

# Package ‘s525’

January 20, 2018

**Type** Package

**Title** Implementation of common Bayesian models

**Version** 1.0

**Date** 2017-11-26

**Author** Daniel Bourgeois

**Maintainer** Daniel Bourgeois <dcb10@rice.edu>

**Description** This package contains implementations of common Bayesian models.

**Imports** truncnorm, MASS, coda

**License** MIT License

## R topics documented:

s525-package . . . . .	2
factor_analysis . . . . .	2
factor_analysis_with_regression . . . . .	3
factor_infinite . . . . .	4
horseshoe_regression . . . . .	6
mat_apply . . . . .	7
metropolis . . . . .	8
plackett_luce . . . . .	9
probit_horseshoe_regression . . . . .	10
probit_regression . . . . .	11
ridge_regression . . . . .	13
rnorm_qinv_l . . . . .	14
rplackett_luce . . . . .	15
rpost_regression_coef . . . . .	15
shrinkage_regression_plot . . . . .	16
<b>Index</b>	<b>18</b>

---

s525-package

*Implementation of common Bayesian models*


---

### Description

This package contains implementations of common Bayesian models.

### Details

The DESCRIPTION file: This package was not yet installed at build time.

Index: This package was not yet installed at build time.

### Author(s)

Daniel Bourgeois

Maintainer: Daniel Bourgeois <dcb10@rice.edu>

### Examples

```
# simple examples of the most important functions
```

---

factor\_analysis

*Runs gibbs sampler for a factor model.*


---

### Description

The model is as follows:

$$y_i = \Lambda \eta_i + \epsilon_i$$

$$\epsilon_i \sim N_p(0, \Sigma)$$

where  $\Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_p^2)$ .

See Joyee Ghosh & David B. Dunson (2009) Default Prior Distributions and Efficient Posterior Computation in Bayesian Factor Analysis, Journal of Computational and Graphical Statistics, 18:2, 306-320, DOI: 10.1198/jcgs.2009.07145 for conditional posteriors.

**Usage**

```
factor_analysis <- function(
  Y,
  k,
  niter = 1000,
  shape_psi = 1/2,
  rate_psi = 1/2,
  shape_sigma2 = 1,
  rate_sigma2 = 0.2,
  nonzero_structure = NULL)
```

**Arguments**

Y	n by p matrix
k	number of factors
niter	number of iterations for the gibbs sampler to run.
shape_psi	shape parameter for psi. Can be a scalar or a k vector
rate_psi	rate parameter for psi. Can be a k vector
shape_sigma2	shape parameter for sigma2. Can be a p vector
rate_sigma2	rate parameter for sigma2. Can be a p vector
nonzero_structure	A boolean p x k matrix. If the i, jth spot is TRUE, then $\lambda_{ij}$ is free. If the i, jth spot is FALSE, then $\lambda_{ij}$ is zero. If not set, then a lower triangular matrix is used.

**Value**

sigma	An niter x p matrix of posterior values
eta	An niter x n x k array of posterior values
lambda	An niter x p x k array of posterior values

---

factor\_analysis\_with\_regression

*Runs gibbs sampler for a factor model with regression on the factors.*

---

**Description**

The model is as follows:

$$\begin{aligned}
 y_i &= \alpha + \Lambda \eta_i + \epsilon_i, \\
 \eta_i &= B x_i + \delta_i \\
 \epsilon_i &\sim N_p(0, \tau^{-1}) \\
 \delta_i &\sim N_k(0, I)
 \end{aligned}$$

where  $\tau = \text{diag}(\tau_1, \dots, \tau_p)$ .

See Joyee Ghosh & David B. Dunson (2009) Default Prior Distributions and Efficient Posterior Computation in Bayesian Factor Analysis, Journal of Computational and Graphical Statistics, 18:2, 306-320, DOI: 10.1198/jcgs.2009.07145.

**Usage**

```
factor_analysis_with_regression <- function(
  Y,
  X,
  k,
  niter = 1,
  shape_psi = 1/2,
  rate_psi = 1/2,
  shape_tau = 1,
  rate_tau = 0.2,
  coef_multiplier = 10,
  nonzero_structure = NULL)
```

**Arguments**

Y	n by p matrix
X	n by f matrix
k	number of factors
niter	number of iterations for the gibbs sampler to run.
shape_psi	shape parameter for psi. Can be a scalar or a k vector
rate_psi	rate parameter for psi. Can be a k vector
shape_tau	shape parameter for sigma2. Can be a p vector
rate_tau	rate parameter for sigma2. Can be a p vector
nonzero_structure	A boolean p x k matrix. If the i, jth spot is TRUE, then $\lambda_{ij}$ is free. If the i, jth spot is FALSE, then $\lambda_{ij}$ is zero. If not set, then a lower triangular matrix is used.

**Value**

alpha	An niter x p matrix of posterior values
lambda	An niter x p x k array of posterior values
tau	An niter x p matrix of posterior values
eta	An niter x n x k array of posterior values
B	An niter x k x f array of posterior values

---

factor_infinite	<i>Runs gibbs sampler for a factor model with potentially infinite k</i>
-----------------	--

---

**Description**

The model is as follows:

$$y_i = \Lambda \eta_i + \epsilon_i$$

$$\epsilon_i \sim N_p(0, \Sigma)$$

where  $\Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_p^2)$ .

See Bhattacharya, Anirban, and David B. Dunson. 'Sparse Bayesian infinite factor models.' *Biometrika* (2011): 291-306.

**Usage**

```
factor_infinite <- function(
  Y,
  k_star,
  niter = 1000,
  a_sig = 1,
  b_sig = 1,
  rho = 3,
  a1 = 1,
  a2 = 3)
```

**Arguments**

Y	n by p matrix
k_star	number of factors to cut off at
niter	number of iterations for the gibbs sampler to run.
a_sig	shape hyper parameter for the $\sigma^2$
b_sig	shape hyper parameter for the $\sigma^2$
rho	hyper parameter
a1	hyper parameter
a2	hyper parameter

**Value**

sigma2	An niter x p matrix of posterior values
lambda	An niter x p x k_star array of posterior values

---

horseshoe\_regression    *Gibbs Sampler for Horseshoe Regression*


---

### Description

The horseshoe regression model is given by

$$[y_i|x_i, \beta, \sigma] \sim N(x_i^t\beta, \sigma^2), i = 1, \dots, n,$$

$$[\beta_j|\sigma, \lambda_j] \sim N(0, \sigma^2\lambda_j^2),$$

$$[\lambda_j|A] \sim C^+(0, A),$$

$$A \sim Uniform(0, 10),$$

$$p(\sigma^2) \propto \frac{1}{\sigma^2}.$$

The half-Cauchy parameter expansion is used; given by

$$[\eta_j|\gamma_j] \sim Gamma(\frac{1}{2}, \gamma_j),$$

$$[\gamma_j] \sim Gamma(\frac{1}{2}, \frac{1}{A^2}).$$

Let  $\eta_j = \lambda_j^{-2}$ ,  $\tau_A = A^{-2}$ ,  $\tau = \frac{1}{\sigma^2}$  and  $\Lambda = diag(\eta_1, \dots, \eta_p)$ . The full conditionals are given by:

$$[\beta|Y, X, \eta, \tau] \sim \mathcal{N}((X'X + \Lambda)^{-1}X'Y, \tau^{-1}(X'X + \Lambda)^{-1}),$$

$$[\eta_j|\beta_j, \gamma_j, \tau] \sim \exp(-\frac{\tau\beta_j^2}{2} + \gamma_j),$$

$$[\gamma_j|\eta_j, \tau_A] \sim \exp(\eta_j + \tau_A),$$

$$[\tau_A|\gamma] \sim Gamma(\frac{p-1}{2}, \sum \gamma_i)I_{(\frac{1}{100}, \infty)},$$

$$[\tau|Y, X, \beta, \eta] \sim Gamma(\frac{n+p}{2}, \frac{(y - X\beta)'(y - X\beta) + \beta'\Lambda\beta}{2}).$$

### Usage

```
horseshoe_regression <- function(
  Y,
  X,
  niter = 10000)
```

### Arguments

Y	n by 1
X	n by p predictor matrix
niter	number of gibbs sampling iterations

**Details**

This function returns the generated parameters from the gibbs sampling markov chain.

**Value**

beta	An niter x p matrix
lambda	An niter x p matrix
sigma	An niter x 1 matrix

**Examples**

```
# Load the data
prostate.data = "https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data"
prostate = read.table(file = prostate.data, sep=" ", header = TRUE)
# Training data:
prostate_train = prostate[which(prostate$train),-10]
# Testing data:
prostate_test = prostate[which(!prostate$train),-10]
# Response:
y = prostate_train$lpsa
# Center and scale the data:
y = scale(y)
# And the predictors
X = scale(prostate_train[,names(prostate_train) != "lpsa"])

gibbs_hs <- horseshoe_regression(y, X, niter=10000)
shrinkage_regression_plot(gibbs_hs$beta, y, X)
```

mat\_apply

*mat apply***Description**

Apply a function to each element in an input and return an arbitrary dimensioned array.

**Usage**

```
mat_apply <- function(vec, fun)
```

**Arguments**

vec	vector of inputs fed into fun
fun	a function to call for each input of vec

**Details**

This function returns an array with first dimension of length(vec) provided length(vec) > 1—the first dimension indexes over the length(vec) outputs.

**Examples**

```
# returns a 10 x 3 x 3 array. ret[i,,] contains i*diag(3).
ret <- mat_apply(1:10, function(i){ i*diag(3) })
```

metropolis

*Metropolis MCMC***Description**

The metropolis algorithm is a special case of the metropolis-hastings algorithm, namely where the proposal distribution is symmetric.

**Usage**

```
metropolis <- function(
  rproposal,
  prob,
  niter,
  init,
  log_prob = FALSE)
```

**Arguments**

rproposal	rproposal(previous_val) generates a value from the proposal distribution
prob	prob(val_proposed)/prob(val_t_minus_1) forms the acceptance probability
niter	number of iterations to perform
init	vector of initial values
log_prob	whether or not prob function specifies log probabilities

**Details**

This function returns a niter x d matrix of values where d is the dimension of init and the dimension of each element from rproposal. The returned matrix contains all generated values of the metropolis walk.

**Examples**

```
vals <- metropolis(
  rproposal = function(val){ rnorm(1, mean = val, sd = 1){} },
  prob = posterior_prob,
  niter = 10000,
  init = 0)

library(coda)
plot(as.mcmc(vals))
```



---

plackett\_luce

*Gibbs Sampler for posterior plackett luce parameters*


---

## Description

The prior on each parameter is given by a gamma distribution.

The parameter expansion is given by Caron, Francois, and Arnaud Doucet. "Efficient Bayesian inference for generalized Bradley-Terry models." Journal of Computational and Graphical Statistics 21.1 (2012): 174-196.

## Usage

```
plackett_luce <- function(
  rank_matrix,
  shape = 1,
  rate = 1,
  niter = 1000)
```

## Arguments

rank_matrix	n by p matrix of integers ranging from 1,...,k where k = max(rank_matrix, na.rm=TRUE). Each row may contain NA values provided that at least 2 values of each row are not NA. The same integer can appear in each row of the rank matrix as well.
shape	scalar or k vector specifying prior shape parameter over pl parameters
rate	scalar or k vector specifying prior rate parameter over pl parameters
niter	number of gibbs sampling iterations

## Details

This function returns a niter x k matrix of the gibbs sampler.

## Examples

```
library(coda)

vals <- rplackett_luce(100, vs = c(100, 1, 1))
post_vals <- plackett_luce(vals)
plot(as.mcmc(post_vals))
```

---

probit\_horseshoe\_regression

*Gibbs Sampler for Probit Regression with Horseshoe Prior*


---

### Description

The probit regression model with horseshoe prior is given by

$$\begin{aligned}
 y_i | \pi_i &\sim \text{Bernoulli}(\pi_i), \\
 \pi_i &= \Phi(x_i^t \beta), \\
 [\beta_j | \lambda_j] &\sim N(0, \lambda_j^2), j = 2, \dots, p, \\
 p(\beta_1) &\propto 1, \\
 [\lambda_j | A] &\sim C^+(0, A), j = 2, \dots, p, \\
 A &\sim \text{Uniform}(0, 10).
 \end{aligned}$$

where  $\Phi$  is given by the gaussian CDF.

The implemented parameter-expanded model is given by

$$\begin{aligned}
 y_i^* &= x_i^t \beta + \epsilon_i, \\
 \epsilon_i &\sim N(0, 1), \\
 y_i &= I(y_i^* > 0).
 \end{aligned}$$

The half-Cauchy parameter expansion is also used; given by

$$\begin{aligned}
 [\eta_j | \gamma_j] &\sim \text{Gamma}(\frac{1}{2}, \gamma_j), \\
 [\gamma_j] &\sim \text{Gamma}(\frac{1}{2}, \frac{1}{A^2})
 \end{aligned}$$

and  $\eta_j = \lambda_j^{-2}$ ,  $\tau_A = A^{-2}$ , The full conditionals are given by:

$$[y_i^* | y_i, \beta, X] \sim \text{sgn}(y_i, y_i^*) N(x_i^t \beta, 1)$$

where  $\text{sgn}$  is 1 if both arguments are of the same sign and zero otherwise,

$$[\beta | Y^*, X, \eta] \sim \mathcal{N}(Q^{-1}l, Q^{-1})$$

where  $Q = X'X + \text{diag}(0, 1/\eta_2, \dots, 1/\eta_p)$  and  $l = X'Y^*$ ,

$$\begin{aligned}
 [\eta_j | \beta_j, \gamma_j] &\sim \exp(\frac{\beta_j^2}{2} + \gamma_j), \\
 [\gamma_j | \eta_j, \tau_A] &\sim \exp(\eta_j + \tau_A), \\
 [\tau_A | \gamma] &\sim \text{Gamma}(\frac{p-2}{2}, \sum \gamma_i) I_{(\frac{1}{100}, \infty)}.
 \end{aligned}$$

**Usage**

```
probit_horseshoe_regression <- function(
  Y,
  X,
  niter,
  init = NULL)
```

**Arguments**

Y                      n by 1 vector of ones and zeros  
X                        n by p predictor matrix, where  $p > 1$  and the first column of X is all 1.  
niter                    number of gibbs sampling iterations  
init                     Initial starting values for beta. If NULL, beta is set to zero.

**Details**

This function returns a niter x p matrix of values where p is the second dimension of the predictor matrix X. The returned matrix contains all generated values of the gibbs sampling markov chain.

**Examples**

```
print("TODO")
```

---

probit_regression	<i>Gibbs Sampler for Probit Regression</i>
-------------------	--

---

**Description**

The probit regression model is given by

$$y_i | \pi_i \sim \text{Bernoulli}(\pi_i),$$

$$\pi_i = \Phi(x_i^t \beta),$$

$$\beta_j \sim N(0, A^2).$$

where  $\Phi$  is given by the gaussian CDF.

The implemented parameter-expanded model is given by

$$y_i^* = x_i^t \beta + \epsilon_i,$$

$$\epsilon_i \sim N(0, 1),$$

$$y_i = I(y_i^* > 0).$$

The full conditional distributions are given by

$$[\beta | y, y^*, X] \sim N(Q^{-1}l, Q^{-1})$$

where  $Q = X^t X + A^{-2} I$  and  $l = X^t y^*$ ,

$$[y_i^* | y_i, \beta, X] \sim \text{sgn}(y_i, y_i^*) N(x_i^t \beta, 1)$$

where  $\text{sgn}$  is 1 if both arguments are of the same sign and zero otherwise.

If  $A$  is a vector,  $Q = X^t X + \text{diag}(A)$ . A flat prior on the intercept at position 1 is thus given by  $A = c(\text{Inf}, \text{rep}(3, p))$ .

## Usage

```
probit_regression <- function(
  Y,
  X,
  niter,
  A = 3,
  init = NULL)
```

## Arguments

<code>Y</code>	n by 1 vector of ones and zeros
<code>X</code>	n by p predictor matrix
<code>niter</code>	number of gibbs sampling iterations
<code>A</code>	A parameter. The default value is chosen to provide a reasonable range of $\pi$ . If $A$ is a vector, then different variances are given for each intercept.
<code>init</code>	Initial starting values for beta. If NULL, beta is set to zero.

## Details

This function returns a `niter x p` matrix of values where `p` is the second dimension of the predictor matrix `X`. The returned matrix contains all generated values of the gibbs sampling markov chain.

## Examples

```
library(LearnBayes)
library(s525)

data(donner)
y = donner$survival
X = cbind(1, donner$age, donner$male)
niter <- 2000

gibbs_results <- probit_regression(y, X, niter)
```

ridge\_regression

*Gibbs Sampler for Ridge Regression***Description**

The ridge regression model coefficients are given by

$$\arg \min_{\beta} \| Y - X\beta \|^2 + \lambda \| \beta \|^2 .$$

This is the implementation of the bayesian interpretation. Namely,

$$[y_i|x_i, \beta, \tau] \sim N(x_i^t \beta, \tau^{-1}), i = 1, \dots, n,$$

$$[\beta|\eta] \sim N(0, \eta^{-1}I),$$

$$\tau \sim \text{Gamma}(\frac{1}{100}, \frac{1}{100}),$$

$$\sigma_{\beta} \sim \text{unif}(0, A)$$

where  $A = 1000$  and  $\tau = \frac{1}{\sigma_{\beta}^2}$ .

**Usage**

```
ridge_regression <- function(
  Y,
  X,
  niter = 10000)
```

**Arguments**

Y	n by 1
X	n by p predictor matrix
niter	number of gibbs sampling iterations

**Details**

This function returns the generated parameters from the gibbs sampling markov chain.

**Value**

beta	An niter by p matrix
sigma	An niter by 1 matrix
sigma_beta	An niter by 1 matrix

## Examples

```
# Load the data
prostate.data = "https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data"
prostate = read.table(file = "prostate.data", sep="", header = TRUE)
# Training data:
prostate_train = prostate[which(prostate$train),-10]
# Testing data:
prostate_test = prostate[which(!prostate$train),-10]
# Response:
y = prostate_train$lpsa
# Center and scale the data:
y = scale(y)
# And the predictors
X = scale(prostate_train[,names(prostate_train) != "lpsa"])

gibbs_ridge <- ridge_regression(y, X, niter=10000)
shrinkage_regression_plot(gibbs_ridge$beta, y, X, main = "Ridge Prior")
```

---

rnorm\_qinv\_l

*Sample from multivariate normal distribution*


---

## Description

Sample from multivariate normal distribution with mean  $Q^{-1}l$  and covariance matrix  $Q^{-1}$ .

## Usage

```
rnorm_qinv_l <- function(
  n,
  Q,
  l)
```

## Arguments

n	number of elements to generate
Q	p by p precision matrix.
l	p by 1 vector

## Details

The algorithm is as follows

1. Cholesky decomposition of  $Q$  into  $LL^t$ .
2. Sample  $z$  from  $\text{rnorm}(p)$ . Let  $y = Lz + l$ .
3. Solve for  $x$  in  $LL^t x = y$  and return.

## Value

beta	An p x 1 vector if $n = 1$ otherwise a n by p matrix
------	--

---

rplackett_luce	<i>Generate random permutations from a plackett luce distribution</i>
----------------	---

---

**Description**

The plackett luce model assigns a probability distribution over permutations.

**Usage**

```
rplackett_luce <- function(n, vs)
```

**Arguments**

n	number of permutations to generate
vs	k vector of probabilities; need not be scaled to sum to 1

**Details**

This function returns a n x k matrix where each row is a permutation generated from a plackett luce distribution with parameters given by vs.

**Examples**

```
library(coda)

vals <- rplackett_luce(100, vs = c(100, 1, 1))
post_vals <- plackett_luce(vals)
plot(as.mcmc(post_vals))
```

---

rpost_regression_coef	<i>Generate posterior for regression coefficients</i>
-----------------------	---

---

**Description**

Generate a  $N(\mu, \Sigma)$  random variable where

$$\Sigma = (X^t X + D^{-1})^{-1},$$

$$\mu = \Sigma X^t \alpha.$$

The algorithm is  $O(np^2)$ ; for large p it performs fast.

**Usage**

```
rpost_regression_coef <- function(
  X,
  D,
  alpha,
  u = NULL)
```

**Arguments**

X	n by p matrix
D	p by p matrix
alpha	n by 1 vector
u	Optional. If specified, don't generate $u \sim N(0, D)$ .

**Details**

The algorithm is from

Bhattacharya, Anirban, Antik Chakraborty, and Bani K. Mallick. "Fast sampling with Gaussian scale mixture priors in high-dimensional regression." *Biometrika* (2016): asw042.

The algorithm is as follows:

1. Sample  $u \sim N(0, D)$ ,  $d \sim N(0, I_n)$
2. Set  $v = Xu + d$
3. Solve  $(XDX^t + I_n)w = (\alpha - v)$
4. Return  $\beta = u + DX^tw$

**Value**

beta	An p x 1 vector
------	-----------------

---

shrinkage\_regression\_plot

*Plot the MC chain of regression coefficients compared to OLS.*

---

**Description**

Plot the MC chain of regression coefficients compared to OLS using 95% HPD confidence intervals.

**Usage**

```
horseshoe_regression_plot <- function(
  beta,
  Y,
  X,
  main = "Horseshoe Prior",
  ylim = NULL)
```

**Arguments**

beta	n x p MCMC chain of beta coefficients.
Y	Y values used to generate beta; used as input to OLS regression
X	X values used to generate beta; used as input to OLS regression
main	Title of the plot
ylim	y range of plot



**Examples**

```
# Load the data
prostate.data = "https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data"
prostate = read.table(file = "prostate.data", sep="", header = TRUE)
# Training data:
prostate_train = prostate[which(prostate$train),-10]
# Testing data:
prostate_test = prostate[which(!prostate$train),-10]
# Response:
y = prostate_train$lpsa
# Center and scale the data:
y = scale(y)
# And the predictors
X = scale(prostate_train[,names(prostate_train) != "lpsa"])

gibbs_hs <- horseshoe_regression(y, X, niter=10000)
shrinkage_regression_plot(gibbs_hs$beta, y, X)
```

# Index

## \*Topic **package**

s525-package, [2](#)

factor\_analysis, [2](#)

factor\_analysis\_with\_regression, [3](#)

factor\_infinite, [4](#)

horseshoe\_regression, [6](#)

mat\_apply, [7](#)

metropolis, [8](#)

plackett\_luce, [9](#)

probit\_horseshoe\_regression, [10](#)

probit\_regression, [11](#)

ridge\_regression, [13](#)

rnorm\_qinv\_l, [14](#)

rplackett\_luce, [15](#)

rpost\_regression\_coef, [15](#)

s525 (s525-package), [2](#)

s525-package, [2](#)

shrinkage\_regression\_plot, [16](#)