Package ‘BayesPostEst’

October 6, 2019

Type Package

Title Generate Postestimation Quantities for Bayesian MCMC Estimation

Version 0.1.0

Date 2019-10-01

Description An implementation of functions to generate and plot postestimation quantities after estimating Bayesian regression models using Markov chain Monte Carlo (MCMC). Functionality includes the estimation of the Precision-Recall curves (see Beger, 2016 <doi:10.2139/ssrn.2765419>), the implementation of the observed values method of calculating predicted probabilities by Hanmer and Kalkan (2013) <doi:10.1111/j.1540-5907.2012.00602.x>, the implementation of the average value method of calculating predicted probabilities (see King, Tomz, and Wittenberg, 2000 <doi:10.2307/2669316>), and the generation and plotting of first differences to summarize typical effects across covariates (see Long 1997, ISBN:9780803973749; King, Tomz, and Wittenberg, 2000 <doi:10.2307/2669316>). This package can be used with MCMC output generated by any Bayesian estimation tool including 'JAGS', 'BUGS', 'MCMCpack', and 'Stan'.

URL https://github.com/ShanaScogin/BayesPostEst

BugReports https://github.com/ShanaScogin/BayesPostEst/issues

License GPL-3

Imports carData, caTools, coda (>= 0.13), dplyr (>= 0.5.0), ggmcmc, ggplot2, ggridges, R2jags, reshape2, rlang, stats, texreg, runjags, tidyr (>= 0.5.1)

Depends R (>= 2.14.0)

Encoding UTF-8

LazyData TRUE

LazyLoad TRUE

Suggests datasets, knitr, MCMCpack, rjags, rmarkdown, rstan (>= 2.10.1), rstanarm, testthat, covr

VignetteBuilder knitr

RoxygenNote 6.1.1

SystemRequirements JAGS (http://mcmc-jags.sourceforge.net)

NeedsCompilation no

Author Johannes Karreth [aut] (<https://orcid.org/0000-0003-4586-7153>),
Shana Scogin [aut, cre] (<https://orcid.org/0000-0002-7801-853X>),
Rob Williams [aut] (<https://orcid.org/0000-0001-9259-3883>),
Andreas Beger [aut] (<https://orcid.org/0000-0003-1883-3169>),
Myunghee Lee [ctb],
Neil Williams [ctb]

Maintainer Shana Scogin <`shanarscogin@gmail.com`>

Repository CRAN

Date/Publication 2019-10-06 04:10:03 UTC

R topics documented:

`BayesPostEst` *BayesPostEst*

Description

BayesPostEst

BayesPostEst functions

This package currently has seven main functions

- `mcmcAveProb()`
- `mcmcObsProb()`
- `mcmcFD()`
- `mcmcFDplot()`
- `mcmcRocPrc()`

- `mcmcTab()`
- `mcmcReg()`

These functions can be used to generate and plot postestimation quantities after estimating Bayesian regression models using MCMC. The package combines functions written originally for Johannes Karreth's workshop on Bayesian modeling at the ICPSR Summer program. For now, the package focuses mostly on generalized linear regression models for binary outcomes (logistic and probit regression). The vignette for this package has a walk-through of each function in action. Please refer to that to get an overview of all the functions, or visit the documentation for a specific function of your choice. Johannes Karreth's website (http://www.jkarreth.net) also has resources for getting started with Bayesian analysis, fitting models, and presenting results.

`jags_logit` *Fitted JAGS logit model*

Description

A fitted JAGS logit model generated with [R2jags::jags()]. See the example code below for how it was created. Used in examples and for testing.

Usage

```
jags_logit
```

Format

A class "rjags" object created by [R2jags::jags()]

Examples

```
data("sim_data")

## formatting the data for jags
datjags <- as.list(data)
datjags$N <- length(datjags$Y)

## creating jags model
model <- function()  {

  for(i in 1:N){
    Y[i] ~ dbern(p[i])  ## Bernoulli distribution of y_i
      logit(p[i]) <- mu[i]    ## Logit link function
      mu[i] <- b[1] +
        b[2] * X1[i] +
        b[3] * X2[i]
  }

  for(j in 1:3){
```

```
    b[j] ~ dnorm(0, 0.001) ## Use a coefficient vector for simplicity
  }

}

params <- c("b")
inits1 <- list("b" = rep(0, 3))
inits2 <- list("b" = rep(0, 3))
inits <- list(inits1, inits2)

## fitting the model with R2jags
set.seed(123)
jags_logit <- R2jags::jags(data = datjags, inits = inits,
                         parameters.to.save = params, n.chains = 2,
                         n.iter = 2000, n.burnin = 1000, model.file = model)
```

---

jags_probit *Fitted JAGS probit model*

Description

A fitted JAGS probit model generated with [R2jags::jags()]. See the example code below for how it was created. Used in examples and for testing.

Usage

```
jags_probit
```

Format

A class "rjags" object created by [R2jags::jags()]

Examples

```
data("sim_data")

## formatting the data for jags
datjags <- as.list(data)
datjags$N <- length(datjags$Y)

## creating jags model
model <- function()  {

  for(i in 1:N){
    Y[i] ~ dbern(p[i])  ## Bernoulli distribution of y_i
      probit(p[i]) <- mu[i]    ## Update with probit link function
```

```
      mu[i] <- b[1] +
        b[2] * X1[i] +
        b[3] * X2[i]
  }

  for(j in 1:3){
    b[j] ~ dnorm(0, 0.001) ## Use a coefficient vector for simplicity
  }

}

params <- c("b")
inits1 <- list("b" = rep(0, 3))
inits2 <- list("b" = rep(0, 3))
inits <- list(inits1, inits2)

## fitting the model with R2jags
set.seed(123)
jags_probit <- R2jags::jags(data = datjags, inits = inits,
                         parameters.to.save = params, n.chains = 2,
                         n.iter = 2000, n.burnin = 1000, model.file = model)
```

`mcmcAveProb` *Predicted Probabilities using Bayesian MCMC estimates for the "Average" Case*

Description

This function calculates predicted probabilities for "average" cases after a Bayesian logit or probit model. As "average" cases, this function calculates the median value of each predictor. For an explanation of predicted probabilities for "average" cases, see e.g. King, Tomz & Wittenberg (2000, American Journal of Political Science 44(2): 347-361).

Usage

```
mcmcAveProb(modelmatrix, mcmcout, xcol, xrange, xinterest,
  link = "logit", ci = c(0.025, 0.975), fullsims = FALSE)
```

Arguments

`modelmatrix` model matrix, including intercept (if the intercept is among the parameters estimated in the model). Create with model.matrix(formula, data). Note: the order of columns in the model matrix must correspond to the order of columns in the matrix of posterior draws in the `mcmcout` argument. See the `mcmcout` argument for more.

`mcmcout`	posterior distributions of all logit coefficients, in matrix form. This can be created from rstan, MCMCpack, R2jags, etc. and transformed into a matrix using the function as.mcmc() from the coda package for `jags` class objects, as.matrix() from base R for `mcmc`, `mcmc.list`, `stanreg`, and `stanfit` class objects, and `object$sims.matrix` for `bugs` class objects. Note: the order of columns in this matrix must correspond to the order of columns in the model matrix. One can do this by examining the posterior distribution matrix and sorting the variables in the order of this matrix when creating the model matrix. A useful function for sorting column names containing both characters and numbers as you create the matrix of posterior distributions is `mixedsort()` from the gtools package.
`xcol`	column number of the posterior draws (`mcmcout`) and model matrices that corresponds to the explanatory variable for which to calculate associated Pr(y = 1). Note that the columns in these matrices must match.
`xrange`	name of the vector with the range of relevant values of the explanatory variable for which to calculate associated Pr(y = 1).
`xinterest`	semi-optional argument. Name of the explanatory variable for which to calculate associated Pr(y = 1). If `xcol` is supplied, this is not needed. If both are supplied, the function defaults to `xcol` and this argument is ignored.
`link`	type of generalized linear model; a character vector set to `"logit"` (default) or `"probit"`.
`ci`	the bounds of the credible interval. Default is `c(0.025,0.975)` for the 95% credible interval.
`fullsims`	logical indicator of whether full object (based on all MCMC draws rather than their average) will be returned. Default is `FALSE`. Note: The longer `xrange` is, the larger the full output will be if `TRUE` is selected.

Details

This function calculates predicted probabilities for "average" cases after a Bayesian logit or probit model. For an explanation of predicted probabilities for "average" cases, see e.g. King, Tomz & Wittenberg (2000, American Journal of Political Science 44(2): 347-361)

Value

if `fullsims` = `FALSE` (default), a tibble with 4 columns:

- x: value of variable of interest, drawn from `xrange`
- median_pp: median predicted Pr(y = 1) when variable of interest is set to x, holding all other predictors to average (median) values
- lower_pp: lower bound of credible interval of predicted probability at given x
- upper_pp: upper bound of credible interval of predicted probability at given x

if `fullsims` = `TRUE`, a tibble with 3 columns:

- Iteration: number of the posterior draw
- x: value of variable of interest, drawn from `xrange`
- pp: average predicted Pr(y = 1) when variable of interest is set to x, holding all other predictors to average (median) values

References

King, Gary, Michael Tomz, and Jason Wittenberg. 2000. "Making the Most of Statistical Analyses: Improving Interpretation and Presentation." American Journal of Political Science 44 (2): 347–61. http://www.jstor.org/stable/2669316

Examples

```
  ## simulating data
  set.seed(123456)
  b0 <- 0.2 # true value for the intercept
  b1 <- 0.5 # true value for first beta
  b2 <- 0.7 # true value for second beta
  n <- 500 # sample size
  X1 <- runif(n, -1, 1)
  X2 <- runif(n, -1, 1)
  Z <- b0 + b1 * X1 + b2 * X2
  pr <- 1 / (1 + exp(-Z)) # inv logit function
  Y <- rbinom(n, 1, pr)
  data <- data.frame(cbind(X1, X2, Y))

  ## formatting the data for jags
  datjags <- as.list(data)
  datjags$N <- length(datjags$Y)

  ## creating jags model
  model <- function()  {

  for(i in 1:N){
    Y[i] ~ dbern(p[i])  ## Bernoulli distribution of y_i
    logit(p[i]) <- mu[i]    ## Logit link function
    mu[i] <- b[1] +
      b[2] * X1[i] +
      b[3] * X2[i]
  }

  for(j in 1:3){
    b[j] ~ dnorm(0, 0.001) ## Use a coefficient vector for simplicity
  }

}

params <- c("b")
inits1 <- list("b" = rep(0, 3))
inits2 <- list("b" = rep(0, 3))
inits <- list(inits1, inits2)

## fitting the model with R2jags
library(R2jags)
set.seed(123)
fit <- jags(data = datjags, inits = inits,
         parameters.to.save = params, n.chains = 2, n.iter = 2000,
```

```
        n.burnin = 1000, model.file = model)

### average value approach
library(coda)
xmat <- model.matrix(Y ~ X1 + X2, data = data)
mcmc <- as.mcmc(fit)
mcmc_mat <- as.matrix(mcmc)[, 1:ncol(xmat)]
X1_sim <- seq(from = min(datjags$X1),
              to = max(datjags$X1),
              length.out = 10)
ave_prob <- mcmcAveProb(modelmatrix = xmat,
                        mcmcout = mcmc_mat,
                        xrange = X1_sim,
                        xcol = 2)
```

---

mcmcFD *First Differences of a Bayesian Logit or Probit model*

Description

R function to calculate first differences after a Bayesian logit or probit model. First differences are a method to summarize effects across covariates. This quantity represents the difference in predicted probabilities for each covariate for cases with low and high values of the respective covariate. For each of these differences, all other variables are held constant at their median. For more, see Long (1997, Sage Publications) and King, Tomz, and Wittenberg (2000, American Journal of Political Science 44(2): 347-361).

Usage

```
mcmcFD(modelmatrix, mcmcout, link = "logit", ci = c(0.025, 0.975),
  percentiles = c(0.25, 0.75), fullsims = FALSE)
```

Arguments

`modelmatrix` model matrix, including intercept (if the intercept is among the parameters estimated in the model). Create with model.matrix(formula, data). Note: the order of columns in the model matrix must correspond to the order of columns in the matrix of posterior draws in the `mcmcout` argument. See the `mcmcout` argument for more.

`mcmcout` posterior distributions of all logit coefficients, in matrix form. This can be created from rstan, MCMCpack, R2jags, etc. and transformed into a matrix using the function as.mcmc() from the coda package for `jags` class objects, as.matrix() from base R for `mcmc`, `mcmc.list`, `stanreg`, and `stanfit` class objects, and `object$sims.matrix` for bugs class objects. Note: the order of columns in this matrix must correspond to the order of columns in the model matrix. One can do this by examining the posterior distribution matrix and sorting the variables in the order of this matrix when creating the model matrix. A useful function for sorting column names containing both characters and numbers as you create the matrix of posterior distributions is `mixedsort()` from the gtools package.

`link` type of generalized linear model; a character vector set to `"logit"` (default) or `"probit"`.

`ci` the bounds of the credible interval. Default is `c(0.025,0.975)` for the 95% credible interval.

`percentiles` values of each predictor for which the difference in Pr(y = 1) is to be calculated. Default is `c(0.25,0.75)`, which will calculate the difference between Pr(y = 1) for the 25th percentile and 75th percentile of the predictor. For binary predictors, the function automatically calculates the difference between Pr(y = 1) for x = 0 and x = 1.

`fullsims` logical indicator of whether full object (based on all MCMC draws rather than their average) will be returned. Default is `FALSE`.

Value

if `fullsims = FALSE` (default), a data frame with four columns:

- median_fd: median first difference
- lower_fd: lower bound of credible interval of the first difference
- upper_fd: upper bound of credible interval of the first difference
- VarName: name of the variable as found in `modelmatrix`
- VarID: identifier of the variable, based on the order of columns in `modelmatrix` and `mcmcout`. Can be adjusted for plotting

if `fullsims = TRUE`, a data frame with as many columns as predictors in the model. Each row is the first difference for that variable based on one set of posterior draws. Column names are taken from the column names of `modelmatrix`.

References

- King, Gary, Michael Tomz, and Jason Wittenberg. 2000. “Making the Most of Statistical Analyses: Improving Interpretation and Presentation.” American Journal of Political Science 44 (2): 347–61. http://www.jstor.org/stable/2669316
- Long, J. Scott. 1997. Regression Models for Categorical and Limited Dependent Variables. Thousand Oaks: Sage Publications

Examples

```
## simulating data
set.seed(123456)
b0 <- 0.2 # true value for the intercept
b1 <- 0.5 # true value for first beta
b2 <- 0.7 # true value for second beta
n <- 500 # sample size
X1 <- runif(n, -1, 1)
X2 <- runif(n, -1, 1)
```

```
Z <- b0 + b1 * X1 + b2 * X2
pr <- 1 / (1 + exp(-Z)) # inv logit function
Y <- rbinom(n, 1, pr)
data <- data.frame(cbind(X1, X2, Y))

## formatting the data for jags
datjags <- as.list(data)
datjags$N <- length(datjags$Y)

## creating jags model
model <- function()  {

  for(i in 1:N){
    Y[i] ~ dbern(p[i])  ## Bernoulli distribution of y_i
    logit(p[i]) <- mu[i]    ## Logit link function
    mu[i] <- b[1] +
      b[2] * X1[i] +
      b[3] * X2[i]
  }

  for(j in 1:3){
    b[j] ~ dnorm(0, 0.001) ## Use a coefficient vector for simplicity
  }

}

params <- c("b")
inits1 <- list("b" = rep(0, 3))
inits2 <- list("b" = rep(0, 3))
inits <- list(inits1, inits2)

## fitting the model with R2jags
set.seed(123)
fit <- R2jags::jags(data = datjags, inits = inits,
                    parameters.to.save = params, n.chains = 2, n.iter = 2000,
                    n.burnin = 1000, model.file = model)

## running function with logit
xmat <- model.matrix(Y ~ X1 + X2, data = data)
mcmc <- coda::as.mcmc(fit)
mcmc_mat <- as.matrix(mcmc)[, 1:ncol(xmat)]
object <- mcmcFD(modelmatrix = xmat,
                 mcmcout = mcmc_mat)
object
```

`mcmcFDplot` *Plot First Differences from MCMC output*

Description

R function to plot first differences generated from MCMC output. For more on this method, see the documentation for `mcmcFD()`, Long (1997, Sage Publications), and King, Tomz, and Wittenberg (2000, American Journal of Political Science 44(2): 347-361). For a description of this type of plot, see Figure 1 in Karreth (2018, International Interactions 44(3): 463-90).

Usage

```
mcmcFDplot(fdfull, ROPE = NULL)
```

Arguments

`fdfull`	Output generated from `mcmcFD(...,full_sims = TRUE)`.
`ROPE`	defaults to NULL. If not NULL, a numeric vector of length two, defining the Region of Practical Equivalence around 0. See Kruschke (2013, Journal of Experimental Psychology 143(2): 573-603) for more on the ROPE.

Details

An R function to plot first differences after a Bayesian logit or probit model.

Value

a density plot of the differences in probabilities. The plot is made with ggplot2 and can be passed on as an object to customize. Annotated numbers show the percent of posterior draws with the same sign as the median estimate (if `ROPE = NULL`) or on the same side of the ROPE as the median estimate (if `ROPE` is specified).

References

- Karreth, Johannes. 2018. "The Economic Leverage of International Organizations in Interstate Disputes." International Interactions 44 (3): 463-90. https://doi.org/10.1080/03050629.2018.1389728.
- King, Gary, Michael Tomz, and Jason Wittenberg. 2000. "Making the Most of Statistical Analyses: Improving Interpretation and Presentation." American Journal of Political Science 44 (2): 347–61. http://www.jstor.org/stable/2669316.
- Kruschke, John K. 2013. "Bayesian Estimation Supersedes the T-Test." Journal of Experimental Psychology: General 142 (2): 573–603. https://doi.org/10.1037/a0029146.
- Long, J. Scott. 1997. Regression Models for Categorical and Limited Dependent Variables. Thousand Oaks: Sage Publications.

Examples

```
## simulating data
set.seed(123456)
b0 <- 0.2 # true value for the intercept
b1 <- 0.5 # true value for first beta
b2 <- 0.7 # true value for second beta
n <- 500 # sample size
```

```
X1 <- runif(n, -1, 1)
X2 <- runif(n, -1, 1)
Z <- b0 + b1 * X1 + b2 * X2
pr <- 1 / (1 + exp(-Z)) # inv logit function
Y <- rbinom(n, 1, pr)
data <- data.frame(cbind(X1, X2, Y))

## formatting the data for jags
datjags <- as.list(data)
datjags$N <- length(datjags$Y)

## creating jags model
model <- function()  {

  for(i in 1:N){
    Y[i] ~ dbern(p[i])  ## Bernoulli distribution of y_i
    logit(p[i]) <- mu[i]    ## Logit link function
    mu[i] <- b[1] +
      b[2] * X1[i] +
      b[3] * X2[i]
  }

  for(j in 1:3){
    b[j] ~ dnorm(0, 0.001) ## Use a coefficient vector for simplicity
  }

}

params <- c("b")
inits1 <- list("b" = rep(0, 3))
inits2 <- list("b" = rep(0, 3))
inits <- list(inits1, inits2)

## fitting the model with R2jags
set.seed(123)
fit <- R2jags::jags(data = datjags, inits = inits,
                    parameters.to.save = params, n.chains = 2, n.iter = 2000,
                    n.burnin = 1000, model.file = model)

## preparing data for mcmcFD()
xmat <- model.matrix(Y ~ X1 + X2, data = data)
mcmc <- coda::as.mcmc(fit)
mcmc_mat <- as.matrix(mcmc)[, 1:ncol(xmat)]

## plotting with mcmcFDplot()
full <- mcmcFD(modelmatrix = xmat,
               mcmcout = mcmc_mat,
               fullsims = TRUE)
mcmcFDplot(full)
```

`mcmcObsProb` *Predicted Probabilities using Bayesian MCMC estimates for the Average of Observed Cases*

Description

Implements R function to calculate the predicted probabilities for "observed" cases after a Bayesian logit or probit model, following Hanmer & Kalkan (2013) (2013, American Journal of Political Science 57(1): 263-277).

Usage

```
mcmcObsProb(modelmatrix, mcmcout, xcol, xrange, xinterest,
  link = "logit", ci = c(0.025, 0.975), fullsims = FALSE)
```

Arguments

`modelmatrix`	model matrix, including intercept (if the intercept is among the parameters estimated in the model). Create with model.matrix(formula, data). Note: the order of columns in the model matrix must correspond to the order of columns in the matrix of posterior draws in the `mcmcout` argument. See the `mcmcout` argument for more.
`mcmcout`	posterior distributions of all logit coefficients, in matrix form. This can be created from rstan, MCMCpack, R2jags, etc. and transformed into a matrix using the function as.mcmc() from the coda package for `jags` class objects, as.matrix() from base R for `mcmc`, `mcmc.list`, `stanreg`, and `stanfit` class objects, and `object$sims.matrix` for `bugs` class objects. Note: the order of columns in this matrix must correspond to the order of columns in the model matrix. One can do this by examining the posterior distribution matrix and sorting the variables in the order of this matrix when creating the model matrix. A useful function for sorting column names containing both characters and numbers as you create the matrix of posterior distributions is `mixedsort()` from the gtools package.
`xcol`	column number of the posterior draws (`mcmcout`) and model matrices that corresponds to the explanatory variable for which to calculate associated Pr(y = 1). Note that the columns in these matrices must match.
`xrange`	name of the vector with the range of relevant values of the explanatory variable for which to calculate associated Pr(y = 1).
`xinterest`	semi-optional argument. Name of the explanatory variable for which to calculate associated Pr(y = 1). If `xcol` is supplied, this is not needed. If both are supplied, the function defaults to `xcol` and this argument is ignored.
`link`	type of generalized linear model; a character vector set to `"logit"` (default) or `"probit"`.
`ci`	the bounds of the credible interval. Default is `c(0.025,0.975)` for the 95% credible interval.

fullsims — logical indicator of whether full object (based on all MCMC draws rather than their average) will be returned. Default is FALSE. Note: The longer xrange is, the larger the full output will be if TRUE is selected.

Details

This function calculates predicted probabilities for "observed" cases after a Bayesian logit or probit model following Hanmer and Kalkan (2013, American Journal of Political Science 57(1): 263-277)

Value

if fullsims = FALSE (default), a tibble with 4 columns:

- x: value of variable of interest, drawn from xrange
- median_pp: median predicted Pr(y = 1) when variable of interest is set to x
- lower_pp: lower bound of credible interval of predicted probability at given x
- upper_pp: upper bound of credible interval of predicted probability at given x

if fullsims = TRUE, a tibble with 3 columns:

- Iteration: number of the posterior draw
- x: value of variable of interest, drawn from xrange
- pp: average predicted Pr(y = 1) of all observed cases when variable of interest is set to x

References

Hanmer, Michael J., & Ozan Kalkan, K. (2013). Behind the curve: Clarifying the best approach to calculating predicted probabilities and marginal effects from limited dependent variable models. American Journal of Political Science, 57(1), 263-277. https://doi.org/10.1111/j.1540-5907.2012.00602.x

Examples

```
## simulating data
set.seed(123456)
b0 <- 0.2 # true value for the intercept
b1 <- 0.5 # true value for first beta
b2 <- 0.7 # true value for second beta
n <- 500 # sample size
X1 <- runif(n, -1, 1)
X2 <- runif(n, -1, 1)
Z <- b0 + b1 * X1 + b2 * X2
pr <- 1 / (1 + exp(-Z)) # inv logit function
Y <- rbinom(n, 1, pr)
data <- data.frame(cbind(X1, X2, Y))

## formatting the data for jags
datjags <- as.list(data)
datjags$N <- length(datjags$Y)
```

```
  ## creating jags model
  model <- function()  {

  for(i in 1:N){
    Y[i] ~ dbern(p[i])  ## Bernoulli distribution of y_i
    logit(p[i]) <- mu[i]    ## Logit link function
    mu[i] <- b[1] +
      b[2] * X1[i] +
      b[3] * X2[i]
  }

  for(j in 1:3){
    b[j] ~ dnorm(0, 0.001) ## Use a coefficient vector for simplicity
  }

}

params <- c("b")
inits1 <- list("b" = rep(0, 3))
inits2 <- list("b" = rep(0, 3))
inits <- list(inits1, inits2)

## fitting the model with R2jags
library(R2jags)
set.seed(123)
fit <- jags(data = datjags, inits = inits,
          parameters.to.save = params, n.chains = 2, n.iter = 2000,
          n.burnin = 1000, model.file = model)

### observed value approach
library(coda)
xmat <- model.matrix(Y ~ X1 + X2, data = data)
mcmc <- as.mcmc(fit)
mcmc_mat <- as.matrix(mcmc)[, 1:ncol(xmat)]
X1_sim <- seq(from = min(datjags$X1),
              to = max(datjags$X1),
              length.out = 10)
obs_prob <- mcmcObsProb(modelmatrix = xmat,
                        mcmcout = mcmc_mat,
                        xrange = X1_sim,
                        xcol = 2)
```

`mcmcReg` *LaTeX or HTML regression tables for MCMC Output*

Description

This function creates LaTeX or HTML regression tables for MCMC Output using the `texreg` function from the `texreg` R package.

Usage

```
mcmcReg(mod, pars = NULL, pointest = "mean", ci = 0.95, hpdi = F,
  coefnames = NULL, gof = numeric(0), gofnames = character(0),
  format = "latex", file, ...)
```

Arguments

`mod`	Bayesian model object generated by R2jags, rjags, R2WinBUGS, R2OpenBUGS, MCMCpack, rstan, rstanarm, and brms, or a list of model objects of the same class.
`pars`	a scalar or vector of the parameters you wish to include in the table. By default, `mcmcReg` includes all parameters saved in a model object. If a model has lots of samples and lots of saved parameters, not explicitly specifying a limited number of parameters to include via `pars` may take a long time. `pars` can either be a vector with the specific parameters to be included in the table e.g. `pars = c("beta[1]","beta[2]","beta[3]")`, or they can be partial names that will be matched using regular expressions e.g. `pars = "beta"`. Both of these will include `beta[1]`, `beta[2]`, and `beta[3]` in the table. When combining models with different parameters in one table, this argument also accepts a list the length of the number of models.
`pointest`	a character indicating whether to use the mean or median for point estimates in the table.
`ci`	a scalar indicating the confidence level of the uncertainty intervals.
`hpdi`	a logical indicating whether to use highest posterior density intervals or equal tailed credible intervals to capture uncertainty.
`coefnames`	an optional vector or list of vectors containing parameter names for each model. If there are multiple models, the list must have the same number of elements as there are models, and the vector of names in each list element must match the number of parameters. If not supplied, the function will use the parameter names in the model object(s). Note that this replaces the standard `custom.coef.names` argument in `texreg` because there is no `extract` method for MCMC model objects, and many MCMC model objects do not have unique parameter names.
`gof`	a named list of goodness of fit statistics, or a list of such lists.
`gofnames`	an optional vector or list of vectors containing goodness of fit statistic names for each model. Like `coefnames` in this function (which replaces the `custom.coef.names` argument in `texreg`), `gofnames` replaces the standard `custom.gof.names` argument in `texreg`. If there are multiple models, the list must have the same number of elements as there are models, and the vector of names in each list element must match the number of goodness of fit statistics.
`format`	a character indicating `latex` or `html` output.
`file`	optional file name to write table to file instead of printing to console.
`...`	optional arguments to `texreg`.

Details

This function creates LaTeX or HTML regression tables for MCMC Output using the `texreg` function from the `texreg` R package

If using `custom.coef.map` with more than one model, you should rename the parameters in the model objects to ensure that different parameters with the same subscript are not conflated by `texreg` e.g. `beta[1]` could represent age in one model and income in another, and `texreg` would combine the two if you do not rename `beta[1]` to more informative names in the model objects.

If `mod` is a `brmsfit` object or list of `brmsfit` objects, note that the default `brms` names for coefficients are `b_Intercept` and `b`, so both of these should be included in `par` if you wish to include the intercept in the table.

Value

A formatted regression table in LaTeX or HTML format.

Author(s)

Rob Williams, <jayrobwilliams@gmail.com>

Examples

```
## simulating data
set.seed(123456)
b0 <- 0.2 # true value for the intercept
b1 <- 0.5 # true value for first beta
b2 <- 0.7 # true value for second beta
n <- 500 # sample size
X1 <- runif(n, -1, 1)
X2 <- runif(n, -1, 1)
Z <- b0 + b1 * X1 + b2 * X2
pr <- 1 / (1 + exp(-Z)) # inv logit function
Y <- rbinom(n, 1, pr)
data <- data.frame(cbind(X1, X2, Y))

## formatting the data for jags
datjags <- as.list(data)
datjags$N <- length(datjags$Y)

## creating jags model
model <- function()  {

  for(i in 1:N){
    Y[i] ~ dbern(p[i])  ## Bernoulli distribution of y_i
    logit(p[i]) <- mu[i]    ## Logit link function
    mu[i] <- b[1] +
      b[2] * X1[i] +
      b[3] * X2[i]
  }
```

```
  for(j in 1:3){
    b[j] ~ dnorm(0, 0.001) ## Use a coefficient vector for simplicity
  }

}

params <- c("b")
inits1 <- list("b" = rep(0, 3))
inits2 <- list("b" = rep(0, 3))
inits <- list(inits1, inits2)

## fitting the model with R2jags
set.seed(123)
fit <- R2jags::jags(data = datjags, inits = inits,
                    parameters.to.save = params, n.chains = 2, n.iter = 2000,
                    n.burnin = 1000, model.file = model)

## generating regression table with all parameters
mcmcReg(fit)

## generating regression table with only betas and custom coefficent names
mcmcReg(fit, pars = c('b'), coefnames = c('Variable 1', 'Variable 2',
                                          'Variable 3'))
## generating regression tables with all betas and custom names
mcmcReg(fit, coefnames = c('Variable 1', 'Variable 2',
                           'Variable 3', 'deviance'))
```

`mcmcRocPrc` *ROC and Precision-Recall Curves using Bayesian MCMC estimates*

Description

Generate ROC and Precision-Recall curves after fitting a Bayesian logit or probit regression.

Usage

```
mcmcRocPrc(object, yname, xnames, curves, fullsims)
```

Arguments

`object`	A "rjags" object (see `R2jags::jags()`) for a fitted binary choice model.
`yname`	(`character(1)`) The name of the dependent variable, should match the variable name in the JAGS data object.
`xnames`	(`base::character()`) A character vector of the independent variable names, should match the corresponding names in the JAGS data object.

`curves` logical indicator of whether or not to return values to plot the ROC or Precision-Recall curves. If set to `FALSE` (default), results are returned as a list without the extra values.

`fullsims` logical indicator of whether full object (based on all MCMC draws rather than their average) will be returned. Default is `FALSE`. Note: If `TRUE` is chosen, the function takes notably longer to execute.

Value

Returns a list; the specific structure depends on the combination of the "curves" and "fullsims" argument values.

References

Beger, Andreas. 2016. “Precision-Recall Curves.” Available at SSRN: http://dx.doi.org/10.2139/ssrn.2765419

Examples

```
# load simulated data and fitted model (see ?sim_data and ?jags_logit)
data("jags_logit")

# using mcmcRocPrc
fit_sum <- mcmcRocPrc(jags_logit,
                      yname = "Y",
                      xnames = c("X1", "X2"),
                      curves = TRUE,
                      fullsims = FALSE)
```

`mcmcTab` *Summarize Bayesian MCMC Output R function for summarizing MCMC output in a regression-style table.*

Description

Summarize Bayesian MCMC Output

R function for summarizing MCMC output in a regression-style table.

Usage

```
mcmcTab(sims, ci = c(0.025, 0.975), pars = NULL, Pr = FALSE,
  ROPE = NULL)
```

Arguments

sims	Bayesian model object generated by R2jags, rjags, R2WinBUGS, R2OpenBUGS, MCMCpack, rstan, and rstanarm.
ci	desired level for credible intervals; defaults to c(0.025, 0.975).
pars	character vector of parameters to be printed; defaults to `NULL` (all parameters are printed). If not `NULL`, the user can either specify the exact names of parameters to be printed (e.g. `c("alpha","beta1","beta2")`) or part of a name so that all parameters containing that name will be printed (e.g. `"beta"` will print `beta1`, `beta2`, etc.).
Pr	print percent of posterior draws with same sign as median; defaults to `FALSE`.
ROPE	defaults to `NULL`. If not `NULL`, a vector of two values defining the region of practical equivalence ("ROPE"); returns % of posterior draws to the left/right of ROPE. For this quantity to be meaningful, all parameters must be on the same scale (e.g. standardized coefficients or first differences). See Kruschke (2013, Journal of Experimental Psychology 143(2): 573-603) for more on the ROPE.

Value

a data frame containing MCMC summary statistics.

References

Kruschke, John K. 2013. "Bayesian Estimation Supersedes the T-Test." Journal of Experimental Psychology: General 142 (2): 573–603. https://doi.org/10.1037/a0029146.

Examples

```
data("jags_logit")

## printing out table
object <- mcmcTab(jags_logit,
          ci = c(0.025, 0.975),
          pars = NULL,
          Pr = FALSE,
          ROPE = NULL)
object
```

---

sim_data *Simulated data for examples*

Description

Simulated data to fit example models against

Usage

```
sim_data
```

Format

a data.frame

Examples

```
## simulating data
set.seed(123456)
b0 <- 0.2 # true value for the intercept
b1 <- 0.5 # true value for first beta
b2 <- 0.7 # true value for second beta
n <- 500 # sample size
X1 <- runif(n, -1, 1)
X2 <- runif(n, -1, 1)
Z <- b0 + b1 * X1 + b2 * X2
pr <- 1 / (1 + exp(-Z)) # inv logit function
Y <- rbinom(n, 1, pr)
sim_data <- data.frame(cbind(X1, X2, Y))
```

Index