

Package: surbayes (via r-universe)

November 5, 2024

Type Package

Title Bayesian Analysis of Seemingly Unrelated Regression Models

Version 0.1.2

Date 2020-08-24

Author Ethan Alt

Maintainer Ethan Alt <ethanalt@live.unc.edu>

Description Implementation of the direct Monte Carlo approach of Zellner and Ando (2010) <[doi:10.1016/j.jeconom.2010.04.005](https://doi.org/10.1016/j.jeconom.2010.04.005)> to sample from posterior of Seemingly Unrelated Regression (SUR) models. In addition, a Gibbs sampler is implemented that allows the user to analyze SUR models using the power prior.

License GPL (>= 2)

Imports Rcpp (>= 1.0.4.6), Matrix, rlist

LinkingTo Rcpp, RcppArmadillo

Encoding UTF-8

RoxygenNote 7.1.0

Collate 'RcppExports.R' 'predict.surbayes.R' 'sur_sample_powerprior.R' 'sur_sample_dmc.R' 'sur_sample.R' 'surbayes-package.R'

URL <https://github.com/ethan-alt/surbayes>

BugReports <https://github.com/ethan-alt/surbayes/issues>

Config/pak/sysreqs libxml2-dev

Repository <https://ethan-alt.r-universe.dev>

RemoteUrl <https://github.com/ethan-alt/surbayes>

RemoteRef HEAD

RemoteSha 293a6ad2833fd0f44baa7adb4dc01103909612e3

Contents

predict.surbayes	2
predict_surbayes_cpp	3
predict_surbayes_helper	4
sample_sigma	4
sur_sample	5
sur_sample_cov_helper_cpp	6
sur_sample_cpp	7
sur_sample_dmc	7
sur_sample_gibbs_cpp	8
sur_sample_powerprior	9

Index	12
--------------	-----------

predict.surbayes	<i>Get predictive posterior samples</i>
------------------	---

Description

This function returns a list of new data sets by sampling from the posterior predictive density of $Y | Y_0, X_{new}$.

Usage

```
## S3 method for class 'surbayes'
predict(object, newdata, nsims = 1, ...)
```

Arguments

object	Result from calling sur_sample
newdata	data.frame of new data
nsims	number of posterior draws to take. The default and minimum is 1. The maximum is the number of simulations in surbayes
...	further arguments passed to or from other methods

Value

$n \times J \times nsims$ array of predicted values

Examples

```
## Taken from bayesm package
if(nchar(Sys.getenv("LONG_TEST")) != 0) {M=1000} else {M=10}
set.seed(66)
## simulate data from SUR
beta1 = c(1,2)
beta2 = c(1,-1,-2)
```

```

nobs = 100
nreg = 2
iota = c(rep(1, nobs))
X1 = cbind(iota, runif(nobs))
X2 = cbind(iota, runif(nobs), runif(nobs))
Sigma = matrix(c(0.5, 0.2, 0.2, 0.5), ncol = 2)
U = chol(Sigma)
E = matrix( rnorm( 2 * nobs ), ncol = 2) %*% U
y1 = X1 %*% beta1 + E[,1]
y2 = X2 %*% beta2 + E[,2]
X1 = X1[, -1]
X2 = X2[, -1]
data = data.frame(y1, y2, X1, X2)
names(data) = c( paste0( 'y', 1:2 ), paste0('x', 1:(ncol(data) - 2) ))
## run DMC sampler
formula.list = list(y1 ~ x1, y2 ~ x2 + x3)

## Fit model
out = sur_sample( formula.list, data, M = M )

## Obtain predictions
pred = predict(out, data, nsims = 1)

```

predict_surbayes_cpp *Sample from predictive posterior density C++ helper*

Description

C++ implementation to obtain a matrix of samples from predictive posterior density

Usage

```
predict_surbayes_cpp(Mu, Sigmalist, n, J, nsims)
```

Arguments

Mu	matrix of means
Sigmalist	list of covariance matrices
n	number of observations
J	number of endpoints
nsims	Number of simulations (number of rows in Mu)

`predict_surbayes_helper`*Get one sample from predictive posterior of SUR*

Description

C++ implementation to obtain one sample from predictive posterior density

Usage

```
predict_surbayes_helper(mu, Sigma, n, J)
```

Arguments

<code>mu</code>	vector of means
<code>Sigma</code>	covariance matrix shared among all observations
<code>n</code>	number of observations
<code>J</code>	number of endpoints

`sample_sigma`*Sample Sigma via Gibbs for SUR model*

Description

This is a c++ implementation of sampling Sigma via Gibbs in SUR model–inverse Wishart

Usage

```
sample_sigma(nu, V, p)
```

Arguments

<code>nu</code>	degrees of freedom
<code>V</code>	scale matrix
<code>p</code>	dimension of covariance matrix

Value

sampld covariance matrix

sur_sample	<i>Sample from seemingly unrelated regression</i>
------------	---

Description

This function is a wrapper function that performs either (1) Direct Monte Carlo or (2) Gibbs sampling of the SUR model depending on whether 1 or 2 data sets are specified.

Usage

```
sur_sample(
  formula.list,
  data,
  M,
  histdata = NULL,
  Sigma0 = NULL,
  a0 = 1,
  burnin = 0,
  thin = 1
)
```

Arguments

formula.list	A list of formulas, each element giving the formula for the corresponding endpoint.
data	A data.frame containing all the variables contained in formula.list]
M	Number of samples to be drawn
histdata	A data.frame of historical data to apply power prior on
Sigma0	<i>optional</i> a $J \times J$ matrix giving the initial covariance matrix. Default is the MLE. Ignored if histdata is null
a0	A scalar between 0 and 1 giving the power prior parameter. Ignored if histdata is null
burnin	A non-negative integer giving the burn-in parameter. Ignored if histdata is null as direct Monte Carlo is performed
thin	A positive integer giving the thin parameter. Ignored if histdata is null as direct Monte Carlo is performed

Value

A list. First element is posterior draws. Second element is list of JxJ covariance matrices.

Examples

```

## Taken from bayesm package
if(nchar(Sys.getenv("LONG_TEST")) != 0) {M=1000} else {M=10}
set.seed(66)
## simulate data from SUR
beta1 = c(1,2)
beta2 = c(1,-1,-2)
nobs = 100
nreg = 2
iota = c(rep(1, nobs))
X1 = cbind(iota, runif(nobs))
X2 = cbind(iota, runif(nobs), runif(nobs))
Sigma = matrix(c(0.5, 0.2, 0.2, 0.5), ncol = 2)
U = chol(Sigma)
E = matrix( rnorm( 2 * nobs ), ncol = 2) %%% U
y1 = X1 %%% beta1 + E[,1]
y2 = X2 %%% beta2 + E[,2]
X1 = X1[, -1]
X2 = X2[, -1]
data = data.frame(y1, y2, X1, X2)
names(data) = c( paste0( 'y', 1:2 ), paste0('x', 1:(ncol(data) - 2) ))
## run DMC sampler
formula.list = list(y1 ~ x1, y2 ~ x2 + x3)

## Fit models
out_dmc = sur_sample( formula.list, data, M = M )           ## DMC used
out_powerprior = sur_sample( formula.list, data, M, data ) ## Gibbs used

```

sur_sample_cov_helper_cpp

Helper function to sample covariance

Description

This function is called by `sur_sample_cov_cpp`. It samples the covariance matrix of a SUR

Usage

```
sur_sample_cov_helper_cpp(Y, Xlist, n, J, pj, sigma11, r1)
```

Arguments

Y	A matrix, each column a vector of responses
Xlist	A list, each element a design matrix
n	Integer giving number of observations
J	Integer giving number of endpoints
pj	A vector giving number of covariates per endpoint

sigma11	A scalar giving a draw for the (1,1) component of the covariance matrix
r1	A vector of residuals for the first endpoint's regression

sur_sample_cpp	<i>Sample from SUR via Direct Monte Carlo (C++ version)</i>
----------------	---

Description

C++ implementation of Zellner and Ando (2010) Direct Monte Carlo method for sampling from the posterior of a Bayesian SUR

Usage

```
sur_sample_cpp(Y, Xlist, y, X, XtX, pj, M)
```

Arguments

Y	matrix (y_1, \dots, y_J)
Xlist	A list, each element a design matrix
y	vector of responses
X	design matrix
XtX	matrix giving crossprod(cbind(X1, ..., XJ))
pj	vector giving number of covariates per endpoint
M	An integer giving the number of desired samples

sur_sample_dmc	<i>Sample SUR model via direct Monte Carlo</i>
----------------	--

Description

This function samples from the posterior of a SUR model using the DMC method of Ando and Zellner (2010)

Usage

```
sur_sample_dmc(formula.list, data, M = 1000)
```

Arguments

formula.list	A list of formulas, each element giving the formula for the corresponding endpoint.
data	A data.frame containing all the variables contained in formula.list]
M	Number of samples to be drawn

Value

A list. First element is posterior draws. Second element is list of JxJ covariance matrices. Other elements are helpful statistics about the SUR model to pass to other functions.

Examples

```
## Taken from bayesm package
if(nchar(Sys.getenv("LONG_TEST")) != 0) {M=1000} else {M=10}
set.seed(66)
## simulate data from SUR
beta1 = c(1,2)
beta2 = c(1,-1,-2)
nobs = 100
nreg = 2
iota = c(rep(1, nobs))
X1 = cbind(iota, runif(nobs))
X2 = cbind(iota, runif(nobs), runif(nobs))
Sigma = matrix(c(0.5, 0.2, 0.2, 0.5), ncol = 2)
U = chol(Sigma)
E = matrix( rnorm( 2 * nobs ), ncol = 2) %*% U
y1 = X1 %*% beta1 + E[,1]
y2 = X2 %*% beta2 + E[,2]
X1 = X1[, -1]
X2 = X2[, -1]
data = data.frame(y1, y2, X1, X2)
names(data) = c( paste0( 'y', 1:2 ), paste0('x', 1:(ncol(data) - 2) ))
## run DMC sampler
formula.list = list(y1 ~ x1, y2 ~ x2 + x3)

## fit using historical data as current data set--never done in practice
out = sur_sample_powerprior( formula.list, data, histdata = data, M = M )
```

sur_sample_gibbs_cpp *Power Prior Gibbs sampling*

Description

This is a c++ implementation of Gibbs sampling SUR model with power prior

Usage

```
sur_sample_gibbs_cpp(
  Sigma,
  M,
  X,
  X0,
  XtX,
  X0tX0,
```



```

    Y,
    Y0,
    y,
    y0,
    a0,
    pvec,
    burnin,
    thin
  )

```

Arguments

Sigma	initial value for covariance matrix
M	number of samples
X	design matrix for current data
X0	design matrix for historical data
XtX	matrix that is <code>crossprod(cbind(X1, ..., XJ))</code>
X0tX0	matrix that is <code>crossprod(cbind(X01, ..., X0J))</code>
Y	future response as matrix (Y1, ..., YJ)
Y0	historical response as matrix (Y01, ..., Y0J)
y	future response as vector
y0	historical response as vector
a0	power prior parameter
pvec	vector giving number of covariates per endpoint
burnin	Burn-in parameter
thin	Thin parameter

Value

sampld covariance matrix

sur_sample_powerprior *Sample from SUR posterior via power prior*

Description

This function uses Gibbs sampling to sample from the posterior density of a SUR model using the power prior.

Usage

```
sur_sample_powerprior(
  formula.list,
  data,
  histdata,
  M,
  Sigma0 = NULL,
  a0 = 1,
  burnin = 0,
  thin = 1
)
```

Arguments

formula.list	A list of formulas, each element giving the formula for the corresponding end-point.
data	A data.frame containing all the variables contained in formula.list]
histdata	A data.frame of historical data to apply power prior on
M	Number of samples to be drawn
Sigma0	A $J \times J$ matrix giving the initial covariance matrix. Default is the MLE.
a0	A scalar between 0 and 1 giving the power prior parameter
burnin	A non-negative integer giving the burn-in parameter
thin	A positive integer giving the thin parameter

Value

A list. First element is posterior draws. Second element is list of JxJ covariance matrices.

Examples

```
## Taken from bayesm package
if(nchar(Sys.getenv("LONG_TEST")) != 0) {M=1000} else {M=10}
set.seed(66)
## simulate data from SUR
beta1 = c(1,2)
beta2 = c(1,-1,-2)
nobs = 100
nreg = 2
iota = c(rep(1, nobs))
X1 = cbind(iota, runif(nobs))
X2 = cbind(iota, runif(nobs), runif(nobs))
Sigma = matrix(c(0.5, 0.2, 0.2, 0.5), ncol = 2)
U = chol(Sigma)
E = matrix( rnorm( 2 * nobs ), ncol = 2) %%% U
y1 = X1 %%% beta1 + E[,1]
y2 = X2 %%% beta2 + E[,2]
X1 = X1[, -1]
X2 = X2[, -1]
```

```
data = data.frame(y1, y2, X1, X2)
names(data) = c( paste0( 'y', 1:2 ), paste0('x', 1:(ncol(data) - 2) ))
## run DMC sampler
formula.list = list(y1 ~ x1, y2 ~ x2 + x3)

## fit using historical data as current data set--never done in practice
out = sur_sample_powerprior( formula.list, data, histdata = data, M = M )
```

Index

`predict.surbayes`, 2
`predict_surbayes_cpp`, 3
`predict_surbayes_helper`, 4

`sample_sigma`, 4
`sur_sample`, 5
`sur_sample_cov_helper_cpp`, 6
`sur_sample_cpp`, 7
`sur_sample_dmc`, 7
`sur_sample_gibbs_cpp`, 8
`sur_sample_powerprior`, 9