Package 'opImputation'

November 7, 2025

Type Package

Title Optimal Selection of Imputation Methods for Pain-Related Numerical Data

Version 0.6

Description A model-agnostic framework for selecting dataset-specific imputation methods for missing values in numerical data related to pain. Lotsch J, Ultsch A (2025) ``A model-agnostic framework for dataset-specific selection of missing value imputation methods in pain-related numerical data'' Canadian Journal of Pain (in minor revision).

Depends R (>= 3.5.0)

Imports parallel, Rfit, methods, stats, caret, ABCanalysis, ggplot2, future, future.apply, progressr, missForest, utils, mice, miceRanger, multiUS, Amelia, mi, reshape2, DataVisualizations, abind, cowplot, twosamples, ggh4x, ggrepel, tools

LazyData true

Suggests testthat (>= 3.0.0)

License GPL-3

URL https://github.com/JornLotsch/opImputation

Encoding UTF-8

Date 2025-11-04

NeedsCompilation no

Author Jorn Lotsch [aut, cre] (ORCID: <https://orcid.org/0000-0002-5818-6958>), Alfred Ultsch [aut] (ORCID: <https://orcid.org/0000-0002-7845-3283>)

Maintainer Jorn Lotsch <j.lotsch@em.uni-frankfurt.de>

Repository CRAN

Date/Publication 2025-11-07 13:50:02 UTC

Contents

`CodeinLogMetabolitesUrine` *Codeine metabolite concentrations in urine.*

Description

Data set from a pharmacogenetic investigation that assessed the formation of morphine from codeine in the presence of variants in cytochrome P450 2D6.

Usage

```
data("CodeinLogMetabolitesUrine")
```

Details

Size 50 x 5 , stored in `CodeinLogMetabolitesUrine["MOR", "M3G", "M6G", "COD", "C6G"]`

Examples

```
data(CodeinLogMetabolitesUrine)
str(CodeinLogMetabolitesUrine)
```

`compare_imputation_methods` *Compare Imputation Methods for Missing Value Analysis*

Description

Performs a comprehensive comparative analysis of different imputation methods on a dataset by artificially inserting missings, applying various imputation techniques, and evaluating their performance through multiple metrics and visualizations. Optionally produces a final imputed dataset using the best-performing method.

Usage

```
compare_imputation_methods(
  data,
  imputation_methods = all_imputation_methods,
  imputation_repetitions = 20,
  perfect_methods_in_ABC = FALSE,
  n_iterations = 20,
  n_proc = getOption("mc.cores", 2L),
  percent_missing = 0.1,
  seed,
  mnar_shape = 1,
  mnar_ity = 0,
  low_only = FALSE,
  fixed_seed_for_inserted_missings = FALSE,
  max_attempts = 1000,
  overall_best_z_delta = FALSE,
  produce_final_imputations = TRUE,
  plot_results = TRUE,
  verbose = TRUE
)
```

Arguments

`data`	Data frame or matrix containing numeric data. May contain existing missing values (NA).
`imputation_methods`	Character vector of imputation method names to compare. Default is `all_imputation_methods`. Must include at least two non-calibrating methods. Available options include: **Univariate methods:** `"median"`, `"mean"`, `"mode"`, `"rSample"`; **Multivariate methods:** `"bag"`, `"bag_repeated"`, `"rf_mice"`, `"rf_mice_repeated"`, `"rf_missForest"`, `"rf_missForest_repeated"`, `"miceRanger"`, `"miceRanger_repeated"`, `"cart"`, `"cart_repeated"`, `"linear"`, `"pmm"`, `"pmm_repeated"`, `"knn3"`, `"knn5"`, `"knn7"`, `"knn9"`, `"knn10"`, `"ameliaImp"`, `"ameliaImp_repeated"`, `"miImp"`; **Diagnostic methods:** `"plus"`, `"plusminus"`, `"factor"`; **Calibrating methods:** `"tinyNoise_0.000001"`, `"tinyNoise_0.00001"`, `"tinyNoise_0.0001"`, `"tinyNoise_0.001"`, `"tinyNoise_0.01"`, `"tinyNoise_0.05"`, `"tinyNoise_0.1"`, `"tinyNoise_0.2"`, `"tinyNoise_0.5"`, `"tinyNoise_1"`. It is recommended that all imputation methods be used in a complete comparison (Default).
`imputation_repetitions`	Integer. Number of times each imputation method is repeated for each iteration. Default is 20.
`perfect_methods_in_ABC`	Whether to include perfect imputation methods in comparative selections. Default is FALSE.
`n_iterations`	Integer. Number of different missing data patterns to test. Default is 20.
`n_proc`	Integer. Number of processor cores to use for parallel processing. Default is `getOption("mc.cores", 2L)`.

`percent_missing`
: Numeric. Proportion of values to randomly set as missing in each iteration (0 to 1). Default is 0.1 (10%).

`seed`
: Integer. Random seed for reproducibility. If missing, reads current system seed. Setting the parameter is recommended for better reproducibility.

`mnar_shape`
: Numeric. Shape parameter for MNAR (Missing Not At Random) mechanism. Default is 1 (MCAR - Missing Completely At Random).

`mnar_ity`
: Numeric. Degree of missingness mechanism (0-1). Default is 0 (completely random).

`low_only`
: Logical. If TRUE, only insert missings in lower values. Default is FALSE.

`fixed_seed_for_inserted_missings`
: Logical. If TRUE, use same seed for inserting missings across all iterations. Default is FALSE.

`max_attempts`
: Integer. Maximum attempts to create valid missing pattern without completely empty cases. Default is 1000.

`overall_best_z_delta`
: Logical. If TRUE, compare all methods against the overall best; if FALSE, compare against best within category. Default is FALSE.

`produce_final_imputations`
: Logical. If TRUE, produce final imputed dataset using the best-performing univariate or multivariate method from the ABC analysis. The function will try methods in order of their ranking until one succeeds in producing a complete dataset with no missing values. Default is TRUE.

`plot_results`
: Logical. If TRUE, show summary plots. Default is TRUE.

`verbose`
: Logical. If TRUE, print best method information and turn on messaging. Default is TRUE.

Details

This function implements a model-agnostic framework for dataset-specific selection of missing value imputation methods. The analysis workflow:

1. Artificially inserts missing values into complete data
2. Applies multiple imputation methods
3. Calculates performance metrics (zDelta values)
4. Ranks methods using ABC analysis
5. Generates comprehensive visualizations
6. Optionally produces final imputed dataset using the best method

The zDelta metric represents standardized absolute differences between original and imputed values, providing a robust measure of imputation quality.

The MNAR mechanism allows testing methods under realistic scenarios:

- `mnar_ity = 0`: Missing Completely At Random (MCAR)
- `mnar_ity > 0`: Missing Not At Random with specified degree

- `low_only = TRUE`: Missings preferentially in lower values
- `mnar_shape`: Controls shape of missingness probability distribution

Final Imputation Process: When `produce_final_imputations = TRUE`, the function automatically:

1. Extracts the ranked list of methods from ABC analysis results
2. Filters to only univariate and multivariate methods (excludes poisoned/calibrating methods)
3. Tries each method in order of performance ranking
4. Stops at the first method that successfully produces a complete dataset with no missing values
5. Prints informative console output showing which method was used, its ABC category, score, and ranking

If all methods fail to produce a complete dataset, the function returns NULL for both `imputed_data` and `method_used_for_imputation` and prints a warning message.

Value

Returns a list containing:

`all_imputation_runs`
: List containing all imputation results generated across repeated simulation runs and missing-data patterns.

`zdelta_metrics`
: Standardized z-delta error metrics, including raw values, medians, and variable-wise summaries quantifying deviations between original and imputed data.

`method_performance_summary`
: Comprehensive performance summary of all imputation methods, including ranking metrics and Activity-Based Classification (ABC) results.

`best_overall_method`
: Character. Name of the best-performing imputation method for the analyzed dataset.

`best_univariate_method`
: Character. Name of the top-performing univariate (single-variable) imputation method.

`best_multivariate_method`
: Character. Name of the top-performing multivariate (multi-variable) imputation method.

`best_uni_or_multivariate_method`
: Character. Name of the leading combined uni/multivariate imputation method.

`best_poisoned_method`
: Character. Name of the top-performing stress-test (formerly "poisoned") method.

`abc_results_table`
: Data frame containing the ABC (Activity-Based Classification) analysis results, including method categories and performance scores.

`fig_zdelta_distributions`
: `ggplot` object displaying the distribution of standardized z-delta values for the best-performing methods.

`fig_summary_comparison`
: `ggplot` object providing a combined summary figure integrating ABC classification and z-delta plots for comparative visualization.

`final_imputed_data`
: Data frame containing the final dataset with all missing values filled in using the best-performing method (only if `produce_final_imputations = TRUE`). Returns `NULL` if no complete dataset could be produced or if imputation was disabled.

`final_imputation_method`
: Character. Name of the imputation algorithm automatically selected and applied to create the final complete dataset. Returns `NULL` if imputation was disabled or failed.

Note

The function requires at least two non-calibrating imputation methods for comparison. Parallel processing can significantly improve performance on multi-core systems. Explicitly setting the `seed` parameter is strongly recommended for reproducibility.

When `produce_final_imputations = TRUE`, the function will display console output indicating which method was used for the final imputation, including its ABC category (A, B, or C), ABC score, and ranking among valid methods. This provides transparency and allows users to understand the quality of the chosen imputation method.

Author(s)

Jorn Lotsch, Alfred Ultsch

References

Lotsch J, Ultsch A. (2025). A model-agnostic framework for dataset-specific selection of missing value imputation methods in pain-related numerical data. Can J Pain (in minor revision)

See Also

`impute_missings` for single imputation operations

`create_diagnostic_missings` for creating diagnostic missing values

Examples

```
# Load example data
data_iris <- iris[,1:4]

# Add some missings
set.seed(42)
for(i in 1:4) data_iris[sample(1:nrow(data_iris), 0.05*nrow(data_iris)), i] <- NA

# Basic comparison with a subset of methods
results <- compare_imputation_methods(
  data = data_iris,
  imputation_methods = c("mean", "median", "rSample"),
```

```
  n_iterations = 2,
  imputation_repetitions = 2,
  produce_final_imputations = FALSE,
  plot_results = FALSE,
  verbose = FALSE
)

# Print results
# print(results)

# Cleanup to avoid open sockets during R CMD check
future::plan(future::sequential)
```

`create_diagnostic_missings` *Create Diagnostic Missing Values in Data*

Description

Introduces additional missing values into a dataset (which may already contain missings) using various missingness mechanisms: Missing Completely at Random (MCAR), Missing at Random (MAR), and Missing Not at Random (MNAR). Missing values are only inserted at positions that currently contain actual values (non-NA).

Usage

```
create_diagnostic_missings(
  x,
  Prob = 0.1,
  mnarity = 0,
  mnarshape = 1,
  lowOnly = FALSE,
  seed,
  maxAttempts = 1000
)
```

Arguments

`x`	Data frame or matrix with numeric data. May contain existing missing values
`Prob`	Numeric between 0 and 1. Proportion of non-missing values to set as missing (default: 0.1)
`mnarity`	Numeric between 0 and 1. Proportion of MNAR (vs MCAR/MAR) missingness (default: 0)
`mnarshape`	Numeric >= 1. Shape parameter for MNAR probability distribution (default: 1)
`lowOnly`	Logical. If TRUE, only creates missings for low values in MNAR case (default: FALSE)
`seed`	Integer. Random seed for reproducibility (default: 42)

maxAttempts	Integer. Maximum number of attempts to generate valid missing pattern (default: 100)

Details

The function creates missing values using a combination of mechanisms:

- MCAR: Random missingness independent of data values (controlled by 1-mnarity)
- MAR/MNAR: Value-dependent missingness (controlled by mnarity)

The shape of the MNAR probability distribution is controlled by mnarshape. When lowOnly = TRUE, MNAR mechanism targets only low values; otherwise it targets extreme values (both high and low).

The function ensures that no row ends up with all values missing by excluding positions from the sampling pool that would create completely missing rows.

Value

A list with two elements:

toDelete	List of row indices where values were set to missing, one vector per column
missData	Data frame with introduced missing values

Examples

```
# Create 10% MCAR missings
result <- create_diagnostic_missings(
  x = iris[,1:4],
  Prob = 0.1,
  mnarity = 0
)

# Create 20% missings with 50% MNAR targeting low values
result <- create_diagnostic_missings(
  x = iris[,1:4],
  Prob = 0.2,
  mnarity = 0.5,
  lowOnly = TRUE
)
```

---

impute_missings *Impute Missing Values Using Specified Method*

Description

Fills in missing values (NA) in numeric data using a specified imputation method. Provides a unified interface to univariate, multivariate, ensemble, and diagnostic imputation approaches. The function automatically handles method-specific parameters and error recovery.

Usage

```
impute_missings(
  x,
  method = "rf_missForest",
  ImputationRepetitions = 10,
  seed = NULL,
  x_orig = NULL
)
```

Arguments

x	Data frame or matrix containing numeric data with missing values (NA). All columns must be numeric.
method	Character string specifying which imputation method to use. Default is "rf_missForest". See Details for all available methods.
ImputationRepetitions	Integer. Number of repetitions for methods ending with "_repeated". These methods perform multiple imputations and return the median across repetitions for increased stability. Default is 10. Ignored for non-repeated methods.
seed	Integer. Random seed for reproducibility. If missing, reads current system seed. Setting the parameter is recommended for better reproducibility. Must be the same as set in compare_imputation_methods for reprodicible results.
x_orig	Data frame or matrix. Original complete data required only for poisoned and calibrating methods (used for validation/benchmarking). Must have same dimensions as x. Default is NULL.

Details

This function provides access to multiple imputation algorithms through a single interface. Simply specify the desired method name via the method parameter.

Available Methods:

Univariate methods (replace each missing value independently):

- "median" - Column median
- "mean" - Column mean
- "mode" - Column mode (most frequent value)
- "rSample" - Random sample from observed values

Bagging methods (bootstrap aggregating with decision trees):

- "bag" - Single bagged tree imputation
- "bag_repeated" - Repeated bagging with median aggregation

Random forest methods (ensemble of decision trees):

- "rf_mice" - Random forest via mice package
- "rf_mice_repeated" - Repeated RF via mice

- "rf_missForest" - Random forest via missForest package (recommended)
- "rf_missForest_repeated" - Repeated RF via missForest
- "miceRanger" - Random forest via miceRanger package
- "miceRanger_repeated" - Repeated RF via miceRanger

Tree-based methods:

- "cart" - Classification and regression trees
- "cart_repeated" - Repeated CART with median aggregation

Regression methods:

- "linear" - Lasso regression (L1-regularized linear model)
- "pmm" - Predictive mean matching
- "pmm_repeated" - Repeated PMM with median aggregation

k-Nearest neighbors methods:

- "knn3", "knn5", "knn7", "knn9", "knn10" - k-NN with specified number of neighbors

Multiple imputation methods:

- "ameliaImp" - Single imputation via Amelia II
- "ameliaImp_repeated" - Multiple imputations via Amelia II
- "miImp" - Multiple imputation via mi package

Poisoned methods (require x_orig, for validation only):

- "plus" - Add systematic positive offset
- "plusminus" - Add alternating positive/negative offset
- "factor" - Multiply by constant factor

Calibrating methods (require x_orig, for benchmarking):

- "tinyNoise_0.000001" through "tinyNoise_1" - Add small random noise with specified magnitude (available magnitudes: 0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.05, 0.1, 0.2, 0.5, 1)

Repeated methods: Methods ending with "_repeated" perform multiple independent imputations and return the median value across all repetitions. This typically provides more stable and robust results but requires more computation time. The number of repetitions is controlled by the ImputationRepetitions parameter.

Method selection guidance:

- For quick results: Use "median" or "mean"
- For moderate missing data: Use "rf_missForest" or "knn5"
- For high-quality results: Use "rf_missForest_repeated" or "pmm_repeated"
- For systematic comparison: Use compare_imputation_methods

Value

Returns a data frame with the same dimensions and column names as the input `x`, but with missing values filled in according to the specified method. If imputation fails, returns a data frame with all values set to NA.

Note

- Setting `seed` is strongly recommended for reproducibility
- Repeated methods provide better results but take longer to compute
- Poisoned and calibrating methods are for validation/benchmarking only
- If a method fails, the function returns NA values rather than throwing an error
- Some methods may be slow on large datasets

Author(s)

Jorn Lotsch, Alfred Ultsch

References

Lotsch J, Ultsch A. (2025). A model-agnostic framework for dataset-specific selection of missing value imputation methods in pain-related numerical data. Can J Pain (in minor revision)

See Also

`compare_imputation_methods`

Examples

```
# Load example data
data_iris <- iris[,1:4]

# Add some misisngs
set.seed(42)
for(i in 1:4) data_iris[sample(1:nrow(data_iris), 0.05*nrow(data_iris)), i] <- NA

# Simple univariate imputation with median
data_iris_imputed_median <- impute_missings(
  data_iris,
  method = "median"
)

# Show data
head(data_iris_imputed_median)
```

LipidsPsychiatricPat *Chromatography mass spectrometry of lipid mediators measured in blood samples.*

Description

Data set from a lipidomic investigation that assessed lipid mediators in blood samples in patients.

Usage

```
data("LipidsPsychiatricPat")
```

Details

Size 94 x 8 , stored in `LipidsPsychiatricPat["S1P", "C16Sphinganin", "C16Cer", "C20Cer", "C24Cer", "C24_1Cer", "C16GluCer", "C16LacCer"]`

Examples

```
data(LipidsPsychiatricPat)
str(LipidsPsychiatricPat)
```

PainThresholds *Psychophysical data from an investigation of pain thresholds.*

Description

Data set from an investigation of pain thresholds to various stimuli in healthy volunteers.

Usage

```
data("PainThresholds")
```

Details

Size 125 x 8 , stored in `PainThresholds["von.Frey", "von.Frey.Caps", "Heat", "Heat.Caps","Cold", "Cold.Menth", "Pressure", "Electric"]`

Examples

```
data(PainThresholds)
str(PainThresholds)
```

QSTpainEJPtransf *Psychophysical data from a clinical quantitative sensory testing study.*

Description

Data set from a psychophysical investigation in a clinical quantitative sensory testing study in healthy subjects.

Usage

```
data("QSTpainEJPtransf")
```

Details

Size 72 x 19 , stored in `QSTpainEJPtransf["PressureThr", "PressureTol", "TSACold", "ElectricThr", "ElectricTol", "Co2Thr","CO2VAS", "LaserThr", "LaserVAS", "CDT", "WDT", "TSL", "CPT", "HPT", "PPT", "MPT", "MPS", "WUR", "MDT"]`

Examples

```
data(QSTpainEJPtransf)
str(QSTpainEJPtransf)
```

Index