Package: simputation (via r-universe)

October 13, 2024

Maintainer Mark van der Loo <mark.vanderloo@gmail.com>

License GPL-3

Title Simple Imputation

Type Package

LazyLoad yes

Description Easy to use interfaces to a number of imputation methods that fit in the not-a-pipe operator of the 'magrittr' package.

Version 0.2.8

Depends R (>= 4.0.0)

Imports stats, utils, MASS, rpart, gower, VIM, randomForest, glmnet, missForest, norm

URL https://github.com/markvanderloo/simputation

BugReports https://github.com/markvanderloo/simputation/issues

Suggests tinytest, knitr, rmarkdown, dplyr

RoxygenNote 7.2.0

VignetteBuilder knitr

NeedsCompilation yes

Author Mark van der Loo [aut, cre]

Repository CRAN

Date/Publication 2022-06-16 14:30:02 UTC

Contents

`deparse` *A* `deparse` *replacement that always returns a length-1 vector*

Description

A `deparse` replacement that always returns a length-1 vector

Usage

```
deparse(...)
```

Arguments

`...`	Arguments passed on to `base::deparse()`

Value

The deparsed string

Examples

```
long_formula <- this_is_a_formula_with_long_variables ~
  the_test_is_checking_if_deparse_will_return +
  multiple_strings_or_not
simputation:::deparse(long_formula)
```

`foretell` *Alternative to 'predict' returning values of correct type.*

Description

Te default `precict` function doesn't always return the predicted variable by default. For example, when estimating a binomial model using `glm`, by default the log-odds are returned. `foretell` wraps `predict` while setting options so that the actual predicted value is returned.

Usage

```
foretell(object, ...)

## Default S3 method:
foretell(object, ...)

## S3 method for class 'glm'
foretell(object, newdata = NULL, type, ...)

## S3 method for class 'rpart'
foretell(object, newdata, type, ...)
```

Arguments

object	A model object,(lm, glm, ...)
...	Furher arguments passed to predict.
newdata	[data.frame] a data frame in which to look for variables with which to predict. For glmnet models this argument is mandatory.
type	[character] Type of output. If missing, the type of predicted variable is returned.

---

glimpse_na *Show the number of (remaining) missing values.*

Description

Quick indication of the amount and location of missing values. The function uses na_status to print the missing values, but returns the original x (invisibly) and therefore can be used in an imputation pipeline to peek at the NA's status.

Usage

```
glimpse_na(x, show_only_missing = TRUE, ...)

lhs %?>% rhs
```

Arguments

x	an R object caryying data (e.g. data.frame)
show_only_missing	if TRUE only columns with NA's will be printed.
...	arguments passed to na_status.
lhs	left hand side of pipe
rhs	right hand side of pipe

Details

glimpse_na is especially helpful when interactively adding imputation methods. glimpse_na is named after glimpse in dplyr.

Operator %?>% is syntactic sugar: it inserts a glimpse_na in the pipe.

Examples

```
irisNA <- iris
irisNA[1:3,1] <- irisNA[3:7,2] <- NA

# How many NA's?
na_status(irisNA)

# add an imputation method one at a time
iris_imputed <-
  irisNA |>
  glimpse_na() # same as above

# ok, glimpse_na says "Sepal.Width" has NA's
# fix that:

iris_imputed <-
  irisNA |>
  impute_const(Sepal.Width ~ 7) |>
  glimpse_na() # end NA

# Sepal.Length is having NA's

iris_imputed <-
  irisNA |>
  impute_const(Sepal.Width ~ 7) |>
  impute_cart(Sepal.Length ~ .) |>
  glimpse_na() # end NA

# in an existing imputation pipeline we can peek with
# glimpse_na or %?>%

iris_imputed <-
  irisNA |>
  glimpse_na() |>     # shows the begin NA
  impute_const(Sepal.Width ~ 7) |>
  glimpse_na() |>     # after 1 imputation
  impute_cart(Sepal.Length ~ .) |>
  glimpse_na()         # end NA

# or
iris_imputed <-
  irisNA %?>%
  impute_const(Sepal.Width ~ 7) %?>%
  impute_cart(Sepal.Length ~ .)
```

```
na_status(iris_imputed)
```

`impute` *Impute using a previously fitted model.*

Description

Impute one or more variables using a single R object representing a previously fitted model.

Usage

```
impute(dat, formula, predictor = foretell, ...)

impute_(dat, variables, model, predictor = foretell, ...)
```

Arguments

`dat`	`[data.frame]` The data to be imputed.
`formula`	`[formula]` object of the form `<imputed variables> ~ <model object>`
`predictor`	`[function]` with signature `object, newdata, ...` that returns predicted values given a model `object` and a new dataset `newdata`. By default `foretell` is used.
`...`	Extra arguments passed to `predictor`
`variables`	`[character]` Names of columns in `dat` to impute.
`model`	A model object.

Model specification

Formulas are of the form

`IMPUTED_VARIABLES ~ MODEL_OBJECT`

The left-hand-side of the formula object lists the variable or variables to be imputed. The right-hand-side must be a model object for which an `S3 predict` method is implemented. Alternatively, one can specify a custom predicting function. This function must accept at least a model and a dataset, and return one predicted value for each row in the dataset.

`foretell` implements usefull `predict` methods for cases where by default the predicted output is not of the same type as the predicted variable (e.g. when using certain link functions in `glm`)

Details

`impute_` is an explicit version of `impute` that works better in programming contexts, especially in cases involving nonstandard evaluation.

See Also

Other imputation: `impute_cart()`, `impute_hotdeck`, `impute_lm()`

Examples

```
irisNA <- iris
iris[1:3,1] <- NA
my_model <- lm(Sepal.Length ~ Sepal.Width + Species, data=iris)
impute(irisNA, Sepal.Length ~ my_model)
```

---

impute_cart *Decision Tree Imputation*

Description

Imputation based on CART models or Random Forests.

Usage

```
impute_cart(
  dat,
  formula,
  add_residual = c("none", "observed", "normal"),
  cp,
  na_action = na.rpart,
  ...
)

impute_rf(
  dat,
  formula,
  add_residual = c("none", "observed", "normal"),
  na_action = na.omit,
  ...
)
```

Arguments

`dat`	`[data.frame]`, with variables to be imputed and their predictors.
`formula`	`[formula]` imputation model description (see Details below).
`add_residual`	`[character]` Type of residual to add. `"normal"` means that the imputed value is drawn from `N(mu, sd)` where `mu` and `sd` are estimated from the model's residuals (`mu` should equal zero in most cases). If `add_residual = "observed"`, residuals are drawn (with replacement) from the model's residuals. Ignored for non-numeric predicted variables.

cp — The complexity parameter used to prune the CART model. If omitted, no pruning takes place. If a single number, the same complexity parameter is used for each imputed variable. If of length # of variables imputed, the complexity parameters used must be in the same order as the predicted variables in the model formula.

na_action — [function] what to do with missings in training data. By default cases with missing values in predicted or predictors are omitted (see 'Missings in training data').

... — further arguments passed to

- rpart for impute_cart
- randomForest for impute_rf

Model specification

Formulas are of the form

IMPUTED_VARIABLES ~ MODEL_SPECIFICATION [| GROUPING_VARIABLES]

The left-hand-side of the formula object lists the variable or variables to be imputed. Variables on the right-hand-side are used as predictors in the CART or random forest model.

If grouping variables are specified, the data set is split according to the values of those variables, and model estimation and imputation occur independently for each group.

Grouping using dplyr::group_by is also supported. If groups are defined in both the formula and using dplyr::group_by, the data is grouped by the union of grouping variables. Any missing value in one of the grouping variables results in an error.

Methodology

CART imputation by impute_cart can be used for numerical, categorical, or mixed data. Missing values are estimated using a Classification and Regression Tree as specified by Breiman, Friedman and Olshen (1984). This means that prediction is fairly robust agains missingess in predictors.

Random Forest imputation with impute_rf can be used for numerical, categorical, or mixed data. Missing values are estimated using a Random Forest model as specified by Breiman (2001).

References

Breiman, L., Friedman, J., Stone, C.J. and Olshen, R.A., 1984. Classification and regression trees. CRC press.

Breiman, L., 2001. Random forests. Machine learning, 45(1), pp.5-32.

See Also

Other imputation: impute_hotdeck, impute_lm(), impute()

`impute_hotdeck` *Hot deck imputation*

Description

Hot-deck imputation methods include random and sequential hot deck, k-nearest neighbours imputation and predictive mean matching.

Usage

```
impute_rhd(
  dat,
  formula,
  pool = c("complete", "univariate", "multivariate"),
  prob,
 backend = getOption("simputation.hdbackend", default = c("simputation", "VIM")),
  ...
)

impute_shd(
  dat,
  formula,
  pool = c("complete", "univariate", "multivariate"),
  order = c("locf", "nocb"),
 backend = getOption("simputation.hdbackend", default = c("simputation", "VIM")),
  ...
)

impute_pmm(
  dat,
  formula,
  predictor = impute_lm,
  pool = c("complete", "univariate", "multivariate"),
  ...
)

impute_knn(
  dat,
  formula,
  pool = c("complete", "univariate", "multivariate"),
  k = 5,
 backend = getOption("simputation.hdbackend", default = c("simputation", "VIM")),
  ...
)
```

Arguments

`dat`	`[data.frame]`, with variables to be imputed and their predictors.

formula — [formula] imputation model description (see Details below).

pool — [character] Specify donor pool when backend="simputation"

- "complete". Only records for which the variables on the left-hand-side of the model formula are complete are used as donors. If a record has multiple missings, all imputations are taken from a single donor.
- "univariate". Imputed variables are treated one by one and independently so the order of variable imputation is unimportant. If a record has multiple missings, separate donors are drawn for each missing value.
- "multivariate". A donor pool is created for each missing data pattern. If a record has multiple missings, all imputations are taken from a single donor.

prob — [numeric] Sampling probability weights (passed through to sample). Must be of length nrow(dat).

backend — [character] Choose the backend for imputation. If backend="VIM" the variables used to sort the data (in case of sequential hot deck) may not coincide with imputed variables.

... — further arguments passed to VIM::hotdeck if VIM is chosen as backend, otherwise they are passed to

- order for impute_shd and backend="simputation"
- VIM::hotdeck for impute_shd and impute_rhd when backend="VIM".
- VIM:kNN for impute_knn when backend="VIM"
- The predictor function for impute_pmm.

order — [character] Last Observation Carried Forward or Next Observarion Carried Backward. Only for backend="simputation"

predictor — [function] Imputation to use for predictive part in predictive mean matching. Any of the impute_ functions of this package (it makes no sense to use a hotdeck imputation).

k — [numeric] Number of nearest neighbours to draw the donor from.

Model specification

Formulas are of the form

IMPUTED_VARIABLES ~ MODEL_SPECIFICATION [| GROUPING_VARIABLES]

The left-hand-side of the formula object lists the variable or variables to be imputed. The interpretation of the independent variables on the right-hand-side depends on the imputation method.

- impute_rhd Variables in MODEL_SPECIFICATION and/or GROUPING_VARIABLES are used to split the data set into groups prior to imputation. Use ~ 1 to specify that no grouping is to be applied.
- impute_shd Variables in MODEL_SPECIFICATION are used to sort the data. When multiple variables are specified, each variable after the first serves as tie-breaker for the previous one.
- impute_knn The predictors are used to determine Gower's distance between records (see gower_topn). This may include the variables to be imputed..

- `impute_pmm` Predictive mean matching. The `MODEL_SPECIFICATION` is passed through to the `predictor` function.

If grouping variables are specified, the data set is split according to the values of those variables, and model estimation and imputation occur independently for each group.

Grouping using `dplyr::group_by` is also supported. If groups are defined in both the formula and using `dplyr::group_by`, the data is grouped by the union of grouping variables. Any missing value in one of the grouping variables results in an error.

Methodology

Random hot deck imputation with `impute_rhd` can be applied to numeric, categorical or mixed data. A missing value is copied from a sampled record. Optionally samples are taken within a group, or with non-uniform sampling probabilities. See Andridge and Little (2010) for an overview of hot deck imputation methods.

Sequential hot deck imputation with `impute_rhd` can be applied to numeric, categorical, or mixed data. The dataset is sorted using the 'predictor variables'. Missing values or combinations thereof are copied from the previous record where the value(s) are available in the case of LOCF and from the next record in the case of NOCF.

Predictive mean matching with `impute_pmm` can be applied to numeric data. Missing values or combinations thereof are first imputed using a predictive model. Next, these predictions are replaced with observed (combinations of) values nearest to the prediction. The nearest value is the observed value with the smallest absolute deviation from the prediction.

K-nearest neighbour imputation with `impute_knn` can be applied to numeric, categorical, or mixed data. For each record containing missing values, the k most similar completed records are determined based on Gower's (1977) similarity coefficient. From these records the actual donor is sampled.

Using the VIM backend

The VIM package has efficient implementations of several popular imputation methods. In particular, its random and sequential hotdeck implementation is faster and more memory-efficient than that of the current package. Moreover, **VIM** offers more fine-grained control over the imputation process then **simputation**.

If you have this package installed, it can be used by setting `backend="VIM"` for functions supporting this option. Alternatively, one can set `options(simputation.hdbackend="VIM")` so it becomes the default.

Simputation will map the simputation call to a function in the **VIM** package. In particular:

- `impute_rhd` is mapped to `VIM::hotdeck` where imputed variables are passed to the `variable` argument and the union of predictor and grouping variables are passed to `domain_var`. Extra arguments in `...` are passed to `VIM::hotdeck` as well. Argument `pool` is ignored.
- `impute_shd` is mapped to `VIM::hotdeck` where imputed variables are passed to the `variable` argument, predictor variables to `ord_var` and grouping variables to `domain_var`. Extra arguments in `...` are passed to `VIM::hotdeck` as well. Arguments `pool` and `order` are ignored. In `VIM` the donor pool is determined on a per-variable basis, equivalent to setting `pool="univariate"` with the simputation backend. **VIM** is LOCF-based. Differences between **simputation** and `VIM` likely occurr when the sorting variables contain missings.

- `impute_knn` is mapped to `VIM::kNN` where imputed variables are passed to `variable`, predictor variables are passed to `dist_var` and grouping variables are ignored with a message. Extra arguments in `...` are passed to `VIM::kNN` as well. Argument `pool` is ignored. Note that simputation adheres stricktly to the Gower's original definition of the distance measure, while **VIM** uses a generalized variant that can take ordered factors into account.

By default, **VIM**'s imputation functions add indicator variables to the original data to trace what values have been imputed. This is switched off by default for consistency with the rest of the simputation package, but it may be turned on again by setting `imp_var=TRUE`.

References

Andridge, R.R. and Little, R.J., 2010. A review of hot deck imputation for survey non-response. International statistical review, 78(1), pp.40-64.

Gower, J.C., 1971. A general coefficient of similarity and some of its properties. Biometrics, pp.857–871.

See Also

Other imputation: `impute_cart()`, `impute_lm()`, `impute()`

`impute_lm` *(Robust) Linear Regression Imputation*

Description

Regression imputation methods including linear regression, robust linear regression with M-estimators, regularized regression with lasso/elasticnet/ridge regression.

Usage

```
impute_lm(
  dat,
  formula,
  add_residual = c("none", "observed", "normal"),
  na_action = na.omit,
  ...
)

impute_rlm(
  dat,
  formula,
  add_residual = c("none", "observed", "normal"),
  na_action = na.omit,
  ...
)
```

```
impute_en(
  dat,
  formula,
  add_residual = c("none", "observed", "normal"),
  na_action = na.omit,
  family = c("gaussian", "poisson"),
  s = 0.01,
  ...
)
```

Arguments

dat	[data.frame], with variables to be imputed and their predictors.
formula	[formula] imputation model description (See Model description)
add_residual	[character] Type of residual to add. "normal" means that the imputed value is drawn from N(mu,sd) where mu and sd are estimated from the model's residuals (mu should equal zero in most cases). If add_residual = "observed", residuals are drawn (with replacement) from the model's residuals. Ignored for non-numeric predicted variables.
na_action	[function] what to do with missings in training data. By default cases with missing values in predicted or predictors are omitted (see 'Missings in training data').
...	further arguments passed to • lm for impute_lm • rlm for impute_rlm • glmnet for impute_en
family	Response type for elasticnet / lasso regression. For family="gaussian" the imputed variables are general numeric variables. For family="poisson" the imputed variables are nonnegative counts. See glmnet for details.
s	The value of λ to use when computing predictions for lasso/elasticnet regression (parameter *s* of predict.glmnet). For impute_en the (optional) parameter *lambda* is passed to glmnet when estimating the model (which is advised against).

Value

dat, but imputed where possible.

Model specification

Formulas are of the form

IMPUTED_VARIABLES ~ MODEL_SPECIFICATION [| GROUPING_VARIABLES]

The left-hand-side of the formula object lists the variable or variables to be imputed. The right-hand side excluding the optional GROUPING_VARIABLES model specification for the underlying predictor.

If grouping variables are specified, the data set is split according to the values of those variables, and model estimation and imputation occur independently for each group.

Grouping using `dplyr::group_by` is also supported. If groups are defined in both the formula and using `dplyr::group_by`, the data is grouped by the union of grouping variables. Any missing value in one of the grouping variables results in an error.

Grouping is ignored for `impute_const`.

Methodology

Linear regression model imputation with `impute_lm` can be used to impute numerical variables based on numerical and/or categorical predictors. Several common imputation methods, including ratio and (group) mean imputation can be expressed this way. See `lm` for details on possible model specification.

Robust linear regression through M-estimation with `impute_rlm` can be used to impute numerical variables employing numerical and/or categorical predictors. In M-estimation, the minimization of the squares of residuals is replaced with an alternative convex function of the residuals that decreases the influence of outliers.

Also see e.g. Huber (1981).

Lasso/elastic net/ridge regression imputation with `impute_en` can be used to impute numerical variables employing numerical and/or categorical predictors. For this method, the regression coefficients are found by minimizing the least sum of squares of residuals augmented with a penalty term depending on the size of the coefficients. For lasso regression (Tibshirani, 1996), the penalty term is the sum of squares of the coefficients. For ridge regression (Hoerl and Kennard, 1970), the penalty term is the sum of absolute values of the coefficients. Elasticnet regression (Zou and Hastie, 2010) allows switching from lasso to ridge by penalizing by a weighted sum of the sum-of-squares and sum of absolute values term.

References

Huber, P.J., 2011. Robust statistics (pp. 1248-1251). Springer Berlin Heidelberg.

Hoerl, A.E. and Kennard, R.W., 1970. Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, 12(1), pp.55-67.

Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), pp.267-288.

Zou, H. and Hastie, T., 2005. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67(2), pp.301-320.

See Also

Getting started with simputation,

Other imputation: `impute_cart()`, `impute_hotdeck`, `impute()`

Examples

```
data(iris)
irisNA <- iris
irisNA[1:4, "Sepal.Length"] <- NA
irisNA[3:7, "Sepal.Width"] <- NA
```

```
# impute a single variable (Sepal.Length)
i1 <- impute_lm(irisNA, Sepal.Length ~ Sepal.Width + Species)

# impute both Sepal.Length and Sepal.Width, using robust linear regression
i2 <- impute_rlm(irisNA, Sepal.Length + Sepal.Width ~ Species + Petal.Length)
```

`impute_median` *Impute (group-wise) medians*

Description

Impute medians of group-wise medians.

Usage

```
impute_median(
  dat,
  formula,
  add_residual = c("none", "observed", "normal"),
  type = 7,
  ...
)
```

Arguments

`dat`	`[data.frame]`, with variables to be imputed and their predictors.
`formula`	`[formula]` imputation model description (See Model description)
`add_residual`	`[character]` Type of residual to add. `"normal"` means that the imputed value is drawn from `N(mu, sd)` where `mu` and `sd` are estimated from the model's residuals (`mu` should equal zero in most cases). If `add_residual = "observed"`, residuals are drawn (with replacement) from the model's residuals. Ignored for non-numeric predicted variables.
`type`	`[integer]` Specifies the algorithm to compute the median. See the 'details' section of `quantile`.
`...`	Currently not used.

Model Specification

Formulas are of the form

`IMPUTED_VARIABLES ~ MODEL_SPECIFICATION [ | GROUPING_VARIABLES ]`

The left-hand-side of the formula object lists the variable or variables to be imputed. Variables in `MODEL_SPECIFICATION` and/or `GROUPING_VARIABLES` are used to split the data set into groups prior to imputation. Use `~ 1` to specify that no grouping is to be applied.

Examples

```
# group-wise median imputation
irisNA <- iris
irisNA[1:3,1] <- irisNA[4:7,2] <- NA
a <- impute_median(irisNA, Sepal.Length ~ Species)
head(a)

# group-wise median imputation, all variables except species

a <- impute_median(irisNA, . - Species ~ Species)
head(a)
```

`impute_multivariate` *Multivariate, model-based imputation*

Description

Models that simultaneously optimize imptuation of multiple variables. Methods include imputation based on EM-estimation of multivariate normal parameters, imputation based on iterative Random Forest estimates and stochastic imptuation based on bootstrapped EM-estimatin of multivariate normal parameters.

Usage

```
impute_em(dat, formula, verbose = 0, ...)

impute_mf(dat, formula, ...)
```

Arguments

`dat`	`[data.frame]` with variables to be imputed.
`formula`	`[formula]` imputation model description
`verbose`	`[numeric]` Control amount of output printed to screen. Higher values mean more output, typically per iteration. • 0 or a number ≥ 1 for `impute_em` • 0, 1, or 2 for `impute_emb`
`...`	Options passed to • `norm::em.norm` for `impute_em` • `missForest::missForest` for `impute_mf`

Model specification

Formulas are of the form

`[IMPUTED_VARIABLES] ~ MODEL_SPECIFICATION [ | GROUPING_VARIABLES ]`

When `IMPUTED_VARIABLES` is empty, every variable in `MODEL_SPECIFICATION` will be imputed. When `IMPUTED_VARIABLES` is specified, all variables in `IMPUTED_VARIABLES` and `MODEL_SPECIFICATION` are part of the model, but only the `IMPUTED_VARIABLES` are imputed in the output.

`GROUPING_VARIABLES` specify what categorical variables are used to split-impute-combine the data. Grouping using `dplyr::group_by` is also supported. If groups are defined in both the formula and using `dplyr::group_by`, the data is grouped by the union of grouping variables. Any missing value in one of the grouping variables results in an error.

Methodology

EM-based imputation with `impute_em` only works for numerical variables. These variables are assumed to follow a multivariate normal distribution for which the means and covariance matrix is estimated based on the EM-algorithm of Dempster Laird and Rubin (1977). The imputations are the expected values for missing values, conditional on the value of the estimated parameters.

Multivariate Random Forest imputation with `impute_mf` works for numerical, categorical or mixed data types. It is based on the algorithm of Stekhoven and Buehlman (2012). Missing values are imputed using a rough guess after which a predictive random forest is trained and used to re-impute themissing values. This is iterated until convergence.

References

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." Journal of the royal statistical society. Series B (methodological) (1977): 1-38.

Stekhoven, D.J. and Buehlmann, P., 2012. MissForest—non-parametric missing value imputation for mixed-type data. Bioinformatics, 28(1), pp.112-118.

`impute_proxy` *Impute by variable derivation*

Description

Impute missing values by a constant, by copying another variable computing transformations from other variables.

Usage

```
impute_proxy(dat, formula, add_residual = c("none", "observed", "normal"), ...)

impute_const(dat, formula, add_residual = c("none", "observed", "normal"), ...)
```

Arguments

dat	[data.frame], with variables to be imputed and their predictors.
formula	[formula] imputation model description (See Model description)
add_residual	[character] Type of residual to add. "normal" means that the imputed value is drawn from N(mu,sd) where mu and sd are estimated from the model's residuals (mu should equal zero in most cases). If add_residual = "observed", residuals are drawn (with replacement) from the model's residuals. Ignored for non-numeric predicted variables.
...	Currently unused

Model Specification

Formulas are of the form

IMPUTED_VARIABLES ~ MODEL_SPECIFICATION [| GROUPING_VARIABLES]

The left-hand-side of the formula object lists the variable or variables to be imputed.

For impute_const, the MODEL_SPECIFICATION is a single value and GROUPING_VARIABLES are ignored.

For impute_proxy, the MODEL_SPECIFICATION is a variable or expression in terms of variables in the dataset that must result in either a single number of in a vector of length nrow(dat).

If grouping variables are specified, the data set is split according to the values of those variables, and model estimation and imputation occur independently for each group.

Grouping using dplyr::group_by is also supported. If groups are defined in both the formula and using dplyr::group_by, the data is grouped by the union of grouping variables. Any missing value in one of the grouping variables results in an error.

Examples

```
irisNA <- iris
irisNA[1:3,1] <- irisNA[3:7,2] <- NA

# impute a constant

a <- impute_const(irisNA, Sepal.Width ~ 7)
head(a)

a <- impute_proxy(irisNA, Sepal.Width ~ 7)
head(a)

# copy a value from another variable (where available)
a <- impute_proxy(irisNA, Sepal.Width ~ Sepal.Length)
head(a)

# group mean imputation
a <- impute_proxy(irisNA
  , Sepal.Length ~ mean(Sepal.Length,na.rm=TRUE) | Species)
head(a)
```

```
# random hot deck imputation
a <- impute_proxy(irisNA, Sepal.Length ~ mean(Sepal.Length, na.rm=TRUE)
, add_residual = "observed")

# ratio imputation (but use impute_lm for that)
a <- impute_proxy(irisNA,
 Sepal.Length ~ mean(Sepal.Length,na.rm=TRUE)/mean(Sepal.Width,na.rm=TRUE) * Sepal.Width)
```

---

na.roughfix *Rough imputation for handling missing predictors.*

Description

This function is re-exported from `randomForest:na.roughfix` when available. Otherwise it will throw a warning and resort to `options("na.action")`

Usage

```
na.roughfix(object, ...)
```

Arguments

`object`	an R object caryying data (e.g. `data.frame`)
`...`	arguments to be passed to other methods.

---

na_status *Show the number of (remaining) missing values.*

Description

Quick indication of the amount and location of missing values.

Usage

```
na_status(
  x,
  show_only_missing = TRUE,
  sort_columns = show_only_missing,
  show_message = TRUE,
  ...
)
```

Arguments

x	an R object caryying data (e.g. data.frame)
show_only_missing	if TRUE only columns with NA's will be printed.
sort_columns	If TRUE the columns are sorted descending by the number of missing values.
show_message	if TRUE message will be printed.
...	arguments to be passed to other methods.

Value

data.frame with the column and number of NA's

See Also

glimpse_na

Examples

```
irisNA <- iris
irisNA[1:3,1] <- irisNA[3:7,2] <- NA
na_status(irisNA)

# impute a constant
a <- impute_const(irisNA, Sepal.Width ~ 7)
na_status(a)
```

---

simputation *simputation*

Description

A package to make imputation simpler.

Details

To get started, see the introductory vignette.

Index