Package ‘bayesdistreg’

October 12, 2022

Type Package

Title Bayesian Distribution Regression

Version 0.1.0

Maintainer Emmanuel Tsyawo <estsyawo@temple.edu>

Description Implements Bayesian Distribution Regression methods. This package contains functions for three estimators (non-asymptotic, semi-asymptotic and asymptotic) and related routines for Bayesian Distribution Regression in Huang and Tsyawo (2018) <doi:10.2139/ssrn.3048658> which is also the recommended reference to cite for this package. The functions can be grouped into three (3) categories. The first computes the logit likelihood function and posterior densities under uniform and normal priors. The second contains Independence and Random Walk Metropolis-Hastings Markov Chain Monte Carlo (MCMC) algorithms as functions and the third category of functions are useful for semi-asymptotic and asymptotic Bayesian distribution regression inference.

Depends R (>= 2.1.0)

License GPL-2

Encoding UTF-8

LazyData true

Imports MASS, sandwich, stats

RoxygenNote 6.1.1

NeedsCompilation no

Author Emmanuel Tsyawo [aut, cre],
Weige Huang [aut]

Repository CRAN

Date/Publication 2019-02-05 22:44:13 UTC

R topics documented:

`asymcnfB` *Asymmetric simultaneous bayesian confidence bands*

Description

`asymcnfB` obtains asymmetric bayesian distribution confidence bands

Usage

```
asymcnfB(DF, DFmat, alpha = 0.05, scale = FALSE)
```

Arguments

`DF`	the target distribution/quantile function as a vector
`DFmat`	the matrix of draws of the distribution, rows correspond to elements in `DF`
`alpha`	level such that `1-alpha` is the desired probability of coverage
`scale`	logical for scaling using the inter-quartile range

Value

cstar - a constant to add and subtract from DF to create confidence bands if no scaling=FALSE else a vector of length DF.

Examples

```
set.seed(14); m=matrix(rbeta(500,1,4),nrow = 5) + 1:5
DF = apply(m,1,mean); plot(1:5,DF,type="l",ylim = c(min(m),max(m)), xlab = "Index")
asyCB<- asymcnfB(DF,DFmat = m)
lines(1:5,DF-asyCB$cmin,lty=2); lines(1:5,DF+asyCB$cmax,lty=2)
```

---

distreg — *Bayesian distribution regression*

Description

`distreg` draws randomly from the density of F(yo) at a threshold value yo

Usage

```
distreg(thresh, data0, MH = "IndepMH", ...)
```

Arguments

`thresh`	threshold value that is used to binarise the continuous outcome variable
`data0`	original data set with the first column being the continuous outcome variable
`MH`	metropolis-hastings algorithm to use; default:"IndepMH", alternative "RWMH"
`...`	any additional inputs to pass to the MH algorithm

Value

fitob a vector of fitted values corresponding to the distribution at threshold thresh

Examples

```
data0=faithful[,c(2,1)]; qnt<-quantile(data0[,1],0.25)
distob<- distreg(qnt,data0,iter = 102, burn = 2);
plot(density(distob,.1),main="Kernel density plot")
```

distreg.asymp *Asymptotic distribution regression*

Description

`distreg.asymp` takes input object from dr_asympar() for asymptotic bayesian distribution.

Usage

```
distreg.asymp(ind, drabj, data, vcovfn = "vcov", ...)
```

Arguments

`ind`	index of object in list `drabj` (i.e. a threshold value) from which to take draws
`drabj`	object from dr_asympar()
`data`	dataframe, first column is the outcome
`vcovfn`	a string denoting the function to extract the variance-covariance. Defaults at "vcov". Other variance-covariance estimators in the sandwich package are usable.
`...`	additional input to pass to `vcovfn`

Value

a mean `Fhat` and a variance `varF`

Examples

```
y = faithful$waiting
x = scale(cbind(faithful$eruptions,faithful$eruptions^2))
qtaus = quantile(y,c(0.05,0.25,0.5,0.75,0.95))
drabj<- dr_asympar(y=y,x=x,thresh = qtaus); data = data.frame(y,x)
(asymp.obj<- distreg.asymp(ind=2,drabj,data,vcovfn="vcov"))
```

distreg.sas *Semi-asymptotic bayesian distribution*

Description

`distreg.sas` takes input object from dr_asympar() for semi asymptotic bayesian distribution. This involves taking random draws from the normal approximation of the posterior at each threshold value.

Usage

```
distreg.sas(ind, drabj, data, vcovfn = "vcov", iter = 100)
```

Arguments

`ind`	index of object in list `drabj` (i.e. a threshold value) from which to take draws
`drabj`	object from dr_asympar()
`data`	dataframe, first column is the outcome
`vcovfn`	a string denoting the function to extract the variance-covariance. Defaults at "vcov". Other variance-covariance estimators in the sandwich package are usable.
`iter`	number of draws to simulate

Value

fitob vector of random draws from density of F(yo) using semi-asymptotic BDR

Examples

```
y = faithful$waiting
x = scale(cbind(faithful$eruptions,faithful$eruptions^2))
qtaus = quantile(y,c(0.05,0.25,0.5,0.75,0.95))
drabj<- dr_asympar(y=y,x=x,thresh = qtaus); data = data.frame(y,x)
drsas1 = lapply(1:5,distreg.sas,drabj=drabj,data=data,iter=100)
drsas2 = lapply(1:5,distreg.sas,drabj=drabj,data=data,vcovfn="vcovHC",iter=100)
par(mfrow=c(3,2));invisible(lapply(1:5,function(i)plot(density(drsas1[[i]],.1))));par(mfrow=c(1,1))
par(mfrow=c(3,2));invisible(lapply(1:5,function(i)plot(density(drsas2[[i]],.1))));par(mfrow=c(1,1))
```

`distreg_cfa` *Counterfactual bayesian distribution regression*

Description

`distreg` draws randomly from the density of counterfactual of F(yo) at a threshold value yo

Usage

```
distreg_cfa(thresh, data0, MH = "IndepMH", cft, cfIND, ...)
```

Arguments

`thresh`	threshold value that is used to binarise the continuous outcome variable
`data0`	original data set with the first column being the continuous outcome variable
`MH`	metropolis-hastings algorithm to use; default:"IndepMH", alternative "RWMH"
`cft`	column vector of counterfactual treatment
`cfIND`	the column index(indices) of treatment variable(s) to replace with `cft` in `data0`
`...`	any additional inputs to pass to the MH algorithm

Value

robj a list of a vector of fitted values corresponding to random draws from F(yo), counterfactual F(yo), and the parameters

Examples

```
data0=faithful[,c(2,1)]; qnt<-quantile(data0[,1],0.25)
cfIND=2 #Note: the first column is the outcome variable.
cft=0.95*data0[,cfIND] # a decrease by 5%
dist_cfa<- distreg_cfa(qnt,data0,cft,cfIND,MH="IndepMH",iter = 102, burn = 2)
par(mfrow=c(1,2)); plot(density(dist_cfa$counterfactual,.1),main="Original")
plot(density(dist_cfa$counterfactual,.1),main="Counterfactual"); par(mfrow=c(1,1))
```

---

distreg_cfa.sas *Semi-asymptotic counterfactual distribution*

Description

distreg_cfa.sas takes input object from dr_asympar() for counterfactual semi asymptotic bayesian distribution. This involves taking random draws from the normal approximation of the posterior at each threshold value.

Usage

```
distreg_cfa.sas(ind, drabj, data, cft, cfIND, vcovfn = "vcov",
  iter = 100)
```

Arguments

ind	index of object in list drabj (i.e. a threshold value) from which to take draws
drabj	object from dr_asympar()
data	dataframe, first column is the outcome
cft	column vector of counterfactual treatment
cfIND	the column index(indices) of treatment variable(s) to replace with cft in data0
vcovfn	a string denoting the function to extract the variance-covariance. Defaults at "vcov". Other variance-covariance estimators in the sandwich package are usable.
iter	number of draws to simulate

Value

fitob vector of random draws from density of F(yo) using semi-asymptotic BDR

Examples

```
y = faithful$waiting
x = scale(cbind(faithful$eruptions,faithful$eruptions^2))
qtaus = quantile(y,c(0.05,0.25,0.5,0.75,0.95))
drabj<- dr_asympar(y=y,x=x,thresh = qtaus); data = data.frame(y,x)
cfIND=2 #Note: the first column is the outcome variable.
cft=0.95*data[,cfIND] # a decrease by 5%
cfa.sasobj<- distreg_cfa.sas(ind=2,drabj,data,cft,cfIND,vcovfn="vcov")
par(mfrow=c(1,2)); plot(density(cfa.sasobj$original,.1),main="Original")
plot(density(cfa.sasobj$counterfactual,.1),main="Counterfactual"); par(mfrow=c(1,1))
```

---

dr_asympar — *Binary glm object at several threshold values*

Description

`dr_asympar` computes a normal approximation of the likelihood at a vector of threshold values

Usage

```
dr_asympar(y, x, thresh, ...)
```

Arguments

`y`	outcome variable
`x`	matrix of covariates
`thresh`	vector of threshold values on the support of outcome y
`...`	additional arguments to pass to `lapl_aprx2`

Value

a list of glm objects corresponding to `thresh`

Examples

```
y = faithful$waiting
x = scale(cbind(faithful$eruptions,faithful$eruptions^2))
qtaus = quantile(y,c(0.05,0.25,0.5,0.75,0.95))
drabj<- dr_asympar(y=y,x=x,thresh = qtaus)
lapply(drabj,coef); lapply(drabj,vcov)
# mean and covariance at respective threshold values
```

`fitdist` *The distribution of mean fitted logit probabilities*

Description

`fitdist` function generates a vector of mean fitted probabilities that constitute the distribution. This involves marginalising out covariates.

Usage

```
fitdist(Matparam, data)
```

Arguments

`Matparam`	an M x k matrix of parameter draws, each being a 1 x k vector
`data`	dataframe used to obtain Matparam

Value

dist fitted (marginalised) distribution

`fitlogit` *Fitted logit probabilities*

Description

`fitlogit` obtains a vector of fitted logit probabilities given parameters (pars) and data

Usage

```
fitlogit(pars, data)
```

Arguments

`pars`	vector of parameters
`data`	data frame. The first column of the data frame ought to be the binary dependent variable

Value

vec vector of fitted logit probabilities

IndepMH — *Independence Metropolis-Hastings Algorithm*

Description

IndepMH computes random draws of parameters using a specified proposal distribution.

Usage

```
IndepMH(data, propob = NULL, posterior = NULL, iter = 1500,
  burn = 500, vscale = 1.5, start = NULL, prior = "Uniform",
  mu = 0, sig = 10)
```

Arguments

data	data required for the posterior distribution
propob	a list of mean and variance-covariance of the normal proposal distribution (default:NULL)
posterior	the posterior distribution. It is set to null in order to use the logit posterior. The user can specify log posterior as a function of parameters and data (pars,data)
iter	number of random draws desired (default: 1500)
burn	burn-in period for the MH algorithm (default: 500)
vscale	a positive value to scale up or down the variance-covariance matrix in the proposal distribution
start	starting values of parameters for the MH algorithm. It is automatically generated but the user can also specify.
prior	the prior distribution (default: "Normal", alternative: "Uniform")
mu	the mean of the normal prior distribution (default:0)
sig	the variance of the normal prior distribution (default:10)

Value

val a list of matrix of draws pardraws and the acceptance rate

Examples

```
y = indicat(faithful$waiting,70)
x = scale(cbind(faithful$eruptions,faithful$eruptions^2))
data = data.frame(y,x); propob<- lapl_aprx(y,x)
IndepMH_n<- IndepMH(data=data,propob,iter = 102, burn = 2) # prior="Normal"
IndepMH_u<- IndepMH(data=data,propob,prior="Uniform",iter = 102, burn = 2) # prior="Uniform"
par(mfrow=c(3,1));invisible(apply(IndepMH_n$Matpram,2,function(x)plot(density(x))))
invisible(apply(IndepMH_u$Matpram,2,function(x)plot(density(x))));par(mfrow=c(1,1))
```

indicat *Indicator function*

Description

This function creates 0-1 indicators for a given threshold y0 and vector y

Usage

```
indicat(y, y0)
```

Arguments

y	vector y
y0	threshold value y0

Value

val

jdpar.asymp *Joint asymptotic mutivariate density of parameters*

Description

`jdpar.asymp` takes input object from dr_asympar() for asymptotic bayesian distribution. It returns objects for joint mutivariate density of parameters across several thresholds. Check for positive definiteness of the covariance matrix, else exclude thresholds yielding negative eigen values.

Usage

```
jdpar.asymp(drabj, data, jdF = FALSE, vcovfn = "vcovHC", ...)
```

Arguments

drabj	object from dr_asympar()
data	dataframe, first column is the outcome
jdF	logical to return joint density of F(yo) across thresholds in drabj
vcovfn	a string denoting the function to extract the variance-covariance. Defaults at "vcov". Other variance-covariance estimators in the sandwich package are usable.
...	additional input to pass to `vcovfn`

Value

mean vector Theta and variance-covariance matrix vcovpar of parameters across thresholds and if `jdF=TRUE`, a mean vector `mnF` and a variance-covariance matrix `vcovF` of F(yo)

Examples

```
y = faithful$waiting
x = scale(cbind(faithful$eruptions,faithful$eruptions^2))
qtaus = quantile(y,c(0.05,0.25,0.5,0.75,0.95))
drabj<- dr_asympar(y=y,x=x,thresh = qtaus); data = data.frame(y,x)
(drjasy = jdpar.asymp(drabj=drabj,data=data,jdF=TRUE))
```

---

jntCBOM — *Montiel Olea and Plagborg-Moller (2018) confidence bands*

Description

`jntCBOM` implements calibrated symmetric confidence bands (algorithm 2) in Montiel Olea and Plagborg-Moller (2018).

Usage

```
jntCBOM(DF, DFmat, alpha = 0.05, eps = 0.001)
```

Arguments

`DF`	the target distribution/quantile function as a vector
`DFmat`	the matrix of draws of the distribution, rows correspond to indices elements in `DF`
`alpha`	level such that `1-alpha` is the desired probability of coverage
`eps`	steps by which the grid on 1-alpha:alpha/2 is searched.

Value

CB - confidence band, zeta - the optimal level

Examples

```
set.seed(14); m=matrix(rbeta(500,1,4),nrow = 5) + 1:5
DF = apply(m,1,mean); plot(1:5,DF,type="l",ylim = c(min(m),max(m)), xlab = "Index")
jOMCB<- jntCBOM(DF,DFmat = m)
lines(1:5,jOMCB$CB[,1],lty=2); lines(1:5,jOMCB$CB[,2],lty=2)
```

lapl_aprx *Laplace approximation of posterior to normal*

Description

This function generates mode and variance-covariance for a normal proposal distribution for the bayesian logit.

Usage

```
lapl_aprx(y, x, glmobj = FALSE)
```

Arguments

`y`	the binary dependent variable y
`x`	the matrix of independent variables.
`glmobj`	logical for returning the logit glm object

Value

val A list of mode variance-covariance matrix, and scale factor for proposal draws from the multivariate normal distribution.

Examples

```
y = indicat(faithful$waiting,mean(faithful$waiting))
x = scale(cbind(faithful$eruptions,faithful$eruptions^2))
gg<- lapl_aprx(y,x)
```

lapl_aprx2 *Laplace approximation of posterior to normal*

Description

`lapl_aprx2` is a more flexible alternative to `lapl_aprx`. This creates `glm` objects from which joint asymptotic distributions can be computed.

Usage

```
lapl_aprx2(y, x, family = "binomial", ...)
```

Arguments

`y`	the binary dependent variable y
`x`	the matrix of independent variables.
`family`	a parameter to be passed `glm()`, defaults to the logit model
`...`	additional parameters to be passed to `glm()`

Value

val A list of mode variance-covariance matrix, and scale factor for proposal draws from the multivariate normal distribution.

Examples

```
y = indicat(faithful$waiting,mean(faithful$waiting))
x = scale(cbind(faithful$eruptions,faithful$eruptions^2))
(gg<- lapl_aprx2(y,x)); coef(gg); vcov(gg)
```

`logit` *Logit likelihood function*

Description

`logit` is the logistic likelihood function given data.

Usage

```
logit(start, data, Log = TRUE)
```

Arguments

`start`	vector of starting values
`data`	dataframe. The first column should be the dependent variable.
`Log`	a logical input (defaults to `True`) to take the log of the likelihood.

Value

like returns the likelihood function value.

Examples

```
y = indicat(faithful$waiting,mean(faithful$waiting))
x = scale(cbind(faithful$eruptions,faithful$eruptions^2))
data = data.frame(y,x)
logit(rep(0,3),data)
```

`LogitLink` *Logit function*

Description

This is the link function for logit regression

Usage

```
LogitLink(x)
```

Arguments

`x`	Random variable

Value

val Probability value from the logistic function

`parLply` *Parallel compute*

Description

`parLply` uses `parlapply` from the `parallel` package with a function as input

Usage

```
parLply(vec, fn, type = "FORK", no_cores = 1, ...)
```

Arguments

`vec`	vector of inputs over which to parallel compute
`fn`	the function
`type`	this option is set to "FORK", use "PSOCK" on windows
`no_cores`	the number of cores to use. Defaults at 1
`...`	extra inputs to `fn()`

Value

out parallel computed output

par_distreg *Parallel compute bayesian distribution regression*

Description

par_distreg uses parallel computation to compute bayesian distribution regression for a given vector of threshold values and a data (with first column being the continuous outcome variable)

Usage

```
par_distreg(thresh, data0, fn = distreg, no_cores = 1,
  type = "PSOCK", ...)
```

Arguments

thresh	vector of threshold values.
data0	the original data set with a continous dependent variable in the first column
fn	bayesian distribution regression function. the default is distreg provided in the package
no_cores	number of cores for parallel computation
type	type passed to makeCluster() in the package parallel
...	any additional input parameters to pass to fn

Value

mat a G x M matrix of output (G is the length of thresh, M is the number of draws)

Examples

```
data0=faithful[,c(2,1)]; qnts<-quantile(data0[,1],c(0.05,0.25,0.5,0.75,0.95))
out<- par_distreg(qnts,data0,no_cores=1,iter = 102, burn = 2)
par(mfrow=c(3,2));invisible(apply(out,1,function(x)plot(density(x,30))));par(mfrow=c(1,1))
```

posterior *Posterior distribution*

Description

posterior computes the value of the posterior at parameter values pars

Usage

```
posterior(pars, data, Log = TRUE, mu = 0, sig = 25,
  prior = "Normal")
```

Arguments

pars	parameter values
data	dataframe. The first column must be the binary dependent variable
Log	logical to take the log of the posterior.(defaults to TRUE)
mu	mean of prior of each parameter value in case the prior is Normal (default: 0)
sig	standard deviation of prior of each parameter in case the prior is Normal (default: 25)
prior	string input of "Normal" or "Uniform" prior distribution to use

Value

val value function of the posterior

Examples

```
y = indicat(faithful$waiting,mean(faithful$waiting))
x = scale(cbind(faithful$eruptions,faithful$eruptions^2))
data = data.frame(y,x)
posterior(rep(0,3),data,Log = FALSE,mu=0,sig = 10,prior = "Normal") # no log
posterior(rep(0,3),data,Log = TRUE,mu=0,sig = 10,prior = "Normal") # log
posterior(rep(0,3),data,Log = TRUE) # use default values
```

---

prior_n *Normal Prior distribution*

Description

This normal prior distribution is a product of univariate N(mu,sig)

Usage

```
prior_n(pars, mu, sig, Log = FALSE)
```

Arguments

pars	parameter values
mu	mean value of each parameter value
sig	standard deviation of each parameter value
Log	logical to take the log of prior or not (defaults to FALSE)

Value

val Product of probability values for each parameter

Examples

```
prior_n(rep(0,6),0,10,Log = TRUE) #log of prior
prior_n(rep(0,6),0,10,Log = FALSE) #no log
```

`prior_u` *Uniform Prior distribution*

Description

This uniform prior distribution proportional to 1

Usage

```
prior_u(pars)
```

Arguments

`pars`	parameter values

Value

val value of joint prior =1 for the uniform prior

`quant_bdr` *Quantile conversion of a bayesian distribution matrix*

Description

`quant_bdr` converts a bayesian distribution regression matrix from `par_distreg()` output to a matrix of quantile distribution.

Usage

```
quant_bdr(taus, thresh, mat)
```

Arguments

`taus`	a vector of quantile indices
`thresh`	a vector of threshold values used in a `par_distreg()` type function
`mat`	bayesian distribution regression output matrix

Value

qmat matrix of quantile distribution

RWMH *Random Walk Metropolis-Hastings Algorithm*

Description

RWMH computes random draws of parameters using a specified proposal distribution. The default is the normal distribution

Usage

```
RWMH(data, propob = NULL, posterior = NULL, iter = 1500,
  burn = 500, vscale = 1.5, start = NULL, prior = "Normal",
  mu = 0, sig = 10)
```

Arguments

data	data required for the posterior distribution. First column is the outcome
propob	a list of mean and variance-covariance of the normal proposal distribution (default: NULL i.e. internally generated)
posterior	the posterior distribution. It is set to null in order to use the logit posterior. The user can specify log posterior as a function of parameters and data (pars,data)
iter	number of random draws desired
burn	burn-in period for the Random Walk MH algorithm
vscale	a positive value to scale up or down the variance-covariance matrix in the proposal distribution
start	starting values of parameters for the MH algorithm. It is automatically generated from the proposal distribution but the user can also specify.
prior	the prior distribution (default: "Normal", alternative: "Uniform")
mu	the mean of the normal prior distribution (default:0)
sig	the variance of the normal prior distribution (default:10)

Value

val a list of matrix of draws Matpram and the acceptance rate

Examples

```
y = indicat(faithful$waiting,70)
x = scale(cbind(faithful$eruptions,faithful$eruptions^2))
data = data.frame(y,x); propob<- lapl_aprx(y,x)
RWMHob_n<- RWMH(data=data,propob,iter = 102, burn = 2) # prior="Normal"
RWMHob_u<- RWMH(data=data,propob,prior="Uniform",iter = 102, burn = 2)
par(mfrow=c(3,1));invisible(apply(RWMHob_n$Matpram,2,function(x)plot(density(x))))
invisible(apply(RWMHob_u$Matpram,2,function(x)plot(density(x))));par(mfrow=c(1,1))
```

simcnfB *Symmetric simultaneous bayesian confidence bands*

Description

`simcnfB` obtains symmetric bayesian distribution confidence bands

Usage

```
simcnfB(DF, DFmat, alpha = 0.05, scale = FALSE)
```

Arguments

`DF`	the target distribution/quantile function as a vector
`DFmat`	the matrix of draws of the distribution, rows correspond to elements in `DF`
`alpha`	level such that `1-alpha` is the desired probability of coverage
`scale`	logical for scaling using the inter-quartile range

Value

cstar - a constant to add and subtract from DF to create confidence bands if no scaling=FALSE else a vector of length DF.

Examples

```
set.seed(14); m=matrix(rbeta(500,1,4),nrow = 5) + 1:5
DF = apply(m,1,mean); plot(1:5,DF,type="l",ylim = c(0,max(m)), xlab = "Index")
symCB<- simcnfB(DF,DFmat = m)
lines(1:5,DF-symCB,lty=2); lines(1:5,DF+symCB,lty=2)
```

Index