Package 'arm '

January 1, 2007

Version 1.0-2

Date 2007-1-1

Title Data Analysis Using Regression and Multilevel/Hierarchical Models

Author Andrew Gelman <gelman@stat.columbia.edu>, Yu-Sung Su <ys463@columbia.edu>, Jennifer Hill <jh1030@columbia.edu>, Maria Grazia Pittau <grazia@stat.columbia.edu>, Jouni Kerman <jouni@kerman.com> and Tian Zheng <tzheng@stat.columbia.edu>

Maintainer Yu-Sung Su <ys463@columbia.edu>

Depends methods, R (>= 2.4.0), MASS, Matrix (>= 0.9975-1), lme4 (>= 0.9975-1), R2WinBUGS, BRugs

Suggests car, coda, nnet

Description R functions for processing lm, glm, mer and polr output.

URL http://www.stat.columbia.edu/~gelman/software/

License GPL (version 2 or later)

R topics documented:

balanceplot 2
bayesglm 3
bayespolr 6
coefplot 8
corrplot 10
display 11
go 14
invlogit 14
lalonde 15
matching 16
mcsamp 17
rescale 19
se.coef 20
sigma.hat 22
sim 23
standardize 24

Index **27**

`balanceplot` *Plot of Balance Statistics*

Description

This function plots the balance statistics before and after matching.

Usage

```
balanceplot(rawdata, matched, pscore.fit,
    longcovnames = NULL,
    main = "Standardized Difference in Means",
    cex.main = 1, cex.vars = 0.8, cex.pts = 0.8,
    mar = c(0, 15, 4, 2), mgp = c(2, 0.25, 0),
    oma = c(0, 0, 0, 0), tcl = -0.2, ...)
```

Arguments

`rawdata`	unmatched rawdata
`matched`	matched data using "matching" function
`pscore.fit`	glm.fit object to get pscore
`longcovnames`	long covariate names. If not provided, plot will use covariate variable name by default
`main`	title of the plot
`cex.main`	font size of main title
`cex.vars`	font size of variabel names
`cex.pts`	point size of the estimates
`mar`	margin of the plot, see ?`par` for details
`mgp`	axis margin of the plot, see ?`par` for details
`oma`	outer margin of the plot, see ?`par` for details
`tcl`	length of ticks, see ?`par` for details
`...`	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.

Author(s)

Jeniffer Hill ⟨jh1030@columbia.edu⟩; Yu-Sung Su ⟨ys463@columbia.edu⟩

References

Andrew Gelman and Jennifer Hill, Data Analysis Using Regression and Multilevel/Hierarchical Models, Cambridge University Press, 2006. (Chater 10)

See Also

matching, par

Examples

```
data(lalonde)
attach(lalonde)
fit <- glm(treat ~ re74 + re75 + educ + black + hisp + married +
            nodegr + u74 + u75, family=binomial(link="logit"))
pscores <- predict(fit, type="link")
matches <- matching(z=lalonde$treat, score=pscores)
matched <- lalonde[matches$matched,]
balanceplot(lalonde, matched, fit)
```

---

bayesglm *Bayesian generalized linear models.*

Description

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

Usage

```
bayesglm (formula, family = gaussian, data,
    weights, subset, na.action,
    start = NULL, etastart, mustart,
    offset, control = glm.control(...),
    model = TRUE, method = "glm.fit",
    x = FALSE, y = TRUE, contrasts = NULL,
    prior.mean = 0, prior.scale = 2.5,
    prior.scale.for.intercept = 10, prior.df = 1,
    scaled = TRUE, n.iter = 50, ...)

bayesglm.fit (x, y, weights = rep(1, nobs),
    start = NULL, etastart = NULL,
    mustart = NULL, offset = rep(0, nobs), family = gaussian(),
    control = glm.control(), intercept = TRUE,
    prior.mean=0, prior.scale=2.5, prior.scale.for.intercept=10,
    prior.df=1, scaled=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.
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 (including 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 2.5. Can be a vector of length equal to the number of predictors (including the intercept, if any). If it is a scalar, it is expanded to the length of this vector.
prior.scale.for.intercept	prior scale for the intercept: default is 10.
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 (including the intercept, if any). If it is a scalar, it is expanded to the length of this vector.
scaled	if scaled = TRUE, then the prior distribution is rescaled: default is TRUE
n.iter	default is 50.
...	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., `offests`, `contrasts`, `deviance` for the null model) all work.

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

Value

See `glm` for details.

Author(s)

Andrew Gelman ⟨gelman@stat.columbia.edu⟩; Yu-Sung Su ⟨ys463@columbia.edu⟩; Maria Grazia Pittau ⟨grazia@stat.columbia.edu⟩; Aleks Jakulin ⟨Jackulin@stat.columbia.edu⟩

References

Andrew Gelman, Aleks Jakulin, Maria Grazia Pittau and Yu-Sung Su, A default prior distribution for logistic and other regression models, unpublished paper available at http://www.stat.columbia.edu/~gelman/standardize/

See Also

glm, bayespolr

Examples

```
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)  # (using the display() function from regression.R)

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.df=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.df=1)  # 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)
```

`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"),
```

```
    prior.mean = 0, prior.scale = 2.5, prior.df = 1,
    scaled = TRUE, prior.mean.for.cutpoints = 0,
    prior.scale.for.cutpoints = 10, prior.df.for.cutpoints = 1)
```

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)'
...	additional 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.
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).
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.
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.
prior.mean.for.cutpoints	prior mean for cutpoints: default is 0. 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.
prior.scale.for.cutpoints	prior scale for cutpoints: default is 10. 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.
prior.df.for.cutpoints	for t distribution: default is 1 (Cauchy). 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.

Details

The program is a simple alteration of `polr()` 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.

Author(s)

Andrew Gelman ⟨gelman@stat.columbia.edu⟩; Yu-Sung Su ⟨ys463@columbia.edu⟩; Maria Grazia Pittau ⟨grazia@stat.columbia.edu⟩

See Also

`bayesglm`, `polr`

Examples

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

`coefplot` *Function for Making Coefficient Plot*

Description

A function that plot the coefficients from a lm or glm fit

Usage

```
coefplot(fit, longnames=NULL,
         x.label="", y.label="", main.label="",
         intercept=FALSE, varnames=TRUE, display.fit=FALSE,
         cex.var=0.8, cex.pts=0.9,...)
```

Arguments

`fit`	fitted objects-lm or glm
`longnames`	long variable names (vector). If not provided, use default variable names
`x.label`	label for x-axis
`y.label`	label for y-axis
`main.label`	label for the whole plot
`intercept`	If TRUE will plot intercept, default=FALSE

varnames	If FALSE will not plot variable names, default=TRUE
display.fit	If TRUE will display fitted output, default=FALSE
cex.var	The fontsize of the varible names, default=0.8
cex.pts	The size of data points, default=0.9
...	Other plot options may be passed to this function

Details

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

Value

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

Author(s)

Yu-Sung Su ⟨ys463@columbia.edu⟩

References

Andrew Gelman and Jennifer Hill, Data Analysis Using Regression and Multilevel/Hierarchical Models, Cambridge University Press, 2006.

See Also

display, par, lm, glm

Examples

```
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)
x6 <- rnorm(1000,100,30)
x7 <- rpois(1000,10)
x8 <- rbinom(1000,1,prob=0.4)
x9 <- rbeta(1000,5,4)
x10 <- runif(1000,-10,-1)
longnames <- c("a long name01","a long name02","a long name03",
               "a long name04","a long name05","a long name06",
               "a long name07","a long name08","a long name09",
               "a long name10")

fit1 <- lm(y1 ~ x1 + x2 + x3 + x4 +x5 +x6 +x7 +x8 + x9 + x10)
fit2 <- glm(y2 ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10,
          family=binomial(link="logit"))

# plot 1
par (mfrow=c(2,2))
coefplot(fit1)
coefplot(fit2)
```

```
# plot 2
par (mar=c(2,8,2,0.5))
coefplot(fit1, longnames=longnames, display=TRUE)

# plot 3
par (mar=c(2,2,2,2))
coefplot(fit2, longnames=longnames, varnames=FALSE)
```

`corrplot` *Function for Making Correlation Plot*

Description

Function for making a correlation plot starting from a data matrix

Usage

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

Arguments

`data`	a data matrix
`var.names`	variable names of the data matrix, if not provided use default variable names
`details`	show more than one digits correlaton 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 theromoeter
`color`	default is FALSE, which uses gray scale
`abs`	if TRUE, transform all correlation values into positive values
`...`	other plot options may be passed to this function

Details

The function adapts the R function for Figure 8 in Tian Zheng, Matthew Salganik, and Andrew Gelman, 2006, "How many people do you know in prison?: using overdispersion in count data to estimate social structure in networks", Journal of the American Statistical Association, Vol.101, N0. 474: p.409-23.

Value

A correlation plot.

Note

see also: http://www.stat.columbia.edu/~gelman/research/published/

Author(s)

Tian Zheng ⟨tzheng@stat.columbia.edu⟩; Yu-Sung Su ⟨ys463@columbia.edu⟩

References

Tian Zheng, Matthew Salganik, and Andrew Gelman, 2006, "How many people do you know in prison?: using overdispersion in count data to estimate social structure in networks", Journal of the American Statistical Association, Vol.101, N0. 474: p.409-23

See Also

cor, par

Examples

```
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")

#test 1
corrplot(test.data)

#test 2
corrplot(test.data,test.names,n.col.legend=7)
```

---

display *Functions for Processing lm, glm, mer and polr Output*

Description

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

Usage

```
display (object, digits = 2)

display.lm (object, digits=2)
display.glm (object, digits=2)
display.mer (object, digits=2)
display.polr (object, digits=2)
```

Arguments

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

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 ⟨ys463@columbia.edu⟩; Maria Grazia Pittau ⟨grazia@stat.columbia.edu⟩

References

Andrew Gelman and Jennifer Hill, Data Analysis Using Regression and Multilevel/Hierarchical Models, Cambridge University Press, 2006.

See Also

summary, lm, glm, lmer, polr

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))
   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))

#
# display a simple linear model
#
   M1 <- lm (y1 ~ x)
   display (M1)

#
# display a simple logit model
#
   M2 <- glm (y2 ~ x, family=binomial(link="logit"))
   display (M2)
#
# Then fit and display a simple varying-intercept model:
#
   M3 <- lmer (y1 ~ x + (1|group))
   display (M3)
   M3.sim <- mcsamp (M3)
   print (M3.sim)
   plot (M3.sim)
#
# Then the full varying-intercept, varying-slope model:
#
   M4 <- lmer (y1 ~ x + (1 + x |group))
   display (M4)
   M4.sim <- mcsamp (M4)
   print (M4.sim)
   plot (M4.sim)
#
# Then the full varying-intercept, logit model:
#
   M5 <- lmer (y2 ~ x + (1|group), family=binomial(link="logit"))
   display (M5)
   M5.sim <- mcsamp (M5)
   print (M5.sim)
   plot (M5.sim)
#
# Then the full varying-intercept, varying-slope logit model:
#
   M6 <- lmer (y2 ~ x + (1 + x |group), family=binomial(link="logit"))
   display (M6)
   M6.sim <- mcsamp (M6)
   print (M6.sim)
   plot (M6.sim)

#
# Then the ordered logit model from polr
#
   M7 <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
   display(M7)

   M8 <- bayespolr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
```

```
display(M8)
```

`go` *Function to Recall Last Source File*

Description

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

Usage

```
go(..., which.source=NULL, add=FALSE, timer=FALSE)
```

Arguments

`...`	list of filenames as character strings
`which.source`	the source file of which the first arugment belongs to
`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
```

`invlogit` *Logistic function*

Description

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

Usage

```
invlogit(x)
```

Arguments

`x`	A vector of continuous values

Details

The Inverse-logit function defined as: $logit^{-}1(x) = e^x/(1 + e^x)$ 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

Andrew Gelman and Jennifer Hill, Data Analysis Using Regression and Multilevel/Hierarchical Models, Cambridge University Press, 2006.

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)
```

---

lalonde *Lalonde Dataset*

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 martial 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://elsa.berkeley.edu/~imbens/matlab/lalonde_exper_04feb2.m. Many of these models are found to produce good balance for the Lalonde data.

Note

This documentation is adapted from `Matching` package.

References

Dehejia, Rajeev and Sadek Wahba. 1999."Causal Effects in Non-Experimental Studies: Re-Evaluating the Evaluation of Training Programs." *Journal of the American Statistical Association* 94 (448): 1053-1062.

LaLonde, Robert. 1986. "Evaluating the Econometric Evaluations of Training Programs." *American Economic Review* 76:604-620.

See Also

`matching`, `GenMatch` `balanceplot`

Examples

```
data(lalonde)
```

`matching` *Matching*

Description

Function for processing matching with propensity score

Usage

```
matching(z, score)
```

Arguments

`z`	vector of indicators for treatment or control.
`score`	vector of the propensity scores in the same order as z.

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)

Jeniffer Hill ⟨jh1030@columbia.edu⟩; Yu-Sung Su ⟨ys463@columbia.edu⟩

References

Andrew Gelman and Jennifer Hill, Data Analysis Using Regression and Multilevel/Hierarchical Models, Cambridge University Press, 2006.

See Also

balanceplot

Examples

```
data(lalonde)
attach(lalonde)
fit <- glm(treat ~ re74 + re75 + educ + black + hisp + married
               + nodegr + u74 + u75, family=binomial(link="logit"))
pscores <- predict(fit, type="link")
matches <- matching(z=lalonde$treat, score=pscores)
matched <- lalonde[matches$matched,]
balanceplot(lalonde, matched, fit)
```

---

mcsamp *Quick Function to Run mcmcsamp() in lme4*

Description

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

Usage

```
mcsamp (object, n.chains=3, n.iter=1000,
    n.burnin=floor(n.iter/2), n.thin=1,
    saveb=TRUE, make.bugs.object=TRUE)
```

Arguments

object	lmer or glmer objects
n.chains	number of MCMC chains
n.iter	number of iteration for each MCMC chain
n.burnin	number of burnin for each MCMC chain
n.thin	number of thin for each MCMC chain
saveb	if 'TRUE', causes the values of the random effects in each sample to be saved.
make.bugs.object	tranform the output into bugs object, default is TRUE

Details

This generic generates a sample from the posterior distribution of the parameters of a fitted model using Markov Chain Monte Carlo methods.

Value

An object of (S3) class '"mcmc"' suitable for use with the functions in the "coda" package.

Author(s)

Andrew Gelman ⟨gelman@stat.columbia.edu⟩; Yu-Sung Su ⟨ys463@columbia.edu⟩

References

Andrew Gelman and Jennifer Hill, Data Analysis Using Regression and Multilevel/Hierarchical Models, Cambridge University Press, 2006.

See Also

display, sim

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))
   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))
#
# 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 + x |group), family=binomial(link="logit"))
   display (M4)
   M4.sim <- mcsamp (M4)
   print (M4.sim)
   plot (M4.sim)
```

`rescale` *Function for Standardizing by Centering and Dividing by 2 sd's*

Description

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

Usage

```
rescale(x, binary.inputs)
```

Arguments

`x`	a vector
`binary.inputs`	options for standardizing binary variables

Value

the standardized vector

Author(s)

Andrew Gelman ⟨gelman@stat.columbia.edu⟩; Yu-Sung Su ⟨ys463@columbia.edu⟩

References

Andrew Gelman, Scaling regression inputs by dividing by two standard deviations. http://www.stat.columbia.edu/~gelman/research/unpublished/standardizing.pdf

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, "full2")
rescale(y, "center")
```

---

se.coef *Extract Standard Errors of Model Coefficients*

Description

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

Usage

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

se.coef.lm (object)
se.coef.glm (object)
se.coef.mer (object)
```

Arguments

object	object of lm, glm, lmer and glmer fit

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 `coef` and `se.ranef` gives a list of standard errors for `coef`.

Author(s)

Andrew Gelman ⟨gelman@stat.columbia.edu⟩; Yu-Sung Su ⟨ys463@columbia.edu⟩

References

Andrew Gelman and Jennifer Hill, Data Analysis Using Regression and Multilevel/Hierarchical Models, Cambridge University Press, 2006.

See Also

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

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))
   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 <- lmer (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)
```

Arguments

`object`	any fitted model object of lm, glm and mer class

Author(s)

Andrew Gelman ⟨gelman@stat.columbia.edu⟩; Yu-Sung Su ⟨ys463@columbia.edu⟩

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 <- glm (y2 ~ x, family=binomial(link="logit"))
sigma.hat(M2)

M3 <- lmer (y1 ~ (1+x|group))
sigma.hat(M3)

M4 <- lmer (y2 ~ (1+x|group), family=binomial(link="logit"))
sigma.hat(M4)
```

`sim` *Functions to Get Posterios Distributions*

Description

This generic function gets posterior simulations of sigma and beta from a lm object, or simulations of beta from a glm object, or simulations of beta from a mer object

Usage

```
sim(object, n.sims = 100)

sim.lm (object, n.sims = 100)
sim.glm (object, n.sims = 100)
sim.mer (object, n.sims = 100)
```

Arguments

`object`	the output of a call to "lm" with n data points and k predictors
`n.sims`	number of independent simulation draws to create

Value

`sigma.sim`	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)
`beta.sim`	matrix (dimensions n.sims x k) of n.sims random draws of beta

Author(s)

Andrew Gelman ⟨gelman@stat.columbia.edu⟩; Yu-Sung Su ⟨ys463@columbia.edu⟩; M.Grazia Pittau ⟨grazia@stat.columbia.edu⟩

References

Andrew Gelman and Jennifer Hill, Data Analysis Using Regression and Multilevel/Hierarchical Models, Cambridge University Press, 2006.

See Also

`display`, `mcsamp`, `lm`, `glm`, `lmer`

Examples

```
#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 <- as.data.frame (round (cbind (y, x, group), 2))

# Linear regression
 x1 <- y123.dat$y
 y1 <- y123.dat$x
 M1 <- lm (y1 ~ x1)
 display(M1)
 M1.sim <- sim(M1)

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

# Using lmer:
# Example 1
 E1 <- lmer (y ~ x + (1 | group))
 display(E1)
 E1.sim <- sim (E1)

# Example 2
 u.full <- u[group]
 E2 <- lmer (y ~ x + u.full + (1 | group))
 display(E2)
 E2.sim <- sim (E2)

# Example 3
# Set up the fake data for Example 3
 y <- rbinom (n, 1, invlogit (a[group] + b*x))
 E3 <- lmer (y ~ x + (1 | group), family=binomial(link="logit"))
 display(E3)
 E3.sim <- 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

```
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 ⟨ys463@columbia.edu⟩

References

Andrew Gelman, Scaling regression inputs by dividing by two standard deviations http://www.stat.columbia.edu/~gelman/research/unpublished/standardizing.pdf

See Also

`rescale`

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))
M1 <- glm (y ~ x, family=binomial(link="logit"))
display(M1)
```

```
M2 <-standardize(M1)
display(M2)
```

Index

∗Topic **datasets**
  lalonde, 15
∗Topic **dplot**
  balanceplot, 1
  coefplot, 8
  corrplot, 10
∗Topic **methods**
  balanceplot, 1
  bayesglm, 3
  bayespolr, 6
  display, 11
  go, 13
  matching, 16
  mcsamp, 17
  se.coef, 20
  sigma.hat, 22
  sim, 23
  standardize, 24
∗Topic **models**
  bayesglm, 3
  bayespolr, 6
  invlogit, 14
  matching, 16
  mcsamp, 17
  rescale, 19
  se.coef, 20
  sim, 23
  standardize, 24

as.data.frame, *3*

balanceplot, 1, *16, 17*
bayesglm, 3, *8*
bayespolr, *5*, 6

coef, *21*
coefplot, 8
cor, *11*
corrplot, 10

display, *9*, 11, *18*, *21–23*
display,glm-method *(display)*, 11
display,lm-method *(display)*, 11
display,mer-method *(display)*, 11
display,polr-method *(display)*, 11
display.glm *(display)*, 11
display.lm *(display)*, 11
display.mer *(display)*, 11
display.polr *(display)*, 11

family, *3*

GenMatch, *16*
glm, *5*, *9*, *12*, *22, 23*
glm.control, *4*
go, 13

invlogit, 14

lalonde, 15
lm, *9*, *12*, *22, 23*
lmer, *12*, *22, 23*

matching, *2*, 16, *16*
mcsamp, 17, *23*
model.offset, *4*

na.exclude, *3*
na.fail, *3*
na.omit, *3*

offset, *4*
options, *3*

par, *2*, *9*, *11*
polr, *8*, *12*

rescale, 19, *25*

se.coef, 20
se.coef,glm-method *(se.coef)*, 20
se.coef,lm-method *(se.coef)*, 20
se.coef,mer-method *(se.coef)*, 20
se.coef.glm *(se.coef)*, 20
se.coef.lm *(se.coef)*, 20
se.coef.mer *(se.coef)*, 20
se.fixef *(se.coef)*, 20
se.ranef *(se.coef)*, 20
sigma.hat, *21*, 22

sigma.hat,glm-method(*sigma.hat*), 22
sigma.hat,lm-method(*sigma.hat*), 22
sigma.hat,mer-method(*sigma.hat*), 22
sigma.hat.glm(*sigma.hat*), 22
sigma.hat.lm(*sigma.hat*), 22
sigma.hat.mer(*sigma.hat*), 22
sim, *18*, 23
sim,glm-method(*sim*), 23
sim,lm-method(*sim*), 23
sim,mer-method(*sim*), 23
sim.glm(*sim*), 23
sim.lm(*sim*), 23
sim.mer(*sim*), 23
standardize, *20*, 24
summary, *12*, *22*