Package 'NestedCategBayesImpute'

November 8, 2016

Type Package

Title Synthetic Household with Structure Zeros

Version 0.1

Date 2016-02-15

Author Quanli Wang, Jerry Reiter, Jingchen Hu and Olanrewaju Akande

Maintainer Quanli Wang <quanli@stat.duke.edu>

Description

The tool set provides a set of functions to fit Dirichlet process mixture models for nested categorical data with structure zeros. It has has direct applications for generate synthetic households data.

License GPL(>=3)

LazyData TRUE

Imports Rcpp (>= 0.12.0)

LinkingTo Rcpp

R topics documented:

`checkconstraints` *Checking a data matrix of households for the possible/impossible status under a predefined set of structural zeros.*

Description

Checking a data matrix of households for the possible/impossible status under a predefined set of structural zeros.

Usage

```
checkconstraints(data, neededpossiblehh, hh_size)
```

Arguments

`data`	A household data matrix generated by calling samplinghouseholds.
`neededpossiblehh`	The number of possible households needed before checking is stopped.
`hh_size`	The household size for the households in data.

Details

Given an input household data matrix, these functions will check the possible/impossible status of each household and also output the desired number of possible and impossible households separately. checkconstraints checks constraints when the household head is included as an individual within the household.

The predefined list of structural zeros currently included is completely data driven. It does not fully represent the true list of impossible combinations in the true population from the American Community Survey (ACS) but rather represents a primary list of combinations not satisfied by our sample of size 10000 from the 2012 ACS. In fact, some of the structural zeros included here do not generalize to more recent ACS data sets. For example, we expect the true population to contain households with same sex couple but do not observe any in our sample and so we leave this in to be able to generate synthetic data with the properties of our sample data.

The structural zeros already included are:

- Valid Head: Each household must contain exactly one head and he/she must be at least 17 years old.
- Valid Spouse: Each household cannot contain more than one spouse and he/she must be at least 17 years old.
- Valid Couple: No same sex couple and age difference between couples cannot exceed 52.
- Valid Children: The household head must be older than the oldest child by at least 12.
- Valid Children-in-Law: The household head must be older than the oldest child-in-law by at least 10.

- Valid Parents: The youngest parent must be older than the household head by at least 13.
- Valid Parents-in-Law: The youngest parent-in-law must be older than the household head by at least 9.
- Valid Siblings: The age difference between the household head and siblings cannot exceed 33.
- Valid Siblings-in-Law: The age difference between the household head and siblings-in-law cannot exceed 33.
- Valid Grandchildren: The household head and spouse must both be at least 34 years old. Also, the household head must be older than the grandchildren by at least 30.

Users can modify this list by downloading the package source, making changes only to the checkconstraints_imp.cpp file and re-building the package.

Value

A list containing information on checking result.

`outcome`	An indicator vector for the possible/impossible household status under constraints.
`Households`	A data matrix for impossible households.
`Index`	A vector for the original indexes of households when possible households are found. Generally not to be used.
`synHouseholds`	A data matrix for possible households.
`possible`	The actual number of possible households returned.

Author(s)

Quanli Wang

---

checkconstraints_HHhead_at_group_level

Checking a data matrix of households for the possible/impossible status under a predefined set of structural zeros.

Description

Checking a data matrix of households for the possible/impossible status under a predefined set of structural zeros.

Usage

```
checkconstraints_HHhead_at_group_level(data, neededpossiblehh, hh_size)
```

Arguments

`data`	A household data matrix generated by calling samplinghouseholds.
`neededpossiblehh`	The number of possible households needed before checking is stopped.
`hh_size`	The household size for the households in data.

Details

Given an input household data matrix, these functions will check the possible/impossible status of each household and also output the desired number of possible and impossible households separately. checkconstraints_HHhead_at_group_level checks contraints when the household head is moved to the household level.

The predefined list of structural zeros currently included is completely data driven. It does not fully represent the true list of impossible combinations in the true population from the American Community Survey (ACS) but rather represents a primary list of combinations not satisfied by our sample of size 10000 from the 2012 ACS. In fact, some of the structural zeros included here do not generalize to more recent ACS data sets. For example, we expect the true population to contain households with same sex couple but do not observe any in our sample and so we leave this in to be able to generate synthetic data with the properties of our sample data.

The structural zeros already included are:

- Valid Head: Each household must contain exactly one head and he/she must be at least 17 years old.
- Valid Spouse: Each household cannot contain more than one spouse and he/she must be at least 17 years old.
- Valid Couple: No same sex couple and age difference between couples cannot exceed 52.
- Valid Children: The household head must be older than the oldest child by at least 12.
- Valid Children-in-Law: The household head must be older than the oldest child-in-law by at least 10.
- Valid Parents: The youngest parent must be older than the household head by at least 13.
- Valid Parents-in-Law: The youngest parent-in-law must be older than the household head by at least 9.
- Valid Siblings: The age difference between the household head and siblings cannot exceed 33.
- Valid Siblings-in-Law: The age difference between the household head and siblings-in-law cannot exceed 33.
- Valid Grandchildren: The household head and spouse must both be at least 34 years old. Also, the household head must be older than the grandchildren by at least 30.

Users can modify this list by downloading the package source, making changes only to the checkconstraints_imp.cpp file and re-building the package.

Value

A list containing information on checking result.

`outcome`	An indicator vector for the possible/impossible household status under constraints.
`Households`	A data matrix for impossible households.
`Index`	A vector for the original indexes of households when possible households are found. Generally not to be used.
`synHouseholds`	A data matrix for possible households.
`possible`	The actual number of possible households returned.

Author(s)

Quanli Wang

GetImpossibleHouseholds *Generate the desired number of impossible households required to observe a given number of possible households.*

Description

Given model parameters, generate the desired number of impossible households required to observe a given number of possible households. Also generate synthetic (and valid) data of the same size as the observed data when required.

Usage

```
GetImpossibleHouseholds(d, n_star_h, lambda, omega, phi, pi, howmany, n, synindex, HHhead_at_group
```

Arguments

d	Vector containing the number of levels for each individual-level variable.
n_star_h	Vector containing the number of observed households for the different household sizes in the original data.
lambda	Multinomial probabilities for each group-level variable.
omega	Latent class probabilities for the group-level and individual-level latent class pairs.
phi	Multinomial probabilities for each individual-level variable by each pair of group-level and individual-level latent classes.
pi	Latent class probabilities for the group-level latent classes.
howmany	Number of households to be generated at a time; batch sampling is used to improve computing speed.
n	Number of households in the original input data and the sum of n_star_h.
synindex	Logical indicator for sampling synthetic data. Set to TRUE when synthetic data is needed.
HHhead_at_group_level	Logical indicator for data structure with respect to the household head. Set to TRUE if the household head has been moved to the household level and FALSE otherwise.

Value

G_Individuals_and_M_extra	A data matrix containing both the group-level (in long format) and individual-level latent classes for the impossible households.
G_extra	A vector containing the group-level latent classes for the impossible households.
IndividualData_extra	A data matrix containing the individual-level data for the impossible households.
HHdata_extra	A data matrix containing the group-level data for the impossible households.
hh_size_new	A vector for the number of impossible households for the different household sizes.
synIndividuals_all	Synthetic data when synindex is TRUE. NULL otherwise.

Author(s)

Quanli Wang

`groupcount` *Generate 2D count table for two integer-valued vectors.*

Description

Similar to 'table' function, this function builts a contingency table of the counts at each combination of all possible values from two integer-valued input vectors.

Usage

```
groupcount(g1, g2, n1, n2)
```

Arguments

g1	The first integer-valued input vector. The max value in g1 is n1.
g2	The second integer-valued input vector. The max value in g1 is n2.
n1	The maximum value in g1.
n2	The maximum value in g2.

Details

This is implemented as an utility function to build a 2D histogram count table. For efficiency, it does not check if the maximum values in input vectors exceed the maximum values specified.

Value

The count table.

Author(s)

Quanli Wang

Examples

```
n1 <- 20
n2 <- 10
g1 <- sample.int(n1,1000, replace = TRUE)
g2 <- sample.int(n2,1000, replace = TRUE)
counts <- groupcount(g1,g2,n1,n2)
```

groupcount1D *Generate histogram count for an integer-valued vector.*

Description

Generate histogram count for an integer-valued vector.

Usage

```
groupcount1D(g, n)
```

Arguments

g	An integer-valued input vector. The max value in g is n.
n	The max value in g.

Details

This is implemented as an utility function for 1D histgram count. For efficiency, it does not check if the maximum value in the input vector exceeds the maximum value specified.

Value

The count values.

Author(s)

Quanli Wang

Examples

```
n <- 20
g <- sample.int(n,1000, replace = TRUE)
counts <- groupcount1D(g,n)
```

household *example dataset to be used for library development.*

Description

This is the example data set that is used in Monica's paper (not exactly the same, but a new random sample of same size).

Usage

```
data("household")
```

Format

A data frame with 10000 observations on the following 9 variables.

`Hhindex` Household index

`pernum` Ask Monica to fill it in

`sex` Gender. 1 = male, 2 = female

`race` Recoded general race code. 1 = white alone, 2 = black alone,3 = American Indian/Alaska Native alone,4 = Asian or Pacific Islander alone,5 = other, 6 = two or more races.

`sthn` Gender. 1 = male, 2 = female

`age` Age. 1=0(lessthenoneyearold),2=1,...,94 = 93

`relate` 1 = head/householder, 2 = spouse, 3 = child, 4 = child-in-law, 5 = parent, 6 = parent-in-law, 7 = sibling, 8 = sibling-in-law,9 = grandchild, 10 = other relatives,11 = partner, friend, visitor,12 = other non-relatives

`ownership` Ownership of dwelling. 1 = owned or being bought (loan), 2 = rented.

`householdsize` Household size. 2=2people,3=3people,4=4people.

Details

Monica please provide the details here about the dataset.

Source

From Monica. Need real sources.

References

Jingchen Hu, Jerome P. Reiter and Quanli Wang. Dirichlet process mixture models for nested categorical data. Vol 00(000). DOI: 0000.

Examples

```
data(household)
```

`households2individuals` *Convert a household data matrix to the corresponding individual member data matrix.*

Description

Convert a household data matrix to the corresponding individual member data matrix.

Usage

```
households2individuals(data, hh_size)
```

Arguments

`data`	Household data matrix.
`hh_size`	The household size for the households in data.

Value

Individual member data matrix.

Author(s)

Quanli Wang

`initData` *Initialize the input data structure.*

Description

Initialize the input data structure.

Usage

```
initData(household, individual_variable_index, household_variable_index)
```

Arguments

`household`	A data matrix for the input household data.
`individual_variable_index`	Vector of column indexes for the individual-level variables.
`household_variable_index`	Vector of column indexes for the group-level variables.

Value

A list object including all the necessary data variables needed by the sampler.

`origdata`	Original data.
`n_i`	Vector containing the number of individuals in each household in the data.
`n`	Number of households in the data
`HHdataorigT`	The transposed household level data – each column now represents each household.
`HHserial`	Vector containing the household index for each individual in the data.
`n_individuals`	The total number of individuals N across all n households in the input data.
`n_individuals_real`	The real total number of individuals N across all n households. The is the same as n_individuals if the household head hasn't been moved to the household level and different otherwise.
`p`	Number of individual-level variables.
`d`	Vector containing the number of levels for each of the p variables.
`dataT`	The transposed individual level data – each column now represents each individual.
`maxd`	The max value in d
`n_star_h`	Vector containing the number of observed households for the different household sizes in the original data.

Author(s)

Quanli Wang

`initOutput` *Set the output structure for saving posterior samples of parameters.*

Description

Set the output structure for saving posterior samples of parameters.

Usage

```
initOutput(data, hyper, mc)
```

Arguments

`data`	Input data (including household info and level of each variable); output of the initData function.
`hyper`	Hyper parameters for priors.
`mc`	MCMC parameters.

Value

A list of output parameters to be saved.

`alphaout`	Vector of posterior samples for the concentration parameter in the Dirichlet process for the group-level latent classes.
`betaout`	Vector of posterior samples for the concentration parameter in the Dirichlet process for the individual-level latent classes. Currently, this is assumed to be the same within all group-level classes.
`piout`	Matrix of posterior samples for the vector of probabilities for the group-level latent classes.
`omegaout`	3D array of posterior samples for the matrix of probabilities for the group-level and individual-level latent class pairs.
`nout`	Vector of posterior samples for the total number of impossible households sampled.
`extrasize`	Matrix of posterior samples for the number of impossible households sampled, split by household size.
`elapsed_time`	Vector of time taken to run each iteration.
`newphiout`	3D array of posterior samples for the individual-level probabilities for each individual-level variable by each pair of group-level and individual-level latent classes.
`lambdaout`	A list of an array of posterior samples for the group-level probabilities for each group-level variable. Each array in the list is for each group-level variable.

Author(s)

Quanli Wang

`initParameters` *Initialize the model parameters for the MCMC.*

Description

Initialize the model parameters for the MCMC.

Usage

```
initParameters(data, hyper, HHhead_at_group_level)
```

Arguments

`data`	The input data to be used in the model.
`hyper`	Hyper parameters for the prior distributions.
`HHhead_at_group_level`	Logical indicator for data structure with respect to the household head. Set to TRUE if the household head has been moved to the household level and FALSE otherwise.

Value

A list of the initial values of the parameters.

`alpha`	Concentration parameter in the Dirichlet process for the group-level latent classes.
`beta`	Concentration parameter in the Dirichlet process for the individual-level latent classes. Currently, this is assumed to be the same within all group-level classes.
`phi`	Matrix of posterior samples for the individual-level probabilities for each individual-level variable by each pair of group-level and individual-level latent classes.
`HHdata_all`	The transposed household level data – each column represents each household.
`lambda`	A list of matrices of the group-level probabilities for each group-level variable by the group-level latent classes. Each matrix in the list is for each group-level variable.
`u`	Vector of the beta-distributed variables in the stick breaking representation of the group-level latent classes.
`pi`	Vector of the probabilities for the group-level latent classes.
`v`	Matrix of the beta-distributed variables in the stick breaking representation of the individual-level latent classes by the group-level latent classes.
`omega`	Matrix of the probabilities for the individual-level latent classes by the group-level latent classes.

Author(s)

Quanli Wang

`sampleG` *Update household (group) level latent class indexes.*

Description

Update household (group) level latent class indexes.

Usage

```
sampleG(phi, data, omega, pi, ni, HHdata, lambda)
```

Arguments

`phi`	Matrix of posterior samples for the individual-level probabilities for each individual-level variable by each pair of group-level and individual-level latent classes.
`data`	Individual level data.
`omega`	Matrix of the probabilities for the individual-level latent classes by the group-level latent classes.
`pi`	Vector of the probabilities for the group-level latent classes.
`ni`	Vector containing the number of individuals in each household in the data..
`HHdata`	Household level data.
`lambda`	A list of matrices of the group-level probabilities for each group-level variable by the group-level latent classes. Each matrix in the list is for each group-level variable.

Details

Function for obtaining a posterior sample of the household-level latent class indexes for all households in the input data based on the corresponding full conditional distribution.

Value

A list with two variables.

`G`	A vector for the updated values of the household-level latent class indexes for all households in the input data.
`G_Individuals`	The vector G expanded to a long format to match the number of individuals in data.

Author(s)

Quanli Wang

`samplehouseholds` *Rcpp implementation for sampling household data without constraints.*

Description

Rcpp implementation for sampling household data without constraints.

Usage

```
samplehouseholds(phi, omega, pi, d, lambda, currrentbatch, nHouseholds, householdsize)
```

Arguments

`phi`	Matrix of posterior samples for the individual-level probabilities for each individual-level variable by each pair of group-level and individual-level latent classes.
`omega`	Matrix of the probabilities for the individual-level latent classes by the group-level latent classes.
`pi`	Vector of the probabilities for the group-level latent classes.
`d`	Vector containing the number of levels for each of the indiviual-level variables.
`lambda`	A list of matrices of the group-level probabilities for each group-level variable by the group-level latent classes. Each matrix in the list is for each group-level variable.
`currrentbatch`	The current batch number for the household data to be generated. The household ID will be generated based on this batch number.
`nHouseholds`	The number of households to be generated by one call to this function.
`householdsize`	The size of the households to be generated.

Details

This function allows the model to generate a batch of nHouseholds with each household of size householdsize. The generated household data will include both possible and impossible households. Use samplehouseholds when the household head is included as an individual within the household.

Value

A data matrix with each row for one household.

Author(s)

Quanli Wang

samplehouseholds_HHhead_at_group_level *Rcpp implementation for sampling household data without constraints.*

Description

Rcpp implementation for sampling household data without constraints.

Usage

```
samplehouseholds_HHhead_at_group_level(phi, omega, pi, d, lambda, currrentbatch, nHouseholds, hou
```

Arguments

phi	Matrix of posterior samples for the individual-level probabilities for each individual-level variable by each pair of group-level and individual-level latent classes.
omega	Matrix of the probabilities for the individual-level latent classes by the group-level latent classes.
pi	Vector of the probabilities for the group-level latent classes.
d	Vector containing the number of levels for each of the indiviual-level variables.
lambda	A list of matrices of the group-level probabilities for each group-level variable by the group-level latent classes. Each matrix in the list is for each group-level variable.
currrentbatch	The current batch number for the household data to be generated. The household ID will be generated based on this batch number.
nHouseholds	The number of households to be generated by one call to this function.
householdsize	The size of the households to be generated.

Details

This function allows the model to generate a batch of nHouseholds with each household of size householdsize. The generated household data will include both possible and impossible households. Use samplehouseholds_HHhead_at_group_level when the household head is moved to the household level.

Value

A data matrix with each row for one household.

Author(s)

Quanli Wang

sampleM *Update individual level latent class indexes.*

Description

Update individual level latent class indexes.

Usage

```
sampleM(phi, data, omega, G, serial)
```

Arguments

phi	Matrix of posterior samples for the individual-level probabilities for each individual-level variable by each pair of group-level and individual-level latent classes.
data	Input individual-level data.
omega	Matrix of the probabilities for the individual-level latent classes by the group-level latent classes.
G	Household-level latent class indexes.
serial	Vector containing the household index for each individual in the data.

Details

Function for obtaining a posterior sample of the individual-level latent class indexes for all individuals in the input data based on the corresponding full conditional distribution.

Value

A vector for the updated values of the individual-level latent class indexes for all individuals in the input data.

Author(s)

Quanli Wang

UpdateAlpha *Update alpha.*

Description

Update alpha – the concentration parameter in the Dirichlet process for the group-level latent classes.

Usage

```
UpdateAlpha(aa, ab, u)
```

Arguments

aa	Hyper-parameter a for alpha.
ab	Hyper-parameter b for alpha.
u	Vector of the beta-distributed variables in the stick breaking representation of the group-level latent classes.

Value

Updated (posterior) value for alpha based on the corresponding full conditional distribution.

Author(s)

Quanli Wang

UpdateBeta *Update beta.*

Description

Update beta – the concentration parameter in the Dirichlet process for the individual-level latent classes. Currently, this is assumed to be the same within all group-level classes.

Usage

```
UpdateBeta(ba, bb, v)
```

Arguments

ba	Hyper-parameter a for beta.
bb	Hyper-parameter b for beta.
v	Matrix of the beta-distributed variables in the stick breaking representation of the individual-level latent classes by the group-level latent classes.

Value

Updated (posterior) value for beta based on the corresponding full conditional distribution..

Author(s)

Quanli Wang

UpdateLambda *Update lambda.*

Description

Update lambda – the list of matrices of the group-level probabilities for each group-level variable by the group-level latent classes when the weighting/capping option is not used. Each matrix in the list is for each group-level variable.

Usage

```
UpdateLambda(dHH, FF, G_all, HHdata_all)
```

Arguments

dHH	A vector containing the number of levels for each household-level variable.
FF	Maximum number of household-level latent classes allowed.
G_all	A vector of the household-level latent class indexes for all households both in the original data and the sampled impossible households.
HHdata_all	Data matrix for the household-level data from both the original data and the sampled impossible households.

Details

Function for obtaining a posterior sample of lambda when the weighting/capping option is not used.

Value

Updated (posterior) value for lambda based on the corresponding full conditional distribution.

Author(s)

Quanli Wang

UpdateLambdaWeighted *Update lambda.*

Description

Update lambda – the list of matrices of the group-level probabilities for each group-level variable by the group-level latent classes – when the weighting/capping option is used. Each matrix in the list is for each group-level variable.

Usage

```
UpdateLambdaWeighted(dHH, FF, G_all, HHdata_all,struc_weight)
```

Arguments

dHH	A vector containing the number of levels for each household-level variable.
FF	Maximum number of household-level latent classes allowed.
G_all	A vector of the household-level latent class indexes for all households both in the original data and the sampled impossible households.
HHdata_all	Data matrix for the household-level data from both the original data and the sampled impossible households.
struc_weight	A vector of weights by household sizes used in capping the number of sampled impossible households.

Details

Function for obtaining a posterior sample of lambda when the weighting/capping option is used.

Value

Updated (posterior) value for lambda based on the corresponding full conditional distribution.

Author(s)

Quanli Wang

UpdateOmega *Update omega and v.*

Description

Update omega – the matrix of the probabilities for the individual-level latent classes by the group-level latent classes – and v – the matrix of the beta-distributed variables in the stick breaking representation of the individual-level latent classes by the group-level latent classes – when the weighting/capping option is not used.

Usage

```
UpdateOmega(beta, M_all, FF, SS)
```

Arguments

beta	Concentration parameter in the Dirichlet process for the individual-level latent classes. Currently, this is assumed to be the same within all group-level classes.
M_all	A vector of both the household-level and individual-level latent class indexes for all households both in the original data and the sampled impossible households.
FF	Maximum number of household-level latent classes allowed.
SS	Maximum number of individual-level latent classes allowed.

Value

A list containing the updated (posterior) values for omega and v based on the corresponding full conditional distributions.

Author(s)

Quanli Wang

`UpdateOmegaWeighted` *Update omega and v.*

Description

Update omega – the matrix of the probabilities for the individual-level latent classes by the group-level latent classes – and v – the matrix of the beta-distributed variables in the stick breaking representation of the individual-level latent classes by the group-level latent classes – when the weighting/capping option is used.

Usage

```
UpdateOmegaWeighted(beta, M_all, FF, SS, struc_weight)
```

Arguments

`beta`	Concentration parameter in the Dirichlet process for the individual-level latent classes. Currently, this is assumed to be the same within all group-level classes.
`M_all`	A vector of both the household-level and individual-level latent class indexes for all households both in the original data and the sampled impossible households.
`FF`	Maximum number of household-level latent classes allowed.
`SS`	Maximum number of individual-level latent classes allowed.
`struc_weight`	A vector of weights by household sizes used in capping the number of sampled impossible households.

Value

A list containing the updated (posterior) values for omega and v based on the corresponding full conditional distributions.

Author(s)

Quanli Wang

`UpdatePhi` *Update phi.*

Description

Update phi – the matrix of posterior samples for the individual-level probabilities for each individual-level variable by each pair of group-level and individual-level latent classes – when the weighting/capping option is not used.

Usage

```
UpdatePhi(IndividualData_all, M_all, FF, SS, p, d, maxd, individual_variable_index)
```

Arguments

`IndividualData_all`	Data matrix for the individual-level data from both the original data and the sampled impossible households.
`M_all`	A vector of both the household-level and individual-level latent class indexes for all households both in the original data and the sampled impossible households.
`FF`	Maximum number of household-level latent classes allowed.
`SS`	Maximum number of individual-level latent classes allowed.
`p`	Number of individual-level variables.
`d`	A vector for the number of levels of each individual-level variable.
`maxd`	Maximum value in d.
`individual_variable_index`	Vector of column indexes for the individual-level variables.

Details

Function for obtaining a posterior sample of phi when the weighting/capping option is not used.

Value

Updated (posterior) value for phi based on the corresponding full conditional distribution.

Author(s)

Quanli Wang

UpdatePhiWeighted *Update phi.*

Description

Update phi – the matrix of posterior samples for the individual-level probabilities for each individual-level variable by each pair of group-level and individual-level latent classes – when the weighting/capping option is used.

Usage

```
UpdatePhiWeighted(IndividualData_all, M_all, FF, SS, p, d, maxd, individual_variable_index, struc
```

Arguments

IndividualData_all	Data matrix for the individual-level data from both the original data and the sampled impossible households.
M_all	A vector of both the household-level and individual-level latent class indexes for all households both in the original data and the sampled impossible households.
FF	Maximum number of household-level latent classes allowed.
SS	Maximum number of individual-level latent classes allowed.
p	Number of individual-level variables.
d	A vector for the number of levels of each individual-level variable.
maxd	Maximum value in d.
individual_variable_index	Vector of column indexes for the individual-level variables.
struc_weight	A vector of weights by household sizes used in capping the number of sampled impossible households.

Details

Function for obtaining a posterior sample of phi when the weighting/capping option is used.

Value

Updated (posterior) value for phi based on the corresponding full conditional distribution.

Author(s)

Quanli Wang

UpdatePi *Update pi and u.*

Description

Update pi – the vector of the probabilities for the group-level latent classes – and u – the vector of the beta-distributed variables in the stick breaking representation of the group-level latent classes – when the weighting/capping option is not used.

Usage

```
UpdatePi(alpha, G_all, FF)
```

Arguments

alpha	Concentration parameter in the Dirichlet process for the group-level latent classes
G_all	A vector of the household-level latent class indexes for all households both in the original data and the sampled impossible households.
FF	Maximum number of household-level latent classes allowed.

Details

Function for obtaining a posterior sample of pi when the weighting/capping option is not used.

Value

A list containing the updated (posterior) values for pi and u based on the corresponding full conditional distributions.

Author(s)

Quanli wang

UpdatePiWeighted *Update pi and u.*

Description

Update pi – the vector of the probabilities for the group-level latent classes – and u – the vector of the beta-distributed variables in the stick breaking representation of the group-level latent classes when the weighting/capping option is used.

Usage

```
UpdatePiWeighted(alpha, G_all, FF, struc_weight)
```

Arguments

alpha	Concentration parameter in the Dirichlet process for the group-level latent classes
G_all	A vector of the household-level latent class indexes for all households both in the original data and the sampled impossible households.
FF	Maximum number of household-level latent classes allowed.
struc_weight	A vector of weights by household sizes used in capping the number of sampled impossible households.

Details

Function for obtaining a posterior sample of pi when the weighting/capping option is used.

Value

A list containing the updated (posterior) values for pi and u based on the corresponding full conditional distributions.

Author(s)

Quanli wang

Index