)
```

---

bayesglm  

Bayesian generalized linear models.

Description

Bayesian functions for generalized linear modeling with independent normal, t, or Cauchy prior distribution for the coefficients.

Usage

```r
bayesglm(formula, family = gaussian, data, weights, subset, na.action,
    start = NULL, etastart, mustart, offset, control = list(...),
    model = TRUE, method = "glm.fit",
    x = FALSE, y = TRUE, contrasts = NULL, drop.unused.levels = TRUE,
    prior.mean = 0,
    prior.scale = NULL,
    prior.df = 1,
    prior.mean.for.intercept = 0,
```
bayesglm

prior.scale.for.intercept = NULL,
prior.df.for.intercept = 1,
min.prior.scale=1e-12,
scaled = TRUE, keep.order=TRUE,
drop.baseline=TRUE,
maxit=100,
print.unnormalized.log.posterior=FALSE,
Warning=TRUE,...)

bayesglm.fit (x, y, weights = rep(1, nobs),
start = NULL, etastart = NULL,
mustart = NULL, offset = rep(0, nobs), family = gaussian(),
control = list(), intercept = TRUE,
prior.mean = 0,
prior.scale = NULL,
prior.df = 1,
prior.mean.for.intercept = 0,
prior.scale.for.intercept = NULL,
prior.df.for.intercept = 1,
min.prior.scale=1e-12, scaled = TRUE,
print.unnormalized.log.posterior=FALSE, Warning=TRUE)

Arguments

formula a symbolic description of the model to be fit. The details of model specification are given below.

family a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.)

data an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which glm is called.

weights an optional vector of weights to be used in the fitting process. Should be NULL or a numeric vector.

subset an optional vector specifying a subset of observations to be used in the fitting process.

na.action a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The “factory-fresh” default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.

start starting values for the parameters in the linear predictor.

etastart starting values for the linear predictor.

mustart starting values for the vector of means.

offset this can be used to specify an a priori known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length
either one or equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if both are specified their sum is used. See model.offset.

control

a list of parameters for controlling the fitting process. See the documentation for glm.control for details.

model

a logical value indicating whether model frame should be included as a component of the returned value.

method

the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). The only current alternative is "model.frame" which returns the model frame and does no fitting.

x, y

For glm: logical values indicating whether the response vector and model matrix used in the fitting process should be returned as components of the returned value.

For glm.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.

contrasts

an optional list. See the contrasts.arg of model.matrix.default.

drop.unused.levels

default TRUE; if FALSE, it interpolates the intermediate values if the data have integer levels.

intercept

logical. Should an intercept be included in the null model?

prior.mean

prior mean for the coefficients: default is 0. Can be a vector of length equal to the number of predictors (not counting the intercept, if any). If it is a scalar, it is expanded to the length of this vector.

prior.scale

prior scale for the coefficients: default is NULL; if is NULL, for a logit model, prior.scale is 2.5; for a probit model, prior scale is 2.5*1.6. Can be a vector of length equal to the number of predictors (not counting the intercept, if any). If it is a scalar, it is expanded to the length of this vector.

prior.df

prior degrees of freedom for the coefficients. For t distribution: default is 1 (Cauchy). Set to Inf to get normal prior distributions. Can be a vector of length equal to the number of predictors (not counting the intercept, if any). If it is a scalar, it is expanded to the length of this vector.

prior.mean.for.intercept

prior mean for the intercept: default is 0. See ‘Details’.

prior.scale.for.intercept

prior scale for the intercept: default is NULL; for a logit model, prior scale for intercept is 10; for probit model, prior scale for intercept is rescaled as 10*1.6.

prior.df.for.intercept

prior degrees of freedom for the intercept: default is 1.

min.prior.scale

Minimum prior scale for the coefficients: default is 1e-12.

scaled

scaled=TRUE, the scales for the prior distributions of the coefficients are determined as follows: For a predictor with only one value, we just use prior.scale. For a predictor with two values, we use prior.scale/range(x). For a predictor with more than two values, we use prior.scale/(2*sd(x)). If the response is Gaussian, prior.scale is also multiplied by 2 * sd(y). Default is TRUE.
keep.order  a logical value indicating whether the terms should keep their positions. If FALSE the terms are reordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on. Effects of a given order are kept in the order specified. Default is TRUE.

drop.baseline  Drop the base level of categorical x’s, default is TRUE.

maxit  integer giving the maximal number of IWLS iterations, default is 100. This can also be controlled by control.

print.unnormalized.log.posterior  display the unnormalized log posterior likelihood for bayesglm, default=FALSE

Warning  default is TRUE, which will show the error messages of not convergence and separation.

...  further arguments passed to or from other methods.

Details

The program is a simple alteration of glm() that uses an approximate EM algorithm to update the betas at each step using an augmented regression to represent the prior information.

We use Student-t prior distributions for the coefficients. The prior distribution for the constant term is set so it applies to the value when all predictors are set to their mean values.

If scaled=TRUE, the scales for the prior distributions of the coefficients are determined as follows: For a predictor with only one value, we just use prior.scale. For a predictor with two values, we use prior.scale/range(x). For a predictor with more than two values, we use prior.scale/(2*sd(x)).

We include all the glm() arguments but we haven’t tested that all the options (e.g., offsets, contrasts, deviance for the null model) all work.

The new arguments here are: prior.mean, prior.scale, prior.scale.for.intercept, prior.df, prior.df.for.intercept and scaled.

Value

See glm for details.

prior.mean  prior means for the coefficients and the intercept.

prior.scale  prior scales for the coefficients

prior.df  prior dfs for the coefficients.

prior.scale.for.intercept  prior scale for the intercept

prior.df.for.intercept  prior df for the intercept

Author(s)

Andrew Gelman <gelman@stat.columbia.edu>; Yu-Sung Su <suyusung@tsinghua.edu.cn>; Daniel Lee <bearlee@alum.mit.edu>; Aleks Jakulin <Jakulin@stat.columbia.edu>
References


See Also

glm, bayespolr

Examples

```r
n <- 100
x1 <- rnorm(n)
x2 <- rbinom(n, 1, .5)
b0 <- 1
b1 <- 1.5
b2 <- 2
y <- rbinom(n, 1, invlogit(b0+b1*x1+b2*x2))
M1 <- glm (y ~ x1 + x2, family=binomial(link="logit"))
display (M1)
M2 <- bayesglm (y ~ x1 + x2, family=binomial(link="logit"),
  prior.scale=Inf, prior.df=Inf)
display (M2) # just a test: this should be identical to classical logit
M3 <- bayesglm (y ~ x1 + x2, family=binomial(link="logit"))
  # default Cauchy prior with scale 2.5
  display (M3)
M4 <- bayesglm (y ~ x1 + x2, family=binomial(link="logit"),
  prior.scale=2.5, prior.df=1)
  # Same as M3, explicitly specifying Cauchy prior with scale 2.5
  display (M4)
M5 <- bayesglm (y ~ x1 + x2, family=binomial(link="logit"),
  prior.scale=2.5, prior.df=7) # t_7 prior with scale 2.5
  display (M5)
M6 <- bayesglm (y ~ x1 + x2, family=binomial(link="logit"),
  prior.scale=2.5, prior.df=Inf) # normal prior with scale 2.5
  display (M6)
# Create separation: set y=1 whenever x2=1
# Now it should blow up without the prior!
y <- ifelse (x2==1, 1, y)
M1 <- glm (y ~ x1 + x2, family=binomial(link="logit"))
display (M1)
```

```
M2 <- bayesglm (y ~ x1 + x2, family=binomial(link="logit"),
    prior.scale=Inf, prior.scale.for.intercept=Inf) # Same as M1
display (M2)

M3 <- bayesglm (y ~ x1 + x2, family=binomial(link="logit"))
display (M3)

M4 <- bayesglm (y ~ x1 + x2, family=binomial(link="logit"),
    prior.scale=2.5, prior.scale.for.intercept=10) # Same as M3
display (M4)

M5 <- bayesglm (y ~ x1 + x2, family=binomial(link="logit"),
    prior.scale=2.5, prior.df=7)
display (M5)

M6 <- bayesglm (y ~ x1 + x2, family=binomial(link="logit"),
    prior.scale=2.5, prior.df=Inf)
display (M6)

# bayesglm with gaussian family (bayes lm)
sigma <- 5
y2 <- rnorm (n, b0+b1*x1+b2*x2, sigma)
M7 <- bayesglm (y2 ~ x1 + x2, prior.scale=Inf, prior.df=Inf)
display (M7)

# bayesglm with categorical variables
z1 <- trunc(runif(n, 4, 9))
levels(factor(z1))
z2 <- trunc(runif(n, 15, 19))
levels(factor(z2))

## drop the base level (R default)
M8 <- bayesglm (y ~ x1 + factor(z1) + factor(z2),
    family=binomial(link="logit"), prior.scale=2.5, prior.df=Inf)
display (M8)

## keep all levels with the intercept, keep the variable order
M9 <- bayesglm (y ~ x1 + x1:x2 + factor(z1) + x2 + factor(z2),
    family=binomial(link="logit"),
    prior.mean=rep(0,12),
    prior.scale=rep(2.5,12),
    prior.df=rep(Inf,12),
    prior.mean.for.intercept=0,
    prior.scale.for.intercept=10,
    prior.df.for.intercept=1,
    drop.baseline=FALSE, keep.order=TRUE)
display (M9)

## keep all levels without the intercept
M10 <- bayesglm (y ~ x1 + factor(z1) + x1:x2 + factor(z2)-1,
    family=binomial(link="logit"),
    prior.mean=rep(0,11),
bayespolr

Bayesian Ordered Logistic or Probit Regression

Description
Bayesian functions for ordered logistic or probit modeling with independent normal, t, or Cauchy prior distribution for the coefficients.

Usage
bayespolr(formula, data, weights, start, ..., subset, na.action, contrasts = NULL, Hess = TRUE, model = TRUE, method = c("logistic", "probit", "cloglog", "cauchit"), drop.unused.levels=TRUE, prior.mean = 0, prior.scale = 2.5, prior.df = 1, prior.counts.for.bins = NULL, min.prior.scale=1e-12, scaled = TRUE, maxit = 100, print.unnormalized.log.posterior = FALSE)

Arguments
formula a formula expression as for regression models, of the form response ~ predictors. The response should be a factor (preferably an ordered factor), which will be interpreted as an ordinal response, with levels ordered as in the factor. A proportional odds model will be fitted. The model must have an intercept: attempts to remove one will lead to a warning and be ignored. An offset may be used. See the documentation of formula for other details.
data an optional data frame in which to interpret the variables occurring in formula.
weights optional case weights in fitting. Default to 1.
start initial values for the parameters. This is in the format c(coefficients, zeta) ...
expression arguments to be passed to optim, most often a control argument.
subset expression saying which subset of the rows of the data should be used in the fit. All observations are included by default.
na.action a function to filter missing data.

prior.scale=rep(2.5,11),
prior.df=rep(Inf,11),
drop.baseline=FALSE)
display (M10)
 contrasts: a list of contrasts to be used for some or all of the factors appearing as variables in the model formula.

 Hess: logical for whether the Hessian (the observed information matrix) should be returned.

 model: logical for whether the model matrix should be returned.

 method: logistic or probit or complementary log-log or cauchit (corresponding to a Cauchy latent variable and only available in R >= 2.1.0).

 drop.unused.levels: default TRUE, if FALSE, it interpolates the intermediate values if the data have integer levels.

 prior.mean: prior mean for the coefficients: default is 0. Can be a vector of length equal to the number of predictors (not counting the intercepts). If it is a scalar, it is expanded to the length of this vector.

 prior.scale: prior scale for the coefficients: default is 2.5. Can be a vector of length equal to the number of predictors (not counting the intercepts). If it is a scalar, it is expanded to the length of this vector.

 prior.df: for t distribution: default is 1 (Cauchy). Set to Inf to get normal prior distributions. Can be a vector of length equal to the number of predictors (not counting the intercepts). If it is a scalar, it is expanded to the length of this vector.

 prior.counts.for.bins: default is NULL, which will augment the data by giving each cut point a 1/levels(y). To use a noninformative prior, assign prior.counts.for.bins = 0. If it is a scalar, it is expanded to the number of levels of y.

 min.prior.scale: Minimum prior scale for the coefficients: default is 1e-12.

 scaled: if scaled = TRUE, then the prior distribution is rescaled. Can be a vector of length equal to the number of cutpoints (intercepts). If it is a scalar, it is expanded to the length of this vector.

 maxit: integer giving the maximal number of IWLS iterations, default is 100. This can also be controlled by control.

 print.unnormalized.log.posterior: display the unnormalized log posterior likelihood for bayesglm fit, default=FALSE

Details

The program is a simple alteration of polr in VR version 7.2-31 that augments the loglikelihood with the log of the t prior distributions for the coefficients.

We use Student-t prior distributions for the coefficients. The prior distributions for the intercepts (the cutpoints) are set so they apply to the value when all predictors are set to their mean values.

If scaled=TRUE, the scales for the prior distributions of the coefficients are determined as follows: For a predictor with only one value, we just use prior.scale. For a predictor with two values, we use prior.scale/range(x). For a predictor with more than two values, we use prior.scale/(2*sd(x)).
Value

See polr for details.

prior.mean  prior means for the coefficients.
prior.scale prior scales for the coefficients.
prior.df    prior dfs for the coefficients.
prior.counts.for.bins  prior counts for the cutpoints.

Author(s)

Andrew Gelman <gelman@stat.columbia.edu>; Yu-Sung Su <suyusung@tsinghua.edu.cn>; Maria Grazia Pittau <grazia@stat.columbia.edu>

See Also

bayesglm, polr

Examples

M1 <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
display (M1)

M2 <- bayespolr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing,
prior.scale=Inf, prior.df=Inf) # Same as M1
display (M2)

M3 <- bayespolr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
display (M3)

M4 <- bayespolr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing,
prior.scale=2.5, prior.df=1) # Same as M3
display (M4)

M5 <- bayespolr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing,
prior.scale=2.5, prior.df=7)
display (M5)

M6 <- bayespolr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing,
prior.scale=2.5, prior.df=Inf)
display (M6)

# Assign priors
M7 <- bayespolr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing,
prior.mean=rep(0,6), prior.scale=rep(2.5,6), prior.df=c(1,1,7,7,7))
display (M7)

### Another example

y <- factor (rep (1:10,1:10))
x <- rnorm (length(y))
x <- x - mean(x)

M8 <- polr(y ~ x)
display(M8)

M9 <- bayespolr(y ~ x, prior.scale=Inf, prior.df=Inf, prior.counts.for.bins=0)
display(M9) # same as M1

M10 <- bayespolr(y ~ x, prior.scale=Inf, prior.df=Inf, prior.counts.for.bins=10000)
display(M10)

### Another example

y <- factor(rep(1:3,1:3))
x <- rnorm(length(y))
x <- x - mean(x)

M11 <- polr(y ~ x)
display(M11)

M12 <- bayespolr(y ~ x, prior.scale=Inf, prior.df=Inf, prior.counts.for.bins=0)
display(M12) # same as M1

M13 <- bayespolr(y ~ x, prior.scale=Inf, prior.df=Inf, prior.counts.for.bins=1)
display(M13)

M14 <- bayespolr(y ~ x, prior.scale=Inf, prior.df=Inf, prior.counts.for.bins=10)
display(M14)

binnedplot

\textit{Binned Residual Plot}

\textbf{Description}

A function that plots averages of \(y\) versus averages of \(x\) and can be useful to plot residuals for logistic regression.

\textbf{Usage}

\begin{verbatim}
binnedplot(x , y, nclass=NULL, 
  xlab="Expected Values", ylab="Average residual", 
  main="Binned residual plot", 
  cex.pts=0.8, col.pts=1, col.int="gray", ...)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \textbf{x} \hspace{1cm} The expected values from the logistic regression.
\end{itemize}
The residuals values from logistic regression (observed values minus expected values).

Number of categories (bins) based on their fitted values in which the data are divided. Default=NULL and will take the value of nclass according to the $\text{nclass} = \lfloor \sqrt{n} \rfloor$ such that if $n \geq 100$, nclass=$\lfloor \sqrt{n} \rfloor$; if $10 < n < 100$, nclass=10; if $n < 10$, nclass=$\lfloor n/2 \rfloor$.

a label for the x axis, default is "Expected Values".

a label for the y axis, default is "Average residual".

a main title for the plot, default is "Binned residual plot". See also title.

The size of points, default=0.8.

color of points, default is black

color of intervals, default is gray

Graphical parameters to be passed to methods

Details

In logistic regression, as with linear regression, the residuals can be defined as observed minus expected values. The data are discrete and so are the residuals. As a result, plots of raw residuals from logistic regression are generally not useful. The binned residuals plot instead, after dividing the data into categories (bins) based on their fitted values, plots the average residual versus the average fitted value for each bin.

Value

A plot in which the gray lines indicate plus and minus 2 standard-error bounds, within which one would expect about 95% of the binned residuals to fall, if the model were actually true.

Note

There is typically some arbitrariness in choosing the number of bins: each bin should contain enough points so that the averaged residuals are not too noisy, but it helps to have also many bins so as to see more local patterns in the residuals (see Gelman and Hill, Data Analysis Using Regression and Multilevel/Hierarchical Models, pag 97).

Author(s)

M. Grazia Pittau <grazia@stat.columbia.edu>; Yu-Sung Su <suyusung@tsinghua.edu.cn>

References


See Also

par, plot
Examples

```r
old.par <- par(no.readonly = TRUE)
data(lalonde)
attach(lalonde)
fit <- glm(treat ~ re74 + re75 + educ + black + hisp + married
+ nodegr + u74 + u75, family=binomial(link="logit"))
x <- predict(fit)
y <- resid(fit)
binnedplot(x,y)
par(old.par)
```

coefplot

Generic Function for Making Coefficient Plot

Description

Functions that plot the coefficients plus and minus 1 and 2 sd from a lm, glm, bugs, and polr fits.

Usage

c coefplot(object,...)

## Default S3 method:
c coefplot(coefs, sds, CI=2,
  lower.conf.bounds, upper.conf.bounds,
  varnames=NULL, vertical=TRUE,
  v.axis=TRUE, h.axis=TRUE,
  cex.var=0.8, cex.pts=0.9,
  col.pts=1, pch.pts=20, var.las=2,
  main=NULL, xlab=NULL, ylab=NULL,
  plot=TRUE, add=FALSE, offset=.1, ...)

## S4 method for signature 'bugs'
c coefplot(object, var.idx=NULL, varnames=NULL,
  CI=1, vertical=TRUE,
  v.axis=TRUE, h.axis=TRUE,
  cex.var=0.8, cex.pts=0.9,
  col.pts=1, pch.pts=20, var.las=2,
  main=NULL, xlab=NULL, ylab=NULL,
  plot=TRUE, add=FALSE, offset=.1,
  mar=c(1,3,5.1,2), ...)

## S4 method for signature 'numeric'
c coefplot(object, ...)

## S4 method for signature 'lm'
c coefplot(object, varnames=NULL, intercept=FALSE, ...)

## S4 method for signature 'glm'
coefplot(object, varnames=NULL, intercept=FALSE, ...)  
## S4 method for signature 'polr'
coefplot(object, varnames=NULL, ...)  

Arguments

object fitted objects-lm, glm, bugs and polr, or a vector of coefficients.

... further arguments passed to or from other methods.

coefficients a vector of coefficients.

de a vector of sds of coefficients.

CI confidence interval, default is 2, which will plot plus and minus 2 sds or 95% CI. If CI=1, plot plus and minus 1 sds or 50% CI instead.

lower.bounds lower bounds of confidence intervals.

upper.bounds upper bounds of confidence intervals.

varnames a vector of variable names, default is NULL, which will use the names of variables; if specified, the length of varnames must be equal to the length of predictors, including the intercept.

vertical orientation of the plot, default is TRUE which will plot variable names in the 2nd axis. If FALSE, plot variable names in the first axis instead.

v.axis default is TRUE, which shows the bottom axis—axis(1).

h.axis default is TRUE, which shows the left axis—axis(2).

cex.var The fontsize of the varible names, default=0.8.

cex.pts The size of data points, default=0.9.

col.pts color of points and segments, default is black.

pch.pts symbol of points, default is solid dot.

var.las the orientation of variable names against the axis, default is 2. see the usage of las in par.

main The main title (on top) using font and size (character expansion) par("font.main") and color par("col.main").

xlab X axis label using font and character expansion par("font.lab") and color par("col.lab").

ylab Y axis label, same font attributes as xlab.

mar A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot. The default is c(1,3,5.1,2).

plot default is TRUE, plot the estimates.

add if add=TRUE, plot over the existing plot. default is FALSE.

offset add extra spaces to separate from the existing dots. default is 0.1.

var.idx the index of the variables of a bugs object, default is NULL which will plot all the variables.

intercept If TRUE will plot intercept, default=FALSE to get better presentation.
Details

This function plots coefficients from bugs, lm, glm and polr with 1 sd and 2 sd interval bars.

Value

Plot of the coefficients from a bugs, lm or glm fit. You can add the intercept, the variable names and display the result of the fitted model.

Author(s)

Yu-Sung Su <suyusung@tsinghua.edu.cn>

References


See Also

display, par, lm, glm, bayesglm, plot

Examples

old.par <- par(no.readonly = TRUE)

y1 <- rnorm(1000,50,23)
y2 <- rbinom(1000,1,prob=0.72)
x1 <- rnorm(1000,50,2)
x2 <- rbinom(1000,1,prob=0.63)
x3 <- rpois(1000,2)
x4 <- runif(1000,40,100)
x5 <- rbeta(1000,2,2)

longnames <- c("a long name01","a long name02","a long name03","a long name04","a long name05")

fit1 <- lm(y1 ~ x1 + x2 + x3 + x4 + x5)
fit2 <- glm(y2 ~ x1 + x2 + x3 + x4 + x5,
    family=binomial(link="logit"))
op <- par()
# plot 1
par (mfrow=c(2,2))
coefplot(fit1)
coefplot(fit2, col.pts="blue")

# plot 2
longnames <- c("(Intercept)", longnames)
coefplot(fit1, longnames, intercept=TRUE, CI=1)

# plot 3
coefplot(fit2, vertical=FALSE, var.las=1, frame.plot=TRUE)
# plot 4: comparison to show bayesglm works better than glm
n <- 100
x1 <- rnorm(n)
x2 <- rbinom(n, 1, .5)
b0 <- 1
b1 <- 1.5
b2 <- 2
y <- rbinom(n, 1, invlogit(b0+b1*x1+b2*x2))
y <- ifelse(x2==1, 1, y)
x1 <- rescale(x1)
x2 <- rescale(x2, "center")

M1 <- glm(y ~ x1 + x2, family=binomial(link="logit")
display(M1)
M2 <- bayesglm(y ~ x1 + x2, family=binomial(link="logit")
display(M2)

# stacked plot
coefplot(M2, xlim=c(-1,5), intercept=TRUE)
coefplot(M1, add=TRUE, col.pts="red")

# arrayed plot
par(mfrow=c(1,2))
x.scale <- c(0, 7.5) # fix x.scale for comparison
coefplot(M1, xlim=x.scale, main="glm", intercept=TRUE)
coefplot(M2, xlim=x.scale, main="bayesglm", intercept=TRUE)

# plot 5: the ordered logit model from polr
M3 <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
display(M3, main="polr")
M4 <- bayespolar(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
display(M4, main="bayespolr", add=TRUE, col.pts="red")

# plot 6: plot bugs & lmer
# par <- op
# M5 <- lmer(Reaction ~ Days + (1|Subject), sleepstudy)
# M5.sim <- mcsamp(M5)
# coefplot(M5.sim, var.idx=5:22, CI=1, ylim=c(18,1), main="lmer model")

# plot 7: plot coefficients & sds vectors
coef.vect <- c(0.2, 1.4, 2.3, 0.5)
sd.vect <- c(0.12, 0.24, 0.23, 0.15)
longnames <- c("var1", "var2", "var3", "var4")
coefplot(coef.vect, sd.vect, varnames=longnames, main="Regression Estimates")
coefplot(coef.vect, sd.vect, varnames=longnames, vertical=FALSE, var.las=1, main="Regression Estimates")
Description

Return a matrix of contrasts used in `bayesglm`.

Usage

```r
contr.bayes.unordered(n, base = 1, contrasts = TRUE)
contr.bayes.ordered (n, scores = 1:n, contrasts = TRUE)
```

Arguments

- `n`: a vector of levels for a factor, or the number of levels.
- `base`: an integer specifying which group is considered the baseline group. Ignored if `contrasts` is `FALSE`.
- `contrasts`: a logical indicating whether contrasts should be computed.
- `scores`: the set of values over which orthogonal polynomials are to be computed.

Details

These functions are adapted from `contr.treatment` and `contr.poly` in `stats` package. The purpose for these functions are to keep the baseline levels of categorical variables and thus to suit the use of `bayesglm`.

- `contr.bayes.unordered` is equivalent to `contr.treatment` whereas `contr.bayes.ordered` is equivalent to `contr.poly`.

Author(s)

Yu-Sung Su <suyusung@tsinghua.edu.cn>

See Also

- `C`, `contr.helmert`, `contr.poly`, `contr.sum`, `contr.treatment`, `glm`, `aov`, `lm`, `bayesglm`.

Examples

```r
cat.var <- rep(1:3, 5)
dim(contr.bayes.unordered(cat.var))
# 15x15 baseline level kept!
dim(contr.treatment(cat.var))
# 15x14
```
corrplot

Correlation Plot

Description

Function for making a correlation plot starting from a data matrix

Usage

corrplot (data, varnames=NULL, cutpts=NULL, 
abs=TRUE, details=TRUE, 
n.col.legend=5, cex.col=0.7, 
cex.var=0.9, digits=1, color=FALSE)

Arguments

data a data matrix
varnames variable names of the data matrix, if not provided use default variable names
abs if TRUE, transform all correlation values into positive values, default=TRUE.
cutpts a vector of cutting points for color legend, default is NULL. The function will decide the cutting points if cutpts is not assigned.
details show more than one digits correlation values. Default is TRUE. FALSE is suggested to get readable output.
n.col.legend number of legend for the color thermometer.
cex.col font size of the color thermometer.
cex.var font size of the variable names.
digits number of digits shown in the text of the color thermometer.
color color of the plot, default is FALSE, which uses gray scale.

Details


Value

A correlation plot.

Author(s)

Tian Zheng <tzheng@stat.columbia.edu>; Yu-Sung Su <suyusung@tsinghua.edu.cn>
References


See Also
cor, par

Examples

old.par <- par(no.readonly = TRUE)

x1 <- rnorm(1000,50,2)
x2 <- rbinom(1000,1,prob=0.63)
x3 <- rpois(1000, 2)
x4 <- runif(1000,40,100)
x5 <- rnorm(1000,100,30)
x6 <- rbeta(1000,2,2)
x7 <- rpois(1000,10)
x8 <- rbinom(1000,1,prob=0.4)
x9 <- rbeta(1000,5,4)
x10 <- runif(1000,-10,-1)

test.data <- data.matrix(cbind(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10))
test.names <- c("a short name01","a short name02","a short name03",
"a short name04","a short name05","a short name06",
"a short name07","a short name08","a short name09",
"a short name10")

# example 1
corrplot(test.data)

# example 2
corrplot(test.data,test.names, abs=FALSE, n.col.legend=7)
corrplot(test.data,test.names, abs=TRUE, n.col.legend=7)

# example 3
data(lalonde)
corrplot(lalonde, details=FALSE, color=TRUE)
corrplot(lalonde, cutpts=c(0,0.25,0.5,0.75), color=TRUE, digits=2)

par(old.par)

discrete.histogram

Histogram for Discrete Distributions

Description

Creates a prettier histogram for discrete distributions
Usage

discrete.histogram (x, prob, prob2=NULL, prob3=NULL, 
    xlab="x", xaxs.label=NULL, yaxs.label=NULL, bar.width=NULL, 
    freq=FALSE, prob.col="blue", prob2.col="red", prob3.col="gray", ...)

Arguments

  x            The vector of x’s
  prob         The probabilities for the x’s
  prob2        A second vector of probabilities of the x’s
  prob3        A third vector of probabilities of the x’s
  xlab         Label for the x axis
  xaxs.label   Label for the x’s
  yaxs.label   Label for the y axis
  bar.width    Width of the bars
  freq         If TRUE, shows a frequency histogram as opposed to probability.
  prob.col     The color of the first set of histogram bars.
  prob2.col    The color of the second set of histogram bars.
  prob3.col    The color of the third set of histogram bars.
  ...          Additional arguments passed to function plot

Details

  This function displays a histogram for discrete probability distributions.

Examples

  a <- c(3,4,0,0,5,1,1,1,1,0)
  discrete.histogram (a)

  x <- c(0,1,3,4,5)
  p <- c(.3,.4,.1,.1,.1)
  discrete.histogram (x,p)

  x <- c(0,1,3,4,5)
  y <- c(3,4,1,1,1)
  discrete.histogram (x,y)
Description

This generic function gives a clean printout of lm, glm, mer, polr and svyglm objects.

Usage

display (object, ...)

## S4 method for signature 'lm'
display(object, digits=2, detail=FALSE)

## S4 method for signature 'bayesglm'
display(object, digits=2, detail=FALSE)

## S4 method for signature 'glm'
display(object, digits=2, detail=FALSE)

## S4 method for signature 'merMod'
display(object, digits=2, detail=FALSE)

## S4 method for signature 'polr'
display(object, digits=2, detail=FALSE)

## S4 method for signature 'svyglm'
display(object, digits=2, detail=FALSE)

Arguments

object The output of a call to lm, glm, mer, polr, svyglm or related regressions function with n data points and k predictors.

... further arguments passed to or from other methods.

digits number of significant digits to display.

detail default is FALSE, if TRUE, display p-values or z-values

Details

This generic function gives a clean printout of lm, glm, mer and polr objects, focusing on the most pertinent pieces of information: the coefficients and their standard errors, the sample size, number of predictors, residual standard deviation, and R-squared. Note: R-squared is automatically displayed to 2 digits, and deviances are automatically displayed to 1 digit, no matter what.

Value

Coefficients and their standard errors, the sample size, number of predictors, residual standard deviation, and R-squared
Note

Output are the model, the regression coefficients and standard errors, and the residual sd and R-squared (for a linear model), or the null deviance and residual deviance (for a generalized linear model).

Author(s)

Andrew Gelman <gelman@stat.columbia.edu>; Yu-Sung Su <suyusung@tsinghua.edu.cn>; Maria Grazia Pittau <grazia@stat.columbia.edu>

References


See Also

summary, lm, glm, lmer, polr, svyglm

Examples

# Here's a simple example of a model of the form, y = a + bx + error,
# with 10 observations in each of 10 groups, and with both the
# intercept and the slope varying by group. First we set up the model and data.
group <- rep(1:10, rep(10,10))
group2 <- rep(1:10, 10)
mu.a <- 0
sigma.a <- 2
mu.b <- 3
sigma.b <- 4
rho <- 0.56
Sigma.ab <- array (c(sigma.a^2, rho*sigma.a*sigma.b,
rho*sigma.a*sigma.b, sigma.b^2), c(2,2))
sigma.y <- 1
ab <- mvrnorm (10, c(mu.a,mu.b), Sigma.ab)
a <- ab[,1]
b <- ab[,2]
d <- rnorm(10)
x <- rnorm (100)
y1 <- rnorm (100, a[group] + b*x, sigma.y)
y2 <- rbinom(100, 1, prob=invlogit(a[group] + b*x))
y3 <- rnorm (100, a[group] + b[group]*x + d[group2], sigma.y)
y4 <- rbinom(100, 1, prob=invlogit(a[group] + b*x + d[group2]))

# display a simple linear model
M1 <- lm (y1 ~ x)
display (M1)
M1.sim <- sim(M1, n.sims=2)

# display a simple logit model
M2 <- glm(y2 ~ x, family=binomial(link="logit"))
display (M2)
M2.sim <- sim(M2, n.sims=2)

# Then fit and display a simple varying-intercept model:
M3 <- lmer(y1 ~ x + (1|group))
display (M3)
M3.sim <- sim(M3, n.sims=2)

# Then the full varying-intercept, varying-slope model:
M4 <- lmer(y1 ~ x + (1 + x |group))
display (M4)
M4.sim <- sim(M4, n.sims=2)

# Then the full varying-intercept, logit model:
M5 <- glmer(y2 ~ x + (1|group), family=binomial(link="logit"))
display (M5)
M5.sim <- sim(M5, n.sims=2)

# Then the full varying-intercept, varying-slope logit model:
M6 <- glmer(y2 ~ x + (1|group) + (0 + x |group),
            family=binomial(link="logit"))
display (M6)
M6.sim <- sim(M6, n.sims=2)

# Then non-nested varying-intercept, varying-slope model:
M7 <- lmer(y3 ~ x + (1 + x |group) + (1|group2))
display(M7)
M7.sim <- sim(M7, n.sims=2)

# Then the ordered logit model from polr
M8 <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
display(M8)

M9 <- bayespolr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
display(M9)
extractDIC

Extract AIC and DIC from a ‘mer’ model

Description
Computes the (generalized) Akaike Information Criterion and Deviance Information Criterion for a mer model.

Usage
extractDIC(fit,...)
## S3 method for class 'merMod'
extractDIC(fit,...)

Arguments
fit fitted merMod mode, usually the result of a fitter like merMod.
...

Further arguments (currently unused).

Author(s)
Andrew Gelman <gelman@stat.columbia.edu>; Yu-Sung Su <suyusung@tsinghua.edu.cn>

Examples
fm1 <- lmer(Reaction ~ Days + (Days|Subject), sleepstudy)
extractAIC(fm1)
extrackDIC(fm1)

fround

Formatting the Rounding of Numbers

Description
fround rounds the values in its first argument to the specified number of decimal places with surrounding quotes.
pfround rounds the values in its first argument to the specified number of decimal places without surrounding quotes.

Usage
fround(x, digits)
pfround(x, digits)
Arguments

x a numeric vector.
digits integer indicating the precision to be used.

Author(s)

Andrew Gelman <gelman@stat.columbia.edu>; Yu-Sung Su <suyusung@tsinghua.edu.cn>

See Also

round

Examples

x <- rnorm(1)
fround(x, digits=2)
pfround(x, digits=2)

Function to Recall Last Source File

Description

A function that like source() but recalls the last source file names by default.

Usage

go(..., add=FALSE, timer=FALSE)

Arguments

... list of filenames as character strings.
add add these names to the current list; if replace, then FALSE.
timer time the execution time of go().

Author(s)

Jouni Kerman <jouni@kerman.com> <kerman@stat.columbia.edu>

Examples

go('myprog')      # will run source('myprog.r')
go()              # will run source('myprog.r') again
go('somelib', add=TRUE)    # will run source('myprog.r') and source('somelib.r')
go('myprog', 'somelib')    # same as above
go('mytest')        # will run source('mytest') only
go()                # runs source('mytest') again
G                   # short cut to call go()
invlogit

Logistic and Inverse logistic functions

Description
Inverse-logit function, transforms continuous values to the range (0, 1)

Usage
logit(x)
invlogit(x)

Arguments
x
A vector of continuous values

Details
The Inverse-logit function defined as: \( \text{logit}^{-1}(x) = \frac{e^x}{1 + e^x} \) transforms continuous values to the range (0, 1), which is necessary, since probabilities must be between 0 and 1 and maps from the linear predictor to the probabilities

Value
A vector of estimated probabilities

Author(s)
Andrew Gelman <gelman@stat.columbia.edu>, M.Grazia Pittau <grazia@stat.columbia.edu>

References

Examples
```
data(frisk)
n <- 100
x1 <- rnorm (n)
x2 <- rbinom (n, 1, .5)
b0 <- 1
b1 <- 1.5
b2 <- 2
Inv.logit <- invlogit(b0+b1*x1+b2*x2)
plot(b0+b1*x1+b2*x2, Inv.logit)
```
Description

Dataset used by Dehejia and Wahba (1999) to evaluate propensity score matching.

Usage

data(lalonde)

Format

A data frame with 445 observations on the following 12 variables.

- **age**: age in years.
- **educ**: years of schooling.
- **black**: indicator variable for blacks.
- **hisp**: indicator variable for Hispanics.
- **married**: indicator variable for marital status.
- **nodegr**: indicator variable for high school diploma.
- **re74**: real earnings in 1974.
- **re75**: real earnings in 1975.
- **re78**: real earnings in 1978.
- **u74**: indicator variable for earnings in 1974 being zero.
- **u75**: indicator variable for earnings in 1975 being zero.
- **treat**: an indicator variable for treatment status.

Details

Two demos are provided which use this dataset. The first, DehejiaWahba, replicates one of the models from Dehejia and Wahba (1999). The second demo, AbadieImbens, replicates the models produced by Abadie and Imbens http://scholar.harvard.edu/imbens/scholar_software/matching-estimators. Many of these models are found to produce good balance for the Lalonde data.

Note

This documentation is adapted from Matching package.

References


See Also

matching, GenMatch balance

Examples

data(lalonde)

---

### Description

Function for processing matching with propensity score

### Usage

```r
matching(z, score, replace=FALSE)
```

### Arguments

- `z`: vector of indicators for treatment or control.
- `score`: vector of the propensity scores in the same order as `z`.
- `replace`: whether the control units could be reused for matching, default is `FALSE`.

### Details

Function for matching each treatment unit in turn the control unit (not previously chosen) with the closest propensity score.

### Value

The function returns a vector of indices that the corresponding unit is matched to. 0 means matched to nothing.

### Author(s)

Jennifer Hill <jh1030@columbia.edu>; Yu-Sung Su <suyusung@tsinghua.edu.cn>

### References


### See Also

balance
Examples

```r
# matching first
data(lalonde)
attach(lalonde)
fit <- glm(treat ~ re74 + re75 + age + factor(educ) +
           black + hisp + married + nodegr + u74 + u75,
           family=binomial(link="logit"))
pscores <- predict(fit, type="response")
matches <- matching(z=lalonde$treat, score=pscores)
matched <- matches$cnts

# balance check!
b.stats <- balance(lalonde, treat, matched)
print(b.stats)
plot(b.stats)
```

---

**mcsamp**

*Generic Function to Run 'mcmcsamp()' in lme4*

**Description**

The quick function for MCMC sampling for lmer and glmer objects and convert to Bugs objects for easy display.

**Usage**

```r
## Default S3 method:
mcsamp(object, n.chains=3, n.iter=1000, n.burnin=floor(n.iter/2),
       n.thin=max(1, floor(n.chains * (n.iter - n.burnin)/1000)),
       saveb=TRUE, deviance=FALSE, make.bugs.object=FALSE)
## S4 method for signature 'merMod'
mcsamp(object, ...)```

**Arguments**

- `object` mer objects from lme4
- `n.chains` number of MCMC chains
- `n.iter` number of iteration for each MCMC chain
- `n.burnin` number of burnin for each MCMC chain, Default is `n.iter/2`, that is, discarding the first half of the simulations.
- `n.thin` keep every kth draw from each MCMC chain. Must be a positive integer. Default is `max(1, floor(n.chains * (n.iter - n.burnin) / 1000))` which will only thin if there are at least 2000 simulations.
- `saveb` if 'TRUE', causes the values of the random effects in each sample to be saved.
- `deviance` compute deviance for mer objects. Only works for `lmer` object
make.bugs.object
    transform the output into bugs object, default is TRUE
... further arguments passed to or from other methods.

Details
This function generates a sample from the posterior distribution of the parameters of a fitted model using Markov Chain Monte Carlo methods. It automatically simulates multiple sequences and allows convergence to be monitored. The function relies on mcmcsamp in lme4.

Value
An object of (S3) class "bugs" suitable for use with the functions in the "R2WinBUGS" package.

Author(s)
Andrew Gelman <gelman@stat.columbia.edu>; Yu-Sung Su <ys463@columbia.edu>

References
Douglas Bates and Deepayan Sarkar, lme4: Linear mixed-effects models using S4 classes.

See Also
display, lmer, sim

Examples

```r
## Here's a simple example of a model of the form, y = a + bx + error,
## with 10 observations in each of 10 groups, and with both the intercept
## and the slope varying by group. First we set up the model and data.
##
## # group <- rep(1:10, rep(10,10))
## # group2 <- rep(1:10, 10)
## # mu.a <- 0
## # sigma.a <- 2
## # mu.b <- 3
## # sigma.b <- 4
## # rho <- 0.56
## # Sigma.ab <- array (c(sigma.a^2, rho*sigma.a*sigma.b,
## # rho*sigma.a*sigma.b, sigma.b^2), c(2,2))
## # sigma.y <- 1
## # ab <- mvrnorm (10, c(mu.a,mu.b), Sigma.ab)
## # a <- ab[,1]
## # b <- ab[,2]
## # d <- rnorm(10)
## # x <- rnorm (100)
## # y1 <- rnorm (100, a[group] + b*x, sigma.y)
```
# y2 <- rbinom(100, 1, prob=invlogit(a[group] + b*x))
# y3 <- rnorm(100, a[group] + b[group]*x + d[group2], sigma.y)
# y4 <- rbinom(100, 1, prob=invlogit(a[group] + b*x + d[group2]))

## Then fit and display a simple varying-intercept model:
##
## M1 <- lmer (y1 ~ x + (1|group))
## display (M1)
## M1.sim <- mcsamp (M1)
## print (M1.sim)
## plot (M1.sim)

## Then the full varying-intercept, varying-slope model:
##
## M2 <- lmer (y1 ~ x + (1 + x |group))
## display (M2)
## M2.sim <- mcsamp (M2)
## print (M2.sim)
## plot (M2.sim)

## Then the full varying-intercept, logit model:
##
## M3 <- lmer (y2 ~ x + (1|group), family=binomial(link="logit"))
## display (M3)
## M3.sim <- mcsamp (M3)
## print (M3.sim)
## plot (M3.sim)

## Then the full varying-intercept, varying-slope logit model:
##
## M4 <- lmer (y2 ~ x + (1|group) + (0+x |group),
## family=binomial(link="logit"))
## display (M4)
## M4.sim <- mcsamp (M4)
## print (M4.sim)
## plot (M4.sim)

## Then non-nested varying-intercept, varying-slope model:
##
## M5 <- lmer (y3 ~ x + (1 + x |group) + (1|group2))
## display (M5)
## M5.sim <- mcsamp (M5)
## print (M5.sim)
## plot (M5.sim)
Description

model.matrixBayes creates a design matrix.

Usage

model.matrixBayes(object, data = environment(object),
contrasts.arg = NULL, xlev = NULL, keep.order = FALSE, drop.baseline=FALSE,...)

Arguments

object an object of an appropriate class. For the default method, a model formula or terms object.
data a data frame created with model.frame. If another sort of object, model.frame is called first.
contrasts.arg A list, whose entries are contrasts suitable for input to the contrasts replacement function and whose names are the names of columns of data containing factors.
xlev to be used as argument of model.frame if data has no "terms" attribute.
keep.order a logical value indicating whether the terms should keep their positions. If FALSE the terms are reordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on. Effects of a given order are kept in the order specified.
drop.baseline Drop the base level of categorical Xs, default is TRUE.
... further arguments passed to or from other methods.

Details

model.matrixBayes is adapted from model.matrix in the stats package and is designed for the use of bayesglm. It is designed to keep baseline levels of all categorical variables and keep the variable names unordered in the output. The design matrices created by model.matrixBayes are unidentifiable using classical regression methods, though; they can be identified using bayesglm.

Author(s)

Yu-Sung Su <suyusung@tsinghua.edu.cn>

References

See Also

\texttt{model.frame, model.extract, terms, terms.formula, bayesglm}.

Examples

\begin{verbatim}
ff <- log(Volume) ~ log(Height) + log(Girth)
str(m <- model.frame(ff, trees))
(model.matrix(ff, m))
class(ff) <- c("bayesglm", "terms", "formula")
(model.matrixBayes(ff, m))
\end{verbatim}

\begin{verbatim}
multicomp.plot

\textbf{Description}

Plots significant difference of simulated array.

\textbf{Usage}

\texttt{multicomp.plot(object, alpha = 0.05, main = "Multiple Comparison Plot",}
\texttt{label = NULL, shortlabel = NULL, show.pvalue = FALSE,}
\texttt{label.as.shortlabel = FALSE, label.on.which.axis = 3,}
\texttt{col.low = "lightsteelblue", col.same = "white", col.high = "lightslateblue",}
\texttt{vertical.line = TRUE, horizontal.line = FALSE,}
\texttt{vertical.line.lty = 1, horizontal.line.lty = 1, mar=c(3.5,3.5,3.5,3.5))}

\textbf{Arguments}

\texttt{object} Simulated array of coefficients, columns being different variables and rows being simulated result.
\texttt{alpha} Level of significance to compare.
\texttt{main} Main label.
\texttt{label} Labels for simulated parameters.
\texttt{shortlabel} Short labels to put into the plot.
\texttt{show.pvalue} Default is FALSE, if set to TRUE replaces short label with Bayesian p value.
\texttt{label.as.shortlabel} Default is FALSE, if set to TRUE takes first 2 character of label and use it as short label.
\texttt{label.on.which.axis} default is the 3rd (top) axis.
\texttt{col.low} Color of significantly low coefficients.
\texttt{col.same} Color of not significant difference.
col.high     Color of significantly high coefficients.
vertical.line Default is TRUE, if set to FALSE does not draw vertical line.
horizontal.line Default is FALSE, if set to TRUE draws horizontal line.
vertical.line.lty Line type of vertical line.
horizontal.line.lty Line type of horizontal line.
mar          A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot. The default is c(3.5, 3.5, 3.5, 3.5).

Value

pvalue      Array of Bayesian p value.
significant Array of significance.

Author(s)

Masanao Yajima <yajima@stat.columbia.edu>, Andrew Gelman <gelman@stat.columbia.edu>

References


See Also

ccoefplot

Examples

old.par <- par(no.readonly = TRUE)

# example 1
simulation.array <- data.frame(coef1=rnorm(100,10,2), coef2=rnorm(100,5,2),
            coef3=rnorm(100,0,1), coef4=rnorm(100,-5,3),
            coef5=rnorm(100,-2,1))
short.lab <- c("c01", "c02", "c03", "c04", "c05")
multicomp.plot(simulation.array[,1:4], label.as.shortlabel=TRUE)

# wraper for multicomp.plot
mcplot(simulation.array, shortlabel = short.lab)

# example 2
data(lalonde)
M1 <- lm(re78 ~ treat + re74 + re75 + age + educ + u74 + u75, data=lalonde)
M1.sim <- sim(M1)
Im.sim <- coef(M1.sim)[:-1]
multicomp.plot(Im.sim, label.as.shortlabel=TRUE, label.on.which.axis=2)
readColumns is a function to read data by columns.

### Description

A function to read data by columns.

### Usage

```r
read.columns(filename, columns)
```

### Arguments

- `filename`: user specified file name including path of the file.
- `columns`: which columns of the data to be read.

### Author(s)

Andrew Gelman <gelman@stat.columbia.edu>

---

rescale is a function for standardizing by centering and dividing by 2 standard deviations (sd's), with exceptions for binary variables.

### Description

This function standardizes a variable by centering and dividing by 2 sd's with exceptions for binary variables.

### Usage

```r
rescale(x, binary.inputs="center")
```

### Arguments

- `x`: a vector.
- `binary.inputs`: options for standardizing binary variables, default is center; 0/1 keeps original scale; 0.5, 0.5 rescales 0 as 0.5 and 1 as 0.5; center subtracts the mean; and full subtracts the mean and divides by 2 sd.

### Value

The standardized vector.
Author(s)
Andrew Gelman <gelman@stat.columbia.edu>; Yu-Sung Su <suyusung@tsinghua.edu.cn>

References

See Also
standardize

Examples
# Set up the fake data
n <- 100
x <- rnorm(n, 2, 1)
x1 <- rnorm(n)
x1 <- (x1-mean(x1))/(2*sd(x1))  # standardization
x2 <- rbinom(n, 1, .5)
b0 <- 1
b1 <- 1.5
b2 <- 2
y <- rbinom(n, 1, invlogit(b0+b1*x1+b2*x2))
rescale(x, "full")
rescale(y, "center")

residual.plot(Expected, Residuals, sigma, main = deparse(substitute(Expected)),
col.pts = "blue", col.ctr = "red", col.sgm = "black", cex = 0.5,
gray.scale = FALSE, xlab = "Predicted", ylab = "Residuals", ...)

Arguments
Expected Expected value.
Residuals Residual value.
sigma Standard error.
main main for the plot. See plot for detail.
se.coef

se.coef

Extract Standard Errors of Model Coefficients

Description

These functions extract standard errors of model coefficients from objects returned by modeling functions.
Usage

```r
se.coef (object, ...)  
se.fixef (object)  
se.ranef (object)
```

```r
## S4 method for signature 'lm'
se.coef(object)
## S4 method for signature 'glm'
se.coef(object)
## S4 method for signature 'merMod'
se.coef(object)
```

Arguments

- `object` : object of `lm`, `glm` and `merMod` fit
- `...` : other arguments

Details

`se.coef` extracts standard errors from objects returned by modeling functions. `se.fixef` extracts standard errors of the fixed effects from objects returned by lmer and glmer functions. `se.ranef` extracts standard errors of the random effects from objects returned by lmer and glmer functions.

Value

- `se.coef` gives lists of standard errors for `coef`, `se.fixef` gives a vector of standard errors for `fixef` and `se.ranef` gives a list of standard errors for `ranef`.

Author(s)

Andrew Gelman <gelman@stat.columbia.edu>; Yu-Sung Su <suyusung@tsinghua.edu.cn>

References


See Also

`display`, `coef`, `sigma.hat`

Examples

```r
# Here's a simple example of a model of the form, y = a + bx + error,  
# with 10 observations in each of 10 groups, and with both the  
# intercept and the slope varying by group. First we set up the model and data.

group <- rep(1:10, rep(10,10))  
mu.a <- 0  
sigma.a <- 2
```
mu.b <- 3
sigma.b <- 4
rho <- 0
Sigma.ab <- array (c(sigma.a^2, rho*sigma.a*sigma.b, rho*sigma.a*sigma.b, sigma.b^2), c(2,2))
sigma.y <- 1
ab <- mvrnorm (10, c(mu.a,mu.b), Sigma.ab)
a <- ab[,1]
b <- ab[,2]

#
x <- rnorm (100)
y1 <- rnorm (100, a[group] + b[group]*x, sigma.y)
y2 <- rbinom(100, 1, prob=invlogit(a[group] + b*x))

# lm fit
M1 <- lm (y1 ~ x)
se.coef (M1)

# glm fit
M2 <- glm (y2 ~ x)
se.coef (M2)

# lmer fit
M3 <- lmer (y1 ~ x + (1 + x |group))
se.coef (M3)
se.fixef (M3)
se.ranef (M3)

# glmer fit
M4 <- glmer (y2 ~ 1 + (0 + x |group), family=binomial(link="logit"))
se.coef (M4)
se.fixef (M4)
se.ranef (M4)

sigma.hat

Extract Residual Errors

Description

This generic function extracts residual errors from a fitted model.

Usage

sigma.hat(object,...)

## S3 method for class 'lm'
sigma.hat(object,...)

## S3 method for class 'glm'
sigma.hat(object,...)

## S3 method for class 'merMod'
sigma.hat(object,...)
sigma.hat(object,...)
## S3 method for class 'sim'
sigma.hat(object,...)
## S3 method for class 'sim.merMod'
sigma.hat(object,...)

Arguments

object any fitted model object of lm, glm and merMod class
...
other arguments

Author(s)
Andrew Gelman <gelman@stat.columbia.edu>; Yu-Sung Su <suyusung@tsinghua.edu.cn>

See Also
display, summary, lm, glm, lmer

Examples

group <- rep(1:10, rep(10,10))
mu.a <- 0
sigma.a <- 2
mu.b <- 3
sigma.b <- 4
rho <- 0
Sigma.ab <- array(c(sigma.a^2, rho*sigma.a*sigma.b,
rho*sigma.a*sigma.b, sigma.b^2), c(2,2))
sigma.y <- 1
ab <- mvrnorm(10, c(mu.a, mu.b), Sigma.ab)
a <- ab[,1]
b <- ab[,2]
x <- rnorm(100)
y1 <- rnorm(100, a[group] + b[group]*x, sigma.y)
y2 <- rbinom(100, 1, prob=invlogit(a[group] + b*x))
M1 <- lm(y1 ~ x)
sigma.hat(M1)

M2 <- bayesglm(y1 ~ x, prior.scale=Inf, prior.df=Inf)
sigma.hat(M2) # should be same to sigma.hat(M1)

M3 <- glm(y2 ~ x, family=binomial(link="logit"))
sigma.hat(M3)

M4 <- lmer(y1 ~ (1+x|group))
sigma.hat(M4)

M5 <- glmer(y2 ~ (1+x|group), family=binomial(link="logit"))
Functions to Get Posterior Distributions

Description

This generic function gets posterior simulations of sigma and beta from a \texttt{lm} object, or simulations of beta from a \texttt{glm} object, or simulations of beta from a \texttt{merMod} object.

Usage

\begin{verbatim}
sim(object, ...)  
## S4 method for signature 'lm'  
sim(object, n.sims = 100)  
## S4 method for signature 'glm'  
sim(object, n.sims = 100)  
## S4 method for signature 'polr'  
sim(object, n.sims = 100)  
## S4 method for signature 'merMod'  
sim(object, n.sims = 100)  
  
## S3 method for class 'sim'  
coef(object,...)  
## S3 method for class 'sim.polr'  
coef(object, slot=c("ALL", "coef", "zeta"),...)  
## S3 method for class 'sim.merMod'  
coef(object,...)  
## S3 method for class 'sim.merMod'  
fixef(object,...)  
## S3 method for class 'sim.merMod'  
ranef(object,...)  
## S3 method for class 'sim.merMod'  
fitted(object, regression,...)  
\end{verbatim}

Arguments

\begin{itemize}
\item \texttt{object} \quad the output of a call to \texttt{lm} with \texttt{n} data points and \texttt{k} predictors.
\item \texttt{slot} \quad return which slot of \texttt{sim.polr}, available options are \texttt{coef}, \texttt{zeta}, \texttt{ALL}.
\item \ldots \quad further arguments passed to or from other methods.
\item \texttt{n.sims} \quad number of independent simulation draws to create.
\item \texttt{regression} \quad the original \texttt{merMod} model.
\end{itemize}
Value

- **coef**: matrix (dimensions n.sims x k) of n.sims random draws of coefficients.
- **zeta**: matrix (dimensions n.sims x k) of n.sims random draws of zetas (cut points in `polr`).
- **fixef**: matrix (dimensions n.sims x k) of n.sims random draws of coefficients of the fixed effects for the `merMod` objects. Previously, it is called `unmodeled`.
- **sigma**: vector of n.sims random draws of sigma (for `glm`'s, this just returns a vector of 1's or else of the square root of the overdispersion parameter if that is in the model)

Author(s)

Andrew Gelman <gelman@stat.columbia.edu>; Yu-Sung Su <suyusung@tsinghua.edu.cn>; Vincent Dorie <vjd4@nyu.edu>

References


See Also

display, lm, glm, lmer

Examples

```r
# Examples of "sim"
set.seed(1)
J <- 15
n <- J*(J+1)/2
group <- rep(1:J, 1:J)
mu.a <- 5
sigma.a <- 2
a <- rnorm(J, mu.a, sigma.a)
b <- -3
x <- rnorm(n, 2, 1)
sigma.y <- 6
y <- rnorm(n, a[group] + b*x, sigma.y)
u <- runif(J, 0, 3)
y123.dat <- cbind(y, x, group)
# Linear regression
x1 <- y123.dat[,2]
y1 <- y123.dat[,1]
M1 <- lm(y1 ~ x1)
display(M1)
M1.sim <- sim(M1)
coef.M1.sim <- coef(M1.sim)
sigma.M1.sim <- sigma.hat(M1.sim)
## to get the uncertainty for the simulated estimates
apply(coef(M1.sim), 2, quantile)
```
quantile(sigma.hat(M1.sim))

# Logistic regression
u.data <- cbind(1:J, u)
dimnames(u.data)[[2]] <- c("group", "u")
u.data <- as.data.frame(u.data)
y <- rbinom(n, 1, invlogit(a[group] + b*x))
M2 <- glm(y ~ x, family=binomial(link="logit"))
display(M2)
M2.sim <- sim(M2)
coef.M2.sim <- coef(M2.sim)
sigma.M2.sim <- sigma.hat(M2.sim)

# Ordered Logistic regression
house.plr <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
display(house.plr)
M.plr <- sim(house.plr)
coef.sim <- coef(M.plr, slot="coef")
zeta.sim <- coef(M.plr, slot="zeta")
coefall.sim <- coef(M.plr)

# Using lmer:
# Example 1
E1 <- lmer(y ~ x + (1 | group))
display(E1)
E1.sim <- sim(E1)
coef.E1.sim <- coef(E1.sim)
fixef.E1.sim <- fixef(E1.sim)
ranef.E1.sim <- ranef(E1.sim)
sigma.E1.sim <- sigma.hat(E1.sim)
yhat <- fitted(E1.sim, E1)

# Example 2
u.full <- u[group]
E2 <- lmer(y ~ x + u.full + (1 | group))
display(E2)
E2.sim <- sim(E2)
coef.E2.sim <- coef(E2.sim)
fixef.E2.sim <- fixef(E2.sim)
ranef.E2.sim <- ranef(E2.sim)
sigma.E2.sim <- sigma.hat(E2.sim)
yhat <- fitted(E2.sim, E2)

# Example 3
y <- rbinom(n, 1, invlogit(a[group] + b*x))
E3 <- glmer(y ~ x + (1 | group), family=binomial(link="logit"))
display(E3)
E3.sim <- sim(E3)
coef.E3.sim <- coef(E3.sim)
fixef.E3.sim <- fixef(E3.sim)
ranef.E3.sim <- ranef(E3.sim)
sigma.E3.sim <- sigma.hat(E3.sim)
yhat <- fitted(E3.sim, E3)
standardize

Function for Standardizing Regression Predictors by Centering and Dividing by 2 sd’s

Description

Numeric variables that take on more than two values are each rescaled to have a mean of 0 and a sd of 0.5; Binary variables are rescaled to have a mean of 0 and a difference of 1 between their two categories; Non-numeric variables that take on more than two values are unchanged; Variables that take on only one value are unchanged

Usage

```r
## S4 method for signature 'lm'
standardize(object, unchanged = NULL,
    standardize.y = FALSE, binary.inputs = "center")

## S4 method for signature 'glm'
standardize(object, unchanged = NULL,
    standardize.y = FALSE, binary.inputs = "center")

## S4 method for signature 'merMod'
standardize(object, unchanged = NULL,
    standardize.y = FALSE, binary.inputs = "center")

## S4 method for signature 'polr'
standardize(object, unchanged = NULL,
    standardize.y = FALSE, binary.inputs = "center")
```

Arguments

- `object`: an object of class `lm` or `glm`
- `unchanged`: vector of names of parameters to leave unstandardized
- `standardize.y`: if TRUE, the outcome variable is standardized also
- `binary.inputs`: options for standardizing binary variables

Details

"0/1" (rescale so that the lower value is 0 and the upper is 1) "-0.5/0.5" (rescale so that the lower value is -0.5 and upper is 0.5) "center" (rescale so that the mean of the data is 0 and the difference between the two categories is 1) "full" (rescale by subtracting the mean and dividing by 2 sd’s) "leave.alone" (do nothing)

Author(s)

Andrew Gelman <gelman@stat.columbia.edu> Yu-Sung Su <suyusung@tsinghua.edu.cn>
References


See Also

rescale

Examples

```r
# Set up the fake data
n <- 100
x <- rnorm(n, 2, 1)
x1 <- rnorm(n)
x1 <- (x1-mean(x1))/(2*sd(x1))  # standardization
x2 <- rbinom(n, 1, .5)
b0 <- 1
b1 <- 1.5
b2 <- 2
y <- rbinom(n, 1, invlogit(b0+b1*x1+b2*x2))
y2 <- sample(1:5, n, replace=TRUE)
M1 <- glm(y ~ x, family=binomial(link="logit"))
display(M1)
M1.1 <- glm(y ~ rescale(x), family=binomial(link="logit"))
display(M1.1)
M1.2 <- standardize(M1)
display(M1.2)
# M1.1 & M1.2 should be the same
M2 <- polr(ordered(y2) ~ x)
display(M2)
M2.1 <- polr(ordered(y2) ~ rescale(x))
display(M2.1)
M2.2 <- standardize(M2.1)
display(M2.2)
# M2.1 & M2.2 should be the same
```

traceplot

*Trace plot of ‘bugs’ object*

Description

Displays a plot of iterations vs. sampled values for each variable in the chain, with a separate plot per variable.
Usage

```r
## S4 method for signature 'bugs'
traceplot( x, mfrow = c(1, 1), varname = NULL,
          match.head = TRUE, ask = TRUE,
          col = rainbow(x$n.chains),
          lty = 1, lwd = 1, ...)
```

Arguments

- `x`: A bugs object
- `mfrow`: graphical parameter (see `par`)
- `varname`: vector of variable names to plot
- `match.head`: matches the variable names by the beginning of the variable names in bugs object
- `ask`: logical; if TRUE, the user is asked before each plot, see `par(ask=.)`
- `col`: graphical parameter (see `par`)
- `lty`: graphical parameter (see `par`)
- `lwd`: graphical parameter (see `par`)
- `...`: further graphical parameters

Author(s)

Masanao Yajima <yajima@stat.columbia.edu>. Yu-Sung Su <suyusung@tsinghua.edu.cn>

See Also

densplot, plot.mcmc, traceplot

triangleplot

Description

Function for making a triangle plot from a square matrix

Usage

```r
triangleplot (x, y=NULL, cutpts=NULL, details=TRUE,
              n.col.legend=5, cex.col=0.7,
              cex.var=0.9, digits=1, color=FALSE)
```
triangleplot

Arguments

- **x**: a square matrix.
- **y**: a vector of names that corresponds to each element of the square matrix x.
- **cutpts**: a vector of cutting points for color legend, default is NULL. The function will decide the cutting points if cutpts is not assigned.
- **details**: show more than one digits correlation values. Default is TRUE. FALSE is suggested to get readable output.
- **n.col.legend**: number of legend for the color thermometer
- **cex.col**: font size of the color thermometer.
- **cex.var**: font size of the variable names.
- **digits**: number of digits shown in the text of the color thermometer.
- **color**: color of the plot, default is FALSE, which uses gray scale.

Details

The function makes a triangle plot from a square matrix, e.g., the correlation plot, see corrplot. If a square matrix contains missing values, the cells of missing values will be marked x.

Author(s)

Yu-Sung Su <suyusung@tsinghua.edu.cn>

See Also

corrplot, par

Examples

```r
old.par <- par(no.readonly = TRUE)

# create a square matrix
x <- matrix(runif(1600, 0, 1), 40, 40)

# fig 1
triangleplot(x)

# fig 2 assign cutting points
triangleplot(x, cutpts=c(0,0.25,0.5,0.75,1), digits=2)

# fig 3 if x contains missing value
x[12,13] <- x[13,12] <- NA
x[25,27] <- x[27,25] <- NA
triangleplot(x)

par(old.par)
```

#library(RColorBrewer)
# cormat <- cor(iris[, -5])
# triangleplot2(cormat, color = brewer.pal(5, "RdBu"),
#               n.col.legend = 5, cex.col = 0.7, cex.var = 0.5)
Index

* array
  contrast.bayes, 19
* datasets
  lalonde, 29
* design
  contrast.bayes, 19
* dplot
  balance, 2
  binnedplot, 13
  coefplot, 15
  corrplot, 20
  discrete.histogram, 21
  triangleplot, 48
* hplot
  balance, 2
  coefplot, 15
  multicomponent, 35
  residual.plot, 38
  traceplot, 47
* manip
  balance, 2
  coefplot, 15
  contrast.bayes, 19
  display, 23
  extractDIC, 26
  fround, 26
  GO, 27
  model.matrixBayes, 33
  rescale, 37
  se.coef, 39
  sigma.hat, 41
  standardize, 46
* methods
  balance, 2
  bayesglm, 4
  bayespolr, 10
  coefplot, 15
  display, 23
  extractDIC, 26
  GO, 27
  matching, 30
  mcsamp, 31
  readColumns, 37
  se.coef, 39
  sigma.hat, 41
  sim, 43
  standardize, 46
* models
  bayesglm, 4
  bayespolr, 10
  invlogit, 28
  matching, 30
  mcsamp, 31
  model.matrixBayes, 33
  se.coef, 39
  sim, 43
  standardize, 46
* print
  fround, 26
* regression
  bayesglm, 4
  bayespolr, 10
  contrast.bayes, 19

aov, 19
as.data.frame, 5
balance, 2, 30
balance-class(balance), 2
bayesglm, 4, 12, 17, 19, 34, 35
bayesglm-class(bayesglm), 4
bayesglm.fit(bayesglm), 4
bayespolr, 8, 10
bayespolr-class(bayespolr), 10
binned.resids(binnedplot), 13
binnedplot, 13

C, 19
coef, 40
<table>
<thead>
<tr>
<th>Function/Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>coef.sim(sim), 43</td>
</tr>
<tr>
<td>coefplot, 15, 36</td>
</tr>
<tr>
<td>coefplot,bugs-method (coefplot), 15</td>
</tr>
<tr>
<td>coefplot,glm-method (coefplot), 15</td>
</tr>
<tr>
<td>coefplot,lm-method (coefplot), 15</td>
</tr>
<tr>
<td>coefplot,numeric-method (coefplot), 15</td>
</tr>
<tr>
<td>coefplot,polr-method (coefplot), 15</td>
</tr>
<tr>
<td>coefplot.default (coefplot), 15</td>
</tr>
<tr>
<td>contr.bayes.ordered (contrast.bayes), 19</td>
</tr>
<tr>
<td>contr.bayes.unordered (contrast.bayes), 19</td>
</tr>
<tr>
<td>contr.helmert, 19</td>
</tr>
<tr>
<td>contr.poly, 19</td>
</tr>
<tr>
<td>contr.sum, 19</td>
</tr>
<tr>
<td>contrasts, 34</td>
</tr>
<tr>
<td>cor, 21</td>
</tr>
<tr>
<td>corrplot, 20, 49</td>
</tr>
<tr>
<td>densplot, 48</td>
</tr>
<tr>
<td>discrete.hist (discrete.histogram), 21</td>
</tr>
<tr>
<td>display, 17, 23, 32, 40, 42, 44</td>
</tr>
<tr>
<td>display,bayesglm-method (display), 23</td>
</tr>
<tr>
<td>display,glm-method (display), 23</td>
</tr>
<tr>
<td>display,lm-method (display), 23</td>
</tr>
<tr>
<td>display,merMod-method (display), 23</td>
</tr>
<tr>
<td>display,polr-method (display), 23</td>
</tr>
<tr>
<td>display,svyglm-method (display), 23</td>
</tr>
<tr>
<td>extractAIC.merMod (extractDIC), 26</td>
</tr>
<tr>
<td>extractDIC, 26</td>
</tr>
<tr>
<td>factor, 34</td>
</tr>
<tr>
<td>family, 5</td>
</tr>
<tr>
<td>fitted.sim.merMod(sim), 43</td>
</tr>
<tr>
<td>fixef.sim.merMod(sim), 43</td>
</tr>
<tr>
<td>fround, 26</td>
</tr>
<tr>
<td>G (GO), 27</td>
</tr>
<tr>
<td>GenMatch, 30</td>
</tr>
<tr>
<td>glm, 7, 8, 17, 19, 24, 42, 44</td>
</tr>
<tr>
<td>glm.control, 6</td>
</tr>
<tr>
<td>GO, 27</td>
</tr>
<tr>
<td>go (GO), 27</td>
</tr>
<tr>
<td>GO-class (GO), 27</td>
</tr>
<tr>
<td>invlogit, 28</td>
</tr>
<tr>
<td>lalonde, 29</td>
</tr>
<tr>
<td>lm, 17, 19, 24, 42, 44</td>
</tr>
<tr>
<td>lmer, 24, 31, 32, 42, 44</td>
</tr>
<tr>
<td>logit (invlogit), 28</td>
</tr>
<tr>
<td>matching, 4, 30, 30</td>
</tr>
<tr>
<td>mcpplot (multicomp.plot), 35</td>
</tr>
<tr>
<td>mcsamp, 31</td>
</tr>
<tr>
<td>mcsamp,merMod-method (mcsamp), 31</td>
</tr>
<tr>
<td>mcsamp.default (mcsamp), 31</td>
</tr>
<tr>
<td>model.extract, 35</td>
</tr>
<tr>
<td>model.frame, 34, 35</td>
</tr>
<tr>
<td>model.matrixBayes, 33</td>
</tr>
<tr>
<td>model.offset, 6</td>
</tr>
<tr>
<td>multicomp.plot, 35</td>
</tr>
<tr>
<td>na.exclude, 5</td>
</tr>
<tr>
<td>na.fail, 5</td>
</tr>
<tr>
<td>na.omit, 5</td>
</tr>
<tr>
<td>offset, 6</td>
</tr>
<tr>
<td>options, 5</td>
</tr>
<tr>
<td>par, 4, 14, 16, 17, 21, 49</td>
</tr>
<tr>
<td>pfround (fround), 26</td>
</tr>
<tr>
<td>plot, 14, 17</td>
</tr>
<tr>
<td>plot.balance (balance), 2</td>
</tr>
<tr>
<td>plot.mcmc, 48</td>
</tr>
<tr>
<td>polr, 11, 12, 24</td>
</tr>
<tr>
<td>predict.bayesglm (bayesglm), 4</td>
</tr>
<tr>
<td>print,bayesglm-method (bayesglm), 4</td>
</tr>
<tr>
<td>print,bayespolr-method (bayespolr), 10</td>
</tr>
<tr>
<td>print.balance (balance), 2</td>
</tr>
<tr>
<td>ranef.sim.merMod(sim), 43</td>
</tr>
<tr>
<td>read.columns (readColumns), 37</td>
</tr>
<tr>
<td>readColumns, 37</td>
</tr>
<tr>
<td>rescale, 37, 47</td>
</tr>
<tr>
<td>residual.plot, 38</td>
</tr>
<tr>
<td>round, 27</td>
</tr>
<tr>
<td>se.coef, 39</td>
</tr>
<tr>
<td>se.coef,glm-method (se.coef), 39</td>
</tr>
<tr>
<td>se.coef,lm-method (se.coef), 39</td>
</tr>
<tr>
<td>se.coef,merMod-method (se.coef), 39</td>
</tr>
<tr>
<td>se.fixef (se.coef), 39</td>
</tr>
<tr>
<td>se.ranef (se.coef), 39</td>
</tr>
<tr>
<td>show,bayesglm-method (bayesglm), 4</td>
</tr>
<tr>
<td>show,bayespolr-method (bayespolr), 10</td>
</tr>
<tr>
<td>sigma.hat, 40, 41</td>
</tr>
</tbody>
</table>
INDEX

sim, 32, 43
sim, glm-method (sim), 43
sim, lm-method (sim), 43
sim, merMod-method (sim), 43
sim, polr-method (sim), 43
sim-class (sim), 43
sim.merMod-class (sim), 43
standardize, 38, 46
standardize, glm-method (standardize), 46
standardize, lm-method (standardize), 46
standardize, merMod-method (standardize), 46
standardize, polr-method (standardize), 46
stats, 19
summary, 24, 42
svyglm, 24
terms, 35
terms.formula, 35
traceplot, 47, 48
traceplot, bugs-method (traceplot), 47
traceplot, mcmc.list-method (traceplot), 47
traceplot.default (traceplot), 47
triangleplot, 48