Package ‘BaBooN’

June 15, 2015

Version 0.2-0

Date 2015-06-15

Title Bayesian Bootstrap Predictive Mean Matching - Multiple and Single Imputation for Discrete Data

Author Florian Meinfelder [aut, cre], Thorsten Schnapp [aut]

Maintainer Florian Meinfelder <florian.meinfelder@uni-bamberg.de>

Description Included are two variants of Bayesian Bootstrap Predictive Mean Matching to multiply impute missing data. The first variant is a variable-by-variable imputation combining sequential regression and Predictive Mean Matching (PMM) that has been extended for unordered categorical data. The Bayesian Bootstrap allows for generating approximately proper multiple imputations. The second variant is also based on PMM, but the focus is on imputing several variables at the same time. The suggestion is to use this variant, if the missing-data pattern resembles a data fusion situation, or any other missing-by-design pattern, where several variables have identical missing-data patterns. Both variants can be run as 'single imputation' versions, in case the analysis objective is of a purely descriptive nature.

License GPL (>= 2)

Depends R (>= 3.1.0), Rcpp (>= 0.11.2)

Imports Hmisc, MASS, nnet, coda

Suggests mice, lattice, norm

LinkingTo Rcpp, RcppArmadillo

ByteCompile true

URL http://www.r-project.org

Repository CRAN

Date/Publication 2015-06-15 17:30:31

NeedsCompilation yes

R topics documented:

`BaBooN-package` *Package for multiple imputation of missing values based on Bayesian Bootstrap with Predictive Mean Matching.*

Description

Included are two variants of Bayesian Bootstrap Predictive Mean Matching to multiply impute missing data. The first variant is a variable-by-variable imputation combining sequential regression and Predictive Mean Matching (PMM) that has been extended for unordered categorical data. The Bayesian Bootstrap allows for generating approximately proper multiple imputations. The second variant is also based on PMM, but the focus is on imputing several variables at the same time. The suggestion is to use this variant, if the missing-data pattern resembles a data fusion situation, or any other missing-by-design pattern, where several variables have identical missing-data patterns. Both variants can be run as 'single imputation' versions, in case the analysis objective is of a purely descriptive nature.

Details

Package:	BaBooN
Type:	Package
Version:	0.2-0
Date:	2015-06-15
License:	GPL (>= 2)

Author(s)

Florian Meinfelder [aut, cre] <florian.meinfelder[AT]uni-bamberg.de>
Thorsten Schnapp [aut] <thorsten.schnapp[AT]uni-bamberg.de>
Maintainer: Florian Meinfelder <florian.meinfelder[AT]uni-bamberg.de>

References

Koller-Meinfelder, F. (2009) *Analysis of Incomplete Survey Data – Multiple Imputation Via Bayesian Bootstrap Predictive Mean Matching*, doctoral thesis.

Cowles, M.K. and Carlin, B.P. (1996) Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review. *Journal of the American Statistical Association*, Vol. **91**, pp. 883–904.

Eddelbuettel, D. and Francois, R. (2011) Rcpp: Seamless R and C++ Integration. *Journal of Statistical Software*, Vol. **40**, No. 8, pp. 1–18. URL http://www.jstatsoft.org/v40/i08/.

Eddelbuettel, D. and Sanderson, C. (2014) RcppArmadillo: Accelerating R with high-performance C++ linear algebra. *Computational Statistics and Data Analysis*, Vol. **71**, March 2014, pp. 1054–1063.

Harrell, F.E., with contributions from Charles Dupont and many others. (2013) *Hmisc: Harrell Miscellaneous*. R package version 3.13-0. http://CRAN.R-project.org/package=Hmisc

Little, R.J.A. (1988) Missing-Data Adjustments in Large Surveys, *Journal of Business and Economic Statistics*, Vol. **6**, No. 3, pp. 287-296.

Plummer, M. and Best, N. and Cowles, K. and Vines, K. (2006) CODA: Convergence Diagnosis and Output Analysis for MCMC, *R News*, Vol. **6**, pp. 7–11

Ported to R by Alvaro A. Novo. Original by Joseph L. Schafer <jls@stat.psu.edu>. (2013). *norm: Analysis of multivariate normal datasets with missing values*. R package version 1.0-9.5. http://CRAN.R-project.org/package=norm

R Core Team (2015) *R: A language and environment for statistical computing. R Foundation for Statistical Computing*, Vienna, Austria. URL http://www.R-project.org/.

Raghunathan T.E. and Lepkowski, J.M. and Van Hoewyk, J. and Solenberger, P (2001) A multivariate technique for multiply imputing missing values using a sequence of regression models. *Survey Methodology*, Vol. **27**, pp. 85–95.

Rubin DB (1981) The Bayesian Bootstrap. *The Annals of Statistics*, Vol. **9**, pp. 130–134.

Rubin, D.B. (1987) *Multiple Imputation for Non-Response in Surveys*. New York: John Wiley & Sons, Inc.

Sarkar, Deepayan (2008) *Lattice: Multivariate Data Visualization with R*. New York: Springer.

Van Buuren, S. (2012) *Flexible imputation of missing data*. Boca Raton: CRC Press.

Van Buuren, S. and Brand, J.P.L. and Groothuis-Oudshoorn, C.G.M. and Rubin, D.B. (2006) Fully conditional specification in multivariate imputation. *Journal of Statistical Computation and Simulation*, Vol. **76**, No. 12, pp. 1049–1064.

Van Buuren, S. and Groothuis-Oudshoorn, K. (2011) mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, Vol. **45**, No. 3, pp. 1–67. URL http://www.jstatsoft.org/v45/i03/.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth Edition. New York: Springer.

BBPMM *(Multiple) Imputation through Bayesian Bootstrap Predictive Mean Matching (BBPMM)*

Description

'BBPMM' performs single and multiple imputation (MI) of mixed-scale variables using a chained equations approach and (Bayesian Bootstrap) Predictive Mean Matching.

Usage

```
BBPMM(Data, M=10, nIter=10, outfile=NULL, ignore=NULL,
vartype=NULL, stepmod="stepAIC", maxit.multi=3, maxit.glm=25,
maxPerc = 0.98, verbose=TRUE, setSeed, chainDiagnostics=TRUE, ...)
```

Arguments

`Data`	A partially incomplete data frame or matrix.
`M`	Number of multiple imputations. If `M=1`, no Bayesian Bootstrap step is carried out. *Default*=`10`.
`nIter`	Number of iterations of the chained equations algorithm before the data set is stored as an 'imputed data set'. If set to "autolin", the numbers of iterations will be selected using a data monotonicity index (based on `dmi`). *Default*=`10`.
`outfile`	A character string that specifies the path and file name for the imputed data sets. If `outfile=NULL` (default), no data set is stored
`ignore`	A character or numerical vector that specifies either column positions or variable names that are to be excluded from the imputation model and process, e.g. an ID variable. If `ignore=NULL` (default), all variables in `Data` are used in the imputation model.
`vartype`	A character vector that flags the class of each variable in `Data` (without the variables defined by the `ignore` argument), with either 'M' for metric-scale or 'C' for categorical. The default (NULL) takes over the classes of `Data`. Overruling these classes can sometimes make sense: e.g., an ordinal-scale variable is originally classified as 'factor', but treating it as metric-scale variable within the imputation process might still be a better choice (considering the robust properties of predictive mean matching to model misspecification).
`stepmod`	Performs variable selection for each imputation model based on the either on Schwarz (Bayes) Information criterion (backward). *Default*=`"stepAIC"`.
`maxit.multi`	Imported argument from the **nnet** package that specifies the maximum number of iterations for the multinomial logit model estimation. *Default*=`3`.
`maxit.glm`	Argument for specification of the maximum number of iterations for the binomial logit model estimation (i.e., glm). *Default*=`25`.

`maxPerc`	The maximum percentage the mode category of a variable is allowed to have in order to try 'regular' imputation. If a variable is approximately Dirac distributed, i.e. if it has (almost) no variance, imputation is carried out by simple hot deck imputation. *Default* = `0.98`.
`verbose`	The algorithm prints information on imputation and iteration numbers. *Default*=`TRUE`.
`setSeed`	Optional argument to fix the pseudo-random number generator in order to allow for reproducible results.
`chainDiagnostics`	Argument specifying if Monte Carlo chains for further diagnostics should be returned as well. *Default*=`TRUE`.
`...`	Further arguments passed to or from other functions.

Details

`BBPMM` is based on a chained equations approach that is using a Bayesian Bootstrap approach and Predictive Mean Matching (PMM) variants for metric-scale, binary, and multi-categorical variables to generate multiple imputations. In order to emulate a monotone missing-data pattern as well as possible, variables are sorted by rate of missingness (in ascending order). If no complete variables exist, the least incomplete variable is imputed via hot-deck. The starting solution then builds the imputation model using the observed values of a particular y variable, and the corresponding observed or already imputed values of the *x* variables (i.e., all variables with fewer missing values than *y*). Due to the PMM element in the algorithm, auto-correlation of subsequent iterations is virtually zero. Therefore, a burn-in period is not required, and there is no need to administer 'high' values (> 20) to '`nIter`' either.

If `M=1`, no Bayesian Bootstrap step is carried out for the chained equations. Note that in this case the algorithm is still unlikely to converge to a stable solution, because of the Predictive Mean Matching step.

Value

`call`	The call of `BBPMM`.
`mis.num`	Vector containing the numbers of missing values per column.
`modelselection`	Chosen model selection method for the function call.
`seed`	Chosen seed value for the function call.
`impdata`	The imputed data set, if `M=1`, or a list containing *M* imputed data sets.
`misOverview`	The percentage of missing values per incomplete variable.
`indMatrix`	A matrix with the same dimensions as `Data` minus `ignore` containing flags for missing values.
`M`	Number of (multiple) imputations.
`nIter`	Number of iterations between two imputations.
`Chains`	List containing the the Gibbs sampler sequences for every variable of every imputation for every iteration.
`FirstSeed`	First `.Random.Seed` before imputation starts.

`LastSeed`	Last `.Random.Seed` after function is done.
`ignoredvariables`	`TRUE` / `FALSE` indicator whether variables were ignored during imputation.

Author(s)

Florian Meinfelder, Thorsten Schnapp [ctb]

References

Koller-Meinfelder, F. (2009) *Analysis of Incomplete Survey Data – Multiple Imputation Via Bayesian Bootstrap Predictive Mean Matching*, doctoral thesis.

Little, R.J.A. (1988) Missing-Data Adjustments in Large Surveys, *Journal of Business and Economic Statistics*, Vol. **6**, No. 3, pp. 287-296.

Raghunathan T.E. and Lepkowski, J.M. and Van Hoewyk, J. and Solenberger, P (2001) A multivariate technique for multiply imputing missing values using a sequence of regression models. *Survey Methodology*, Vol. **27**, pp. 85–95.

Rubin DB (1981) The Bayesian Bootstrap. *The Annals of Statistics*, Vol. **9**, pp. 130–134.

Rubin, D.B. (1987) *Multiple Imputation for Non-Response in Surveys*. New York: John Wiley & Sons, Inc.

Van Buuren, S. and Brand, J.P.L. and Groothuis-Oudshoorn, C.G.M. and Rubin, D.B. (2006) Fully conditional specification in multivariate imputation. *Journal of Statistical Computation and Simulation*, Vol. **76**, No. 12, pp. 1049–1064.

Van Buuren, S. and Groothuis-Oudshoorn, K. (2011) mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, Vol. **45**, No. 3, pp. 1–67. URL http://www.jstatsoft.org/v45/i03/.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth Edition. New York: Springer.

See Also

`BBPMM.row`, `dmi`

Examples

```
### sample data set with non-normal variables
set.seed(1000)
n <- 50
x1 <- round(runif(n,0.5,3.5))
x2 <- as.factor(c(rep(1,10),rep(2,25),rep(3,15)))
x3 <- round(rnorm(n,0,3))
y1 <- round(x1-0.25*(x2==2)+0.5*x3+rnorm(n,0,1))
y1 <- ifelse(y1<1,1,y1)
y1 <- as.factor(ifelse(y1>4,5,y1))
y2 <- x1+rnorm(n,0,0.5)
y3 <- round(x3+rnorm(n,0,2))
data1 <- as.data.frame(cbind(x1,x2,x3,y1,y2,y3))
misrow1 <- sample(n,20)
```

```
misrow2 <- sample(n,15)
misrow3 <- sample(n,10)
is.na(data1[misrow1, 4]) <- TRUE
is.na(data1[misrow2, 5]) <- TRUE
is.na(data1[misrow2, 6]) <- TRUE

### imputation
imputed.data <- BBPMM(data1, nIter=5, M=5)
```

`BBPMM.row` *(Multiple) Imputation of variable vectors*

Description

'BBPMM.row' performs single and multiple imputation (MI) of metric scale variable vectors. For MI, parameter draws from a posterior distribution are replaced by a Bayesian Bootstrap step. Imputations are generated using Predictive Mean Matching (PMM) as described in Little (1988).

Usage

```
BBPMM.row(misDataPat, blockImp=length(misDataPat$blocks),
 M=10, outfile=NULL, manWeights=NULL, stepmod="stepAIC", verbose=TRUE,
 tol=0.25, setSeed=NULL, ...)
```

Arguments

`misDataPat`	An object created by `rowimpPrep` that contains information on all identified missing-data patterns.
`blockImp`	A scalar or vector containing the number(s) of the block(s) considered for imputation. Per default only the last block is imputed.
`M`	Number of multiple imputations. If M=1, no Bayesian Bootstrap step is carried out.
`outfile`	A character string that specifies the path and file name for the imputed data sets. If `outfile=NULL` (default), no data set is stored.
`manWeights`	Optional argument containing manual (non-negative) weights for the PMM step. `manWeights` can either be a list containing a vector for each missingness pattern, or just a vector, if only one missingness pattern/block exists. In either case, the number of elements in the vector(s) must match the number of variables in the corresponding block. Note that the higher the weight the higher the importance of a good match for the corresponding variable's predictive means.
`stepmod`	Performs variable selection for each imputation model based on the either on Schwarz (Bayes) Information criterion (backward). *Default*=`"stepAIC"`.
`verbose`	The algorithm prints information on weighting matrices and imputation numbers. *Default*=`TRUE`.
`tol`	Imported argument from function `qr` that specifies the tolerance level for linear dependencies among the complete variables and defaults to 0.25.

`setSeed`	Optional argument to fix the pseudo-random number generator in order to allow for reproducible results.
`...`	Further arguments passed to or from other functions.

Details

The simultaneous imputation of several variables is useful for missing-by-design patterns, such as data fusion or split questionnaire designs. The predictive means of the imputation variables are weighted by the inverse of the covariance matrix of the residuals from the regression of these variables on the complete variables. The intuitive idea behind is that distances between predictive means should be punished more severely, if the particular variable can be explained well by the (completely observed) imputation model variables. Through partialization and subsequent usage of the residuals the weight matrix is transformed into a diagonal matrix. The calculated weights can be adjusted by manual weights. Since the weight matrix is a Mahalanobis type of distance matrix, the weights are in the denominator and therefore the lower the weight, the higher the influence. As this is somewhat counterintuitive, the reciprocal of the manual weights is taken. Therefore, the higher the manual weight the higher in the influence of the corresponding variable's predictor on the overall distance. To identify the missing-by-design patterns it is necessary to transform the raw data set via `rowimpPrep`. The donor/recipient ID pairlist for each imputation and identified pattern ('block') is stored. In general, `weightMatrix`, `model` and `pairlist` are list objects named 'M1' to 'M<M>', and each in return is a list object named 'block1' to 'block<length(blockImp)>'. `model` contains another list object with `lm`-objects for all variables in a particular block. Unlike `BBPMM` this algorithm is not based on sequential regression. Therefore, imputed variables are conditionally independent given the completely observed variables (of which at least one must exist).

Value

`call`	The call of `BBPMM.row`
`mis.num`	Vector containing the numbers of missing values per column.
`modelselection`	Chosen model selection method for the function call.
`seed`	Chosen seed value for the function call.
`impdata`	A list containing *M* completed data sets.
`weightMatrix`	A list containing weight matrices for all imputations and blocks.
`model`	A list containing the lm-objects for all imputations and blocks.
`pairlist`	A list containing the donor/recipient pairlist data frames for all imputations and blocks.
`indMatrix`	A matrix with the same dimensions as the incomplete data containing flags for missing values.
`FirstSeed`	First `.Random.Seed` before imputation starts.
`LastSeed`	Last `.Random.Seed` after function is done.
`ignoredvariables`	`TRUE` / `FALSE` indicator whether variables were ignored during imputation.

Author(s)

Florian Meinfelder, Thorsten Schnapp [ctb]

References

Eddelbuettel, D. and Francois, R. (2011) Rcpp: Seamless R and C++ Integration. *Journal of Statistical Software*, Vol. **40**, No. 8, pp. 1–18. URL http://www.jstatsoft.org/v40/i08/.

Eddelbuettel, D. and Sanderson, C. (2014) RcppArmadillo: Accelerating R with high-performance C++ linear algebra. *Computational Statistics and Data Analysis*, Vol. **71**, March 2014, pp. 1054–1063.

Koller-Meinfelder, F. (2009) *Analysis of Incomplete Survey Data – Multiple Imputation Via Bayesian Bootstrap Predictive Mean Matching*, doctoral thesis.

Little, R.J.A. (1988) Missing-Data Adjustments in Large Surveys, *Journal of Business and Economic Statistics*, Vol. **6**, No. 3, pp. 287-296.

Raghunathan T.E. and Lepkowski, J.M. and Van Hoewyk, J. and Solenberger, P (2001) A multivariate technique for multiply imputing missing values using a sequence of regression models. *Survey Methodology*, Vol. **27**, pp. 85–95.

Rubin DB (1981) The Bayesian Bootstrap. *The Annals of Statistics*, Vol. **9**, pp. 130–134.

Rubin, D.B. (1987) *Multiple Imputation for Non-Response in Surveys*. New York: John Wiley & Sons, Inc.

Van Buuren, S. and Brand, J.P.L. and Groothuis-Oudshoorn, C.G.M. and Rubin, D.B. (2006) Fully conditional specification in multivariate imputation. *Journal of Statistical Computation and Simulation*, Vol. **76**, No. 12, pp. 1049–1064.

Van Buuren, S. and Groothuis-Oudshoorn, K. (2011) mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, Vol. **45**, No. 3, pp. 1–67. URL http://www.jstatsoft.org/v45/i03/.

Venables, W. N. and Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth Edition. New York: Springer.

See Also

`rowimpPrep`, `BBPMM`

Examples

```
### sample data set with non-normal variables and a single
### missingness pattern
set.seed(1000)
n <- 50
x1 <- round(runif(n,0.5,3.5))
x2 <- as.factor(c(rep(1,10),rep(2,25),rep(3,15)))
x3 <- round(rnorm(n,0,3))
y1 <- round(x1-0.25*(x2==2)+0.5*x3+rnorm(n,0,1))
y1 <- ifelse(y1<1,1,y1)
y1 <- ifelse(y1>4,5,y1)
y2 <- y1+rnorm(n,0,0.5)
y3 <- round(x3+rnorm(n,0,2))
data <- as.data.frame(cbind(x1,x2,x3,y1,y2,y3))
misrow1 <- sample(n,20)
data[misrow1, c(4:6)] <- NA
```

```
### preparation step
impblock <- rowimpPrep(data)

### imputation
imputed.data <- BBPMM.row(impblock, M=5)
```

---

dmi *Data monotonicity index for missing values*

Description

'dmi' calculates a monotonicity index for data with missing values.

Usage

```
dmi(Data)
```

Arguments

`Data`	A data frame containing missing values.

Details

The *data monotonicity index* examines the ratio of missing values with non-monotonicity and complete monotonicity in all variables. To denote full monotonicity with 1 and no monotonicity with 0 this ratio is subtracted from 1.

$$dmi = 1 - \frac{\sum_{j=1}^{p} \sum_{i=1}^{n-\sum_{h=1}^{n} I(r_{hj}=0)} \sum_{h=1}^{n} I(r_{hi} = 0)}{\sum_{h=1}^{n} \sum_{j=1}^{p} I(r_{hj} = 0)}$$

Value

Returns a value between 1 (fully monotone) and 0 (no monotonicity).

Author(s)

Florian Meinfelder, Thorsten Schnapp

References

Harrell, F.E., with contributions from Charles Dupont and many others. (2013) *Hmisc: Harrell Miscellaneous*. R package version 3.13-0. http://CRAN.R-project.org/package=Hmisc

Venables, W. N. & Ripley, B. D. (2002) *Modern Applied Statistics with S*. Fourth Edition. Springer, New York. ISBN 0-387-95457-0

Ported to R by Alvaro A. Novo. Original by Joseph L. Schafer <jls@stat.psu.edu>. (2013). *norm: Analysis of multivariate normal datasets with missing values*. R package version 1.0-9.5. http://CRAN.R-project.org/package=norm

See Also

BBPMM, prelim.norm

Examples

```
if(!require(MASS)) install.packages("MASS")
library(MASS)  ## see references
data(survey)

## Sorting via 'norm's prelim.norm
if(!require(Hmisc)) install.packages("Hmisc")
library(Hmisc) ## see references
survey.numeric <- asNumericMatrix(survey)

if(!require(norm)) install.packages("norm")
library(norm) ## see references
su.sort    <- prelim.norm(survey.numeric)
new.survey <- survey[order(su.sort$ro),
                     sort(su.sort$nmis,index.return=TRUE)$ix]

## Comparison
dmi(survey)     # original
dmi(new.survey) # sorted
```

---

impdiagnosticconversion

Conversion from BBPMM output to mice's mids object or prepares imputed data for coda's mcmc or mcmc.list objects

---

Description

'impdiagnosticconversion' prepares the output of BBPMM for **coda**'s (Plummer et al. 2006) mcmc or mcmc.list object. It is also possible to convert the output into an object of the type mids provided by the package **mice** (van Buuren, Groothuis-Oudshoorn 2011).

Usage

```
impdiagnosticconversion(imputed.data, type=c("mcmc.list","mcmc","mids"))
```

Arguments

imputed.data	An object of the type 'imp' returned by **BaBooN**'s BBPMM imputation method.
type	A string choosing the type of output: Either "mcmc.list", "mcmc" or "mids". *Default*="mcmc.list".

Details

To provide a wider variety of analysis and diagnostic tools and facilitate the collaboration between different imputation packages, we started to create a conversion tool. At the moment it is possible to transform an output created by **BaBooN**'s `BBPMM` function into a mids-like object returned from `mice` (van Buuren, Groothuis-Oudshoorn 2011). For diagnostics with the help of the functions provided by the package **coda** (Plummer et al. 2006), this function prepares the output. It returns a list, that isn't directly of the desired class, but results in a list with four elements, containing the chains for the means, variances, medians and standard deviations of the imputed values transformed to the specified type respectively. Hence, accessing the first element of the list returns actual the converted means, the second element contains the converted variances and so on.

We hope it is somehow an impulse for future standardisation or interchangeability of outputs generated by different multiple imputation routines and analysing the imputed values.

Value

See for details `mcmc`, `mcmc.list` or `mids`. Remark: Converting to `mcmc` or `mcmc.list` results in a list with four elements (medians, vars, medians, sds), containing the chains for the means, variances, medians and standard deviations of the imputed values transformed to the specified type. See the examples.

Author(s)

Thorsten Schnapp, Florian Meinfelder [ctb]

References

Cowles, M.K. and Carlin, B.P. (1996) Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review. *Journal of the American Statistical Association*, Vol. **91**, pp. 883–904.

Koller-Meinfelder, F. (2009) *Analysis of Incomplete Survey Data – Multiple Imputation Via Bayesian Bootstrap Predictive Mean Matching*, doctoral thesis.

Plummer, M. and Best, N. and Cowles, K. and Vines, K. (2006) CODA: Convergence Diagnosis and Output Analysis for MCMC, *R News*, Vol. **6**, pp. 7–11

Van Buuren, S. and Groothuis-Oudshoorn, K. (2011) mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, Vol. **45**, No. 3, pp. 1–67. URL http://www.jstatsoft.org/v45/i03/.

Van Buuren, S. (2012) *Flexible imputation of missing data*. Boca Raton: CRC Press.

Sarkar, Deepayan (2008) *Lattice: Multivariate Data Visualization with R*. New York: Springer.

See Also

`mcmc`, `mcmc.list`, `mids`, `mids-class`, `mice`, `plot.mids`, `BBPMM`

Examples

```
## Not run:
### sample data set with non-normal variables
set.seed(1000)
n <- 50
```

```
x1 <- round(runif(n,0.5,3.5))
x2 <- as.factor(c(rep(1,10),rep(2,25),rep(3,15)))
x3 <- round(rnorm(n,0,3))
y1 <- round(x1-0.25*(x2==2)+0.5*x3+rnorm(n,0,1))
y1 <- ifelse(y1<1,1,y1)
y1 <- as.factor(ifelse(y1>4,5,y1))
y2 <- x1+rnorm(n,0,0.5)
y3 <- round(x3+rnorm(n,0,2))
data1 <- as.data.frame(cbind(x1,x2,x3,y1,y2,y3))
misrow1 <- sample(n,20)
misrow2 <- sample(n,15)
misrow3 <- sample(n,10)
is.na(data1[misrow1, 4]) <- TRUE
is.na(data1[misrow2, 5]) <- TRUE
is.na(data1[misrow2, 6]) <- TRUE

### imputation
imputed.data <- BBPMM(data1, nIter=3, M=3)

### Test Conversion
if(!require(coda)) install.packages("coda")
if(!require(mice)) install.packages("mice")

require(coda) ## see references
require(mice) ## see references
require(lattice) ## see references

## conversion to mcmc
imp.to.mcmc <- impdiagnosticconversion(imputed.data,
                                       type="mcmc")

## conversion to mcmc.list
imp.to.mcmc.list <- impdiagnosticconversion(imputed.data,
                                            type="mcmc.list")

## conversion to mids
imp.to.mids <- impdiagnosticconversion(imputed.data,
                                       type="mids")

### Test

## mcmc:
plot(imp.to.mcmc$means[[1]])
acfplot(imp.to.mcmc$vars[[1]])
plot(imp.to.mcmc$medians[[1]])
acfplot(imp.to.mcmc$sds[[1]])

## mcmc.list:
xyplot(imp.to.mcmc.list[[1]]) ## Mean
qqmath(imp.to.mcmc.list[[2]]) ## Variance
xyplot(imp.to.mcmc.list[[3]]) ## Median
qqmath(imp.to.mcmc.list[[4]]) ## Std.dev.
```

```
## mids:
# Chain-plot from mice
mice:::plot.mids(imp.to.mids)


## End(Not run)
```

MI.inference *Multiple Imputation inference*

Description

'MI.inference' applies Rubin's combining rules to estimated quantities of interest that are based on multiply imputed data sets. The function requires as input two vectors of length M for the estimate and its variance.

Usage

```
MI.inference(thetahat, varhat.thetahat, alpha=0.05)
```

Arguments

`thetahat`	A vector of length M containing estimates of the quantity of interest based on multiply imputed data sets.
`varhat.thetahat`	A vector of length M containing the corresponding variances of `thetahat`.
`alpha`	The significance level at which lower and upper bound are calculated. DEFAULT=0.05

Details

Multiple Imputation (Rubin, 1987) of missing data is a generally accepted way to get correct variance estimates for a particular quantity of interest in the presence of missing data. `MI.inference` estimates the *within variance* W and *between variance* B, and combines them to the *total variance* T. Based on the output, further analysis figures, such as the *fraction of missing information* can be calculated.

Value

`MI.Est`	A scalar containing the MI estimate of the quantity of interest (i.e. an estimator averaged over all M data sets).
`MI.Var`	The Multiple Imputation variance.
`CI.low`	The lower bound of the MI confidence interval.
`CI.up`	The upper bound of the MI confidence interval.
`BVar`	The estimated *between variance*.
`WVar`	The estimated *within variance*.

References

Rubin, D.B. (1987) *Multiple Imputation for Non-Response in Surveys*. New York: John Wiley & Sons, Inc.

Examples

```
## Not run:
### example 1
n <- 100
x1 <- round(runif(n,0.5,3.5))
x2 <- round(runif(n,0.5,4.5))
x3 <- runif(n,1,6)
y1 <- round(x1-0.25*x2+0.5*x3+rnorm(n,0,1))
y1 <- ifelse(y1<2,2,y1)
y1 <- as.factor(ifelse(y1>4,5,y1))
y2 <- x3+rnorm(n,0,2)
y3 <- as.factor(ifelse(x2+rnorm(n,0,2)>2,1,0))
mis1 <- sample(100,20)
mis2 <- sample(100,30)
mis3 <- sample(100,25)
data1 <- data.frame("x1"=x1,"x2"=x2,"x3"=x3,
                    "y1"=y1,"y2"=y2,"y3"=y3)
is.na(data1$y1[mis1]) <- TRUE
is.na(data1$y2[mis2]) <- TRUE
is.na(data1$y3[mis3]) <- TRUE
imputed.data <- BBPMM(data1, M=5, nIter=5)

MI.m.meany2.hat <- sapply(imputed.data$impdata,
                          FUN=function(x) mean(x$y2))

MI.v.meany2.hat <- sapply(imputed.data$impdata,
                          FUN=function(x) var(x$y2)/length(x$y2))

### MI inference
MI.y2 <- MI.inference(MI.m.meany2.hat,
                      MI.v.meany2.hat, alpha=0.05)

MI.y2$MI.Est
MI.y2$MI.Var


#################################################################
### example 2: a small simulation example

### simple additional function to calculate coverages:         #

coverage <- function(value, bounds) {
  ifelse(min(bounds) <= value && max(bounds) >= value, 1, 0)
}
### value            : true value                              #
### bounds           : vector with two elements (upper and     #
###                    lower bound of the CI)                  #
### sample size
n <- 100
### true value for the mean of y2
m.y2 <- 3.5
y2.cover <- vector(length=n)
set.seed(1000)

### 100 data generations
time1 <- Sys.time()
for (i in 1:100) {
  x1 <- round(runif(n,0.5,3.5))
  x2 <- round(runif(n,0.5,4.5))
  x3 <- runif(n,1,6)
  y1 <- round(x1-0.25*x2+0.5*x3+rnorm(n,0,1))
  y1 <- ifelse(y1<2,2,y1)
  y1 <- as.factor(ifelse(y1>4,5,y1))
  y2 <- x3+rnorm(n,0,2)
  y3 <- as.factor(ifelse(x2+rnorm(n,0,2)>2,1,0))
  mis1 <- sample(n,20)
  mis2 <- sample(n,30)
  mis3 <- sample(n,25)
  data1 <- data.frame("x1"=x1,"x2"=x2,"x3"=x3,
                      "y1"=y1,"y2"=y2,"y3"=y3)
  is.na(data1$y1[mis1]) <- TRUE
  is.na(data1$y2[mis2]) <- TRUE
  is.na(data1$y3[mis3]) <- TRUE

  sim.imp <- BBPMM(data1, M=3, nIter=2,
                   stepmod="", verbose=FALSE)

  MI.m.meany2.hat <- sapply(sim.imp$impdata,
                            FUN=function(x) mean(x$y2))

  MI.v.meany2.hat <- sapply(sim.imp$impdata,
                            FUN=function(x)
                            var(x$y2)/length(x$y2))
### MI inference
  MI.y2 <- MI.inference(MI.m.meany2.hat, MI.v.meany2.hat,
                        alpha=0.05)

  y2.cover[i] <- coverage(m.y2, c(MI.y2$CI.low,MI.y2$CI.up))
}
time2 <- Sys.time()
difftime(time2, time1, unit="secs")

### coverage estimator (alpha=0.05):
mean(y2.cover)


## End(Not run)
```

`rowimpPrep` *Missing-data pattern identifier*

Description

'rowimpPrep' identifies all missingness patterns within an incomplete data set. Running `rowimpPrep` is a prerequisite for `BBPMM.row`.

Usage

```
rowimpPrep(data, ID=NULL, verbose=TRUE)
```

Arguments

`data`	Either a data frame or matrix with missing values.
`ID`	A numeric or character string vector indicating the column positions or names of the ID variable (if two data sets were stacked that have a joint subset of variables). The first element refers to the 'donor ID', the second element refers to the 'recipient ID'. This distinction is only of relevance, if the data set is 'L-shaped', i.e. if the data contains only one missing-data pattern (where incomplete cases are 'recipients'). If `ID` has only one element, The function assumes that the identifier variables of the two data sets are packed into a single variable. Default=NULL is used, if no ID variable is specified.
`verbose`	Prints information on identified missing-data patterns. Default=TRUE.

Details

`rowimpPrep` identifies all patterns, and allows to decide, whether to impute all missing-data patterns with `BBPMM.row`or just some of them. This comes in handy if variables that were assumed to be completely observed have missing values. These variables are then likely to define an unexpected 'block' of their own. Of course, `BBPMM.row` can be used to impute missing data that are not missing-by-design as well, but `BBPMM` would probably be the better option. Note that all variables listed in `compNames` are used for the imputation model in `BBPMM.row`, i.e. completely observed variables (ID variables aside) which are not to be used in the imputation model, have to be removed from the data set beforehand.

Value

`data`	The original data set minus the ID variable(s).
`key`	The ID variable(s) from the original data set.
`blocks`	A list containing the column positions of all identified missing-data patterns.
`blockNames`	A list containing the variable names corresponding to object `blocks`.
`compNames`	A character vector containing the variable names of the (completely observed) imputation model variables.
`ignore`	Contains positions of ignored variables.

`ignored_data`	Contains ignored variables.
`indMatrix`	A matrix with the same dimensions as the incomplete data containing flags for missing values.

Author(s)

Florian Meinfelder, Thorsten Schnapp [ctb]

Examples

```
### sample data set with non-normal variables and a single
### missingness pattern
set.seed(1000)
n <- 50
x1 <- round(runif(n,0.5,3.5))
x2 <- as.factor(c(rep(1,10),rep(2,25),rep(3,15)))
x3 <- round(rnorm(n,0,3))
y1 <- round(x1-0.25*(x2==2)+0.5*x3+rnorm(n,0,1))
y1 <- ifelse(y1<1,1,y1)
y1 <- ifelse(y1>4,5,y1)
y2 <- y1+rnorm(n,0,0.5)
y3 <- round(x3+rnorm(n,0,2))
data1 <- as.data.frame(cbind(x1,x2,x3,y1,y2,y3))
misrow1 <- sample(n,20)
is.na(data1[misrow1, c(4:6)]) <- TRUE

### preparation step
impblock <- rowimpPrep(data1)

impblock$blockNames
```

`summary.imp` *Summary method for objects of class 'imp'*

Description

Returns some information about the incomplete data set and the imputation process.

Usage

```
## S3 method for class 'imp'
summary(object,...)
```

Arguments

`object`	Either with `BBPMM` or `BBPMM.row` generated object.
`...`	Arguments to be passed to or from other functions.

Details

Returns information about the percentage of missing data as well as about the imputation variant, the number of (multiple) imputations and the number of iterations between two imputations.

Author(s)

Florian Meinfelder

See Also

`BBPMM`, `BBPMM.row`

Examples

```
### sample data set with non-normal variables and two different
### missingness patterns
n <- 50
x1 <- round(runif(n,0.5,3.5))
x2 <- as.factor(c(rep(1,10),rep(2,25),rep(3,15)))
x3 <- round(rnorm(n,0,3))
y1 <- round(x1-0.25*(x2==2)+0.5*x3+rnorm(n,0,1))
y1 <- ifelse(y1<1,1,y1)
y1 <- as.factor(ifelse(y1>4,5,y1))
y2 <- x1+rnorm(n,0,0.5)
y3 <- round(x3+rnorm(n,0,2))
data1 <- as.data.frame(cbind(x1,x2,x3,y1,y2,y3))
misrow1 <- sample(n,20)
misrow2 <- sample(n,15)
misrow3 <- sample(n,10)
is.na(data1[misrow1, 4]) <- TRUE
is.na(data1[misrow2, 5]) <- TRUE
is.na(data1[misrow2, 6]) <- TRUE

### imputation
imputed.data <- BBPMM(data1, nIter=5, M=5)
summary(imputed.data)
```

`summary.impprep` *Summary method for objects of class 'impprep'*

Description

Returns an overview of missing-data patterns, in particular of missing-by-design patterns.

Usage

```
## S3 method for class 'impprep'
summary(object, nNames=10L,...)
```

Arguments

`object`	An object generated by `rowimpPrep`.
`nNames`	Number of variable names per block to be printed (Default = 10).
`...`	Arguments to be passed to or from other functions.

Details

Returns the number of identified missing-data patterns, the first `nNames` variable names per block and the names of the completely observed variables.

Author(s)

Florian Meinfelder

See Also

`rowimpPrep`

Examples

```
### sample data set with non-normal variables and a single
### missingness pattern
set.seed(1000)
n <- 50
x1 <- round(runif(n,0.5,3.5))
x2 <- as.factor(c(rep(1,10),rep(2,25),rep(3,15)))
x3 <- round(rnorm(n,0,3))
y1 <- round(x1-0.25*(x2==2)+0.5*x3+rnorm(n,0,1))
y1 <- ifelse(y1<1,1,y1)
y1 <- ifelse(y1>4,5,y1)
y2 <- y1+rnorm(n,0,0.5)
y3 <- round(x3+rnorm(n,0,2))
data1 <- as.data.frame(cbind(x1,x2,x3,y1,y2,y3))
misrow1 <- sample(n,20)
is.na(data1[misrow1, c(4:6)]) <- TRUE

### preparation step
impblock <- rowimpPrep(data1)

summary(impblock)
```

Index