Package 'mlim'

May 8, 2026

Type Package

Title Single and Multiple Imputation with Automated Machine Learning

Version 0.3.0

Depends R (>= 3.5.0)

Description Machine learning algorithms have been used for performing
single missing data imputation and most recently, multiple imputations.
However, this is the first attempt for using automated machine learning algorithms
for performing both
single and multiple imputation. Automated machine learning is a procedure for
fine-tuning the model automatic, performing a random search for a model that
results in less error, without overfitting the data. The main idea is
to allow the model to set its own parameters for imputing each variable separately
instead of setting fixed predefined parameters to impute all variables
of the dataset.
Using automated machine learning, the package fine-tunes an Elastic
Net (default) or Gradient Boosting, Random Forest, Deep Learning, Extreme Gradient Boosting,
or Stacked Ensemble machine learning model (from one or a combination of other
supported algorithms) for imputing the missing
observations. This procedure has been implemented for the
first time by this package and is expected to outperform other packages for
imputing missing data that do not fine-tune their models. The multiple imputation
is implemented via bootstrapping without letting the duplicated observations to
harm the cross-validation procedure, which is the way imputed variables are evaluated.
Most notably, the package implements automated procedure for handling imputing imbalanced
data (class rarity problem), which happens when a factor variable has a level that is far more
prevalent than the other(s). This is known to result in biased predictions, hence, biased
imputation of missing data. However, the autobalancing procedure ensures that instead of
focusing on maximizing accuracy (classification error) in imputing factor variables,
a fairer procedure and imputation method is practiced.

License MIT + file LICENSE

Encoding UTF-8

Imports h2o (>= 3.34.0.0), curl (>= 4.3.2), mice, missRanger, memuse,
md.log (>= 0.2.0)

RoxygenNote 7.2.1

LazyData true

URL https://github.com/haghish/mlim,
https://www.sv.uio.no/psi/english/people/aca/haghish/

BugReports https://github.com/haghish/mlim/issues

NeedsCompilation no

Author E. F. Haghish [aut, cre, cph]

Maintainer E. F. Haghish <haghish@uio.no>

Repository CRAN

Date/Publication 2022-12-16 13:00:02 UTC

Contents

`charity` *some items about attitude towards charity*

Description

A dataset containing likert-scale items about attitude towards charity

Usage

```
charity
```

Format

A data frame with 832 rows and 5 variables:

ta1 Charitable Organizations More Effective

ta2 Degree of Trust

ta3 Charitable Organizations Honest/Ethical

ta4 Role Improving Communities

ta5 Job Delivering Services

Source

https://www.stata.com/

`manifest` *Manifest Anxiety Scale*

Description

The Taylor Manifest Anxiety Scale was first developed in 1953 to identify individuals who would be good subjects for studies of stress and other related psychological phenomenon. Since then it has been used as a measure of anxiety as general personality trait. Anxiety is a complex psychological construct that includes a multiple of different facets related to extensive worrying that may impair normal functioning. The test has been widely studied and used in research, however there are some concerns that it does not measure a single trait, but instead, measures a basket of loosely related ones and so the score is not that meaningful.

Usage

```
manifest
```

Format

A data frame with 4469 rows and 52 variables:

gender participants' gender

age participants' age in years

Q1 I do not tire quickly.

Q2 I am troubled by attacks of nausea.

Q3 I believe I am no more nervous than most others.

Q4 I have very few headaches.

Q5 I work under a great deal of tension.

Q6 I cannot keep my mind on one thing.

Q7 I worry over money and business.

Q8 I frequently notice my hand shakes when I try to do something.

Q9 I blush no more often than others.

Q10 I have diarrhea once a month or more.

Q11 I worry quite a bit over possible misfortunes.

Q12 I practically never blush.

Q13 I am often afraid that I am going to blush.

Q14 I have nightmares every few nights.

Q15 My hands and feet are usually warm.

Q16 I sweat very easily even on cool days.

Q17 Sometimes when embarrassed, I break out in a sweat.

Q18 I hardly ever notice my heart pounding and I am seldom short of breath.

Q19 I feel hungry almost all the time.

Q20 I am very seldom troubled by constipation.

Q21 I have a great deal of stomach trouble.

Q22 I have had periods in which I lost sleep over worry.

Q23 My sleep is fitful and disturbed.

Q24 I dream frequently about things that are best kept to myself.

Q25 I am easily embarrassed.

Q26 I am more sensitive than most other people.

Q27 I frequently find myself worrying about something.

Q28 I wish I could be as happy as others seem to be.

Q29 I am usually calm and not easily upset.

Q30 I cry easily.

Q31 I feel anxiety about something or someone almost all the time.

Q32 I am happy most of the time.

Q33 It makes me nervous to have to wait.

Q34 I have periods of such great restlessness that I cannot sit long I a chair.

Q35 Sometimes I become so excited that I find it hard to get to sleep.

Q36 I have sometimes felt that difficulties were piling up so high that I could not overcome them.

Q37 I must admit that I have at times been worried beyond reason over something that really did not matter.

Q38 I have very few fears compared to my friends.

Q39 I have been afraid of things or people that I know could not hurt me.

Q40 I certainly feel useless at times.

Q41 I find it hard to keep my mind on a task or job.

Q42 I am usually self-conscious.

Q43 I am inclined to take things hard.

Q44 I am a high-strung person.

Q45 Life is a trial for me much of the time.

Q46 At times I think I am no good at all.

Q47 I am certainly lacking in self-confidence.

Q48 I sometimes feel that I am about to go to pieces.

Q49 I shrink from facing crisis of difficulty.

Q50 I am entirely self-confident.

Details

The data comes from an online offering of the Taylor Manifest Anxiety Scale. At the end of the test users were asked if their answers were accurate and could be used for research, 76 https://openpsychometrics.org/.

#' items 1 to 50 were rated 1=True and 2=False. gender, chosen from a drop down menu (1=male, 2=female, 3=other) and age was entered as a free response (ages<14 have been removed)

Source

https://openpsychometrics.org/tests/TMAS/

References

Taylor, J. (1953). "A personality scale of manifest anxiety". The Journal of Abnormal and Social Psychology, 48(2), 285-290.

`mlim` *missing data imputation with automated machine learning*

Description

imputes data.frame with mixed variable types using automated machine learning (AutoML)

Usage

```
mlim(
  data = NULL,
  m = 1,
  algos = c("ELNET"),
  postimpute = FALSE,
  stochastic = m > 1,
  ignore = NULL,
  tuning_time = 900,
  max_models = NULL,
  maxiter = 10L,
  cv = 10L,
  matching = "AUTO",
  autobalance = TRUE,
  balance = NULL,
  seed = NULL,
  verbosity = NULL,
  report = NULL,
  tolerance = 0.001,
  doublecheck = TRUE,
  preimpute = "RF",
  cpu = -1,
```

```
  ram = NULL,
  flush = FALSE,
  preimputed.data = NULL,
  save = NULL,
  load = NULL,
  shutdown = TRUE,
  java = NULL,
  ...
)
```

Arguments

data	a data.frame (strictly) with missing data to be imputed. if 'load' argument is provided, this argument will be ignored.
m	integer, specifying number of multiple imputations. the default value is 1, carrying out a single imputation.
algos	character vector, specifying algorithms to be used for missing data imputation. supported algorithms are "ELNET", "RF", "GBM", "DL", "XGB", and "Ensemble". if more than one algorithm is specified, mlim changes behavior to save on runtime. for example, the default is "ELNET", which fine-tunes an Elastic Net model. In general, "ELNET" is expected to be the best algorithm because it fine-tunes very fast, it is very robust to over-fitting, and hence, it generalizes very well. However, if your data has many factor variables, each with several levels, it is recommended to have c("ELNET", "RF") as your imputation algorithms (and possibly add "Ensemble" as well, to make the most out of tuning the models). Note that code"XGB" is only available in Mac OS and Linux. moreover, "GBM", "DL" and "XGB" take the full given "tuning_time" (see below) to tune the best model for imputing he given variable, whereas "ELNET" will produce only one fine-tuned model, often at less time than other algorithms need for developing a single model, which is why "ELNET" is work horse of the mlim imputation package.
postimpute	(EXPERIMENTAL FEATURE) logical. if TRUE, mlim uses algorithms rather than 'ELNET' for carrying out postimputation optimization. however, if FALSE, all specified algorihms will be used in the process of 'reimputation' together. the 'Ensemble' algorithm is encouraged when other algorithms are used. However, for general users unspecialized in machine learning, postimpute is NOT recommended because this feature is currently experimental, prone to over-fitting, and highly computationally extensive.
stochastic	logical. by default it is set to TRUE for multiple imputation and FALSE for single imputation. stochastic argument is currently under testing and is intended to avoid inflating the correlation between imputed valuables.
ignore	character vector of column names or index of columns that should should be ignored in the process of imputation.
tuning_time	integer. maximum runtime (in seconds) for fine-tuning the imputation model for each variable in each iteration. the default time is 900 seconds but for a large dataset, you might need to provide a larger model development time. this

argument also influences max_models, see below. If you are using 'ELNET' algorithm (default), you can be generous with the 'tuning_time' argument because 'ELNET' tunes much faster than the rest and will only produce one model.

max_models — integer. maximum number of models that can be generated in the proecess of fine-tuning the parameters. this value default to 100, meaning that for imputing each variable in each iteration, up to 100 models can be fine-tuned. increasing this value should be consistent with increasing max_model_runtime_secs, allowing the model to spend more time in the process of individualized fine-tuning. as a result, the better tuned the model, the more accurate the imputed values are expected to be

maxiter — integer. maximum number of iterations. the default value is 15, but it can be reduced to 3 (not recommended, see below).

cv — logical. specify number of k-fold Cross-Validation (CV). values of 10 or higher are recommended. default is 10.

matching — logical. if TRUE, imputed values are coerced to the closest value to the non-missing values of the variable. if set to "AUTO", 'mlim' decides whether to match or not, based on the variable classes. the default is "AUTO".

autobalance — logical. if TRUE (default), binary and multinomial factor variables will be balanced before the imputation to obtain fairer and less-biased imputations, which are typically in favor of the majority class. if FALSE, imputation fairness will be sacrificed for overall accuracy, which is not recommended, although it is commonly practiced in other missing data imputation software. MLIM is highly concerned with imputation fairness for factor variables and autobalancing is generally recommended. in fact, higher overall accuracy does not mean a better imputation as long as minority classes are neglected, which increases the bias in favor of the majority class. if you do not wish to autobalance all the factor variables, you can manually specify the variables that should be balanced using the 'balance' argument (see below).

balance — character vector, specifying variable names that should be balanced before imputation. balancing the prevalence might decrease the overall accuracy of the imputation, because it attempts to ensure the representation of the rare outcome. this argument is optional and intended for advanced users that impute a severely imbalance categorical (nominal) variable.

seed — integer. specify the random generator seed

verbosity — character. controls how much information is printed to console. the value can be "warn" (default), "info", "debug", or NULL. to FALSE.

report — filename. if a filename is specified (e.g. report = "mlim.md"), the "md.log" R package is used to generate a Markdown progress report for the imputation. the format of the report is adopted based on the 'verbosity' argument. the higher the verbosity, the more technical the report becomes. if verbosity equals "debug", then a log file is generated, which includes time stamp and shows the function that has generated the message. otherwise, a reduced markdown-like report is generated. default is NULL.

tolerance — numeric. the minimum rate of improvement in estimated error metric of a variable to qualify the imputation for another round of iteration, if the maxiter is

not yet reached. any improvement of imputation is desirable. however, specifying values above 0 can reduce the number of required iterations at a marginal increase of imputation error. for larger datasets, value of "1e-3" is recommended to reduce number of iterations. the default value is '1e-3'.

`doublecheck` logical. default is TRUE (which is conservative). if FALSE, if the estimated imputation error of a variable does not improve, the variable will be not reimputed in the following iterations. in general, deactivating this argument will slightly reduce the imputation accuracy, however, it significantly reduces the computation time. if your dataset is large, you are advised to set this argument to FALSE. (EXPERIMENTAL: consider that by avoiding several iterations that marginally improve the imputation accuracy, you might gain higher accuracy by investing your computational resources in fine-tuning better algorithms such as "GBM")

`preimpute` character. specifies the 'primary' procedure of handling the missing data. before 'mlim' begins imputing the missing observations, they should be prepared for the imputation algorithms and thus, they should be replaced with some values. the default procedure is a quick "RF", which models the missing data with parallel Random Forest model. this is a very fast procedure, which later on, will be replaced within the "reimputation" procedure (see below). possible other alternative is `"mm"`, which carries out mean/mode replacement, as practiced by most imputation algorithms. "mm" is much faster than "RF". if your dataset is very large, consider pre-imputing it before hand using 'mlim.preimpute()' function and passing the preimputed dataset to mlim (see "preimputed.data" argument).

`cpu` integer. number of CPUs to be dedicated for the imputation. the default takes all of the available CPUs.

`ram` integer. specifies the maximum size, in Gigabytes, of the memory allocation. by default, all the available memory is used for the imputation. large memory size is particularly advised, especially for multicore processes. the more you give the more you get!

`flush` logical (experimental). if TRUE, after each model, the server is cleaned to retrieve RAM. this feature is in testing mode and is currently set to FALSE by default, but it is recommended if you have limited amount of RAM or large datasets.

`preimputed.data`

data.frame. if you have used another software for missing data imputation, you can still optimize the imputation by handing the data.frame to this argument, which will bypass the "preimpute" procedure.

`save` filename (with .mlim extension). if a filename is specified, an `mlim` object is saved after the end of each variable imputation. this object not only includes the imputed dataframe and estimated cross-validation error, but also includes the information needed for continuing the imputation, which is very useful feature for imputing large datasets, with a long runtime. this argument is activated by default and an mlim object is stored in the local directory named `"mlim.rds"`.

`load` filename (with .mlim extension). an object of class "mlim", which includes the data, arguments, and settings for re-running the imputation, from where it was previously stopped. the "mlim" object saves the current state of the imputation and is particularly recommended for large datasets or when the user specifies a

	computationally extensive settings (e.g. specifying several algorithms, increasing tuning time, etc.).
shutdown	logical. if TRUE, h2o server is closed after the imputation. the default is TRUE
java	character, specifying path to the executable 64bit Java JDK on the Microsoft Windows machines, if JDK is installed but the path environment variable is not set.
...	arguments that are used internally between 'mlim' and 'mlim.postimpute'. these arguments are not documented in the help file and are not intended to be used by end user.

Value

a data.frame, showing the estimated imputation error from the cross validation within the data.frame's attribution

Author(s)

E. F. Haghish

Examples

```
## Not run:
data(iris)

# add stratified missing observations to the data. to make the example run
# faster, I add NAs only to a single variable.
dfNA <- iris
dfNA$Species <- mlim.na(dfNA$Species, p = 0.1, stratify = TRUE, seed = 2022)

# run the ELNET single imputation (fastest imputation via 'mlim')
MLIM <- mlim(dfNA, shutdown = FALSE)

# in single imputation, you can estimate the imputation accuracy via cross validation RMSE
mlim.summarize(MLIM)

### or if you want to carry out ELNET multiple imputation with 5 datasets.
### next, to carry out analysis on the multiple imputation, use the 'mlim.mids' function
### minimum of 5 datasets
MLIM2 <- mlim(dfNA, m = 5)
mids <- mlim.mids(MLIM2, dfNA)
fit <- with(data=mids, exp=glm(Species ~ Sepal.Length, family = "binomial"))
res <- mice::pool(fit)
summary(res)

# you can check the accuracy of the imputation, if you have the original dataset
mlim.error(MLIM2, dfNA, iris)

## End(Not run)
```

`mlim.error` *imputation error*

Description

calculates NRMSE, missclassification rate, and miss-ranking absolute mean distance, scaled between 0 to 1, where 1 means maximum distance between the actual rank of a level and the imputed level.

Usage

```
mlim.error(
  imputed,
  incomplete,
  complete,
  transform = NULL,
  varwise = FALSE,
  ignore.missclass = TRUE,
  ignore.rank = FALSE
)
```

Arguments

`imputed`	the imputed dataframe
`incomplete`	the dataframe with missing values
`complete`	the original dataframe with no missing values
`transform`	character. it can be either "standardize", which standardizes the numeric variables before evaluating the imputation error, or "normalize", which change the scale of continuous variables to range from 0 to 1. the default is NULL.
`varwise`	logical, default is FALSE. if TRUE, in addition to mean accuracy for each variable type, the algorithm's performance for each variable (column) of the datast is also returned. if TRUE, instead of a numeric vector, a list is retuned.
`ignore.missclass`	logical. the default is TRUE. if FALSE, the overall missclassification rate for imputed unordered factors will be returned. in general, missclassification is not recommended, particularly for multinomial factors because it is not robust to imbalanced data. in other words, an imputation might show a very high accuracy, because it is biased towards the majority class, ignoring the minority levels. to avoid this error, Mean Per Class Error (MPCE) is returned, which is the average missclassification of each class and thus, it is a fairer criteria for evaluating multinomial classes.
`ignore.rank`	logical (default is FALSE, which is recommended). if TRUE, the accuracy of imputation of ordered factors (ordinal variables) will be evaluated based on 'missclassification rate' instead of normalized euclidean distance. this practice is not recommended because higher classification rate for ordinal variables does

not guarantee lower distances between the imputed levels, despite the popularity of evaluating ordinal variables based on missclassification rate. in other words, assume an ordinal variable has 5 levels (1. strongly disagree, 2. disagree, 3. uncertain, 4. agree, 5.strongly agree). in this example, if "ignore.rank = TRUE", then an imputation that imputes level "5" as "4" is equally inaccurate as other algorithm that imputes level "5" as "1". therefore, if you have ordinal variables in your dataset, make sure you declare them as "ordered" factors to get the best imputation accuracy.

Value

numeric vector

Author(s)

E. F. Haghish

Examples

```
## Not run:
data(iris)

# add 10% missing values, ensure missingness is stratified for factors
irisNA <- mlim.na(iris, p = 0.1, stratify = TRUE, seed = 2022)

# run the default imputation
MLIM <- mlim(irisNA)
mlim.error(MLIM, irisNA, iris)

# get error estimations for each variable
mlim.error(MLIM, irisNA, iris, varwise = TRUE)

## End(Not run)
```

---

mlim.mids *prepare "mids" class object*

Description

takes "mlim" object and prepares a "mids" class for data analysis with multiple imputation.

Usage

```
mlim.mids(mlim, incomplete)
```

Arguments

mlim	array of class "mlim", returned by "mlim" function
incomplete	the original data.frame with NAs

Value

object of class 'mids', as required by 'mice' package for analyzing multiple imputation data

Author(s)

E. F. Haghish, based on code from 'prelim' frunction in missMDA R package

Examples

```
## Not run:
data(iris)
require(mice)
irisNA <- mlim.na(iris, p = 0.1, seed = 2022)

# adding unstratified NAs to all variables of a data.frame
MLIM <- mlim(irisNA, m=5, tuning_time = 180, doublecheck = T, seed = 2022)

# create the mids object for MICE package
mids <- mlim.mids(MLIM, irisNA)

# run an analysis on the mids data (just as example)
fit <- with(data=mids, exp=glm(Species~ Sepal.Length, family = "binomial"))

# then, pool the results!
summary(pool(fit))

## End(Not run)
```

---

mlim.na *add stratified/unstratified artificial missing observations*

Description

to examine the performance of imputation algorithms, artificial missing data are added to datasets and then imputed, to compare the original observations with the imputed values. this function can add stratified or unstratified artificial missing data. stratified missing data can be particularly useful if your categorical or ordinal variables are imbalanced, i.e., one category appears at a much higher rate than others.

Usage

```
mlim.na(x, p = 0.1, stratify = FALSE, classes = NULL, seed = NULL)
```

Arguments

x	data.frame. x must be strictly a data.frame and any other data.table classes will be rejected
p	percentage of missingness to be added to the data

stratify	logical. if TRUE (default), stratified sampling will be carried out, when adding NA values to 'factor' variables (either ordered or unordered). this feature makes evaluation of missing data imputation algorithms more fair, especially when the factor levels are imbalanced.
classes	character vector, specifying the variable classes that should be selected for adding NA values. the default value is NULL, meaning all variables will receive NA values with probability of 'p'. however, if you wish to add NA values only to a specific classes, e.g. 'numeric' variables or 'ordered' factors, specify them in this argument. e.g. write "classes = c('numeric', 'ordered')" if you wish to add NAs only to numeric and ordered factors.
seed	integer. a random seed number for reproducing the result (recommended)

Value

data.frame

Author(s)

E. F. Haghish

Examples

```
## Not run:
# adding stratified NA to an atomic vector
x <- as.factor(c(rep("M", 100), rep("F", 900)))
table(mlim.na(x, p=.5, stratify = TRUE))

# adding unstratified NAs to all variables of a data.frame
data(iris)
mlim.na(iris, p=0.5, stratify = FALSE, seed = 1)

# or add stratified NAs only to factor variables, ignoring other variables
mlim.na(iris, p=0.5, stratify = TRUE, classes = "factor", seed = 1)

# or add NAs to numeric variables
mlim.na(iris, p=0.5, classes = "numeric", seed = 1)

## End(Not run)
```

---

mlim.preimpute *carries out preimputation*

Description

instead of replacing missing data with mean and mode, a smarter start-point would be to use fast imputation algorithms and then optimize the imputed dataset with mlim. this procedure usually requires less iterations and will savea lot of computation resources.

Usage

```
mlim.preimpute(data, preimpute = "RF", seed = NULL)
```

Arguments

data	data.frame with missing values
preimpute	character. specify the algorithm for preimputation. the supported options are "RF" (Random Forest), "mm" (mean-mode replacement), and "random" (random sampling from available data). the default is "RF", which carries a parallel random forest imputation, using all the CPUs available. the other alternative is "mm" which performs mean/mode imputation.
seed	integer. specify the random generator seed

Value

imputed data.frame

Author(s)

E. F. Haghish

Examples

```
## Not run:
data(iris)

# add 10% stratified missing values to one factor variable
irisNA <- iris
irisNA$Species <- mlim.na(irisNA$Species, p = 0.1, stratify = TRUE, seed = 2022)

# run the default random forest preimputation
MLIM <- mlim.preimpute(irisNA)
mlim.error(MLIM, irisNA, iris)

## End(Not run)
```

---

mlim.summarize *mlim imputation summary*

Description

provides information about estimated accuracy of the imputation as well as the overall procedure of the imputation.

Usage

```
mlim.summarize(data)
```

Arguments

data	dataset imputed with mlim

Value

estimated imputation accuracy via cross-valdiation procedure

Author(s)

E. F. Haghish

Examples

```
## Not run:
data(iris)

# add 10% stratified missing values to one factor variable
irisNA <- iris
irisNA$Species <- mlim.na(irisNA$Species, p = 0.1, stratify = TRUE, seed = 2022)

# run the ELNET single imputation (fastest imputation via 'mlim')
MLIM <- mlim(irisNA)

# in single imputation, you can estimate the imputation accuracy via cross validation RMSE
mlim.summarize(MLIM)

## End(Not run)
```

Index