Package 'FMMcsVS'

February 8, 2023

Type Package

Title Bayesian Finite Mixure Regression Models with Cluster-Specific Variable Selections

Version 0.1.0

Author Zhen Wang

Maintainer Zhen Wang <zwangiowa@gmail.com>

Description Different MCMC algorithms for different Bayesian mixture models in a regression setup, including clustering, variable selection and regression coeficcient estimations.

Depends Matrix,
caret,
MCMCprecision,
truncdist,
faux,
mcclust.ext,
philentropy,
aricode,
gsl,
label.switching,
mclust,
clustvarsel,
coda,
flexmix,
BNPmix,
factoextra,
MASS,
R (¿= 2.10)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

R topics documented:

coffee *Coffee Data*

Description

Data on the chemical composition of coffee samples collected from around the world, comprising 43 samples from 29 countries. Each sample is either of the Arabica or Robusta variety. Twelve of the thirteen chemical constituents reported in the study are given. The omitted variable is total chlorogenic acid; it is generally the sum of the chlorogenic, neochlorogenic and isochlorogenic acid values.

Usage

```
coffee
```

Format

A data frame with 43 observations and 14 variables. The first two columns contain Variety and Country, respectively, while the remaining 12 columns contain the chemical properties. The Variety is either (1) Arabica or (2) Robusta

Note

The German to English translations of the variable names were carried out by Dr. Sharon M. McNicholas.

Source

Streuli, H. (1973). Der heutige stand der kaffeechemie. In Association Scientifique International du Cafe, 6th International Colloquium on Coffee Chemisrty, Bogata, Columbia, pp. 61–72.

Examples

```
coffee
variety_coffee <- coffee$Variety
```

`country` *Country Data*

Description

Country data with socio-economic and health factors that determine the overall development of the country.

Usage

```
country
```

Format

A data frame with 167 observations and 10 variables:

country country name

child_mort Death of children under 5 years of age per 1000 live births

exports Exports of goods and services per capita. Given as percentage of the GDP per capita

health Total health spending per capita. Given as percentage of GDP per capita

imports Imports of goods and services per capita. Given as percentage of the GDP per capita

income Net income per person

inflation The measurement of the annual growth rate of the Total GDP

life_expec The average number of years a new born child would live if the current mortality patterns are to remain the same

total_fer The number of children that would be born to each woman if the current age-fertility rates remain the same.

gdpp The GDP per capita. Calculated as the Total GDP divided by the total population.

Source

Kaggle Datasets, https://www.kaggle.com/datasets/rohan0301/unsupervised-learning-on-country-data

Examples

```
country
name_country <- country$country
```

`data_gen_func` *Generate Simulation Data*

Description

Generate the simulation data for various models. The data consists of single-dim response y and multi-dim covariates X. For subjects in different clusters, (alpha, beta, zeta, lambda) are different. The function can generate correlated data, by specifying the correlation matrix among X, and balanced/inbalanced data by specifying the cluster probabilities.

For the split model, the data consists of y, fixed covariates W and random covariates Z. For subjects in different clusters, (alpha, beta, psi, zeta, lambda) are different.

Usage

```
data_gen_func(n = 500, alpha_true =  c(0.1, -0.6, 0.5),
             beta_true = rbind(c(0, 0, -0.5, 0, 0.5, 0),
                               c(-0.7, 0, 0.4, 0, 0, 0),
                               c(0.6, 0, 0, 0, -0.4, 0)),
             lambda = c(2, 2, 2), zeta_sep = 1,
             eta = 1, sample_prob = c(1, 1, 1),
             cor_mtx = NULL, rho = rep(0, 3))

data_gen_split(n = 500, alpha_true = c(0.1, -0.6, 0.5),
              beta_true = rbind(c(0, 0, -0.5, 0, 0.5, 0),
                                c(-0.7, 0, 0.4, 0, 0, 0),
                                c(0.6, 0, 0, 0, -0.4, 0)),
              psi_true = rep(c(-1,0,1),3),
              W_mean = 0, lambda = c(2, 2, 2), zeta_sep = 1,
              eta = 1, sample_prob = c(1, 1, 1), rho = 0.5)
```

Arguments

`n`	sample size
`alpha_true`	numeric vector, each element represents intercept for each cluster, length should equal the number of clusters
`beta_true`	numeric matrix, each row represents regression coefficients for each cluster, number of rows should equal the number of clusters
`lambda`	numeric vector, each element represent precision of response y in each cluster
`zeta_sep`	numeric, the difference of zeta values in different clusters, the true zeta values are set to be seq(0, true_M-1) * zeta_sep
`eta`	numeric, precision of covariates X, for split model, the precision of Z
`sample_prob`	numeric vector, ratios of sizes of clusters
`cor_mtx`	numeric matrix, correlation matrix of X, default value is for D=6, M=3, X1-X2, X3-X4, X5-X6 have blocked-wise correlation with coefficients given by rho (no specified in definition)
`rho`	numeric vector, correlation coefficients among X in the case described above. For split model, rho is a numeric value, which is the pair-wise correlation among columns of W

Details

For data_gen_func:

The regression coefficients alpha, beta, the mean of X, zeta, and the precision of y, lambda are cluster-specific.

If no cor_mtx value is specified: 1) if rho is a vector of zero's, generated samples have independent X's; 2) if non-zero values are specified in rho, it's assumed to be the D=6, M=3 case, with a blocked-wise correlations of X1-X2, X3-X4 and X5-X6, and correlation coefficients given by rho, if a different setup (eg, different D or M) is desired, cor_mtx must be specified.

For data_gen_split:

The regression coefficients alpha, beta for W, psi for Z, the mean of Z, zeta, and the precision of y, lambda, are cluster-specific.

W has a pair-wise correlation of rho.

Value

A list is returned:

`y`	numeric vector, response with length n
`X`	numeric matrix, covariates matrix with dimension M*D
`W`	numeric matrix, fixed covariates matrix in split model
`Z`	numeric matrix, random covariates matrix in split model
`index`	numeric vector with length n, group membership indicator ranging from 1 to M for each subject
`alpha_true`	specified true alpha values
`beta_true`	specified true beta values
`psi_true`	specified true psi values for split model

Author(s)

Zhen Wang ¡zwangiowa@gmail.com¿

Examples

```
##generate data with independent X
sample_data_1 <- data_gen_func()

##generate inbalanced data with ratio 1:2:10
sample_data_2 <- data_gen_func(sample_prob = c(1,2,10))

##generate data where D=6, M=3, and X has blocked-wise correlations of X1-X2, X3-X4 and X5-X6, and correlation c
sample_data_3 <- data_gen_func(rho = c(0.2,0.5,0.8))

##generate data where correlation matrix among X is given by:
require(Matrix)
cor_mtx = bdiag(matrix(c(1/1, 0.8,0.8, 1/1), 2, 2),
                matrix(c(1/1, 0.2,0.2, 1/1), 2, 2),
                matrix(c(1/1, 0.9,0.9, 1/1), 2, 2))
sample_data_4 <- data_gen_func(cor_mtx=cor_mtx)
```

```
##generate data for the split model, where W has pair-wise correlation of 0.3
sample_split_data_1 <- data_gen_split(rho=0.3)
```

`flea` *Flea Beatles Measurements*

Description

This data is from a paper by A. A. Lubischew, "On the Use of Discriminant Functions in Taxonomy", Biometrics, Dec 1962, pp.455-477.

Usage

```
flea
```

Format

A data frame with 74 observations and 7 variables:

tars1 width of the first joint of the first tarsus in microns (the sum of measurements for both tarsi)

tars2 the same for the second joint

head the maximal width of the head between the external edges of the eyes in 0.01 mm

aede1 the maximal width of the aedeagus in the fore-part in microns

aede2 the front angle of the aedeagus (1 unit = 7.5 degrees)

aede3 the aedeagus width from the side in microns

species which species is being examined - concinna, heptapotamica, heikertingeri

Examples

```
head(flea)
```

posterior_inf *Posterior Inference for Various Models*

Description

Calculate different metrics defined in the paper to evaluate performance of different models in : clustering accuracy, parameter estimation accuracy and variable selection accuracy. Besides the N-IFPP models, functions for inference of RPMS, BNPmix (P-Y mixture models, M9 and M10) and mclust (model-based clustering, M7) are also included.

Usage

```
post_inf(sim_res, scl = 10001:1e5, data)

post_inf_rpms(sim_res, scl = 10001:1e5, data)

post_inf_bnpmix(sim_res, data)

mclust_vs(data)
```

Arguments

`sim_res`	list, MCMC simulation results, return from the simulation functions defined in the package
`scl`	numeric vector, index of remaining samples after burn-in
`data`	list, the data input used to run the simulations

Details

post_inf runs inference for N-IFPP models, post_inf_rpms runs inference results for the RPMS, post_inf_bnpmix runs inference for the two P-Y mixture models (M9, M10). The current versions of the functions can only handle the case of M_true = 3 as defined in the default simulation data setups.

mclust_vs does inference for the model-based clustering model (M7) introduced by Fraley and Raftrey (2002, JASA), two different types of VS procedures were implemented to adapt to the regression setup, which is not considered in the original model. The two methods are: 1) implemented the variable selection methods by the clustvarsel package, obtain the VS and clustering results, then a common set of variables are selected for all clusters, fit a linear regression model within each cluster to obtain estimates of alpha and beta; 2) run the mclust package for clustering with both the response and covariates, based on the clustering results with fixed K=3, run the variable selection procedure by the clustvarsel package for each cluster so a cluster-specific VS can be achieved, then within each cluster, fit a linear regression model to obtain estimates of alpha and beta;

Value

post_inf, post_inf_rpms and post_inf_bnpmix return:

`true_size`	numeric vector, the true cluster sizes in the data

- `measure_km_ari` numeric, a measurement defined to measure how far away the clusters are from each other, calculated by the ARI values of the true clutering membership and the clustering result of the K-means method, a larger value generally means more distant clusters, thus easier clustering problem
- `auto_corr_k` numeric vector, auto-correlations for posterior of K
- `geweke_k` numeric, the Z-statistic for the geweke diagnostics for posterior of K
- `post_k` numeric vector, posterior distribution of K
- `mse_a1, mse_a2, mse_a3` numeric vectors, the quartiles of the mean squared errors of alpha for the 3 clusters
- `mse_b1, mse_b2, mse_b3` numeric vectors, the quartiles of the mean squared errors of beta for the 3 clusters
- `ARI` numeric vector, the (0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.5, 0.75, 1) quantiles of ARI values calculated by all posterior samples of c
- `c_bin` numeric vector, sizes of clusters obtained by the Binder loss
- `c_vi` numeric vector, sizes of clusters obtained by the VI loss
- `ari_point_bin` numeric, ARI values calculated by c_bin and the truth
- `ari_point_vi` numeric, ARI values calculated by c_vi and the truth
- `FS` numeric vector, the mean False Spaecity for each cluster
- `MS` numeric vector, the mean Missed Spaecity for each cluster
- `ARI_bnpmix` numeric vector, the ARI values calculated from all posterior samples of c by BNPmix

mclust_vs returns:

- `clust_no_g` numeric vector, the sizes of clusters by mclust when G is not pre-specified
- `ari_no_g` numeric, ARI values for clustering results by mclust when G is not pre-specified
- `cluster_no_g_vs` numeric, the sizes of clusters by clustvarsel when G is not pre-specified
- `ari_no_g_vs` numeric, ARI values for clustering results by clustvarsel when G is not pre-specified
- `clust_g3` numeric vector, the sizes of clusters by mclust when G is fixed at 3
- `ari_g3` numeric, ARI values for clustering results by mclust when G is fixed at 3
- `clust_g3_vs` numeric, ARI values for clustering results by clustvarsel when G is fixed at 3
- `ari_g3_vs` numeric, ARI values for clustering results by clustvarsel when G is fixed at 3
- `se_a1.1, se_a2.1, se_a3.1` squared-errors of alpha values for M7, method 1)
- `se_b1.1, se_b2.1, se_b3.1` squared-errors of alpha values for M7, method 1)
- `FS1.1, FS2.1, FS3.1` FS values for each cluster for M7, method 1)
- `MS1.1, MS2.1, MS3.1` MS values for each cluster for M7, method 1)

se_a1.2, se_a2.2, se_a3.2
squared-errors of alpha values for M7, method 2)

se_b1.2, se_b2.2, se_b3.2
squared-errors of alpha values for M7, method 2)

FS1.2, FS2.2, FS3.2
FS values for each cluster for M7, method 2)

MS1.2, MS2.2, MS3.2
MS values for each cluster for M7, method 2)

Note

The MSE values, FS, MS values for VS accuracy are calculated with samples of K = M = 3. ARI values are calculated with all posterior samples.

Author(s)

Zhen Wang ¡zwangiowa@gmail.com¿

References

Chris Fraley and Adrian E Raftery. Model-based clustering, discriminant analysis, and density esti- mation. Journal of the American Statistical Association, 97(458):611–631, 2002.

Riccardo Corradin, Antonio Canale, and Bernardo Nipoti. Bnpmix: An r package for bayesian non- parametric modeling via pitman-yor mixtures. Journal of Statistical Software, 100(15):1–33, 2021.

Examples

```
##generate simulation data
simulation_data <- data_gen_func()

##FBMM with VS, hyper-prior for beta_bel
simulation_1 <- simulation_func(simulation_data$X, simulation_data$y, prior="Bessel")

##run posterior inference, burn-in the first 20k samples
post_fbmm_vs_1 <- post_inf(simulation_1, 20001:1e5, simulation_data)
```

`ppd_sim` *Calculate the Posterior Predictive Density (ppd) for New Data Subjects*

Description

A function to calculate the posterior predictive density for a new subject with response y* and covariates X*.

Usage

```
ppd_sim(sim.res, x.new, y.new, scl=10001:1e5, thin=20)
```

Arguments

`sim.res`	list, MCMC simulation results returned by simulation_func, simulation_split or simulation_func_rpms
`x.new`	numeric vector of length D, covariates for the new subject
`y.new`	numeric, the new y value to estimate the ppd at
`scl`	numeric vector, the index of remaining samples after burn-in
`thin`	numeric, the thinning factor to speed up calculations, posterior samples for every thin-th iterations are used to calculate ppd

Details

See the Appendix of the paper for formulas of the ppd.

Value

The ppd of x.new at y.new is returned.

Author(s)

Zhen Wang ¡zwangiowa@gmail.com¿

Examples

```
##generate simulation data
simulation_data <- data_gen_func()

##FBMM without VS, hyper-prior for beta_bel
simulation_1 <- simulation_func(simulation_data$X, simulation_data$y, prior="Bessel", SS=F)

x.new <- rnorm(6)
y.new <- 0.5
##calculate ppd
ppd.new.sample <- ppd_sim(simulation_1, x.new, y.new, scl=10001:1e5, thin=20)
```

r_sq_post — *Calculate R-square Values for all MCMC Samples*

Description

A function to calculate the R-square values for all posterior samples returned by MCMC simulations.

Usage

```
r_sq_post(sim_res, data, scl=10001:1e5)
```

Arguments

`sim_res`	list, MCMC simulation results returned by simulation_func, simulation_split or simulation_func_rpms
`data`	dataframe, consists of response y and covariates X
`scl`	numeric vector, the index of remaining samples after burn-in

Details

Given the posterior samples of K, and c, for each iteration, within each cluster, fit a linear regression, obtain the R^2 value, then for each iteration, we obtain k_post R^2 values for k_post different clusters. Do such calculations for all posterior samples.

Value

The R-square values for all clusters in all posterior samples are returned as a numeric vector.

Author(s)

Zhen Wang ¡zwangiowa@gmail.com¿

Examples

```
##generate simulation data
simulation_data <- data_gen_func()

##FBMM without VS, hyper-prior for beta_bel
simulation_1 <- simulation_func(simulation_data$X, simulation_data$y, prior="Bessel", SS=F)

##calculate the R^2 values
r_sq_samples <- r_sq_post(simulation_1, simulation_data)
```

simulation_func *MCMC Simulations for the Bayesian Finite Mixture Regression Models with Cluster-Specific Variable Selection*

Description

A simulation function running MCMC simulations for various models with different algorithms. Accepted data should be one-dim response y and multi-dim (D ¿= 1) covariates X expect for the split model, for split model, W and Z should be specified instead of a single covariates matrix X.

Usage

```
simulation_func(X, y, prior = "Dirichlet", SS = TRUE, N = 1e5, gamma_hyperprior = TRUE,
                gamma_fixed = 1, a_gamma = 10, b_gamma = 10, a_unif = 0,
                a_w = 1, b_w = 1, Lambda = 3, a_bessel = 2,
           b_bessel_hyperprior = TRUE, b_bessel_fixed = 1.1, a_b_bessel = 1, b_b_bessel = 10,
                mu = 0, a_tau = 1, b_tau = 1, a_zeta = 0, b_zeta = 1,
                a_lambda = 5, b_lambda = 2, a_alpha = 0, b_alpha = 0.01,
                a_eta = 5, b_eta = 2, L_dynamic = 10, M_init = 6,
                lambda_init = 2, alpha_init = 0)


simulation_split(W, Z, y, prior = "Dirichlet", SS = TRUE, N = 1e5, gamma_hyperprior = TRUE,
                 gamma_fixed = 1, a_gamma = 1, b_gamma = 1,
                 a_w = 1, b_w = 1, Lambda = 3, a_bessel = 2,
           b_bessel_hyperprior = TRUE, b_bessel_fixed = 1.1, a_b_bessel = 1, b_b_bessel = 10,
                 mu = 0, a_tau = 1, b_tau = 1, a_zeta = 0, b_zeta = 1,
                 a_psi = 0, a_4 = 1, b_4 = 1,
                 a_lambda = 5, b_lambda = 2, a_alpha = 0, b_alpha = 0.01,
                 a_eta = 5, b_eta = 2, M_init = 6,
                 lambda_init = 2, alpha_init = 0)
```

Arguments

`X`	numeric matrix, covariates matrix
`W`	numeric matrix, covariates matrix that is fixed in a split model
`X`	numeric matrix, covariates matrix that is modeled with Gaussian distributions in a split model
`y`	numeric vector, response
`prior`	char, priors on mixture weights, or different algorithm. "Dirichlet" for FDMM with conditional algorithm, "Bessel" for FBMM with conditional algorithm, "Dynamic_FDMM" for the Dynamic FDMM model with conditional algorithm, "Dirichlet_Marginal" for FDMM with marginal algorithm, "Uniform" for FUMM with conditional algorithm. For split model, only "Dirichlet" and "Bessel" are enabled

`SS` logical, if TRUE, spike and slab prior is specified on beta for cluster-specific variable selection; if FALSE, a continuous Gaussian prior is specified to beta, no variable selection in implemented

`N` numeric, number of iterations

`gamma_hyperprior`
logical, if TRUE, for FDMMM with conditional and marginal algorithm, a hyper-prior is specified to the concentration parameter gamma, otherwise, gamma is fixed

`gamma_fixed` numeric, if gamma_hyperprior=FALSE, the fixed value for gamma

`a_gamma, b_gamma`
numeric, if gamma_hyperprior=TRUE, a Gamma(a_gamma, b_gamma) is assigned to gamma

`a_unif` numeric, for FUMM, the unnormalised mixture weights S follows Unif(a_unif, 1), 0¡=a_unif¡1

`a_w, b_w` numeric, a Beta(a_w, b_w) is assigned to the SS weights w

`Lambda` numeric, M ˜ Posson_1(Lambda), a shifted Poisson distribution

`a_bessel` numeric, for FBMM, alpha_bel is fixed at this value

`b_bessel_hyperprior`
logical, if TRUE, a hyperprior is specified to beta_bel

`b_bessel_fixed` numeric, if b_bessel_hyperprior=FALSE, beta_bel is fixed at this value

`a_b_bessel, b_b_bessel`
numeric, if b_bessel_hyperprior=TRUE, a Gamma(a_b_bessel, b_b_bessel) hyper-prior is assigned to beta_bel-1

`mu` numeric, mean of the slab part of the SS prior on beta

`a_tau, b_tau` numeric, a Gamma(a_tau, b_tau) hyper-prior for the precision parameter tau of the slab part of the SS prior on beta

`a_zeta, b_zeta` numeric, a Normal(a_tau, b_tau) hyper-prior for the mean parameter of X, zeta

`a_lambda, b_lambda`
numeric, a Gamma(a_lambda, b_lambda) hyper-prior for the precision parameter of y, lambda

`a_alpha, b_alpha`
numeric, a Normal(a_alpha, b_alpha) hyper-prior for the intercept parameter alpha

`a_eta, b_eta` numeric, a Gamma(a_eta, b_eta) hyper-prior for the precision parameter of X, eta

`a_psi` numeric, the prior mean for psi, the regression coefficients corresponding to W

`a_4, b_4` numeric, a Gamma(a_4, b_4) hyper-prior is assigned to b_psi, the prior precison of psi

`L_dynamic` numeric, the number of the auxiliary states, L, in Algorithm 8 of Neal (20000, JCGS)

`M_init` numeric, initial value of M

`lambda_init` numeric, initial value of lambda

`alpha_init` numeric, initial value of alpha

Details

For simulation_func, a total of four different models are considered: within the N-IFPP category: FDMM, FBMM and FUMM, not in N-IFPP: Dynamic FDMM model introduced in Fr\"uhwirth-Schnatter et al (2021, BA).

For the FDMM, both the conditional and marginal algorithms are implemented, for other models, only the conditional algorithm is implemented.

For the marginal algorithm, Algorithm 8 of Neal (2000, JCGS) is applied.

For simulation_split, condistional algorithms are implemented for the split model, for FDMM and FBMM only.

Value

`M_post`	numeric vector of length N, posterior samples of M
`c_post`	numeric matrix of dim N*n, each row represents posterior samples of c for one iteration
`k_post`	numeric vector of length N, posterior samples of K
`U`	numeric vector of length N, posterior samples of U, the auxiliary variable for the conditional algorithm of N-IFPP models
`gamma_post`	numeric vector of length N, posterior samples of the concentration parameter gamma in both FDMM models
`Beta_post`	list, each element of the list represents posterior samples of all beta components in each iteration, of length M_post*D
`alpha_post`	list, each element of the list represents posterior samples of all alpha components in each iteration, of length M_post
`zeta_post`	list, each element of the list represents posterior samples of all zeta components in each iteration, of length M_post*D
`lambda_post`	numeric vector, posterior samples of lambda, precision of y
`eta_post`	numeric vector, posterior samples of eta, precision of X
`tau_post`	numeric vector, posterior samples of tau, precision of the slab part in SS prior
`w_post`	numeric matrix of dim N*D, each row represents posterior smaples of the SS weights, w, in one iteration
`weight_post`	list, each element represents posterior samples of the mixture weights, omega, of length M_post
`psi_post`	numeric matrix, each row represents posterior smaples of psi for each iteration, for split model only
`b_psi_post`	numeric vector, posterior smaples of b_psi, for split model only
`a_lambda`	pre-specified values
`b_lambda`	pre-specified values
`a_eta`	pre-specified values
`b_eta`	pre-specified values
`X`	covariates input
`y`	response input
`W`	covariates input that is fixed, for split model only
`Z`	covariates input that is modeled, for split model only

Note

For current version of simulation_split, it's required that dim(W) ¿= 2 and dim(Z) ¿= 3.

Author(s)

Zhen Wang ¡zwangiowa@gmail.com¿

References

Generalized Mixtures of Finite Mixtures and Telescoping Sampling. Sylvia Fr\"uhwirth-Schnatter, Gertraud Malsiner-Walli, and Bettina Gr\"un, Bayesian Analysis, 16: 1279-1307, 2021.

Radford M. Neal. Markov chain sampling methods for Dirichlet process mixture models. Journal of Computational and Graphical Statistics, 9(2):249–265, 2000.

Is infinity that far? A Bayesian nonparametric perspective of finite mixture models. Raffaele Argiento and Maria De Iorio, Ann. Statist. 50(5): 2641-2663, 2022.

Examples

```
##generate simulation data
simulation_data <- data_gen_func()

##FBMM without VS, hyper-prior for beta_bel
simulation_1 <- simulation_func(simulation_data$X, simulation_data$y, prior="Bessel", SS=F)

##Dynamic FDMM with VS, with fixed gamma value of 2
simulation_2 <- simulation_func(simulation_data$X, simulation_data$y, prior="Dynamic_FDMM",
                                gamma_hyperprior = FALSE,gamma_fixed = 2)

##FUMM where S ~ Unif(0.2, 1)
simulation_3 <- simulation_func(simulation_data$X, simulation_data$y, prior="Uniform", a_unif=0.2)

##generate data for the split model
data_split <- data_gen_split()

###split model with FBMM
simulation_split_fbmm <- simulation_split(data_split$W, data_split$Z, data_split$y, prior = "Bessel")
```

`simulation_func_rpms` *MCMC Simulations for the RPMS Model*

Description

A simulation function running MCMC simulations for the RPMS model introduced in Barcella et al (SIM, 2016). Accepted data should be one-dim response y and multi-dim (D ¿= 1) covariates X.

Usage

```
simulation_func_rpms(X, y, SS = TRUE, N = 1e5, a_w = 1, b_w = 1, mu = 0, a_tau = 1, b_tau = 1,
                  a_zeta = 0, b_zeta = 1, a_lambda = 5, b_lambda = 2, a_alpha = 0,
                     b_alpha = 0.01, a_eta = 5, b_eta = 2, a_adp = 1, b_adp = 1,
              k_init = 6, lambda_init = 2, alpha_dp_init = 1, alpha_init = 0, m_aux = 10)
```

Arguments

X	numeric matrix, covariates matrix
y	numeric vector, response
SS	logical, if TRUE, spike and slab prior is specified on beta for cluster-specific variable selection; if FALSE, a continuous Gaussian prior is specified to beta, no variable selection in implemented
N	numeric, number of iterations
a_w, b_w	numeric, a Beta(a_w, b_w) is assigned to the SS weights w
mu	numeric, mean of the slab part of the SS prior on beta
a_tau, b_tau	numeric, a Gamma(a_tau, b_tau) hyper-prior for the precision parameter tau of the slab part of the SS prior on beta
a_zeta, b_zeta	numeric, a Normal(a_tau, b_tau) hyper-prior for the mean parameter of X, zeta
a_lambda, b_lambda	numeric, a Gamma(a_lambda, b_lambda) hyper-prior for the precision parameter of y, lambda
a_alpha, b_alpha	numeric, a Normal(a_alpha, b_alpha) hyper-prior for the intercept parameter alpha
a_eta, b_eta	numeric, a Gamma(a_eta, b_eta) hyper-prior for the precision parameter of X, eta
a_adp, b_adp	numeric, a Gamma(a_adp, b_adp) hyper-prior for the concentration parameter, alpha, of DP
k_init	numeric, initial value of K
lambda_init	numeric, initial value of lambda
alpha_dp_init	numeric, initial value of alpha of DP
alpha_init	numeric, initial value of alpha
m_aux	numeric, the number of the auxiliary states, m, in Algorithm 8 of Neal (20000, JCGS)

Details

A marginal algorithm is implemented, Algorithm 8 of Neal (2000, JCGS) is applied in this algorithm.

Value

`c_post`	numeric matrix of dim N*n, each row represents posterior samples of c for one iteration
`k_post`	numeric vector of length N, posterior samples of K
`alpha_dp_post`	numeric vector of length N, posterior samples of concentration parameter alpha of DP
`Beta_post`	list, each element of the list represents posterior samples of all beta components in each iteration, of length M_post*D
`alpha_post`	list, each element of the list represents posterior samples of all alpha components in each iteration, of length M_post
`zeta_post`	list, each element of the list represents posterior samples of all zeta components in each iteration, of length M_post*D
`lambda_post`	numeric vector, posterior samples of lambda, precision of y
`eta_post`	numeric vector, posterior samples of eta, precision of X
`tau_post`	numeric vector, posterior samples of tau, precision of the slab part in SS prior
`w_post`	numeric matrix of dim N*D, each row represents posterior smaples of the SS weights, w, in one iteration
`a_lambda`	pre-specified values
`b_lambda`	pre-specified values
`a_eta`	pre-specified values
`b_eta`	pre-specified values
`X`	covariates input
`y`	response input

Author(s)

Zhen Wang ¡zwangiowa@gmail.com¿

References

William Barcella, Maria De Iorio, Gianluca Baio, and James Malone-Lee. Variable selection in co- variate dependent random partition models: an application to urinary tract infection. Statistics in Medicine, 35(8):1373–1389, 2016.

Radford M. Neal. Markov chain sampling methods for Dirichlet process mixture models. Journal of Computational and Graphical Statistics, 9(2):249–265, 2000.

Examples

```
##generate simulation data
simulation_data <- data_gen_func()

##RPMS with VS
simulation_rpms_1 <- simulation_func_rpms(simulation_data$X, simulation_data$y)

##RPMS without VS
simulation_rpms_2 <- simulation_func_rpms(simulation_data$X, simulation_data$y, SS=F)
```