Package ‘NestedCategBayesImpute’

October 12, 2022

Type Package

Title Modeling, Imputing and Generating Synthetic Versions of Nested Categorical Data in the Presence of Impossible Combinations

Version 1.2.1

Date 2019-02-07

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

Maintainer Olanrewaju Akande <akandelanre13@gmail.com>

Description This tool set provides a set of functions to fit the nested Dirichlet process mixture of products of multinomial distributions (NDPMPM) model for nested categorical household data in the presence of impossible combinations. It has direct applications in imputing missing values for and generating synthetic versions of nested household data.

License GPL (>= 3)

LazyData TRUE

Imports stats, coda, dplyr, Rcpp (>= 0.12.0), RcppParallel

LinkingTo Rcpp, RcppParallel

SystemRequirements GNU make

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-02-07 21:43:40 UTC

R topics documented:

cHECKCONSTRAINS ................................. 2
checkconstraints_HHhead_at_group_level .......................... 4
cHECKSZ ........................................ 5
cHECKSZ2 ....................................... 6
GetImpossibleHouseholds ................................. 7
groupcount ...................................... 8
groupcount1D ................................... 9
households2individuals ............................... 10
initData ........................................ 10
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.
checkconstraints

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 should be viewed as an example of a system of constraints. It was derived by treating a subset of the 2012 American Community Survey as a population, and identifying combinations involving the relationship variable that do not appear in the data. This list should not be interpreted as a “true” list of impossible combinations in the target population. We force the combinations of variables in this list to have zero probability to be consistent with the 2012 ACS public use file that we used in the example.

The structural zeros included are:

- Each household must contain exactly one head and he/she must be at least 16 years old.
- Each household cannot contain more than one spouse and he/she must be at least 16 years old.
- Married couples are of opposite sex, and age difference between individuals in the couples cannot exceed 49.
- The household head must be older than the oldest child by at least 7.
- The youngest parent must be older than the household head by at least 10.
- The youngest parent-in-law must be older than the household head by at least 4.
- The age difference between the household head and siblings cannot exceed 37.
- The household head must be at least 34 years old. Also, the household head must be older than the oldest grandchild by at least 26.

Users can modify the list of structural zeros by downloading the package source, making changes only to the checkconstraints_imp.cpp file and re-building the package. Please note that the structural zeros have been specified according to the structure of our example data so that the specific column indexes and levels of age, gender and relationship to household head variables in subsequent data sets must match those in our example data. For more information on the structure of the data, see the documentation of the RunModel function.

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, Olanrewaju Akande
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, parallel)

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.
parallel Logical indicator for running the function in parallel mode.

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 constraints when the household head is moved to the household level. For the list of structural zeros currently included, see the documentation for checkconstraints.

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, Olanrewaju Akande
checkSZ

Description

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

Usage

checkSZ(Data_to_check, h)

Arguments

Data_to_check  The household data matrix that is to be checked for structure zero constraints.
h         The household size for the households to be checked.

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 should be viewed as an example of a system of constraints. It was derived by treating a subset of the 2012 American Community Survey as a population, and identifying combinations involving the relationship variable that do not appear in the data. This list should not be interpreted as a “true” list of impossible combinations in the target population. We force the combinations of variables in this list to have zero probability to be consistent with the 2012 ACS public use file that we used in the example.

The structural zeros included are:

- Each household must contain exactly one head and he/she must be at least 16 years old.
- Each household cannot contain more than one spouse and he/she must be at least 16 years old.
- Married couples are of opposite sex, and age difference between individuals in the couples cannot exceed 49.
- The household head must be older than the oldest child by at least 7.
- The youngest parent must be older than the household head by at least 10.
- The youngest parent-in-law must be older than the household head by at least 4.
- The age difference between the household head and siblings cannot exceed 37.
- The household head must be at least 34 years old. Also, the household head must be older than the oldest grandchild by at least 26.
Users can modify the list of structural zeros by downloading the package source, making changes only to the checkconstraints_imp.cpp file and re-building the package. Please note that the structural zeros have been specified according to the structure of our example data so that the specific column indexes and levels of age, gender and relationship to household head variables in subsequent data sets must match those in our example data. For more information on the structure of the data, see the documentation of the RunModel function.

**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, Olanrewaju Akande

---

**Description**

Michael: Edit here

**Usage**

checkSZ2(Data_to_check, h)

**Arguments**

- **Data_to_check**: Michael: Edit here
- **h**: Michael: Edit here

**Details**

Michael: Edit here

**Value**

Michael: Edit here
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

```r
GetImpossibleHouseholds(d, n_star_h, lambda, omega, phi, pi, blocksize, n, synindex, HHhead_at_group_level, Parallel)
```

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.
blocksize
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.
Parallel
Logical indicator for running the function in parallel mode.

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

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 histogram 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)
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(md)

Arguments

md: A list holds all the input data with optional missing data info.
**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

---

**initMissing**

*Initialize the misising data structure from input data*

**Description**

Initialize the misising data structure from input data

**Usage**

```r
initMissing(data, struc_zero_variables, miss_batch)
```

**Arguments**

- **data**: A list that holds all input data info.
- **struc_zero_variables**: column indexes for the variables that define structural zeros like age and relate (including those for the household head).
- **miss_batch**: initial number of batches to sample for each household with missing data.
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
A list object including all the necessary data variables needed by the sampler; 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.

F_occupied
Vector of posterior samples for the number of occupied household-level latent classes.

S_occupied_max
Vector of posterior samples for the max number of occupied individual-level latent classes.

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.
**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**: A list object including all the necessary data variables needed by the sampler; output of the `initData` function.
- **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.
RunModel

Author(s)
Quanli Wang

Run the mcmc sampler for the model.

Description
Run the mcmc sampler for the model.

Usage
RunModel(orig,mc,hyper,para,output,synindex,individual_variable_index,
  household_variable_index,HHhead_at_group_level,weight_option,struc_weight,MissData,
  Parallel)

Arguments
orig A list object including all the necessary data variables needed by the sampler.;
output of the initData function.
mc A list specifying the number of mcmc iterations, burn-in, thinning and the effective sample size.
hyper Hyper parameters for the prior distributions.
para A list of the initial values of the parameters; output of the initParameters function.
output A list of output parameters to be saved; output of the initOutput function.
synindex A vector of iteration indexes for sampling synthetic data. length(synindex) is
the number of synthetic data needed.
individual_variable_index Vector of column indexes for the individual-level variables.
household_variable_index Vector of column indexes for the group-level variables.
HHhead_at_group_level Logical indicator for whether or not to move the household head to the household level. Set to TRUE to move the household head and FALSE otherwise.
weight_option Logical indicator for whether or not to cap the number of impossible households to sample and re-weight the multinomial counts within each latent class back to the expected truth. Set to TRUE to use the weighting option nd FALSE otherwise.
struc_weight Vector specifying the weights to be used for each household size. The weights must be ordered by household sizes and no household must be excluded.
MissData A list that stores all the info related to missing data. Default to NULL for no missing data.
Parallel Logical indicator for running the function in parallel mode.
Details

This function runs the mcmc sampler for the NDPMPM model and generates posterior samples of parameters. It also generates synthetic data when needed.

Please note that:

• The minimum household size for this mcmc sampler is 2 because households of size 1 do not violate the structural zeros specified in this package. Also, moving the household head to the household level is not possible for households of size 1.
• Each variable included must be recoded to start from 1.
• Moving the household head to the household level and setting the HHhead_at_group_level option to TRUE speeds up the sampler significantly.
• Setting the weight_option to TRUE and specifying weights also speeds up the sampler but the exact rate of speedup depends on the specific weights.

Our example data set contains a sample of 2000 households and seven variables from the 2012 American Community Survey data. The variables are described below:

• ownership (ownership of dwelling): 1 = owned or being bought (loan), 2 = rented.
• householdsize (household size): 2 = 2 people, 3 = 3 people, 4 = 4 people, 5 = 5 people, 6 = 6 people.
• sex (gender): 1 = male, 2 = female.
• race: 1 = white, 2 = black, 3 = American Indian or Alaska Native, 4 = Chinese, 5 = Japanese, 6 = other Asian/Pacific Islander, 7 = other race, 8 = two major races, 9 = three/more major races.
• hisp (Hispanic origin). 1 = not Hispanic, 2 = Mexican, 3 = Puerto Rican, 4 = Cuban, 5 = other.
• age: 1 = 0 (less then one year old), 2 = 1, 3 = 2, . . . , 94 = 93
• relate (relationship to the household head): 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

Subsequent data sets must follow this structure because of the predefined list of structural zeros or users can modify the list of structural zeros by downloading the package source, making changes only to the checkconstraints_imp.cpp file and re-building the package.

Value

| synData | The list of synthetic data when the length(synindex) > 0. |
| output  | The list of posterior samples for the parameters included in output. |

Author(s)

Quanli Wang, Olanrewaju Akande
sampleG  

_Update household (group) level latent class indexes._

**Description**

Update household (group) level latent class indexes.

**Usage**

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

**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.
- `Parallel`: Logical indicator for running the function in parallel mode.

**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,  
HeadAtGroupLevel, Parallel)
```

**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 individual-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.
- `HeadAtGroupLevel`: Logical indicator for running the model that codes household head at the group level.
- `Parallel`: Logical indicator for running the function in parallel mode.

**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
sampleM

Update individual level latent class indexes.

Description
Update individual level latent class indexes.

Usage
sampleM(phi, data, omega, G, serial, Parallel)

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.
Parallel  Logical indicator for running the function in parallel mode.

Details
Function for obtaining a posterior sample of the individual-level latent class indexes for all individ-
uals 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
SampleMissing  

Sample and update missing data

Description

Sample and update missing data if missing data are presented in the input

Usage

SampleMissing(MissData, para, orig, G_household, M, hyper)

Arguments

- **MissData**: The missing data structure that provides all info related to missing data
- **para**: A list of the initial values of the parameters; output of the initParameters function.
- **orig**: A list object including all the necessary data variables needed by the sampler.
- **G_household**: group level household index
- **M**: individual level latent class indexes
- **hyper**: Hyper parameters for the prior distributions.

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.
**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**

\[ \text{UpdateBeta}(ba, bb, v) \]

**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**

\[ \text{UpdateLambda}(HHdata_all, G_all, dHH, FF) \]

**Author(s)**

Quanli Wang
Arguments

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

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

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. The weighting options allows capping the number of impossible households to sample and re-weight the multinomial counts within each latent class back to the expected truth. Each matrix in the list is for each group-level variable.

Usage

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

Arguments

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

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, Olanrewaju Akande

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. The weighting options allows capping the number of impossible households to sample and re-weight the multinomial counts within each latent class back to the expected truth.

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, Olanrewaju Akande
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(data, M_all, FF, SS, d, maxd)

Arguments

data: 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.
d: A vector for the number of levels of each individual-level variable.
maxd: Maximum value in d.

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
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. The weighting options allows capping the number of impossible households to sample and re-weight the multinomial counts within each latent class back to the expected truth.

Usage

UpdatePhiWeighted(data, M_all, FF, SS, d, maxd, struc_weight)

Arguments

data  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.
d A vector for the number of levels of each individual-level variable.
maxd Maximum value in d.
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, Olanrewaju Akande
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 condi-
tional 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. The weighting options allows capping the number of
impossible households to sample and re-weight the multinomial counts within each latent class back
to the expected truth.
**UpdatePiWeighted**

**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, Olanrewaju Akande
Index

* constraints
  checkconstraints, 2
  checkconstraints_HHhead_at_group_level, 4
  checkSZ, 5
* household data without constraint
  samplehouseholds, 17
* household level
  sampleG, 16
* impossible household
  checkconstraints, 2
  checkconstraints_HHhead_at_group_level, 4
  checkSZ, 5
  GetImpossibleHouseholds, 7
* individual level
  sampleM, 18
* mcmc
  RunModel, 14
* model
  RunModel, 14
* possible household
  checkconstraints, 2
  checkconstraints_HHhead_at_group_level, 4
  checkSZ, 5
* sampler
  GetImpossibleHouseholds, 7
  sampleG, 16
  samplehouseholds, 17
  sampleM, 18
  UpdateAlpha, 19
  UpdateBeta, 20
  UpdateLambda, 20
  UpdateLambdaWeighted, 21
  UpdateOmega, 22
  UpdateOmegaWeighted, 23
  UpdatePhi, 24
  UpdatePhiWeighted, 25
  UpdatePi, 26
  UpdatePiWeighted, 26
* synthetic data
  GetImpossibleHouseholds, 7
* utility function
  households2individuals, 10
  checkconstraints, 2
  checkconstraints_HHhead_at_group_level, 4
  checkSZ, 5
  checkSZ2, 6
  GetImpossibleHouseholds, 7
  groupcount, 8
  groupcount1D, 9
  households2individuals, 10
  initData, 10
  initMissing, 11
  initOutput, 12
  initParameters, 13
  RunModel, 14
  sampleG, 16
  samplehouseholds, 17
  sampleM, 18
  SampleMissing, 19
  UpdateAlpha, 19
  UpdateBeta, 20
  UpdateLambda, 20
  UpdateLambdaWeighted, 21
  UpdateOmega, 22
  UpdateOmegaWeighted, 23
  UpdatePhi, 24
  UpdatePhiWeighted, 25
  UpdatePi, 26
  UpdatePiWeighted, 26