# Package 'mixgb'

April 7, 2025

**Title** Multiple Imputation Through 'XGBoost'

**Version** 1.5.3

**Description** Multiple imputation using 'XGBoost', subsampling, and predictive mean matching as described in Deng and Lumley (2023) <doi:10.1080/10618600.2023.2252501>. The package supports various types of variables, offers flexible settings, and enables saving an imputation model to impute new data. Data processing and memory usage have been optimised to speed up the imputation process.

**URL** `https://github.com/agnesdeng/mixgb`

**BugReports** `https://github.com/agnesdeng/mixgb/issues`

**License** GPL (>= 3)

**Encoding** UTF-8

**LazyData** true

**Imports** data.table, Matrix, mice, Rcpp, Rfast, stats, utils, xgboost (>= 1.7.5.1), magrittr

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0)

**Depends** R (>= 3.6.0)

**VignetteBuilder** knitr

**RoxygenNote** 7.3.2

**Config/testthat/edition** 3

**LinkingTo** Rcpp, RcppArmadillo

**NeedsCompilation** yes

**Author** Yongshi Deng [aut, cre] (<`https://orcid.org/0000-0001-5845-859X`>), Thomas Lumley [ths]

**Maintainer** Yongshi Deng <`agnes.yongshideng@gmail.com`>

**Repository** CRAN

**Date/Publication** 2025-04-06 23:30:02 UTC

# Contents

---

## createNA *Create missing values for a dataset*

---

### Description

This function creates missing values under the missing complete at random (MCAR) mechanism. It is for demonstration purposes only.

### Usage

```
createNA(data, var.names = NULL, p = 0.3)
```

### Arguments

| | |
|---|---|
| `data` | A complete data frame. |
| `var.names` | The names of variables where missing values will be generated. |
| `p` | The proportion of missing values in the data frame or the proportions of missing values corresponding to the variables specified in `var.names`. |

### Value

A data frame with artificial missing values

### Examples

```
# Create 30% MCAR data across all variables in a dataset
withNA.df <- createNA(data = iris, p = 0.3)

# Create 30% MCAR data in a specified variable in a dataset
withNA.df <- createNA(data = iris, var.names = c("Sepal.Length"), p = 0.3)

# Create MCAR data in several specified variables in a dataset
withNA.df <- createNA(
  data = iris,
  var.names = c("Sepal.Length", "Petal.Width", "Species"),
  p = c(0.3, 0.2, 0.1)
)
```

## data_clean *Data cleaning*

### Description

The function ‘data_clean()‘ serves the purpose of performing a preliminary check and fix some evident issues. However, the function cannot resolve all data quality-related problems.

### Usage

```
data_clean(rawdata, levels.tol = 0.2)
```

### Arguments

| | |
|---|---|
| `rawdata` | A data frame. |
| `levels.tol` | Tolerant proportion of the number of levels to the number of observations in a multiclass variable. Default: 0.2 |

### Value

A preliminary cleaned dataset

### Examples

```
rawdata <- nhanes3

rawdata[4, 4] <- NaN
rawdata[5, 5] <- Inf
rawdata[6, 6] <- -Inf

cleandata <- data_clean(rawdata = rawdata)
```

## default_params *Auxiliary function for validating xgb.params*

### Description

Auxiliary function for setting up the default XGBoost-related hyperparameters for mixgb and checking the `xgb.params` argument in `mixgb()`. For more details on XGBoost hyperparameters, please refer to XGBoost documentation on parameters.

## Usage

```
default_params(
  device = "cpu",
  tree_method = "hist",
  eta = 0.3,
  gamma = 0,
  max_depth = 3,
  min_child_weight = 1,
  max_delta_step = 0,
  subsample = 0.7,
  sampling_method = "uniform",
  colsample_bytree = 1,
  colsample_bylevel = 1,
  colsample_bynode = 1,
  lambda = 1,
  alpha = 0,
  max_leaves = 0,
  max_bin = 256,
  num_parallel_tree = 1,
  nthread = -1
)
```

## Arguments

| | |
|---|---|
| `device` | Can be either `"cpu"` or `"cuda"`. For ther options please refer to XGBoost documentation on parameters. |
| `tree_method` | Options: `"auto"`, `"exact"`, `"approx"`, and `"hist"`. Default: `"hist"`. |
| `eta` | Step size shrinkage. Default: 0.3. |
| `gamma` | Minimum loss reduction required to make a further partition on a leaf node of the tree. Default: 0 |
| `max_depth` | Maximum depth of a tree. Default: 3. |
| `min_child_weight` | Minimum sum of instance weight needed in a child. Default: 1. |
| `max_delta_step` | Maximum delta step. Default: 0. |
| `subsample` | Subsampling ratio of the data. Default: 0.7. |
| `sampling_method` | The method used to sample the data. Default: `"uniform"`. |
| `colsample_bytree` | Subsampling ratio of columns when constructing each tree. Default: 1. |
| `colsample_bylevel` | Subsampling ratio of columns for each level. Default: 1. |
| `colsample_bynode` | Subsampling ratio of columns for each node. Default: 1. |
| `lambda` | L2 regularization term on weights. Default: 1. |
| `alpha` | L1 regularization term on weights. Default: 0. |

| | |
|---|---|
| `max_leaves` | Maximum number of nodes to be added (Not used when `tree_method = "exact"`). Default: 0. |
| `max_bin` | Maximum number of discrete bins to bucket continuous features (Only used when `tree_method` is either `"hist"`, `"approx"` or `"gpu_hist"`). Default: 256. |
| `num_parallel_tree` | The number of parallel trees used for boosted random forests. Default: 1. |
| `nthread` | The number of CPU threads to be used. Default: -1 (all available threads). |

## Value

A list of hyperparameters.

## Examples

```
default_params()

xgb.params <- list(device = "cuda", subsample = 0.9, nthread = 2)
default_params(device = xgb.params$device,
               subsample = xgb.params$subsample,
               nthread = xgb.params$nthread)

xgb.params <- do.call("default_params", xgb.params)
xgb.params
```

---

`default_params_cran` *Auxiliary function for validating xgb.params compatible with XGBoost CRAN version*

---

## Description

Auxiliary function for setting up the default XGBoost-related hyperparameters for mixgb and checking the `xgb.params` argument in `mixgb()`. For more details on XGBoost hyperparameters, please refer to XGBoost documentation on parameters.

## Usage

```
default_params_cran(
  eta = 0.3,
  gamma = 0,
  max_depth = 3,
  min_child_weight = 1,
  max_delta_step,
  subsample = 0.7,
  sampling_method = "uniform",
  colsample_bytree = 1,
  colsample_bylevel = 1,
  colsample_bynode = 1,
  lambda = 1,
  alpha = 0,
  tree_method = "auto",
  max_leaves = 0,
  max_bin = 256,
  predictor = "auto",
  num_parallel_tree = 1,
  gpu_id = 0,
  nthread = -1
)
```

## Arguments

| | |
|---|---|
| `eta` | Step size shrinkage. Default: 0.3. |
| `gamma` | Minimum loss reduction required to make a further partition on a leaf node of the tree. Default: 0 |
| `max_depth` | Maximum depth of a tree. Default: 3. |
| `min_child_weight` | Minimum sum of instance weight needed in a child. Default: 1. |
| `max_delta_step` | Maximum delta step. Default: 0. |
| `subsample` | Subsampling ratio of the data. Default: 0.7. |
| `sampling_method` | The method used to sample the data. Default: `"uniform"`. |
| `colsample_bytree` | Subsampling ratio of columns when constructing each tree. Default: 1. |
| `colsample_bylevel` | Subsampling ratio of columns for each level. Default: 1. |
| `colsample_bynode` | Subsampling ratio of columns for each node. Default: 1. |
| `lambda` | L2 regularization term on weights. Default: 1. |
| `alpha` | L1 regularization term on weights. Default: 0. |
| `tree_method` | Options: `"auto"`, `"exact"`, `"approx"`, and `"hist"`. Default: `"hist"`. |
| `max_leaves` | Maximum number of nodes to be added (Not used when `tree_method = "exact"`). Default: 0. |
| `max_bin` | Maximum number of discrete bins to bucket continuous features (Only used when `tree_method` is either `"hist"`, `"approx"` or `"gpu_hist"`). Default: 256. |
| `predictor` | Default: `"auto"` |
| `num_parallel_tree` | The number of parallel trees used for boosted random forests. Default: 1. |
| `gpu_id` | Which GPU device should be used. Default: 0. |
| `nthread` | The number of CPU threads to be used. Default: -1 (all available threads). |

## Value

A list of hyperparameters.

## Examples

```
default_params_cran()

xgb.params <- list(subsample = 0.9, gpu_id = 1)
default_params_cran(subsample = xgb.params$subsample, gpu_id = xgb.params$gpu_id)

xgb.params <- do.call("default_params_cran", xgb.params)
xgb.params
```

---

## impute_new — *Impute new data with a saved* mixgb *imputer object*

---

### Description

Impute new data with a saved mixgb imputer object

### Usage

```
impute_new(
  object,
  newdata,
  initial.newdata = FALSE,
  pmm.k = NULL,
  m = NULL,
  verbose = FALSE
)
```

### Arguments

| | |
|---|---|
| object | A saved imputer object created by mixgb(..., save.models = TRUE) |
| newdata | A data.frame or data.table. New data with missing values. |
| initial.newdata | Whether to use the information from the new data to initially impute the missing values of the new data. By default, this is set to FALSE, the original data passed to mixgb() will be used for initial imputation. |
| pmm.k | The number of donors for predictive mean matching. If NULL (the default), the pmm.k value in the saved imputer object will be used. |
| m | The number of imputed datasets. If NULL (the default), the m value in the saved imputer object will be used. |
| verbose | Verbose setting for mixgb. If TRUE, will print out the progress of imputation. Default: FALSE. |

### Value

A list of m imputed datasets for new data.

## Examples

```
set.seed(2022)
n <- nrow(nhanes3)
idx <- sample(1:n, size = round(0.7 * n), replace = FALSE)
train.data <- nhanes3[idx, ]
test.data <- nhanes3[-idx, ]

params <- list(max_depth = 3, subsample = 0.7, nthread = 2)
mixgb.obj <- mixgb(data = train.data, m = 2, xgb.params = params, nrounds = 10,
                   save.models = TRUE, save.models.folder = tempdir())

# obtain m imputed datasets for train.data
train.imputed <- mixgb.obj$imputed.data
train.imputed

# use the saved imputer to impute new data
test.imputed <- impute_new(object = mixgb.obj, newdata = test.data)
test.imputed
```

---

`mixgb` *Multiple imputation through XGBoost*

---

## Description

This function is used to generate multiply-imputed datasets using XGBoost, subsampling and predictive mean matching (PMM).

## Usage

```
mixgb(
  data,
  m = 5,
  maxit = 1,
  ordinalAsInteger = FALSE,
  pmm.type = NULL,
  pmm.k = 5,
  pmm.link = "prob",
  initial.num = "normal",
  initial.int = "mode",
  initial.fac = "mode",
  save.models = FALSE,
  save.vars = NULL,
  save.models.folder = NULL,
  verbose = F,
  xgb.params = list(),
  nrounds = 100,
  early_stopping_rounds = NULL,
  print_every_n = 10L,
  xgboost_verbose = 0,
  ...
)
```

## Arguments

| | |
|---|---|
| `data` | A data.frame or data.table with missing values |
| `m` | The number of imputed datasets. Default: 5 |
| `maxit` | The number of imputation iterations. Default: 1 |
| `ordinalAsInteger` | Whether to convert ordinal factors to integers. By default, `ordinalAsInteger = FALSE`. Setting `ordinalAsInteger = TRUE` may speed up the imputation process for large datasets. |
| `pmm.type` | The type of predictive mean matching (PMM). Possible values: <br>• `NULL` (default): Imputations without PMM; <br>• `0`: Imputations with PMM type 0; <br>• `1`: Imputations with PMM type 1; <br>• `2`: Imputations with PMM type 2; <br>• `"auto"`: Imputations with PMM type 2 for numeric/integer variables; imputations without PMM for categorical variables. |
| `pmm.k` | The number of donors for predictive mean matching. Default: 5 |
| `pmm.link` | The link for predictive mean matching in binary variables <br>• `"prob"` (default): use probabilities; <br>• `"logit"`: use logit values. |
| `initial.num` | Initial imputation method for numeric type data: <br>• `"normal"` (default); <br>• `"mean"`; <br>• `"median"`; <br>• `"mode"`; <br>• `"sample"`. |
| `initial.int` | Initial imputation method for integer type data: <br>• `"mode"` (default); <br>• `"sample"`. |
| `initial.fac` | Initial imputation method for factor type data: <br>• `"mode"` (default); <br>• `"sample"`. |
| `save.models` | Whether to save imputation models for imputing new data later on. Default: `FALSE` |
| `save.vars` | For the purpose of imputing new data, the imputation models for response variables specified in `save.vars` will be saved. The values in `save.vars` can be a vector of names or indices. By default, only the imputation models for variables with missing values in the original data will be saved (`save.vars = NULL`). To save imputation models for all variables, users can specify `save.vars = colnames(data)`. |

| | |
|---|---|
| `save.models.folder` | Users can specify a directory to save all imputation models. Models will be saved in JSON format by internally calling `xgb.save()`, which is recommended by XGBoost. |
| `verbose` | Verbose setting for mixgb. If `TRUE`, will print out the progress of imputation. Default: `FALSE`. |
| `xgb.params` | A list of XGBoost parameters. For more details, please check XGBoost documentation on parameters. |
| `nrounds` | The maximum number of boosting iterations for XGBoost. Default: 100 |
| `early_stopping_rounds` | An integer value `k`. XGBoost training will stop if the validation performance has not improved for `k` rounds. Default: 10. |
| `print_every_n` | Print XGBoost evaluation information at every nth iteration if `xgboost_verbose > 0`. |
| `xgboost_verbose` | Verbose setting for XGBoost training: 0 (silent), 1 (print information) and 2 (print additional information). Default: 0 |
| `...` | Extra arguments to be passed to XGBoost |

## Value

If `save.models = FALSE`, this function will return a list of `m` imputed datasets. If `save.models = TRUE`, it will return an object with imputed datasets, saved models and parameters.

## Examples

```
# obtain m multiply datasets without saving models
params <- list(max_depth = 3, subsample = 0.7, nthread = 2)
mixgb.data <- mixgb(data = nhanes3, m = 2, xgb.params = params, nrounds = 10)

# obtain m multiply imputed datasets and save models for imputing new data later on
mixgb.obj <- mixgb(data = nhanes3, m = 2, xgb.params = params, nrounds = 10,
                   save.models = TRUE, save.models.folder = tempdir())
```

---

`mixgb_cv` *Use cross-validation to find the optimal* `nrounds`

---

## Description

Use cross-validation to find the optimal `nrounds` for an `Mixgb` imputer. Note that this method relies on the complete cases of a dataset to obtain the optimal `nrounds`.

## Usage

```
mixgb_cv(
  data,
  nfold = 5,
  nrounds = 100,
  early_stopping_rounds = 10,
  response = NULL,
  select_features = NULL,
  xgb.params = list(),
  stringsAsFactors = FALSE,
  verbose = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `data` | A data.frame or a data.table with missing values. |
| `nfold` | The number of subsamples which are randomly partitioned and of equal size. Default: 5 |
| `nrounds` | The max number of iterations in XGBoost training. Default: 100 |
| `early_stopping_rounds` | An integer value `k`. Training will stop if the validation performance has not improved for `k` rounds. |
| `response` | The name or the column index of a response variable. Default: `NULL` (Randomly select an incomplete variable). |
| `select_features` | The names or the indices of selected features. Default: `NULL` (Select all the other variables in the dataset). |
| `xgb.params` | A list of XGBoost parameters. For more details, please check XGBoost documentation on parameters. |
| `stringsAsFactors` | A logical value indicating whether all character vectors in the dataset should be converted to factors. |
| `verbose` | A logical value. Whether to print out cross-validation results during the process. |
| `...` | Extra arguments to be passed to XGBoost. |

## Value

A list of the optimal `nrounds`, `evaluation.log` and the chosen `response`.

## Examples

```
params <- list(max_depth = 3, subsample = 0.7, nthread = 2)
cv.results <- mixgb_cv(data = nhanes3, xgb.params = params)
cv.results$best.nrounds

imputed.data <- mixgb(data = nhanes3, m = 3, xgb.params = params,
                      nrounds = cv.results$best.nrounds)
```

`nhanes3` *A small subset of the NHANES III (1988-1994) newborn data*

## Description

This dataset is a small subset of `nhanes3_newborn`. It is for demonstration purposes only. More information on NHANES III data can be found on https://wwwn.cdc.gov/Nchs/Data/Nhanes3/7a/doc/mimodels.pdf

## Usage

```
data(nhanes3)
```

## Format

A data frame of 500 rows and 6 variables. Three variables have missing values.

**HSAGEIR** Age at interview (screener) - qty (months). An integer variable from 2 to 11.

**HSSEX** Sex. A factor variable with levels 1 (Male) and 2 (Female).

**DMARETHN** Race-ethnicity. A factor variable with levels 1 (Non-Hispanic white), 2 (Non-Hispanic black), 3 (Mexican-American) and 4 (Other).

**BMPHEAD** Head circumference (cm). Numeric.

**BMPRECUM** Recumbent length (cm). Numeric.

**BMPWT** Weight (kg). Numeric.

## Source

https://wwwn.cdc.gov/nchs/nhanes/nhanes3/datafiles.aspx

## References

U.S. Department of Health and Human Services (DHHS). National Center for Health Statistics. Third National Health and Nutrition Examination Survey (NHANES III, 1988-1994): Multiply Imputed Data Set. CD-ROM, Series 11, No. 7A. Hyattsville, MD: Centers for Disease Control and Prevention, 2001. Includes access software: Adobe Systems, Inc. Acrobat Reader version 4.

## `nhanes3_newborn` *NHANES III (1988-1994) newborn data*

### Description

This dataset is extracted from the NHANES III (1988-1994) for the age class `Newborn (under 1 year)`. Please note that this example dataset only contains selected variables and is for demonstration purposes only.

### Usage

```
data(nhanes3_newborn)
```

### Format

A data frame of 2107 rows and 16 variables. Nine variables have missing values.

**HSHSIZER** Household size. An integer variable from 1 to 10.

**HSAGEIR** Age at interview (screener) - qty (months). An integer variable from 2 to 11.

**HSSEX** Sex. A factor variable with levels 1 (Male) and 2 (Female).

**DMARACER** Race. A factor variable with levels 1 (White), 2 (Black) and 3 (Other).

**DMAETHNR** Ethnicity. A factor variable with levels 1 (Mexican-American), 2 (Other Hispanic) and 3 (Not Hispanic).

**DMARETHN** Race-ethnicity. A factor variable with levels 1 (Non-Hispanic white), 2 (Non-Hispanic black), 3 (Mexican-American) and 4 (Other).

**BMPHEAD** Head circumference (cm). Numeric.

**BMPRECUM** Recumbent length (cm). Numeric.

**BMPSB1** First subscapular skinfold (mm). Numeric.

**BMPSB2** Second subscapular skinfold (mm). Numeric.

**BMPTR1** First triceps skinfold (mm). Numeric.

**BMPTR2** Second triceps skinfold (mm). Numeric.

**BMPWT** Weight (kg). Numeric.

**DMPPIR** Poverty income ratio. Numeric.

**HFF1** Does anyone who lives here smoke cigarettes in the home? A factor variable with levels 1 (Yes) and 2 (No).

**HYD1** How is the health of subject person in general? An ordinal factor with levels 1 (Excellent), 2 (Very good), 3 (Good), 4 (Fair) and 5 (Poor).

### Source

https://wwwn.cdc.gov/nchs/nhanes/nhanes3/datafiles.aspx

## References

U.S. Department of Health and Human Services (DHHS). National Center for Health Statistics. Third National Health and Nutrition Examination Survey (NHANES III, 1988-1994): Multiply Imputed Data Set. CD-ROM, Series 11, No. 7A. Hyattsville, MD: Centers for Disease Control and Prevention, 2001. Includes access software: Adobe Systems, Inc. Acrobat Reader version 4.

---

## `show_var` *Show multiply imputed values for a single variable*

---

### Description

Show m sets of imputed values for a specified variable.

### Usage

```
show_var(imputation.list, var.name, original.data, true.values = NULL)
```

### Arguments

| | |
|---|---|
| `imputation.list` | A list of `m` imputed datasets returned by the `mixgb` imputer. |
| `var.name` | The name of a variable of interest. |
| `original.data` | The original data with missing data. |
| `true.values` | A vector of the true values (if known) of the missing values. In general, this is unknown. |

### Value

A data.table with `m` columns, each column represents the imputed values of all missing entries in the specified variable. If `true.values` is provided, the last column will be the true values of the missing values.

### Examples

```
#obtain m multiply datasets
library(mixgb)
mixgb.data <- mixgb(data = nhanes3, m = 3)

imputed.BMPHEAD <- show_var(imputation.list = mixgb.data, var.name = "BMPHEAD",
  original.data = nhanes3)
```

# Index