Package ‘missCompare’

October 13, 2022

Type Package

Title Intuitive Missing Data Imputation Framework

Version 1.0.3

Maintainer Tibor V. Varga <tirgit@hotmail.com>

Description Offers a convenient pipeline to test and compare various missing data
imputation algorithms on simulated and real data. These include simpler meth-
ods, such as mean and median
imputation and random replacement, but also include more sophisticated algorithms already im-
plemented in popular
R packages, such as 'mi', described by Su et al. (2011) <doi:10.18637/jss.v045.i02>; 'mice', de-
scribed by van Buuren
and Groothuis-Oudshoorn (2011) <doi:10.18637/jss.v045.i03>; 'missForest', de-
scribed by Stekhoven and Buhlmann (2012)
<doi:10.1093/bioinformatics/btr597>; 'missMDA', described by Josse and Hus-
son (2016) <doi:10.18637/jss.v070.i01>; and
'pcaMethods', described by Stack-
lies et al. (2007) <doi:10.1093/bioinformatics/btm069>. The central assumption behind
'missCompare' is that structurally different datasets (e.g. larger datasets with a large num-
ber of correlated variables
vs. smaller datasets with non correlated variables) will benefit differently from different miss-
ing data imputation
algorithms. 'missCompare' takes measurements of your dataset and sets up a sandbox to try a cu-
rated list of standard and
sophisticated missing data imputation algorithms and compares them assuming custom missing-
ness patterns.
'missCompare' will also impute your real-
life dataset for you after the selection of the best performing algorithm
in the simulations. The package also provides various post-
imputation diagnostics and visualizations to help you
assess imputation performance.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

BugReports https://github.com/Tirgit/missCompare/issues

Depends R (>= 3.5.0)

biocViews

Imports Amelia, data.table, dplyr, ggdendro, ggplot2, Hmisc, ltm, magrittr, MASS, Matrix, mi, mice, missForest, missMDA, pcaMethods, plyr, rlang, stats, utils, tidyr, VIM

Suggests testthat, knitr, rmarkdown, devtools

VignetteBuilder knitr

NeedsCompilation no

Author Tibor V. Varga [aut, cre] (<https://orcid.org/0000-0002-2383-699X>), David Westergaard [aut] (<https://orcid.org/0000-0003-0128-8432>)

Repository CRAN

Date/Publication 2020-12-01 08:50:03 UTC

R topics documented:

`all_patterns` *Missing data spike-in in various missing data patterns*

Description

`all_patterns` spikes in missingness using MCAR, MAR, MNAR (default) and MAP (optional) patterns

Usage

```
all_patterns(
  X_hat,
  MD_pattern,
  NA_fraction,
  min_PDM = 10,
  assumed_pattern = NA
)
```

Arguments

`X_hat`	Simulated matrix with no missingness (Simulated_matrix output from the `simulate` function)
`MD_pattern`	Missing data pattern in the original dataset (MD_Pattern output from the `get_data` function)
`NA_fraction`	Fraction of missingness in the original dataset (Fraction_missingness output from the `get_data` function)
`min_PDM`	All patterns with number of observations less than this number will be removed from the missing data generation. This argument is necessary to be carefully set, as the function will fail or generate erroneous missing data patterns with very complicated missing data patterns. The default is 10, but for large datasets this number needs to be set higher to avoid errors. Please select a value based on the min_PDM_thresholds output from the `get_data` function
`assumed_pattern`	Vector of missingness types (must be same length as missingness fraction per variable). If this input is specified, the function will spike in missing datapoints in a MAP pattern as well.

Details

This function uses the generated simulated matrix and generates missing datapoints in MCAR, MAR and MNAR patterns. Optionally, in case the user defines an assumed pattern, the `all_patterns` function will also generate a MAP missingness pattern. It is suggested that the user carefully examines the missing data fractions, excludes variables with high missingness using the `clean` function. For more information on the functions that spike in missing data in MCAR, MAR, MNAR and MAP patterns, please see the functions `MCAR`, `MAR`, `MNAR` and `MAP`.

Value

`MCAR_matrix`	Matrix with MCAR pre-defined missingness pattern (default output)
`MAR_matrix`	Matrix with MAR pre-defined missingness pattern (default output)
`MNAR_matrix`	Matrix with MNAR pre-defined missingness pattern (default output)
`MAP_matrix`	Matrix with MAP pre-defined missingness pattern (optional output)

Examples

```
cleaned <- clean(clindata_miss, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)

miss_list <- all_patterns(simulated$Simulated_matrix,
                    MD_pattern = metadata$MD_Pattern,
                    NA_fraction = metadata$Fraction_missingness,
                    min_PDM = 20)

miss_list <- all_patterns(simulated$Simulated_matrix,
                    MD_pattern = metadata$MD_Pattern,
                    NA_fraction = metadata$Fraction_missingness,
                    min_PDM = 10,
                    assumed_pattern = c('MAR', 'MCAR', 'MCAR', 'MAR',
                                        'MNAR', 'MCAR', 'MAR', 'MCAR',
                                        'MCAR', 'MAR', 'MNAR'))
```

`clean` *Dataframe cleaning for missing data handling*

Description

`clean` helps in the conversion of missing values, variable types and removes rows and columns above pre-specified missingness

Usage

```
clean(
  X,
  var_remove = NULL,
  var_removal_threshold = 0.5,
  ind_removal_threshold = 1,
  missingness_coding = NA
)
```

Arguments

`X`	Original dataframe with samples in rows and variables as columns
`var_remove`	Variables to remove (e.g. ID). Define by character vector, e.g. c('ID', 'character_variable')
`var_removal_threshold`	Variable removal threshold with default 0.5 (range between 0 and 1). Variables (columns) above this missingness fraction will be removed during the cleaning process
`ind_removal_threshold`	Individual removal threshold with default 1 (range between 0 and 1). Individuals (rows) above this missingness fraction will be removed during the cleaning process
`missingness_coding`	Non NA coding in original dataframe that should be changed to NA (e.g. -9). Can take a single value (define by: missingness_coding = -9) or multiple values (define by: missingness_coding = c(-9, -99, -999))

Details

For better imputation performance, a clean, filtered dataframe is needed. Variables and samples with very high missingness fractions will negatively impact most missing data imputation algorithms. This function cleans the original dataframe by removing rows (samples) and columns (variables) above pre-specified missingness thresholds. The function will also convert any prespecified, strangely coded missing data to NAs. Note that all factor variables will be converted or coerced to numeric variables.

Value

Clean dataset with NAs as missing values and rows/columns above the pre-specified missingness thresholds removed

Examples

```
# basic settings
cleaned <- clean(clindata_miss, missingness_coding = -9)

# setting very conservative removal thresholds
cleaned <- clean(clindata_miss,
                 var_removal_threshold = 0.10,
                 ind_removal_threshold = 0.9,
                 missingness_coding = -9)
```

clindata_miss *Clinical dataset with missingness*

Description

clindata_miss is a custom made dataframe that resembles a real-life clinical dataset. The correlations between variables, the data means, SDs and ranges are realistic, but the dataset is constructed by simulations and manual data input. The dataset contains missing values (approximately 10% missing overall), and values are missing in a realistic pattern.

Usage

```
clindata_miss
```

Format

A data frame with 2500 rows and 12 variables:

age numeric, age, in years, 2.88% missing - in general, age is not likely have lots of missing data in a realistic dataset, therefore only a few values are missing here randomly, e.g. due to mistakes in data input

sex factor, male=1 and female=2, 2.88% missing - similar to age, sex information is also not likely have missing data in a realistic dataset, no values are missing here

waist numeric, waist circumference, in cm, 4.12% missing - anthropometric data is easy to collect, therefore only a small fraction is missing here, often missing together with BMI, the other anthropometric variable

BMI numeric, body mass index, in kg/m2, 4.16% missing - anthropometric data is easy to collect, therefore only a small fraction is missing here, often missing together with waist, the other anthropometric variable

SBP numeric, systolic blood pressure, in mmHg, 8.84% missing - in a realistic fashion, SBP is almost always missing together with DBP

DBP numeric, diastolic blood pressure, in mmHg, 8.84% missing - in a realistic fashion, DBP is almost always missing together with SBP

FG numeric, blood fasting glucose concentration, in mmol/dl, 5.84% missing - often missing together with other clinical variables

PPG numeric, blood postprandial glucose concentration, in mmol/dl, 53.2% missing - in this simulated dataset, only less than half of the participants had postprandial glucose measurements

TC numeric, blood total cholesterol concentration, in mmol/dl, 7.2% missing - often missing together with other lipids, TG and HDL-C

TG numeric, blood triglycerides concentration, in mmol/dl, 7.48% missing - often missing together with other lipids, TC and HDL-C, due to the sensitivity of a hypothetical machine, values below 0.6 are set to -9, upon conversion from -9s to NAs, the missingness fraction is 10.6%

HDL numeric, blood high density lipoprotein cholesterol concentration, in mmol/dl, 10.76% missing - often missing together with other lipids, TG and TC, due to the sensitivity of a hypothetical machine, values below 0.05 are set to -9, upon conversion from -9s to NAs, the missingness fraction is 13.72%

education factor, primary school=1, secondary school=2, bsc degree=3, msc degree=4, phd degree=5, 7.16% missing - self reported education missing in a not random fashion, those with lower education are less likely to report their education status

Source

The dataset is simulated and undergone manual configuration.

`get_data` *Extraction of metadata from dataframes*

Description

`get_data` extracts descriptive metadata from the dataframe including information on missing data

Usage

```
get_data(X, matrixplot_sort = TRUE, plot_transform = TRUE)
```

Arguments

`X`	Original dataframe with samples in rows and variables as columns. Can also use the resulting object from the `clean` function
`matrixplot_sort`	Boolean with default TRUE. If TRUE, the matrix plot will be sorted by missing/non-missing status. If FALSE, the original order of rows will be retained
`plot_transform`	Boolean with default TRUE. If TRUE, the matrix plot will plot all variables scaled (mean = 0, SD = 1). If FALSE, the matrix plot will show the variables on their original scale

Details

This function uses the original dataframe and extracts descriptive metadata including dimensions, missingness fractions overall and by variable, number of missing values overall and by variable, missing data patterns, missing data correlations and missing data visualizations

Value

`Complete_cases`	Number of complete cases (samples with no missing data in any columns)
`Rows`	Total number of rows (samples) in the dataframe
`Columns`	Total number of columns (variables) in the dataframe

`Corr_matrix`	Correlation matrix of all variables. The correlation matrix contains Pearson correlation coefficients based on pairwise correlations between variable pairs
`Fraction_missingness`	Total fraction of missingness expressed as a number between 0 and 1, where 1 means 100% of data is missing and 0 means there are no missing values
`Fraction_missingness_per_variable`	Fraction of missingness per variable. A (named) numeric vector of length the number of columns. Each variable missingness values are expressed as numbers between 0 and 1, where 1 means 100% of data is missing and 0 means there are no missing values
`Total_NA`	Total number of missing values in the dataframe
`NA_per_variable`	Number of missing values per variables in the dataframe. A (named) numeric vector of length the number of columns
`MD_Pattern`	Missing data pattern calculated using mice::md_pattern (see `md.pattern` in the mice package)
`NA_Correlations`	Correlation matrix of variables vs. variables converted to boolean based on missingness status (yes/no). Point-biserial correlation coefficients based on variable pairs is obtained using complete observations in the respective variable pairs. Higher correlation coefficients can indicate MAR missingness pattern
`NA_Correlation_plot`	Plot based on NA_Correlations
`min_PDM_thresholds`	Small dataframe offering clues on how to set min_PDM thresholds in the next steps of the pipeline. The first column represents min_PDM thresholds, while the second column represents percentages that would be retained by setting min_PDM to the respective values. These values are the percentages of the total rows with any number of missing data (excluding complete observations), so a value of e.g. 80% would mean that 80% of rows with missing data with the most common patterns are represented in the simulation step
`Vars_above_half`	Character vector of variables names with missingness higher than 50%
`Matrix_plot`	Matrix plot where missing values are colored gray and available values are colored based on value range
`Cluster_plot`	Cluster plot of co-missingness. Variables demonstrating shared missingness patterns will branch at closer to the bottom of the plot, while no patterns will be represented by branches high in the plot

Examples

```
cleaned <- clean(clindata_miss, missingness_coding = -9)
metadata <- get_data(cleaned)
metadata <- get_data(cleaned, matrixplot_sort = FALSE)
```

`impute_data` *Missing data imputation with various methods*

Description

`impute_data` imputes a dataframe with missing values with selected algorithm(s)

Usage

```
impute_data(X, scale = TRUE, n.iter = 10, sel_method = c(1:16))
```

Arguments

`X`	Dataframe - the original data that contains missing values.
`scale`	Boolean with default TRUE. Scaling will scale and center all numeric variables to mean = 0 and standard deviation = 1. This is strongly suggested for all PCA-based methods, and for the sake of comparison (and in case all methods are run), for the other methods too. Please note, however, that some methods (e.g. pcaMethods NLPCA, missForest, etc.) are equipped to handle non-linear data. In these cases scaling is up to the user. Factor variables will not be scaled.
`n.iter`	Number of iterations to perform with default 10. This will only affect the probabilistic methods that allow for a multiple imputation framework. The rest of the methods (if specified to run) will only generate 1 imputed dataframe.
`sel_method`	Numeric vector that specifies which methods to run. Default is all methods (1-16), but any combinations, including selecting a single method, are allowed.

1	random replacement
2	median imputation
3	mean imputation
4	missMDA Regularized
5	missMDA EM
6	pcaMethods PPCA
7	pcaMethods svdImpute
8	pcaMethods BPCA
9	pcaMethods NIPALS
10	pcaMethods NLPCA
11	mice mixed
12	mi Bayesian
13	Amelia II
14	missForest
15	Hmisc aregImpute
16	VIM kNN

Details

This function assumes that the user has performed simulations using the `impute_simulated` function and arrived to some conclusions regarding which functions would be the best performing on their datasets. This function offers a convenient way to impute datasets with a curated list of functions. Some of the functions allow for a multiple imputation framework (they operate with probabilistic models, hence there is uncertainty in the imputed values), so this function allows to generate multiple imputed datasets. The user can decide to impute their dataframe with a selected method or with multiple methods.

Value

A nested list of imputed datasets. In case only a subset of methods was selected the non-selected list elements will be empty.

`random_replacement`	Imputed dataset using random replacement
`mean_imputation`	Imputed dataset using mean imputation
`median_imputation`	Imputed dataset using median imputation
`missMDA_reg_imputation`	Imputed dataset using the missMDA regularized imputation algorithm
`missMDA_EM_imputation`	Imputed dataset using the missMDA EM imputation algorithm
`pcaMethods_PPCA_imputation`	Imputed dataset using the pcaMethods PPCA imputation algorithm
`pcaMethods_svdImpute_imputation`	Imputed dataset using the pcaMethods svdImpute imputation algorithm
`pcaMethods_BPCA_imputation`	Imputed dataset using the pcaMethods BPCA imputation algorithm
`pcaMethods_Nipals_imputation`	Imputed dataset using the pcaMethods NIPALS imputation algorithm
`pcaMethods_NLPCA_imputation`	Imputed dataset using the pcaMethods NLPCA imputation algorithm
`mice_mixed_imputation`	Imputed dataset using the mice mixed imputation algorithm
`mi_Bayesian_imputation`	Imputed dataset using the mi Bayesian imputation algorithm
`ameliaII_imputation`	Imputed dataset using the Amelia2 imputation algorithm replacement
`missForest_imputation`	Imputed dataset using the missForest imputation algorithm replacement
`Hmisc_aregImpute_imputation`	Imputed dataset using the Hmisc aregImpute imputation algorithm
`VIM_kNN_imputation`	Imputed dataset using the VIM kNN imputation algorithm replacement

Examples

```
## running 10 iterations of all algorithms (that allow for multiple imputation) and
## one copy of those that do not allow for multiple imputations
# impute_data(df, scale = TRUE, n.iter = 10,
#             sel_method = c(1:16))
## running 20 iterations of missForest (e.g. this was the best performing algorithm
## in simulations) on a non-scaled dataframe
# impute_data(df, scale = FALSE, n.iter = 20,
#             sel_method = c(14))
## running 1 iterations of four selected non-probabilistic algorithms on a scaled dataframe
# impute_data(df, scale = TRUE, n.iter = 1,
#             sel_method = c(2:3, 5, 7))
```

`impute_simulated` *Imputation algorithm tester on simulated data*

Description

`impute_simulated` tests the imputation quality of all missing data imputation algorithms on matrices with various missing data patterns, using various metrics

Usage

```
impute_simulated(
  rownum,
  colnum,
  cormat,
  n.iter = 10,
  MD_pattern,
  NA_fraction,
  min_PDM = 10,
  assumed_pattern = NA
)
```

Arguments

`rownum`	Number of rows (samples) in the original dataframe (Rows output from the `get_data` function)
`colnum`	Number of rows (variables) in the original dataframe (Columns output from the `get_data` function)
`cormat`	Correlation matrix of the original dataframe (Corr_matrix output from the `get_data` function)
`n.iter`	Number of iterations to perform with default 10.
`MD_pattern`	Missing data pattern in the original dataset (MD_Pattern output from the `get_data` function)

`NA_fraction`	Fraction of missingness in the original dataset (Fraction_missingness output from the `get_data` function)
`min_PDM`	All patterns with number of observations less than this number will be removed from the missing data generation. This argument is necessary to be carefully set, as the function will fail or generate erroneous missing data patterns with very complicated missing data patterns. The default is 10, but for large datasets this number needs to be set higher to avoid errors.
`assumed_pattern`	Vector of missingness types (must be same length as missingness fraction per variable). If this input is specified, the function will spike in missing datapoints in a MAP pattern as well.

Details

This function tests the imputation accuracy of the a curated list of missing data imputation algorithms (16 algorithms at the moment) by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov-Smirnov test statistics D (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing) for each missing data imputation algorithm. The function will calculate average computation time per method as well. The function will automatically detect whether there is a MAP matrix in the list and calculate metrics for all matrices provided in the list. Important! All statistics output by this function are calculated for ALL missing values across the dataset, not by variable.

Value

`Imputation_metrics_raw`	Raw RMSE, MAE, KS and computation time values per method, per missingness pattern, per iteration)
`Imputation_metrics_means`	RMSE, MAE, KS and computation time means per method and missingness pattern
`Plot_TIME`	Boxplot of computation time values per missing data imputation algorithm
`Plot_RMSE`	Faceted boxplot of RMSE values per missingness pattern and missing data imputation algorithm
`Plot_MAE`	Faceted boxplot of MAE values per missingness pattern and missing data imputation algorithm
`Plot_KS`	Faceted boxplot of KS values per missingness pattern and missing data imputation algorithm

Examples

```
## in case there is no assumed missingness pattern per variable
# wrap <- impute_simulated(rownum = metadata$Rows,
#        colnum = metadata$Columns,
#        cormat = metadata$Corr_matrix,
#        MD_pattern = metadata$MD_Pattern,
#       NA_fraction = metadata$Fraction_missingness,
#       min_PDM = 10,
#       n.iter = 50)

## in case there is a pre-defined assumed pattern
# wrap <- impute_simulated(rownum = metadata$Rows,
#       colnum = metadata$Columns,
#       cormat = metadata$Corr_matrix,
#       MD_pattern = metadata$MD_Pattern,
#       NA_fraction = metadata$Fraction_missingness,
#       min_PDM = 10,
#       assumed_pattern = c('MAR','MAR','MCAR','MCAR',
#                         'MNAR','MCAR','MAR','MNAR',
#                         'MCAR','MNAR','MCAR'),
#       n.iter = 50)
```

MAP *Missing data spike-in in MAP pattern*

Description

MAP spikes in missingness using missing-at-assumed (MAP) pattern

Usage

```
MAP(
  X_hat,
  MD_pattern,
  NA_fraction,
  min_PDM = 10,
 assumed_pattern = c("MAR", "MCAR", "MCAR", "MAR", "MNAR", "MCAR", "MCAR", "MAR",
    "MNAR", "MCAR", "MCAR")
)
```

Arguments

`X_hat`	Simulated matrix with no missingness (Simulated_matrix output from the `simulate` function)
`MD_pattern`	Missing data pattern in the original dataset (MD_Pattern output from the `get_data` function)
`NA_fraction`	Fraction of missingness in the original dataset (Fraction_missingness output from the `get_data` function)
`min_PDM`	All patterns with number of observations less than this number will be removed from the missing data generation. This argument is necessary to be carefully set, as the function will fail or generate erroneous missing data patterns with very complicated missing data patterns. The default is 10, but for large datasets this number needs to be set higher to avoid errors. Please select a value based on the min_PDM_thresholds output from the `get_data` function

`assumed_pattern`	Vector of missingness types (must be same length as missingness fraction per variable)

Details

This function uses the generated simulated matrix and generates missing datapoints in a missing-at-assumed pattern for each variable using the `ampute` function, considering the fraction of missingness in the original dataset and the original missingness pattern. In the `MAP` function, the user needs to define a character vector (of length the same as the fraction the number of columns in the dataset) that specifies which missingness pattern corresponds to the variables. In case the first four columns are assumed missing at random, the next one missing completely at random and the last two column not at random, the input vector will be: `c(rep('MAR', 4), 'MCAR', rep('MNAR',2))` The algorithm will spike in missing values according to the specified pattern. Please note that after the missing data spike-in, the function will remove rows with 100% missing data.

Value

`MAP_matrix`	Matrix with MAP pre-defined missingness pattern
`Summary`	Summary of MAP_matrix including number of missing values per variable

Examples

```
cleaned <- clean(clindata_miss, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)

MAP(simulated$Simulated_matrix,
    MD_pattern = metadata$MD_Pattern,
    NA_fraction = metadata$Fraction_missingness,
    min_PDM = 10,
    assumed_pattern = c('MAR', 'MCAR', 'MCAR', 'MAR', 'MNAR', 'MCAR',
                        'MAR', 'MCAR', 'MCAR', 'MAR', 'MNAR'))
```

`MAR` *Missing data spike-in in MAR pattern*

Description

`MAR` spikes in missingness using missing-at-random (MAR) pattern

Usage

```
MAR(X_hat, MD_pattern, NA_fraction, min_PDM = 10)
```

Arguments

`X_hat`	Simulated matrix with no missingness (Simulated_matrix output from the `simulate` function)
`MD_pattern`	Missing data pattern in the original dataset (MD_Pattern output from the `get_data` function)
`NA_fraction`	Fraction of missingness in the original dataset (Fraction_missingness output from the `get_data` function)
`min_PDM`	All patterns with number of observations less than this number will be removed from the missing data generation. This argument is necessary to be carefully set, as the function will fail or generate erroneous missing data patterns with very complicated missing data patterns. The default is 10, but for large datasets this number needs to be set higher to avoid errors. Please select a value based on the min_PDM_thresholds output from the `get_data` function

Details

This function uses the generated simulated matrix and generates missing datapoints in a missing-at-random pattern for each variable using the `ampute` function, considering the fraction of missingness in the original dataset and the original missingness pattern. The characteristic of the MAR pattern is that the missingness in a variable is dependent on the distribution of other variable(s). Please note that after the missing data spike-in, the function will remove rows with 100% missing data.

Value

`MAR_matrix`	Matrix with MAR pre-defined missingness pattern
`Summary`	Summary of MAR_matrix including number of missing values per variable

Examples

```
cleaned <- clean(clindata_miss, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)

MAR(simulated$Simulated_matrix,
    MD_pattern = metadata$MD_Pattern,
    NA_fraction = metadata$Fraction_missingness,
    min_PDM = 10)
```

MCAR *Missing data spike-in in MCAR pattern*

Description

`MCAR` spikes in missingness using missing-completely-at-random (MCAR) pattern

Usage

```
MCAR(X_hat, MD_pattern, NA_fraction, min_PDM = 10)
```

Arguments

`X_hat`	Simulated matrix with no missingness (Simulated_matrix output from the `simulate` function)
`MD_pattern`	Missing data pattern in the original dataset (MD_Pattern output from the `get_data` function)
`NA_fraction`	Fraction of missingness in the original dataset (Fraction_missingness output from the `get_data` function)
`min_PDM`	All patterns with number of observations less than this number will be removed from the missing data generation. This argument is necessary to be carefully set, as the function will fail or generate erroneous missing data patterns with very complicated missing data patterns. The default is 10, but for large datasets this number needs to be set higher to avoid errors. Please select a value based on the min_PDM_thresholds output from the `get_data` function

Details

This function uses the generated simulated matrix and generates missing datapoints in a missing-completely-at-random pattern for each variable, considering the fraction of missingness for each variable, so potential missing data fraction imbalances between variables in the original data will be retained. The missing data spike-in is completely at random. Please note that after the missing data spike-in, the function will remove rows with 100% missing data.

Value

`MCAR_matrix`	Matrix with MCAR pre-defined missingness pattern
`Summary`	Summary of MCAR_matrix including number of missing values per variable

Examples

```
cleaned <- clean(clindata_miss, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)

MCAR(simulated$Simulated_matrix,
    MD_pattern = metadata$MD_Pattern,
    NA_fraction = metadata$Fraction_missingness,
    min_PDM = 10)
```

`missCompare` *'missCompare': Missing Data Imputation Comparison Framework*

Description

The **'missCompare'** package offers a convenient pipeline to test and compare various missing data imputation algorithms on simulated data. The central assumption behind 'missCompare' is that structurally different datasets (e.g. larger datasets with a large number of correlated variables vs. smaller datasets with non correlated variables and other combinations) will benefit differently from different missing data imputation algorithms. **'missCompare'** takes measurements of your dataset and sets up a sandbox to try a curated list of standard and sophisticated missing data imputation algorithms and compares them assuming custom set missingness patterns. **'missCompare'** will give you a comparative analysis of missing data imputation algorithms, offer a report with the best performing algorithms assuming various missing data patterns and publication ready visualizations, impute your dataset for you, assess imputation performance using a validation framework and help you better understand missing data in your dataset.

Details

Package:	missCompare
Depends:	R (>= 3.5.0)
Type:	Package
Version:	1.0.3
Date:	2020-11-30
License:	MIT
LazyLoad:	Yes

Author(s)

- Tibor V. Varga `<tirgit@hotmail.com>`
- David Westergaard `<david.westergaard@cpr.ku.dk>`

See Also

`https://github.com/Tirgit/missCompare`

`MNAR` *Missing data spike-in in MNAR pattern*

Description

`MNAR` spikes in missingness using missing-not-at-random (MNAR) pattern

Usage

```
MNAR(X_hat, MD_pattern, NA_fraction, min_PDM = 10)
```

Arguments

`X_hat`	Simulated matrix with no missingness (Simulated_matrix output from the `simulate` function)
`MD_pattern`	Missing data pattern in the original dataset (MD_Pattern output from the `get_data` function)
`NA_fraction`	Fraction of missingness in the original dataset (Fraction_missingness output from the `get_data` function)
`min_PDM`	All patterns with number of observations less than this number will be removed from the missing data generation. This argument is necessary to be carefully set, as the function will fail or generate erroneous missing data patterns with very complicated missing data patterns. The default is 10, but for large datasets this number needs to be set higher to avoid errors. Please select a value based on the min_PDM_thresholds output from the `get_data` function

Details

This function uses the generated simulated matrix and generates missing datapoints in a missing-not-at-random pattern for each variable using the `ampute` function, considering the fraction of missingness in the original dataset and the original missingness pattern. The characteristic of the MNAR pattern is that the missingness in a variable is dependent on its own distribution. Please note that after the missing data spike-in, the function will remove rows with 100% missing data.

Value

`MNAR_matrix`	Matrix with MNAR pre-defined missingness pattern
`Summary`	Summary of MNAR_matrix including number of missing values per variable

Examples

```
cleaned <- clean(clindata_miss, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)

MNAR(simulated$Simulated_matrix,
    MD_pattern = metadata$MD_Pattern,
    NA_fraction = metadata$Fraction_missingness,
    min_PDM = 10)
```

`post_imp_diag` *Post imputation diagnostics*

Description

`post_imp_diag` serves as post imputation diagnostics. The function compares the original dataset (with missing data) with the imputed dataset. The function outputs statistics and visualizations that will help the user compare the original and the imputed datasets.

Usage

```
post_imp_diag(X_orig, X_imp, scale = TRUE, n.boot = 100)
```

Arguments

`X_orig`	Dataframe - the original data that contains missing values.
`X_imp`	Dataframe - the imputed data with no missing values.
`scale`	Boolean with default TRUE. Scaling will scale and center all variables to mean = 0 and standard deviation = 1 in the **original dataframe with missingness**. The user should select TRUE or FALSE here depending on whether the imputed dataframe has scaled or unscaled values (which is controlled by the scale argument in `impute_data`. Factor variables will not be scaled.
`n.boot`	Number of bootstrap iterations to generate mean pairwise Pearson correlation coefficients and 95% confidence intervals for variable pairs from the original and the imputed dataframes.

Details

This function uses the original dataframe and produces plots that allows the user to compare the distributions of the original values and the imputed values for each numeric variables. If there are factors present in the dataframes, the function will recognize this and create bar charts for these. In addition, the function will calculate bootstrapped pairwise Pearson correlation coefficients between numeric variables in the original dataframe (with missingness) and the imputed dataframe and plot these for the user to assess whether the imputation distorted the original data structure or not. The function will also visualize variable clusters in the original dataframe and the imputed one. Should the imputation algorithm perform well, the variable distributions and the variable clusters should be similar.

Value

`Histograms`	List of histograms of all numeric variables. The histograms show the original values and the imputed values overlaid for each variables in the dataframe
`Boxplots`	List of boxplots of all numeric variables. The boxplots show the original values and the imputed values for each variables in the dataframe. As normally, the boxplots show the median values, the IQR and the range of values

`Barcharts`	List of bar charts of all categorical (factor) variables. The bar charts show the original categories and the imputed categories for each categorical variables in the dataframe. Bar charts will only be output if scale is set to FALSE and both the original and imputed data contain the same factor variables
`Statistics`	List of output statistics for all variables. A named vector containing means and standard deviations of the original and imputed values, P value from Welch's t test and D test statistic from a Kolmogorov–Smirnov test comparing the original and the imputed values by variable
`Variable_clusters_orig`	Variable clusters based on the original dataframe (with missingness). Regardless of the argument scale being set to TRUE or FALSE, the clusters are assessed based on normalized data
`Variable_clusters_imp`	Variable clusters based on the imputed dataframe. Regardless of the argument scale being set to TRUE or FALSE, the clusters are assessed based on normalized data
`Correlation_stats`	Mean pairwise Pearson's correlation coefficients and 95% confidence intervals from the original dataframe (with missingness) and the imputed dataframe
`Correlation_plot`	Scatter plot of mean pairwise Pearson's correlation coefficients from the original dataframe (with missingness) and the imputed dataframe. The blue line represents a line with slope 1 and intercept 0. The red line is a fitted line of the correlation coefficient pairs. The error bars around the points represent the individual 95% confidence intervals drawn from bootstrapping the correlation coefficients

Examples

```
# diagnostics <- post_imp_diag(X_orig = df_miss, X_imp = df_imputed, scale=TRUE)
# diagnostics$Histograms$variable_X
# diagnostics$Boxplots$variable_Z
# diagnostics$Statistics$variable_Y
```

`simulate` *Simulation of matrix with no missingness*

Description

`simulate` simulates a clean matrix with no missingness based on the original data structure where all variables have the same mean and standard deviation and are normally distributed.

Usage

```
simulate(rownum, colnum, cormat, meanval = 0, sdval = 1)
```

Arguments

`rownum`	Number of rows (samples) in the original dataframe (Rows output from the `get_data` function)
`colnum`	Number of rows (variables) in the original dataframe (Columns output from the `get_data` function)
`cormat`	Correlation matrix of the original dataframe (Corr_matrix output from the `get_data` function)
`meanval`	Desired mean value for the simulated variables, default = 0
`sdval`	Desired standard deviation value for the simulated variables, default = 1

Details

This function requires the metadata from the original dataframe and simulates a matrix with no missingness with the same number of rows and columns and with the same or very similar correlation matrix as observed in the original dataframe. When the correlation matrix is a non positive definitive matrix, the nearPD function estimates the closest positive definitive matrix. Outputs from the function makes it easy to compare the original correlation matrix with the nearPD correlation matrix. In the simulated matrix all variables have normal distribution and fixed mean and standard deviation. This matrix will be subsequently used for spiking in missing values and for the testing of various missing data imputation algorithms.

Value

`Simulated_matrix`
: Simulated matrix with no missingness. The simulated matrix resembles the original dataframe in size and correlation structure, but has normally distributed variables with fixed means and SDs

`Original_correlation_sample`
: Sample of the original correlation structure (for comparison)

`NearPD_correlation_sample`
: Sample of the nearPD (nearest positive definitive matrix) correlation structure of the simulated matrix (for comparison)

Examples

```
cleaned <- clean(clindata_miss, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)
```

test_AmeliaII	*Testing the 'Amelia II' missing data imputation algorithm*

Description

`test_AmeliaII` tests the imputation accuracy of the 'Amelia II' missing data imputation algorithm on matrices with various missing data patterns

Usage

```
test_AmeliaII(X_hat, list)
```

Arguments

`X_hat`	Simulated matrix with no missingness (this matrix will be used to obtain the error between the original and imputed values). (Simulated_matrix output from the `simulate` function)
`list`	List of matrices with various missingness patterns (MCAR, MAR, MNAR and optionally, MAP). (The input is ideally the R object that was generated using the `all_patterns` function)

Details

This function tests the imputation accuracy of the 'Amelia II' missing data imputation algorithm by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov–Smirnov D test statistic (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing). The function will also calculate the cumulative computation time for imputing all datasets. The function will automatically detect whether there is a MAP matrix in the list and calculate RMSE for all matrices provided in the list.

Value

`Comp_time`	Computation time of imputation using method (default output)
`MCAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)

MAR_MAE	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
MNAR_MAE	Mean absolute error (MAE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
MAP_MAE	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
MCAR_KS	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
MAR_KS	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
MNAR_KS	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
MAP_KS	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)

Examples

```
clindata_miss_mini <- clindata_miss[1:80,1:4]
cleaned <- clean(clindata_miss_mini, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)
miss_list <- all_patterns(simulated$Simulated_matrix,
                    MD_pattern = metadata$MD_Pattern,
                    NA_fraction = metadata$Fraction_missingness,
                    min_PDM = 2)

test_AmeliaII(X_hat = simulated$Simulated_matrix, list = miss_list)
```

---

test_aregImpute *Testing the 'Hmisc' aregImpute missing data imputation algorithm*

Description

test_aregImpute tests the imputation accuracy of the 'Hmisc' aregImpute missing data imputation algorithm on matrices with various missing data patterns

Usage

```
test_aregImpute(X_hat, list)
```

Arguments

`X_hat`	Simulated matrix with no missingness (this matrix will be used to obtain the error between the original and imputed values). (Simulated_matrix output from the `simulate` function)
`list`	List of matrices with various missingness patterns (MCAR, MAR, MNAR and optionally, MAP). (The input is ideally the R object that was generated using the `all_patterns` function)

Details

This function tests the imputation accuracy of the 'Hmisc' aregImpute missing data imputation algorithm by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov–Smirnov D test statistic (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing). The function will also calculate the cumulative computation time for imputing all datasets. The function will automatically detect whether there is a MAP matrix in the list and calculate RMSE for all matrices provided in the list.

Value

`Comp_time`	Computation time of imputation using method (default output)
`MCAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)

Examples

```
clindata_miss_mini <- clindata_miss[1:80,1:4]
cleaned <- clean(clindata_miss_mini, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)
miss_list <- all_patterns(simulated$Simulated_matrix,
                    MD_pattern = metadata$MD_Pattern,
                    NA_fraction = metadata$Fraction_missingness,
                    min_PDM = 2)

test_aregImpute(X_hat = simulated$Simulated_matrix, list = miss_list)
```

`test_kNN` *Testing the 'VIM' kNN missing data imputation algorithm*

Description

`test_kNN` tests the imputation accuracy of the 'VIM' kNN missing data imputation algorithm on matrices with various missing data patterns

Usage

```
test_kNN(X_hat, list)
```

Arguments

`X_hat`	Simulated matrix with no missingness (this matrix will be used to obtain the error between the original and imputed values). (Simulated_matrix output from the `simulate` function)
`list`	List of matrices with various missingness patterns (MCAR, MAR, MNAR and optionally, MAP). (The input is ideally the R object that was generated using the `all_patterns` function)

Details

This function tests the imputation accuracy of the 'VIM' kNN missing data imputation algorithm by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov–Smirnov D test statistic (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing). The function will also calculate the cumulative computation time for imputing all datasets. The function will automatically detect whether there is a MAP matrix in the list and calculate RMSE for all matrices provided in the list.

Value

`Comp_time`	Computation time of imputation using method (default output)
`MCAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)

Examples

```
clindata_miss_mini <- clindata_miss[1:80,1:4]
cleaned <- clean(clindata_miss_mini, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)
miss_list <- all_patterns(simulated$Simulated_matrix,
                   MD_pattern = metadata$MD_Pattern,
                   NA_fraction = metadata$Fraction_missingness,
                   min_PDM = 2)

test_kNN(X_hat = simulated$Simulated_matrix, list = miss_list)
```

`test_mean_imp` *Testing the mean imputation algorithm*

Description

`test_mean_imp` tests the imputation accuracy of the mean imputation algorithm on matrices with various missing data patterns

Usage

```
test_mean_imp(X_hat, list)
```

Arguments

`X_hat`	Simulated matrix with no missingness (this matrix will be used to obtain the error between the original and imputed values). (Simulated_matrix output from the `simulate` function)
`list`	List of matrices with various missingness patterns (MCAR, MAR, MNAR and optionally, MAP). (The input is ideally the R object that was generated using the `all_patterns` function)

Details

This function tests the imputation accuracy of the mean imputation algorithm by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov–Smirnov D test statistic (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing). The function will also calculate the cumulative computation time for imputing all datasets. The function will automatically detect whether there is a MAP matrix in the list and calculate RMSE for all matrices provided in the list.

Value

`Comp_time`	Computation time of imputation using method (default output)
`MCAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)

MAR_MAE	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
MNAR_MAE	Mean absolute error (MAE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
MAP_MAE	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
MCAR_KS	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
MAR_KS	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
MNAR_KS	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
MAP_KS	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)

Examples

```
clindata_miss_mini <- clindata_miss[1:80,1:4]
cleaned <- clean(clindata_miss_mini, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)
miss_list <- all_patterns(simulated$Simulated_matrix,
                    MD_pattern = metadata$MD_Pattern,
                    NA_fraction = metadata$Fraction_missingness,
                    min_PDM = 2)

test_mean_imp(X_hat = simulated$Simulated_matrix, list = miss_list)
```

test_median_imp *Testing the median imputation algorithm*

Description

test_median_imp tests the imputation accuracy of the median imputation algorithm on matrices with various missing data patterns

Usage

```
test_median_imp(X_hat, list)
```

Arguments

`X_hat`	Simulated matrix with no missingness (this matrix will be used to obtain the error between the original and imputed values). (Simulated_matrix output from the `simulate` function)
`list`	List of matrices with various missingness patterns (MCAR, MAR, MNAR and optionally, MAP). (The input is ideally the R object that was generated using the `all_patterns` function)

Details

This function tests the imputation accuracy of the median imputation algorithm by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov–Smirnov D test statistic (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing). The function will also calculate the cumulative computation time for imputing all datasets. The function will automatically detect whether there is a MAP matrix in the list and calculate RMSE for all matrices provided in the list.

Value

`Comp_time`	Computation time of imputation using method (default output)
`MCAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)

Examples

```
clindata_miss_mini <- clindata_miss[1:80,1:4]
cleaned <- clean(clindata_miss_mini, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)
miss_list <- all_patterns(simulated$Simulated_matrix,
                    MD_pattern = metadata$MD_Pattern,
                    NA_fraction = metadata$Fraction_missingness,
                    min_PDM = 2)

test_median_imp(X_hat = simulated$Simulated_matrix, list = miss_list)
```

`test_mi` *Testing the 'mi' missing data imputation algorithm*

Description

`test_mi` tests the imputation accuracy of the 'mi' missing data imputation algorithm on matrices with various missing data patterns

Usage

```
test_mi(X_hat, list)
```

Arguments

`X_hat`	Simulated matrix with no missingness (this matrix will be used to obtain the error between the original and imputed values). (Simulated_matrix output from the `simulate` function)
`list`	List of matrices with various missingness patterns (MCAR, MAR, MNAR and optionally, MAP). (The input is ideally the R object that was generated using the `all_patterns` function)

Details

This function tests the imputation accuracy of the 'mi' missing data imputation algorithm by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov–Smirnov D test statistic (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing). The function will also calculate the cumulative computation time for imputing all datasets. The function will automatically detect whether there is a MAP matrix in the list and calculate RMSE for all matrices provided in the list.

Value

`Comp_time`	Computation time of imputation using method (default output)
`MCAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)

Examples

```
# clindata_miss_mini <- clindata_miss[1:80,1:4]
# cleaned <- clean(clindata_miss_mini, missingness_coding = -9)
# metadata <- get_data(cleaned)
# simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
# cormat = metadata$Corr_matrix)
# miss_list <- all_patterns(simulated$Simulated_matrix,
#                    MD_pattern = metadata$MD_Pattern,
#                    NA_fraction = metadata$Fraction_missingness,
#                    min_PDM = 2)

# test_mi(X_hat = simulated$Simulated_matrix, list = miss_list)
```

test_mice_mixed *Testing the 'mice' mixed missing data imputation algorithm*

Description

test_mice_mixed tests the imputation accuracy of the 'mice' mixed missing data imputation algorithm on matrices with various missing data patterns

Usage

```
test_mice_mixed(X_hat, list)
```

Arguments

X_hat	Simulated matrix with no missingness (this matrix will be used to obtain the error between the original and imputed values). (Simulated_matrix output from the simulate function)
list	List of matrices with various missingness patterns (MCAR, MAR, MNAR and optionally, MAP). (The input is ideally the R object that was generated using the all_patterns function)

Details

This function tests the imputation accuracy of the 'mice' mixed missing data imputation algorithm by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov–Smirnov D test statistic (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing). The function will also calculate the cumulative computation time for imputing all datasets. The function will automatically detect whether there is a MAP matrix in the list and calculate RMSE for all matrices provided in the list.

Value

Comp_time	Computation time of imputation using method (default output)
MCAR_RMSE	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
MAR_RMSE	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
MNAR_RMSE	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
MAP_RMSE	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
MCAR_MAE	Mean absolute error (MAE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)

MAR_MAE	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
MNAR_MAE	Mean absolute error (MAE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
MAP_MAE	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
MCAR_KS	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
MAR_KS	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
MNAR_KS	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
MAP_KS	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)

Examples

```
clindata_miss_mini <- clindata_miss[1:80,1:4]
cleaned <- clean(clindata_miss_mini, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)
miss_list <- all_patterns(simulated$Simulated_matrix,
                    MD_pattern = metadata$MD_Pattern,
                    NA_fraction = metadata$Fraction_missingness,
                    min_PDM = 2)

test_mice_mixed(X_hat = simulated$Simulated_matrix, list = miss_list)
```

---

test_missForest *Testing the 'missForest' missing data imputation algorithm*

Description

test_missForest tests the imputation accuracy of the 'missForest' missing data imputation algorithm on matrices with various missing data patterns

Usage

```
test_missForest(X_hat, list)
```

Arguments

`X_hat`	Simulated matrix with no missingness (this matrix will be used to obtain the error between the original and imputed values). (Simulated_matrix output from the `simulate` function)
`list`	List of matrices with various missingness patterns (MCAR, MAR, MNAR and optionally, MAP). (The input is ideally the R object that was generated using the `all_patterns` function)

Details

This function tests the imputation accuracy of the 'missForest' missing data imputation algorithm by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov–Smirnov D test statistic (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing). The function will also calculate the cumulative computation time for imputing all datasets. The function will automatically detect whether there is a MAP matrix in the list and calculate RMSE for all matrices provided in the list.

Value

`Comp_time`	Computation time of imputation using method (default output)
`MCAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)

Examples

```
clindata_miss_mini <- clindata_miss[1:80,1:4]
cleaned <- clean(clindata_miss_mini, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)
miss_list <- all_patterns(simulated$Simulated_matrix,
                    MD_pattern = metadata$MD_Pattern,
                    NA_fraction = metadata$Fraction_missingness,
                    min_PDM = 2)

test_missForest(X_hat = simulated$Simulated_matrix, list = miss_list)
```

`test_missMDA_EM` *Testing the 'missMDA' EM missing data imputation algorithm*

Description

`test_missMDA_EM` tests the imputation accuracy of the 'missMDA' EM missing data imputation algorithm on matrices with various missing data patterns

Usage

```
test_missMDA_EM(X_hat, list)
```

Arguments

`X_hat`	Simulated matrix with no missingness (this matrix will be used to obtain the error between the original and imputed values). (Simulated_matrix output from the `simulate` function)
`list`	List of matrices with various missingness patterns (MCAR, MAR, MNAR and optionally, MAP). (The input is ideally the R object that was generated using the `all_patterns` function)

Details

This function tests the imputation accuracy of the 'missMDA' EM missing data imputation algorithm by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov–Smirnov D test statistic (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing). The function will also calculate the cumulative computation time for imputing all datasets. The function will automatically detect whether there is a MAP matrix in the list and calculate RMSE for all matrices provided in the list.

Value

`Comp_time`	Computation time of imputation using method (default output)
`MCAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)

Examples

```
clindata_miss_mini <- clindata_miss[1:80,1:4]
cleaned <- clean(clindata_miss_mini, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)
miss_list <- all_patterns(simulated$Simulated_matrix,
                   MD_pattern = metadata$MD_Pattern,
                   NA_fraction = metadata$Fraction_missingness,
                   min_PDM = 2)

test_missMDA_EM(X_hat = simulated$Simulated_matrix, list = miss_list)
```

test_missMDA_reg — *Testing the 'missMDA' regularized missing data imputation algorithm*

Description

test_missMDA_reg tests the imputation accuracy of the 'missMDA' regularized missing data imputation algorithm on matrices with various missing data patterns

Usage

```
test_missMDA_reg(X_hat, list)
```

Arguments

X_hat	Simulated matrix with no missingness (this matrix will be used to obtain the error between the original and imputed values). (Simulated_matrix output from the simulate function)
list	List of matrices with various missingness patterns (MCAR, MAR, MNAR and optionally, MAP). (The input is ideally the R object that was generated using the all_patterns function)

Details

This function tests the imputation accuracy of the 'missMDA' regularized missing data imputation algorithm by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov–Smirnov D test statistic (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing). The function will also calculate the cumulative computation time for imputing all datasets. The function will automatically detect whether there is a MAP matrix in the list and calculate RMSE for all matrices provided in the list.

Value

Comp_time	Computation time of imputation using method (default output)
MCAR_RMSE	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
MAR_RMSE	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
MNAR_RMSE	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
MAP_RMSE	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
MCAR_MAE	Mean absolute error (MAE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)

`MAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)

Examples

```
clindata_miss_mini <- clindata_miss[1:80,1:4]
cleaned <- clean(clindata_miss_mini, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)
miss_list <- all_patterns(simulated$Simulated_matrix,
                    MD_pattern = metadata$MD_Pattern,
                    NA_fraction = metadata$Fraction_missingness,
                    min_PDM = 2)

test_missMDA_reg(X_hat = simulated$Simulated_matrix, list = miss_list)
```

`test_pcaMethods_BPCA` *Testing the 'pcaMethods' BPCA missing data imputation algorithm*

Description

`test_pcaMethods_BPCA` tests the imputation accuracy of the 'pcaMethods' BPCA missing data imputation algorithm on matrices with various missing data patterns

Usage

```
test_pcaMethods_BPCA(X_hat, list)
```

Arguments

`X_hat`	Simulated matrix with no missingness (this matrix will be used to obtain the error between the original and imputed values). (Simulated_matrix output from the `simulate` function)
`list`	List of matrices with various missingness patterns (MCAR, MAR, MNAR and optionally, MAP). (The input is ideally the R object that was generated using the `all_patterns` function)

Details

This function tests the imputation accuracy of the 'pcaMethods' BPCA missing data imputation algorithm by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov–Smirnov D test statistic (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing). The function will also calculate the cumulative computation time for imputing all datasets. The function will automatically detect whether there is a MAP matrix in the list and calculate RMSE for all matrices provided in the list.

Value

`Comp_time`	Computation time of imputation using method (default output)
`MCAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)

Examples

```
clindata_miss_mini <- clindata_miss[1:80,1:4]
cleaned <- clean(clindata_miss_mini, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)
miss_list <- all_patterns(simulated$Simulated_matrix,
                    MD_pattern = metadata$MD_Pattern,
                    NA_fraction = metadata$Fraction_missingness,
                    min_PDM = 2)

test_pcaMethods_BPCA(X_hat = simulated$Simulated_matrix, list = miss_list)
```

---

test_pcaMethods_Nipals — *Testing the 'pcaMethods' NIPALS missing data imputation algorithm*

Description

`test_pcaMethods_Nipals` tests the imputation accuracy of the 'pcaMethods' NIPALS missing data imputation algorithm on matrices with various missing data patterns

Usage

```
test_pcaMethods_Nipals(X_hat, list)
```

Arguments

`X_hat`	Simulated matrix with no missingness (this matrix will be used to obtain the error between the original and imputed values). (Simulated_matrix output from the `simulate` function)
`list`	List of matrices with various missingness patterns (MCAR, MAR, MNAR and optionally, MAP). (The input is ideally the R object that was generated using the `all_patterns` function)

Details

This function tests the imputation accuracy of the 'pcaMethods' NIPALS missing data imputation algorithm by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov–Smirnov D test statistic (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing). The function will also calculate the cumulative computation time for imputing all datasets. The function will automatically detect whether there is a MAP matrix in the list and calculate RMSE for all matrices provided in the list.

Value

`Comp_time`	Computation time of imputation using method (default output)
`MCAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)

Examples

```
clindata_miss_mini <- clindata_miss[1:80,1:4]
cleaned <- clean(clindata_miss_mini, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)
miss_list <- all_patterns(simulated$Simulated_matrix,
                   MD_pattern = metadata$MD_Pattern,
                   NA_fraction = metadata$Fraction_missingness,
                   min_PDM = 2)

test_pcaMethods_Nipals(X_hat = simulated$Simulated_matrix, list = miss_list)
```

test_pcaMethods_NLPCA *Testing the 'pcaMethods' NLPCA missing data imputation algorithm*

Description

test_pcaMethods_NLPCA tests the imputation accuracy of the 'pcaMethods' NLPCA missing data imputation algorithm on matrices with various missing data patterns

Usage

```
test_pcaMethods_NLPCA(X_hat, list)
```

Arguments

`X_hat`	Simulated matrix with no missingness (this matrix will be used to obtain the error between the original and imputed values). (Simulated_matrix output from the simulate function)
`list`	List of matrices with various missingness patterns (MCAR, MAR, MNAR and optionally, MAP). (The input is ideally the R object that was generated using the all_patterns function)

Details

This function tests the imputation accuracy of the 'pcaMethods' NLPCA missing data imputation algorithm by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov–Smirnov D test statistic (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing). The function will also calculate the cumulative computation time for imputing all datasets. The function will automatically detect whether there is a MAP matrix in the list and calculate RMSE for all matrices provided in the list.

Value

`Comp_time`	Computation time of imputation using method (default output)
`MCAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)

`MAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)

Examples

```
# clindata_miss_mini <- clindata_miss[1:80,1:4]
# cleaned <- clean(clindata_miss_mini, missingness_coding = -9)
# metadata <- get_data(cleaned)
# simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
# cormat = metadata$Corr_matrix)
# miss_list <- all_patterns(simulated$Simulated_matrix,
#                     MD_pattern = metadata$MD_Pattern,
#                     NA_fraction = metadata$Fraction_missingness,
#                     min_PDM = 2)

# test_pcaMethods_NLPCA(X_hat = simulated$Simulated_matrix, list = miss_list)
```

`test_pcaMethods_PPCA` *Testing the 'pcaMethods' PPCA missing data imputation algorithm*

Description

`test_pcaMethods_PPCA` tests the imputation accuracy of the 'pcaMethods' PPCA missing data imputation algorithm on matrices with various missing data patterns

Usage

```
test_pcaMethods_PPCA(X_hat, list)
```

Arguments

`X_hat`	Simulated matrix with no missingness (this matrix will be used to obtain the error between the original and imputed values). (Simulated_matrix output from the `simulate` function)
`list`	List of matrices with various missingness patterns (MCAR, MAR, MNAR and optionally, MAP). (The input is ideally the R object that was generated using the `all_patterns` function)

Details

This function tests the imputation accuracy of the 'pcaMethods' PPCA missing data imputation algorithm by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov–Smirnov D test statistic (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing). The function will also calculate the cumulative computation time for imputing all datasets. The function will automatically detect whether there is a MAP matrix in the list and calculate RMSE for all matrices provided in the list.

Value

`Comp_time`	Computation time of imputation using method (default output)
`MCAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)

Examples

```
clindata_miss_mini <- clindata_miss[1:80,1:4]
cleaned <- clean(clindata_miss_mini, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)
miss_list <- all_patterns(simulated$Simulated_matrix,
                    MD_pattern = metadata$MD_Pattern,
                    NA_fraction = metadata$Fraction_missingness,
                    min_PDM = 2)

test_pcaMethods_PPCA(X_hat = simulated$Simulated_matrix, list = miss_list)
```

---

test_pcaMethods_svdImpute

Testing the 'pcaMethods' svdImpute missing data imputation algorithm

---

Description

`test_pcaMethods_svdImpute` tests the imputation accuracy of the 'pcaMethods' svdImpute missing data imputation algorithm on matrices with various missing data patterns

Usage

```
test_pcaMethods_svdImpute(X_hat, list)
```

Arguments

`X_hat`	Simulated matrix with no missingness (this matrix will be used to obtain the error between the original and imputed values). (Simulated_matrix output from the `simulate` function)
`list`	List of matrices with various missingness patterns (MCAR, MAR, MNAR and optionally, MAP). (The input is ideally the R object that was generated using the `all_patterns` function)

Details

This function tests the imputation accuracy of the 'pcaMethods' svdImpute missing data imputation algorithm by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov–Smirnov D test statistic (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing). The function will also calculate the cumulative computation time for imputing all datasets. The function will automatically detect whether there is a MAP matrix in the list and calculate RMSE for all matrices provided in the list.

Value

`Comp_time`	Computation time of imputation using method (default output)
`MCAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_RMSE`	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_MAE`	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
`MCAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
`MAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
`MNAR_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
`MAP_KS`	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)

Examples

```
clindata_miss_mini <- clindata_miss[1:80,1:4]
cleaned <- clean(clindata_miss_mini, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)
miss_list <- all_patterns(simulated$Simulated_matrix,
                   MD_pattern = metadata$MD_Pattern,
                   NA_fraction = metadata$Fraction_missingness,
                   min_PDM = 2)

test_pcaMethods_svdImpute(X_hat = simulated$Simulated_matrix, list = miss_list)
```

test_random_imp — *Testing the random replacement imputation algorithm*

Description

test_random_imp tests the imputation accuracy of the random replacement imputation algorithm on matrices with various missing data patterns

Usage

```
test_random_imp(X_hat, list)
```

Arguments

X_hat	Simulated matrix with no missingness (this matrix will be used to obtain the error between the original and imputed values). (Simulated_matrix output from the simulate function)
list	List of matrices with various missingness patterns (MCAR, MAR, MNAR and optionally, MAP). (The input is ideally the R object that was generated using the all_patterns function)

Details

This function tests the imputation accuracy of the random replacement imputation algorithm by comparing the original simulated matrix with no missingness and the imputed matrices generated by the algorithm using the matrices with MCAR, MAR, MNAR and (optionally) MAP missingness patterns. The function calculates root-mean-square error (RMSE), mean absolute error (MAE), Kolmogorov–Smirnov D test statistic (KS) between the imputed datapoints and the original datapoints (that were subsequently set to missing). The function will also calculate the cumulative computation time for imputing all datasets. The function will automatically detect whether there is a MAP matrix in the list and calculate RMSE for all matrices provided in the list.

Value

Comp_time	Computation time of imputation using method (default output)
MCAR_RMSE	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
MAR_RMSE	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
MNAR_RMSE	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
MAP_RMSE	Root-mean-square error (RMSE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
MCAR_MAE	Mean absolute error (MAE) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)

MAR_MAE	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
MNAR_MAE	Mean absolute error (MAE) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
MAP_MAE	Mean absolute error (MAE) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)
MCAR_KS	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MCAR missingness pattern (default output)
MAR_KS	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAR missingness pattern (default output)
MNAR_KS	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MNAR missingness pattern (default output)
MAP_KS	Kolmogorov–Smirnov test statistic (KS) between the indexed original values and the imputed values in an MAP missingness pattern (optional output)

Examples

```
clindata_miss_mini <- clindata_miss[1:80,1:4]
cleaned <- clean(clindata_miss_mini, missingness_coding = -9)
metadata <- get_data(cleaned)
simulated <- simulate(rownum = metadata$Rows, colnum = metadata$Columns,
cormat = metadata$Corr_matrix)
miss_list <- all_patterns(simulated$Simulated_matrix,
                    MD_pattern = metadata$MD_Pattern,
                    NA_fraction = metadata$Fraction_missingness,
                    min_PDM = 2)

test_random_imp(X_hat = simulated$Simulated_matrix, list = miss_list)
```

Index