Package 'EditImputeCont'

March 3, 2020

Type Package

Title Simultaneous Edit-Imputation for Continuous Microdata

Version 1.1.6

Date 2020-02-08

Author Quanli Wang, Hang J. Kim, Jerome P. Reiter, Lawrence H. Cox and Alan F. Karr

Maintainer Hang J. Kim <hangkim0@gmail.com>

Description An integrated editing and imputation method for continuous microdata under linear constraints is implemented. It relies on a Bayesian nonparametric hierarchical modeling approach as described in Kim et al. (2015) <doi:10.1080/01621459.2015.1040881>. In this approach, the joint distribution of the data is estimated by a flexible joint probability model. The generated edit-imputed data are guaranteed to satisfy all imposed edit rules, whose types include ratio edits, balance edits and range restrictions.

License GPL (>= 3)

Depends Rcpp, methods, editrules, graphics, utils, igraph

LinkingTo Rcpp

RcppModules cbei

URL https://github.com/QuanliWang/EditImputeCont

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-03-03 06:50:02 UTC

R topics documented:

`EditImputeCont-package`

Simultaneous Edit-Imputation for Continuous Microdata Package

---

Description

The package implements an integrated editing and imputation for continuous microdata under linear constraints. It relies on a Bayesian nonparametric hierarchical modeling approach in which the joint distribution of the data is estimated by a flexible joint probability model. The generated edit-imputed data are guaranteed to satisfy all imposed edit rules, whose types include ratio edits, balance edits and range restrictions.

Details

Package:	EditImputeCont
Type:	Package
License:	GPL (>= 3)

Author(s)

Quanli Wang, Hang J. Kim, Jerome P. Reiter, Lawrence H. Cox and Alan F. Karr

Maintainer: Hang J. Kim <hangkim0@gmail.com>

References

Hang J. Kim, Lawrence H. Cox, Alan F. Karr, Jerome P. Reiter and Quanli Wang (2015). "Simultaneous Edit-Imputation for Continuous Microdata", Journal of the American Statistical Association, DOI: 10.1080/01621459.2015.1040881.

See Also

`readData`, `createModel`, `multipleEI`

Examples

```
library(EditImputeCont)

## read the toy example data, which has two ratio edits and a balance edit
data(SimpleEx)
data1 = readData(Y.original=SimpleEx$D.obs, ratio=SimpleEx$Ratio.edit,
range=NULL, balance=SimpleEx$Balance.edit)

## create and initialize the model with 15 DP mixture components
```

```
# model1 = createModel(data.obj=data1, K=15)

## Run an iteration of MCMC

# model1$Iterate()

# dim(model1$Y.edited)
## [1]   1000   4  # Edit-imputed datasets of n=1000 records with p=4 variables

## Please see the example in the demo folder for more detailed explanation
```

`bei` *RCPP Implementation of the Library*

Description

Rcpp_bei-class

`createModel` *Create and Initialize the Rcpp_bei Model Object*

Description

createModel creates and initializes an `Rcpp_bei` object.

Usage

```
createModel(data.obj,K)
```

Arguments

`data.obj`	`EditIn.data` object generated from `readData`.
`K`	maximum number of DP mixture components.

Value

createModel returns an `Rcpp_bei` model-object. The returned model object will be referenced in all subsequent calls.

See Also

`Rcpp_bei`

multipleEI *Generate Multiple Edit-imputed Datasets*

Description

multipleEI returns m multiple edit-imputed datasets.

Usage

```
multipleEI(model.obj, n.burnin, m, int.btw.EI, show.iter=TRUE)
```

Arguments

model.obj	Rcpp_bei model-object generated from createModel.
n.burnin	number of burn in iterations.
m	number of multiple edit-imputed datasets.
int.btw.EI	interval (number of iterations) between EI datasets.
show.iter	logical specifying if the iteration number of burning-in is displayed.

Details

The total number of MCMC iterations is (n.burnin + m * int.btw.EI). Please see the example in the demo folder for more detailed explanation.

Value

array of (m, n, p) dimension where m is the number of edit-imputed data sets, n is the number of records and p is the number of variables.

See Also

createModel

Examples

```
data(SimpleEx)
data1 = readData(SimpleEx$D.obs, SimpleEx$Ratio.edit, NULL, SimpleEx$Balance.edit)
# model1 = createModel(data1, 15)

## get 3 edit-imputed data from MCMC by storing every 100 iterations after 50 burn-in

# result1 = multipleEI(model1, n.burnin=50, m=3, int.btw.EI=100)

# dim(result1)
## [1]   3 1000   4
## m=3 Edit-imputed datasets of n=1000 records with p=4 variables
```

NestedEx *Data Example With 11 Variables*

Description

This data set gives an example consisting of observed data, ratio edits, range restrictions and nested balance edits.

Usage

```
data(NestedEx)
```

Format

A list containing the followings:

- **D.obs**: `data.frame` of observed data with $n = 1000$ records and $p = 11$ variables.
- **Ratio.edit**: `data.frame` of ratio edits.
- **Range.edit**: `data.frame` of range restrictions.
- **Balance.edit**: `data.frame` of balance edits of nested forms, which represent V1=V2+V3+V4, V5=V6+0.4 V10+0.6 V11 and V7=0.4 V10+0.6 V11.

See Also

`readData`

Rcpp_bei-class *Class "Rcpp_bei"*

Description

This class implements the MCMC sampler for a joint modeling approach to multiple edit-imputation for continuous data. It provides methods for updating and monitoring the sampler.

Details

Rcpp_bei objects should be created with `createModel`. Please see the example in the demo folder for more detailed explanation.

Extends

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

Fields

- `Y.input`: input dataset generated from `readData` (replacing NA in `Y.original` by -999 and zero values by 0.01).
- `Y.edited`: current edit-imputed dataset.
- `K`: number of mixture components (latent classes).
- `n.occ`: effective number of mixture components.
- `Prob.A`: ratio of the size of the observed sample to the size of the augmented sample.
- `RandomSeed`: random seed.
- `msg.level`: integer in {0,1,2} specifying the level of displayed message; 0: errors only, 1: errors and warnings, 2: all messages. Defaults to 0.
- `FaultyRecordID`: record IDs of `Y.orig` whose values violate edit rules.

Methods

- `Iterate()`: run a single iteration of MCMC.
- `Run(iter)`: run `iter` iterations of MCMC.

References

Hang J. Kim, Lawrence H. Cox, Alan F. Karr, Jerome P. Reiter and Quanli Wang (2015). "Simultaneous Edit-Imputation for Continuous Microdata", Journal of the American Statistical Association, DOI: 10.1080/01621459.2015.1040881.

Examples

```
data(SimpleEx)

## read the data
data1 = readData(SimpleEx$D.obs, SimpleEx$Ratio.edit, NULL,
  SimpleEx$Balance.edit)

## create and initialize the model
# model1 = createModel(data1, K=15)

### run 10 iterations
# model1$Run(10)
# EI_data1 = model1$Y.edited    # store the edit-imputed dataset
```

`readData` *Read Data and Edit Rules*

Description

add description about this function

Usage

```
readData(Y.original, ratio = NULL, range = NULL, balance = NULL, eps.bal = 0.6)
```

Arguments

`Y.original`	original dataset of (n, p) dimension with missing and edit-failing values where n is the number of records and p is the number of variables.
`ratio`	ratio edit.
`range`	range restriction.
`balance`	balance edit.
`eps.bal`	threshold for balance edit. Defaults to 0.6.

Details

`Y.original` has n records and p variables. The variable names (column names) of `Y.original` are used to specify `ratio` edits.

The edit rules are either imported from text files or written by **editrules** package's syntax.

A balance edit is considered as two inequality constraints with the threshold, i.e., 'A = B' is converted to '-eps.bal < A - B < eps.bal' before computation.

For accurate computation, nested balances are written as 'total variable = sum of component variables'. For example, it is recommended to replace 'X1 = X2 + X3' and 'X3 = X4 + X5' with 'X1 = X2 + X4 + X5' and 'X3 = X4 + X5' so that 'X3' does not appear both sides of the balance edits.

Value

readData returns an `EditIn.data` object which consists of

- `Y.input`: input dataset which replaces NA in `Y.original` with -999 and zero values with 0.01.
- `Edit.editmatrix`: `editmatrix` of edit rules. It can be used for functions of **editrules** package.
- `Edit.matrix`:`matrix` of edit rules.
- `Bound.LU`: range restrictions. For variable X whose range is not specified in `range`, the default values are set as max(0.1min(X), 1e-5) for the lower bound and 10max(x) for the upper bound.
- `ratio`: ratio edits.
- `n.balance`: number of balance edit, i.e., the row number of `balance`.
- `FaultyRecordID`: record IDs of `Y.orig` whose values violate edit rules.

References

Hang J. Kim, Lawrence H. Cox, Alan F. Karr, Jerome P. Reiter and Quanli Wang (2015). "Simultaneous Edit-Imputation for Continuous Microdata", Journal of the American Statistical Association, DOI: 10.1080/01621459.2015.1040881.

Edwin de Jonge and Mark van der Loo (2013). editrules: R package for parsing, applying, and manipulating data cleaning rules. R package version 2.7.2.

See Also

editmatrix

Examples

```
### option 1. import from text files ###

data(NestedEx)

D_obs1 = NestedEx$D.obs
Ratio1 = NestedEx$Ratio.edit
Range1 = NestedEx$Range.edit
Balance1 = NestedEx$Balance.edit

data1 = readData(Y.original=D_obs1, ratio=Ratio1, range=Range1,
balance=Balance1, eps.bal=0.6)

# print(data1$Edit.editmatrix)
# plot(data1$Edit.editmatrix)   ## function of 'editrules' package

### option 2. Using the syntax of R package 'editrules' ###

data(NestedEx) ; D_obs2 = NestedEx$D.obs

Ratio2 <- editmatrix(c(
 "X1 <= 1096.63*X5", "X1 <= 2980.96*X7", "X1 <= 148.41*X8", "X1 <= 7.39*X9",
 "X5 <= 0.37*X1", "X5 <= 54.60*X7", "X5 <= 2.72*X8", "X5 <= 0.14*X9",
 "X7 <= 0.14*X1", "X7 <= 1.65*X5", "X7 <= 7.39*X8", "X7 <= 0.05*X9",
 "X8 <= 1.65*X1", "X8 <= 54.60*X5", "X8 <= 403.43*X7", "X8 <= 1.65*X9",
 "X9 <= 20.09*X1", "X9 <= 403.43*X5", "X9 <= 13359.73*X7", "X9 <= 148.41*X8"
))
Range2 <- editmatrix(c(
 "X1 >= 2", "X2 <= 1.2e+06", "X11 >= 0.002", "X11 <= 1.2e+04"
))
Balance2 <- editmatrix(c(
 "X1 == X2+X3+X4", "X5 == X6 + 0.4*X10 + 0.6*X11", "X7 == 0.4*X10 + 0.6*X11"
))

data2 = readData(D_obs2, Ratio2, Range2, Balance2)

# print(data2$Edit.editmatrix)
  # Note: data2 is equivalent to data1
# plot(data2$Edit.editmatrix)   ## function of 'editrules' package
```

---

scatterPlot *Draw Scatter Plots of Edited Dataset*

Description

Compare scatter plots of the input dataset and the edited dataset.

Usage

```
scatterPlot(model.obj, data.obj=NULL, xvar=NULL, yvar=NULL)
```

Arguments

model.obj	Rcpp_bei model-object generated from createModel.
data.obj	EditIn.data object generated from readData. This is used to draw the lines to indicate ratio edits.
xvar	variable to draw on X-axis. If NULL, plots are drawn for all variables in turn.
yvar	variable to draw on Y-axis. If NULL, plots are drawn for all variables in turn.

Details

Draw the scatter plots of log transformed values of the input data (model.obj$Y.input, left panel) and the edit-imputed data (model.obj$Y.edit, right panel). The sky-blue dots on the background represent edit-passing records which are identical for both datasets. The blue dots on the left panel shows the edit-failing records in the original data and those on the right panel shows their edited values. The red dotted lines (if any) show the ratio edits of two log-transformed variables.

SimpleEx *Simple Data Example With Four Variables*

Description

This data set gives a simple example consisting of observed data, ratio edits and a (simple) balance edit.

Usage

```
data(SimpleEx)
```

Format

A list containing the followings:

- **D.obs**: data.frame of observed data with $n = 1000$ records and $p = 4$ variables.
- **Ratio.edit**: data.frame of two ratio edits.
- **Balance.edit**: data.frame of a balance edit.

See Also

readData

Index