Package: synMicrodata (via r-universe)

May 25, 2026

Type Package

Title Synthetic Microdata Generator

Version 2.1.3

Date 2025-11-26

Maintainer Juhee Lee <ljh988488@gmail.com>

Description This tool fits a non-parametric Bayesian model called a ``hierarchically coupled mixture model with local dependence (HCMM-LD)'' to the original microdata in order to generate synthetic microdata for privacy protection. The non-parametric feature of the adopted model is useful for capturing the joint distribution of the original input data in a highly flexible manner, leading to the generation of synthetic data whose distributional features are similar to that of the input data. The package allows the original input data to have missing values and impute them with the posterior predictive distribution, so no missing values exist in the synthetic data output. The method builds on the work of Murray and Reiter (2016) <doi:10.1080/01621459.2016.1174132>.

License GPL (>= 3)

Imports methods, stats, graphics, utils, Rcpp

LinkingTo Rcpp, RcppArmadillo

RcppModules IO_module

NeedsCompilation yes

Author Juhee Lee [aut, cre], Hang J. Kim [aut], Young-Min Kim [aut], Jared Murray [aut]

Repository https://jhlee-projects.r-universe.dev

Date/Publication 2025-11-26 07:10:02 UTC

RemoteUrl https://github.com/cran/synMicrodata

RemoteRef HEAD

RemoteSha 17cb1879d60fe051f4100cc9bbef4b0053573aeb

Contents

createModel 2
modelobject 2
multipleSyn 3
plot.synMicro_object 4
Rcpp_modelobject-class 5
readData 6
summary.synMicro_object 8

Index **10**

`createModel` *Create a model object*

Description

Create a model object for `multipleSyn`.

Usage

```
createModel(data_obj, max_R_S_K = c(30, 50, 20))
```

Arguments

`data_obj`	data object produced by `readData`
`max_R_S_K`	maximum value of the number of mixture component index (r, s, k).

Value

`createModel` returns a `Rcpp_modelobject`

See Also

`multipleSyn`, `readData`

`modelobject` *RCPP Implementation of the Library*

Description

`Rcpp_modelobject-class`

Value

No return value

multipleSyn *Generate synthetic micro datasets*

Description

Generate synthetic micro datasets using a hierarchically coupled mixture model with local dependence (HCMM-LC).

Usage

```
multipleSyn(data_obj, model_obj, n_burnin, m, interval_btw_Syn, show_iter = TRUE)

## S3 method for class 'synMicro_object'
print(x, ...)
```

Arguments

data_obj	data object produced by readData.
model_obj	model object produced by createModel.
n_burnin	size of burn-in.
m	number of synthetic micro datasets to be generated.
interval_btw_Syn	interval between MCMC iterations for generating synthetic micro datasets.
show_iter	logical value. If TRUE, multipleSyn will print history of (r,s,k) components on console.
x	object of class synMicro_object; a result of a call to multipleSyn().
...	further arguments passed to or from other methods.

Value

multipleSyn returns a list of the following conmponents:

synt_data	list of m synthetic micro datasets.
comp_mat	list of matrices of the mixture component indices.
orig_data	original dataset.

References

Murray, J. S. and Reiter, J. P. (2016). Multiple imputation of missing categorical and continuous values via Bayesian mixture models with local dependence. *Journal of the American Statistical Association*, **111(516)**, pp.1466-1479.

See Also

readData, createModel, plot.synMicro_object

Examples

```
## preparing to generate synthetic datsets
Y_demo <- data.frame(
  Sepal.Length = iris$Sepal.Length,
  Sepal.Width  = iris$Sepal.Width,
  Petal.Length = iris$Petal.Length,
  Petal.Width  = iris$Petal.Width
)

X_demo <- data.frame(
  Species = iris$Species
)

dat_obj <- readData(Y_input = Y_demo, X_input = X_demo)
mod_obj <- createModel(dat_obj, max_R_S_K=c(30,50,20))

## generating synthetic datasets
res_obj <- multipleSyn(dat_obj, mod_obj, n_burnin = 100, m = 5,
                       interval_btw_Syn = 50, show_iter = FALSE)

print(res_obj)
```

---

plot.synMicro_object *Plot Comparing Synthetic Data with Original Input Data*

Description

The `plot` method for `synMicro_object` object. This method compares synthetic datasets with original input data.

Usage

```
## S3 method for class 'synMicro_object'
plot(x, vars, plot_num = NULL, ...)
```

Arguments

`x`	`synMicro_object` object.
`vars`	vector of names or indices of the variables to compare.
`plot_num`	if `plot_num` is a number, returns a plot of the corresponding synthetic datset.
`...`	other parameters to be passed through to plotting functions.

Details

The `plot` takes input variables and draws the graph. The type of graph produced is contingent upon the number of categories in selected variables.

- Putting a continuous variable produces a *box plot* of the selected variable.

- Putting more than two continuous variables produces *pairwise scatter plots* for each pair of selected variables.
- Putting categorical variables produce *bar plot* of each selected variable.

If `plot_num=NULL`, the function output plots for all generated synthetic datasets.

See Also

`multipleSyn`

Examples

```
## preparing to generate synthetic datsets
dat_obj <- readData(Y_input = iris[,1:4],
                    X_input = data.frame(Species = iris[,5]))
mod_obj <- createModel(dat_obj, max_R_S_K=c(30,50,20))

## generating synthetic datasets
res_obj <- multipleSyn(dat_obj, mod_obj, n_burnin = 100, m = 2,
                       interval_btw_Syn = 50, show_iter = FALSE)

print(res_obj)

## plotting synthesis datasets
### box plot
par(mfrow=c(3,2))
plot(res_obj, vars = "Sepal.Length") ## variable names


### pairwise scatter plot
plot(res_obj, vars = c(1,2)) ## or variable index


### bar plot
plot(res_obj, vars = "Species")


### specify the synthetic dattaset
par(mfrow=c(1,1))
plot(res_obj, vars = "Petal.Length", plot_num=1)
```

Rcpp_modelobject-class

Class "Rcpp_modelobject"

Description

This class implements a joint modeling approach to generate synthetic microdata with continuous and categorical variables with possibly missing values. The method builds on the work of Murray and Reiter (2016)

Details

Rcpp_modelobject should be created with `createModel`. Please see the example below.

Extends

Class `"C++Object"`, directly.

Fields

- `data_obj` input dataset generated from `readData`.

Methods

- `multipleSyn` generates synthetic micro datasets.

References

Murray, J. S. and Reiter, J. P. (2016). Multiple imputation of missing categorical and continuous values via Bayesian mixture models with local dependence. *Journal of the American Statistical Association*, **111(516)**, pp.1466-1479.

See Also

`Rcpp`, `C++Object-class`

Examples

```
## preparing to generate synthetic datsets
dat_obj <- readData(Y_input = iris[,1:4],
                    X_input = data.frame(Species = iris[,5]))
mod_obj <- createModel(dat_obj, max_R_S_K=c(30,50,20))

## generating synthetic datasets
res_obj <- multipleSyn(dat_obj, mod_obj, n_burnin = 100, m = 5,
                       interval_btw_Syn = 50, show_iter = FALSE)

print(res_obj)
```

`readData` *Read the original datasets*

Description

Read the original input datasets to be learned for synthetic data generation. The package allows the input data to have missing values and impute them with the posterior predictive distribution, so no missing values exist in the synthetic data output.

Usage

```
readData(Y_input, X_input, RandomSeed = 99)

## S3 method for class 'readData_passed'
print(x, ...)
```

Arguments

Y_input	data.frame consisting of continuous variables of the original data. It must contain only variables of class numeric. Non-numeric columns will cause an error.
X_input	data.frame consisting of categorical variables of the original data. It must contain only variables of class factor (ordered factors are allowed). Character or numeric variables are *not* converted automatically; non-factor columns will cause an error. Convert them to factors in advance, e.g., X_input[] <- lapply(X_input, factor).
RandomSeed	random seed number.
x	object of class readData_passed; a result of a call to readData().
...	further arguments passed to or from other methods.

Value

readData returns an object of "readData_passed" class.

An object of class "readData_passed" is a list containing the following components:

n_sample	number of records in the input dataset.
p_Y	number of continuous variables.
Y_mat_std	matrix with standardized values of Y_input, with mean 0 and standard deviation 1.
mean_Y_input	mean vectors of original Y_input.
sd_Y_input	standard deviation vectors of original Y_input.
NA_Y_mat	matrix indicating missing values in Y_input.
p_X	number of categorical variables.
D_l_vec	numbers of levels of each categorical variable.
X_mat_std	matrix with the numeric-transformed values of X_input.
levels_X_input	list of levels of each categorical variable.
NA_X_mat	matrix indicating missing values in X_input.
var_names	list containing variable names of X_input and Y_input.
orig_data	original dataset.

See Also

multipleSyn, createModel

Examples

```
## Example data: split into continuous (Y_input) and categorical (X_input)

### Continuous variables (numeric only)
Y_demo <- data.frame(
  Sepal.Length = iris$Sepal.Length,
  Sepal.Width  = iris$Sepal.Width,
  Petal.Length = iris$Petal.Length,
  Petal.Width  = iris$Petal.Width
)

### Categorical variables (factor)
X_demo_char <- data.frame(
  Species = as.character(iris$Species)
)

## Not run:
## This will produce an error because X_input is not a factor:
# readData(Y_input = Y_demo, X_input = X_demo_char)
## End(Not run)

## Proper conversion of X_input to factor:
X_demo <- data.frame(
  Species = factor(iris$Species)
)

dat_obj <- readData(Y_input = Y_demo, X_input = X_demo)
print(dat_obj)
```

---

summary.synMicro_object

Summarizing synthesis results

Description

summary method for class "summary.synMicro_object".

Usage

```
## S3 method for class 'synMicro_object'
summary(object, max_print = 4, ...)
```

Arguments

object	synMicro_object object.
max_print	maximum number of synthetic datset to print summaries
...	other parameters to be passed through to other functions.

Details

summary reports the synthesis results for each variable. summary reports the synthesis results for each variable. It compares the summary statistics of each variable for the original dataset(Orig.) and synthetic datasets(synt.#), their averaging(Q_bar), and between variance(B_m).

See Also

multipleSyn

Examples

```
## preparing to generate synthetic datsets
dat_obj <- readData(Y_input = iris[,1:4],
                    X_input = data.frame(Species = iris[,5]))
mod_obj <- createModel(dat_obj, max_R_S_K=c(30,50,20))

## generating synthetic datasets
res_obj <- multipleSyn(dat_obj, mod_obj, n_burnin = 100, m = 2,
                       interval_btw_Syn = 50, show_iter = FALSE)

summary(res_obj)
```

Index

∗ **classes**
Rcpp_modelobject-class, 5

createModel, 2, *3*, *6, 7*

modelobject, 2
multipleSyn, 2, 3, *5*, *7*, *9*

plot.synMicro_object, *3*, 4
print.readData_passed *(*readData*)*, 6
print.synMicro_object *(*multipleSyn*)*, 3

Rcpp, *6*
Rcpp_modelobject, *2*
Rcpp_modelobject
*(*Rcpp_modelobject-class*)*, 5
Rcpp_modelobject-class, 5
readData, *2, 3*, 6, *6*

summary.synMicro_object, 8