Package ‘GHS’

October 12, 2022

Title Graphical Horseshoe MCMC Sampler Using Data Augmented Block Gibbs Sampler

Version 0.1

Author Ashutosh Srivastava<srivas48@purdue.edu>, Anindya Bhadra<bhadra@purdue.edu>

Maintainer Ashutosh Srivastava<srivas48@purdue.edu>

Description Draw posterior samples to estimate the precision matrix for multivariate Gaussian data. Posterior means of the samples is the graphical horseshoe estimate by Li, Bhadra and Craig(2017) <arXiv:1707.06661>. The function uses matrix decomposition and variable change from the Bayesian graphical lasso by Wang(2012) <doi:10.1214/12-BA729>, and the variable augmentation for sampling under the horseshoe prior by Makalic and Schmidt(2016) <arXiv:1508.03884>. Structure of the graphical horseshoe function was inspired by the Bayesian graphical lasso function using blocked sampling, authored by Wang(2012) <doi:10.1214/12-BA729>.

Depends R (>= 3.4.0), stats, MASS

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1.9000

NeedsCompilation no

Repository CRAN

Date/Publication 2018-10-30 18:00:07 UTC

R topics documented:

GHS_est *GHS MCMC sampler using data-augmented block (column-wise) Gibbs sampler*

Description

`GHS_est` returns a tuple whose first element is a p by p by nmc matrices of saved posterior samples of precision matrix, second element is the p*(p-1)/2 by nmc vector of saved samples of the local tuning parameter and the third element is the 1 by nmc vector of saved samples of the global tuning parameter

Usage

```
GHS_est(S, n, burnin, nmc)
```

Arguments

S	sample covariance matrix
n	sample size
burnin	number of MCMC burnins
nmc	number of saved samples

Examples

```
# This function generates positive definite matrices for testing purposes
# with specificied eigenvalues
Posdef <- function (n,ev)
{
  Z <- matrix(ncol=n, rnorm(n^2))
  decomp <- qr(Z)
  Q <- qr.Q(decomp)
  R <- qr.R(decomp)
  d <- diag(R)
  ph <- d / abs(d)
  O <- Q %*% diag(ph)
  Z <- t(O) %*% diag(ev) %*% O
  return(Z)
}
eig1 <- rep(1,2)
eig2 <- rep(0.75,3)
#eig3 <- rep(0.25,3)
eig_val <- c(eig1,eig2)
z <- Posdef(5,eig_val)
Mu <- rep(0,5)
Sigma <- solve(z)
Y <- mvrnorm(n=5,mu=Mu,Sigma=Sigma)
S <- t(Y)%*%Y
out <- GHS_est(S,50,100,5000)
```

```
est_matrix <- apply(out[[1]],c(1,2),mean)
image(est_matrix)
```

Index