Package ‘multinomialLogitMix’

July 17, 2023

Type Package

Title Clustering Multinomial Count Data under the Presence of Covariates

Version 1.1

Date 2023-07-13

Maintainer Panagiotis Papastamoulis <papapast@yahoo.gr>

Description Methods for model-based clustering of multinomial counts under the presence of covariates using mixtures of multinomial logit models, as implemented in Papastamoulis (2023) <DOI:10.1007/s11634-023-00547-5>. These models are estimated under a frequentist as well as a Bayesian setup using the Expectation-Maximization algorithm and Markov chain Monte Carlo sampling (MCMC), respectively. The (unknown) number of clusters is selected according to the Integrated Completed Likelihood criterion (for the frequentist model), and estimating the number of non-empty components using overfitting mixture models after imposing suitable sparse prior assumptions on the mixing proportions (in the Bayesian case), see Rousseau and Mengersen (2011) <DOI:10.1111/j.1467-9868.2011.00781.x>. In the latter case, various MCMC chains run in parallel and are allowed to switch states. The final MCMC output is suitably post-processed in order to undo label switching using the Equivalence Classes Representatives (ECR) algorithm, as described in Papastamoulis (2016) <DOI:10.18637/jss.v069.c01>.

License GPL-2

Imports Rcpp (>= 1.0.8.3), MASS, doParallel, foreach, label.switching, ggplot2, coda, matrixStats, mvtnorm, RColorBrewer

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Author Panagiotis Papastamoulis [aut, cre]

Repository CRAN

Date/Publication 2023-07-17 05:00:02 UTC

R topics documented:

  multinomialLogitMix-package ........................................ 2
multinomialLogitMix-package

Clustering Multinomial Count Data under the Presence of Covariates

Description

Methods for model-based clustering of multinomial counts under the presence of covariates using mixtures of multinomial logit models, as implemented in Papastamoulis (2023) <DOI:10.1007/s11634-023-00547-5>. These models are estimated under a frequentist as well as a Bayesian setup using the Expectation-Maximization algorithm and Markov chain Monte Carlo sampling (MCMC), respectively. The (unknown) number of clusters is selected according to the Integrated Completed Likelihood criterion (for the frequentist model), and estimating the number of non-empty components using overfitting mixture models after imposing suitable sparse prior assumptions on the mixing proportions (in the Bayesian case), see Rousseau and Mengersen (2011) <DOI:10.1111/j.1467-9868.2011.00781.x>. In the latter case, various MCMC chains run in parallel and are allowed to switch states. The final MCMC output is suitably post-processed in order to undo label switching using the Equivalence Classes Representatives (ECR) algorithm, as described in Papastamoulis (2016) <DOI:10.18637/jss.v069.c01>.

Details

The DESCRIPTION file:

```
Package:        multinomialLogitMix
Type:           Package
Title:          Clustering Multinomial Count Data under the Presence of Covariates
Version:        1.1
Date:           2023-07-13
Authors@R:      c(person(given = "Panagiotis", family = "Papastamoulis", email = "papapast@yahoo.gr", role = c( "aut", "cre"))
Maintainer:     Panagiotis Papastamoulis <papapast@yahoo.gr>
Index
```

Description: Methods for model-based clustering of multinomial counts under the presence of covariates using mixtures of inference. Equivalence Classes Representatives (ECR) algorithm, as described in Papastamoulis (2016) <DOI:10.18637/jss.v069.c01>.

License: GPL-2

Imports: Rcpp (>= 1.0.8.3), MASS, doParallel, foreach, label.switching, ggplot2, coda, matrixStats, mvtnorm, RColorBrewer

LinkingTo: Rcpp, RcppArmadillo

Author: Panagiotis Papastamoulis [aut, cre] (<https://orcid.org/0000-0001-9468-7613>)

Index of help topics:

- **dealWithLabelSwitching**: Post-process the generated MCMC sample in order to undo possible label switching.
- **expected_complete_LL**: Expected complete LL
- **gibbs_mala_sampler**: The core of the Hybrid Gibbs/MALA MCMC sampler for the multinomial logit mixture.
- **gibbs_mala_sampler_ppt**: Prior parallel tempering scheme of hybrid Gibbs/MALA MCMC samplers for the multinomial logit mixture.
- **log_dirichlet_pdf**: Log-density function of the Dirichlet distribution
- **mala_proposal**: Proposal mechanism of the MALA step.
- **mixLoglikelihood_GLM**: Log-likelihood of the multinomial logit.
- **mix_mnm_logistic**: EM algorithm
- **multinomialLogitMix**: Main function
- **multinomialLogitMix-package**: Clustering Multinomial Count Data under the Presence of Covariates
- **multinomial_logistic_EM**: Part of the EM algorithm for multinomial logit mixture
- **myDirichlet**: Simulate from the Dirichlet distribution
- **newton_raphson_mstep**: M-step of the EM algorithm
- **shakeEM_GLM**: Shake-small EM
- **simulate_multinomial_data**: Synthetic data generator
- **splitEM_GLM**: Split-small EM scheme.

See the main function of the package: **multinomialLogitMix**, which wraps automatically calls to the MCMC sampler **gibbs_mala_sampler_ppt** and the EM algorithm **mix_mnm_logistic**.

Author(s)

NA

Maintainer: Panagiotis Papastamoulis <papapast@yahoo.gr>
dealWithLabelSwitching

Post-process the generated MCMC sample in order to undo possible label switching.

Description

This function implements the Equivalence Classes Representatives (ECR) algorithm from the label.switching package in order to undo the label switching phenomenon.

Usage

devalWithLabelSwitching(gs, burn, thin = 10, zPivot = NULL, returnRaw = FALSE, maxM = NULL)

Arguments

gs An object generated by the main function of the package.
burn Number of draws that will be discarader as burn-in.
thin Thinning of the MCMC sample.
zPivot Optional vector of allocations that will be used as the pivot of the ECR algorithm. If this is not supplied, the pivot will be selected as the allocation vector that corresponds to the iteration that maximized the log-likelihood of the model.
returnRaw Boolean. If true, the function will also return the raw output.
maxM Not used.

Details

See Papastamoulis (2016).

References


See Also

multinomialLogitMix, gibbs_mala_sampler_ppt, mix_mnm_logistic
**Value**

- **cluster** Single best clustering of the data, according to the Maximum A Posteriori rule.
- **nClusters_posterior** Estimated posterior distribution of the number of clusters.
- **mcmc** Post-processed mcmc output.
- **posteriorProbabilities** Estimated posterior membership probabilities.

**Author(s)**

Panagiotis Papastamoulis

**References**


---

**expected_complete_LL** *Expected complete LL*

**Description**

This function is not used at the moment.

**Usage**

```r
expected_complete_LL(y, X, b, w, pr)
```

**Arguments**

- **y** count data.
- **X** design matrix.
- **b** Logit coefficients.
- **w** mixing proportions.
- **pr** mixing proportions.

**Value**

Complete log-likelihood of the model.

**Author(s)**

Panagiotis Papastamoulis
**gibbs_mala_sampler**  
*The core of the Hybrid Gibbs/MALA MCMC sampler for the multinomial logit mixture.*

**Description**

This function implements Gibbs sampling to update the mixing proportions and latent allocations variables of the mixture model. The coefficients of the logit model are updated according to Metropolis-Hastings type move, based on a Metropolis adjusted Langevin (MALA) proposal.

**Usage**

```r
gibbs_mala_sampler(y, X, tau = 3e-05, nu2, K, mcmc_iter = 100,  
alpha_prior = NULL, start_values = "EM", em_iter = 10,  
thin = 10, verbose = FALSE, checkAR = NULL,  
probsSave = FALSE, ar_low = 0.4, ar_up = 0.6)
```

**Arguments**

- `y` matrix of counts.
- `X` design matrix (including constant term).
- `tau` the variance of the normal prior distribution of the logit coefficients.
- `nu2` scale of the MALA proposal (positive).
- `K` number of components of the (overfitting) mixture model.
- `mcmc_iter` Number of MCMC iterations.
- `alpha_prior` Parameter of the Dirichlet prior distribution for the mixing proportions.
- `start_values` Optional list of starting values. Random initialization is used if this is not provided.
- `em_iter` Maximum number of iterations if an EM initialization is enabled.
- `thin` optional thinning of the generated MCMC output.
- `verbose` Boolean.
- `checkAR` Number of iterations to adjust the scale of the proposal in MALA mechanism during the initial warm-up phase of the sampler.
- `probsSave` Optional.
- `ar_low` Lowest threshold for the acceptance rate of the MALA proposal (optional).
- `ar_up` Highest threshold for the acceptance rate of the MALA proposal (optional).

**Value**

- `nClusters` sampled values of the number of clusters (non-empty mixture components).
- `allocations` sampled values of the latent allocation variables.
- `logLikelihood` Log-likelihood values per MCMC iteration.
mixing_proportions
  sampled values of mixing proportions.
coefficients
  sampled values of the coefficients of the multinomial logit.
complete_logLikelihood
  Complete log-likelihood values per MCMC iteration.
class_probs
  Classification probabilities per iteration (optional).
AR
  Acceptance rate of the MALA proposal.

Note

This function is used inside the prior tempering scheme, which is the main function.

Author(s)

Panagiotis Papastamoulis

See Also

gibbs_mala_sampler_ppt

Examples

# Generate synthetic data
K <- 2
p <- 2
D <- 2
n <- 2
set.seed(116)
simData <- simulate_multinomial_data(K = K, p = p, D = D, n = n, size = 20, prob = 0.025)

gs <- gibbs_mala_sampler(y = simData$count_data, X = simData$design_matrix,
tau = 0.00035, nu2 = 100, K = 2, mcmc_iter = 3,
alpha_prior = rep(1,K), start_values = "RANDOM",
thin = 1, verbose = FALSE, checkAR = 100)

Prior parallel tempering scheme of hybrid Gibbs/MALA MCMC samplers for the multinomial logit mixture.

Description

The main MCMC scheme of the package. Multiple chains are run in parallel and swaps between are proposed. Each chain uses different parameters on the Dirichlet prior of the mixing proportion. The smaller concentration parameter should correspond to the first chain, which is the one that used for inference. Subsequent chains should have larger values of concentration parameter for the Dirichlet prior.
Usage

\[
gibbs_mala_sampler_ppt(y, X, \tau = 3e-05, \nu2, K, \text{mcmc_cycles = 100, iter_per_cycle = 10, dirPriorAlphas, start_values = “EM”, em_iter = 10, nChains = 4, nCores = 4, warm_up = 100, checkAR = 50, probsSave = FALSE, showGraph = 50, ar_low = 0.4, ar_up = 0.6, withRandom = TRUE)}
\]

Arguments

\text{y} \quad \text{matrix of counts.}

\text{X} \quad \text{design matrix (including constant term).}

\text{tau} \quad \text{the variance of the normal prior distribution of the logit coefficients.}

\text{nu2} \quad \text{scale of the MALA proposal (positive).}

\text{K} \quad \text{number of components of the (overfitting) mixture model.}

\text{mcmc_cycles} \quad \text{Number of MCMC cycles. At the end of each cycle, a swap between chains is attempted.}

\text{iter_per_cycle} \quad \text{Number of iterations per cycle.}

\text{dirPriorAlphas} \quad \text{Vector of concentration parameters for the Dirichlet priors in increasing order.}

\text{start_values} \quad \text{Optional list of start values. Randomly generated values are used if this is not provided.}

\text{em_iter} \quad \text{Maximum number of iterations if an EM initialization is enabled.}

\text{nChains} \quad \text{Total number of parallel chains.}

\text{nCores} \quad \text{Total number of CPU cores for parallel processing of the nChains.}

\text{warm_up} \quad \text{Initial warm-up period of the sampler, in order to adaptively tune the scale of the MALA proposal (optional).}

\text{checkAR} \quad \text{Number of iterations to adjust the scale of the proposal in MALA mechanism during the initial warm-up phase of the sampler.}

\text{probsSave} \quad \text{Optional.}

\text{showGraph} \quad \text{Optional.}

\text{ar_low} \quad \text{Lowest threshold for the acceptance rate of the MALA proposal (optional).}

\text{ar_up} \quad \text{Highest threshold for the acceptance rate of the MALA proposal (optional).}

\text{withRandom} \quad \text{Logical. If TRUE (default) then a random permutation is applied to the supplied starting values.}

Details

See the paper for details.
Value

- `nClusters` sampled values of the number of clusters (non-empty mixture components).
- `allocations` sampled values of the latent allocation variables.
- `logLikelihood` Log-likelihood values per MCMC iteration.
- `mixing_proportions` sampled values of mixing proportions.
- `coefficients` sampled values of the coefficients of the multinomial logit.
- `complete_logLikelihood` Complete log-likelihood values per MCMC iteration.
- `class_probs` Classification probabilities per iteration (optional).
- `AR` Acceptance rate of the MALA proposal.

Note

The output of the MCMC sampler is not identifiable, due to possible label switching. In order to draw meaningful inferences, the output should be post-processed by `dealWithLabelSwitching`.

Author(s)

Panagiotis Papastamoulis

References


Examples

# Generate synthetic data

```r
K <- 2
p <- 2
D <- 3
n <- 2
set.seed(116)
simData <- simulate_multinomial_data(K = K, p = p, D = D, n = n, size = 20, prob = 0.025)
```

# apply mcmc sampler based on random starting values

```r
Kmax = 2
nChains = 2
dirPriorAlphas = c(1, 1 + 5*exp((seq(2, 14, length = nChains - 1)))/100)/(200)
ncores <- 2
mcmc_cycles <- 2
iter_per_cycle = 2
warm_up <- 2

mcmc_random1 <- gibbs_mala_sampler_ppt(y = simData$count_data, X = simData$design_matrix,
```
tau = 0.00035, nu2 = 100, K = Kmax, dirPriorAlphas = dirPriorAlphas,
mcmc_cycles = mcmc_cycles, iter_per_cycle = iter_per_cycle,
start_values = 'RANDOM',
nChains = nChains, nCores = nCores, warm_up = warm_up, showGraph = 1000,
checkAR = 1000)

#sampled values for the number of clusters (non-empty mixture components) per chain (columns)
mcmc_random1$nClusters

---

### log_dirichlet_pdf

**Log-density function of the Dirichlet distribution**

**Description**

Log-density function of the Dirichlet distribution

**Usage**

```r
log_dirichlet_pdf(alpha, weights)
```

**Arguments**

- `alpha`: Parameter vector
- `weights`: Vector of weights.

**Value**

Log-density of the \( D(\alpha_1, \ldots, \alpha_k) \) evaluated at \( w_1, \ldots, w_k \).

**Author(s)**

Panagiotis Papastamoulis

---

### mala_proposal

**Proposal mechanism of the MALA step.**

**Description**

Only the `mala_proposal_cpp` function is used in the package - which is written as an RCPP function.

**Usage**

```r
mala_proposal(y, X, b, z, tau, A = FALSE, pr, nu2)
```
mixLoglikelihood_GLM

**Arguments**

- **y** count data
- **X** design matrix
- **b** coefficients (array
- **z** allocation vector
- **tau** prior variance
- **A** (array
- **pr** mixing proportions
- **nu2** parameter nu2

**Value**

- **theta** theta values
- **b** coefficients
- **acceptance** log-likelihood.
- **gradient** log-likelihood.

**Author(s)**

Panagiotis Papastamoulis

---

**mixLoglikelihood_GLM**  
*Log-likelihood of the multinomial logit.*

**Description**

Log-likelihood of the multinomial logit.

**Usage**

```r
mixLoglikelihood_GLM(y, theta, pi)
```

**Arguments**

- **y** matrix of counts
- **theta** a three-dimensional array containing the multinomial probabilities per cluster, for each observation.
- **pi** a numeric vector of length K (the number of mixture components) containing the mixing proportions.

**Value**

Log-likelihood value.
mix_mnm_logistic

Author(s)

Panagiotis Papastamoulis

Description

Estimation of the multinomial logit mixture using the EM algorithm. The algorithm exploits a careful initialization procedure (Papastamoulis et al., 2016) combined with a ridge-stabilized implementation of the Newton-Raphson method (Goldfeld et al., 1966) in the M-step.

Usage

mix_mnm_logistic(y, X, Kmax = 10, maxIter = 100, emthreshold = 1e-08, maxNR = 5, nCores, tsplit = 8, msplit = 5, split = TRUE, shake = TRUE, random = TRUE, criterion = "ICL", plotting = FALSE, R0 = 0.1, method = 5)

Arguments

y matrix of counts
X design matrix (including constant term).
Kmax Maximum number of mixture components.
maxIter Maximum number of iterations.
emthreshold Minimum loglikelihood difference between successive iterations in order to terminate.
maxNR maximum number of Newton Raphson iterations
nCores number of cores for parallel computations.
tsplit positive integer denoting the number of different runs for each call of the splitting small EM used by split-small EM initialization procedure.
msplit positive integer denoting the number of different runs for each call of the splitting small EM.
split Boolean indicating if the split initialization should be enabled in the small-EM scheme.
shake Boolean indicating if the shake initialization should be enabled in the small-EM scheme.
random Boolean indicating if random initializations should be enabled in the small-EM scheme.
criterion set to "ICL" to select the number of clusters according to the ICL criterion.
plotting Boolean for displaying intermediate graphical output.
R0 controls the step size of the update: smaller values result to larger step sizes. See description in paper.
method this should be set to 5.
Value

- `estimated_K` selected value of the number of clusters.
- `all_runs` detailed output per run.
- `BIC_values` values of bayesian information criterion.
- `ICL_BIC_values` values of ICL-BIC.
- `estimated_clustering` Single best-clustering of the data, according to the MAP rule.

Author(s)
Panagiotis Papastamoulis

References

Examples

```r
# Generate synthetic data
K <- 2
p <- 2
D <- 3
n <- 2
set.seed(116)
simData <- simulate_multinomial_data(K = K, p = p, D = D, n = n, size = 20, prob = 0.025)

SplitShakeSmallEM <- mix_mnm_logistic(y = simData$count_data,
X = simData$design_matrix, Kmax = 2, maxIter = 1,
emthreshold = 1e-8, maxNR = 1, nCores = 2, tsplit = 1,
msplit = 2, split = TRUE, R0 = 0.1, method = 5,
plotting = FALSE)

# selected number of clusters
SplitShakeSmallEM$estimated_K

# estimated single best-clustering, according to MAP rule
SplitShakeSmallEM$estimated_clustering

# detailed output for all parameters of the selected number of clusters
SplitShakeSmallEM$all_runs[[SplitShakeSmallEM$estimated_K]]
```

multinomialLogitMix | Main function

Description
The main function of the package.
Usage

\texttt{multinomialLogitMix(response, design_matrix, method, Kmax = 10, mcmc_parameters = NULL, em_parameters = NULL, nCores, splitSmallEM = TRUE)}

Arguments

- **response**: matrix of counts.
- **design_matrix**: design matrix (including constant term).
- **method**: character with two possible values: "EM" or "MCMC" indicating the desired method in order to estimate the model.
- **Kmax**: number of components of the (overfitting) mixture model.
- **nCores**: Total number of CPU cores for parallel processing.
- **mcmc_parameters**: List with the parameter set-up of the MCMC sampler. See details for changing the defaults.
- **em_parameters**: List with the parameter set-up of the EM algorithm. See details for changing the defaults.
- **splitSmallEM**: Boolean value, indicating whether the split-small EM scheme should be used to initialize the method. Default: true (suggested).

Details

The details of the parameter setup of the EM algorithm and MCMC sampler. The following specification correspond to the minimal default settings. Larger values of tsplit will result to better performance.

\begin{verbatim}
em_parameters <- list(maxIter = 100, emthreshold = 1e-08, maxNR = 10, tsplit = 16, msplit = 10, split = TRUE, R0 = 0.1, plotting = TRUE)
mcmc_parameters <- list(tau = 0.00035, nu2 = 100, mcmc_cycles = 2600, iter_per_cycle = 20, nChains = 8, dirPriorAlphas = c(1, 1 + 5 * exp((seq(2, 14, length = nChains - 1)))/100)/(200), warm_up = 48000, checkAR = 500, probsSave = FALSE, showGraph = 100, ar_low = 0.15, ar_up = 0.25, burn = 100, thin = 1, withRandom = TRUE)
\end{verbatim}

Value

- **EM**: List with the results of the EM algorithm.
- **MCMC_raw**: List with the raw output of the MCMC sampler - not identifiable MCMC output.
  - **MCMC_post_processed**: Post-processed MCMC, used for the inference.

Author(s)

Panagiotis Papastamoulis
Reference


Examples

# Generate synthetic data

K <- 2 #number of clusters
p <- 2 #number of covariates (constant incl)
D <- 5 #number of categories
n <- 20 #generated number of observations
set.seed(1)
simData <- simulate_multinomial_data(K = K, p = p, D = D, n = n, size = 20, prob = 0.025)

# EM parameters
em_parameters <- list(maxIter = 100, emthreshold = 1e-08,
maxNR = 10, tsplit = 16, msplit = 10, split = TRUE,
R0 = 0.1, plotting = TRUE)

# MCMC parameters – just for illustration
# typically, set `mcmc_cycles` and `warm_up` to a larger values
# such as `mcmc_cycles = 2500` or more
# and `warm_up = 40000` or more.
nChains <- 2 #(set this to a larger value, such as 8 or more)
mcmc_parameters <- list(tau = 0.00035, nu2 = 100, mcmc_cycles = 260,
iter_per_cycle = 20, nChains = nChains, dirPriorAlphas = c(1,
1 + 5 * exp((seq(2, 14, length = nChains - 1))/100)/(200),
warm_up = 4800, checkAR = 500, probsSave = FALSE,
showGraph = 100, ar_low = 0.15, ar_up = 0.25, burn = 100,
thin = 1, withRandom = TRUE)

# run EM with split-small-EM initialization, and then use the output to
# initialize MCMC algorithm for an overfitting mixture with
# Kmax = 5 components (max number of clusters – usually this is
# set to a larger value, e.g. 10 or 20).
# Note:
# 1. the MCMC output is based on the non-empty components
# 2. the EM algorithm clustering corresponds to the selected
# number of clusters according to ICL.
# 3. `nCores` should by adjusted according to your available cores.

mlm <- multinomialLogitMix(response = simData$count_data,
design_matrix = simData$design_matrix, method = "MCMC",
Kmax = 5, nCores = 2, splitSmallEM = TRUE,
mcmc_parameters = mcmc_parameters, em_parameters = em_parameters)

# retrieve clustering according to EM
mlm$EM$estimated_clustering

# retrieve clustering according to MCMC
mlm$MCMC_post_processed$cluster
multinomial_logistic_EM

Part of the EM algorithm for multinomial logit mixture

Description

Part of the EM algorithm for multinomial logit mixture

Usage

```
multinomial_logistic_EM(y, x, K, w_start, b_start,
maxIter = 1000, emthreshold = 1e-08, maxNR = 5,
nCores = NULL, verbose = FALSE, R0, method)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>y</td>
</tr>
<tr>
<td>x</td>
<td>X</td>
</tr>
<tr>
<td>K</td>
<td>K</td>
</tr>
<tr>
<td>w_start</td>
<td>w</td>
</tr>
<tr>
<td>b_start</td>
<td>b</td>
</tr>
<tr>
<td>maxIter</td>
<td>max</td>
</tr>
<tr>
<td>emthreshold</td>
<td>em</td>
</tr>
<tr>
<td>maxNR</td>
<td>maxnr</td>
</tr>
<tr>
<td>nCores</td>
<td>nc</td>
</tr>
<tr>
<td>verbose</td>
<td>verb</td>
</tr>
<tr>
<td>R0</td>
<td>or</td>
</tr>
<tr>
<td>method</td>
<td>method</td>
</tr>
</tbody>
</table>

Value

value

Author(s)

Panagiotis Papastamoulis
**myDirichlet**  
*Simulate from the Dirichlet distribution*

**Description**
Generate a random draw from the Dirichlet distribution $D(\alpha_1, \ldots, \alpha_k)$.

**Usage**
```r
myDirichlet(alpha)
```

**Arguments**
- `alpha`: Parameter vector

**Value**
Simulated vector

**Author(s)**
Panagiotis Papastamoulis

---

**newton_raphson_mstep**  
*M-step of the EM algorithm*

**Description**
Implements the maximization step of the EM algorithm based on a ridge-stabilized version of the Newton-Raphson algorithm, see Goldfeld et al. (1966).

**Usage**
```r
newton_raphson_mstep(y, X, b, w, maxNR = 5, R0 = 0.1, method = 5, verbose = FALSE)
```

**Arguments**
- `y`: count data matrix
- `X`: design matrix (including const).
- `b`: coefficients of the multinomial logit mixture
- `w`: mixing proportions
- `maxNR`: threshold
- `R0`: initial value for the parameter that controls the step-size of the update.
- `method`: set to 5. Always.
- `verbose`: Boolean.
Value

- b coefficients
- theta theta values
- ll log-likelihood.

Author(s)

Panagiotis Papastamoulis

References


Description

Assume that there are at least two clusters in the fitted model. We randomly select 2 of them and propose to randomly re-allocate the assigned observations within those 2 clusters.

Usage

```r
shakeEM_GLM(y, x, K, equalModel, tsplit = 10, maxIter = 20, emthreshold = 1e-08, maxNR = 5, nCores, split = TRUE, R0, method)
```

Arguments

- y y
- x X
- K K
- equalModel eq
- tsplit tsplit
- maxIter maxiter
- emthreshold em
- maxNR max
- nCores nc
- split spl
- R0 ro
- method met
simulate_multinomial_data

**Value**

valu

**Author(s)**

Panagiotis Papastamoulis

---

**simulate_multinomial_data**

*Synthetic data generator*

**Description**

This function simulates data from mixture of multinomial logistic regression models.

**Usage**

```
simulate_multinomial_data(K, p, D, n, size = 20, prob = 0.025, betaTrue = NULL)
```

**Arguments**

- **K**: Number of clusters.
- **p**: Number of covariates, including constant.
- **D**: Number of multinomial categories.
- **n**: Number of data points to simulate.
- **size**: Negative Binomial parameter (number of successes). Default: 20.
- **prob**: Negative Binomial parameter (probability of success). Default: 0.025.
- **betaTrue**: An array which contains the true values of the logit coefficients per cluster. Default: randomly generated values.

**Value**

- **count_data**: matrix of data counts.
- **design_matrix**: design matrix.
- **clustering**: Ground-truth partition of the data.

**Author(s)**

Panagiotis Papastamoulis
**splitEM_GLM**

*Split-small EM scheme.*

**Description**

Split two randomly selected clusters based on a model with one component smaller than the current one. This procedure is repeated within a small-EM scheme. The best split is chose to initialize the model.

**Usage**

```r
splitEM_GLM(y, x, K, smallerModel, tsplit = 10, maxIter = 20,
    emthreshold = 1e-08, maxNR = 5, nCores,
    split = TRUE, R0, method)
```

**Arguments**

- `y`  
- `x`  
- `K`  
- `smallerModel`  
- `tsplit`  
- `maxIter`  
- `emthreshold`  
- `maxNR`  
- `nCores`  
- `split`  
- `R0`  
- `method`

**Value**

`val`

**Author(s)**

Panagiotis Papastamoulis

**References**

Index

* package
  multinomialLogitMix-package, 2

dealWithLabelSwitching, 4, 9
expected_complete_LL, 5
gibbs_mala_sampler, 6
gibbs_mala_sampler_ppt, 3, 4, 7, 7
log_dirichlet_pdf, 10
log_mix_prior_derivative
  (mala_proposal), 10
log_prior_mix (mala_proposal), 10
mala_proposal, 10
mala_proposal_cpp (mala_proposal), 10
mix_mmm_logistic, 3, 4, 12
mixLoglikelihood_GLM, 11
multinomial_logistic_EM, 16
multinomialLogitMix, 3, 4, 13
multinomialLogitMix-package, 2
myDirichlet, 17

newton_raphson_mstep, 17

shakeEM_GLM, 18
simulate_multinomial_data, 19
splitEM_GLM, 20