

# Package ‘VIM’

January 10, 2026

**Version** 7.0.0

**Title** Visualization and Imputation of Missing Values

**Depends** R (>= 4.1.0), colorspace, grid

**Imports** car, grDevices, robustbase, stats, sp, vcd, nnet, e1071, methods, Rcpp, utils, graphics, laeken, ranger, MASS, xgboost, data.table(>= 1.9.4), mlr3, mlr3pipelines, R6, paradox, mlr3tuning, mlr3learners, future

**Suggests** dplyr, tinytest, knitr, mgcv, rmarkdown, reactable, covr, withr, pdist, enetLTS, robmixglm, stringr, glmnet

**Description** Provides methods for imputation and visualization of missing values. It includes graphical tools to explore the amount, structure and patterns of missing and/or imputed values, supporting exploratory data analysis and helping to investigate potential missingness mechanisms (details in Alfons, Templ and Filzmoser, <[doi:10.1007/s11634-011-0102-y](https://doi.org/10.1007/s11634-011-0102-y)>). The quality of imputations can be assessed visually using a wide range of univariate, bivariate and multivariate plots. The package further provides several imputation methods, including efficient implementations of k-nearest neighbour and hot-deck imputation (Kowarik and Templ 2013, <[doi:10.18637/jss.v074.i07](https://doi.org/10.18637/jss.v074.i07)>, iterative robust model-based multiple imputation (Templ 2011, <[doi:10.1016/j.csda.2011.04.012](https://doi.org/10.1016/j.csda.2011.04.012)>; Templ 2023, <[doi:10.3390/math11122729](https://doi.org/10.3390/math11122729)>), and machine learning-based approaches such as robust GAM-based multiple imputation (Templ 2024, <[doi:10.1007/s11222-024-10429-1](https://doi.org/10.1007/s11222-024-10429-1)>) as well as gradient boosting (XGBoost) and transformer-based methods (Niederhametner et al., <[doi:10.1177/18747655251339401](https://doi.org/10.1177/18747655251339401)>). General background and practical guidance on imputation are provided in the Springer book by Templ (2023) <[doi:10.1007/978-3-031-30073-8](https://doi.org/10.1007/978-3-031-30073-8)>.

**LazyData** TRUE

**ByteCompile** TRUE

**License** GPL (>= 2)

**URL** <https://github.com/statistikat/VIM>

**Repository** CRAN

**LinkingTo** Rcpp

**RoxygenNote** 7.3.3

**Encoding** UTF-8

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Matthias Templ [aut, cre],  
 Alexander Kowarik [aut] (ORCID:  
<https://orcid.org/0000-0001-8598-4130>),  
 Andreas Alfons [aut],  
 Johannes Gussenbauer [aut],  
 Nina Niederhametner [aut],  
 Eileen Vattheuer [aut],  
 Gregor de Cillia [aut],  
 Bernd Prantner [ctb],  
 Wolfgang Rannetbauer [aut]

**Maintainer** Matthias Templ <matthias.templ@gmail.com>

**Date/Publication** 2026-01-10 07:52:08 UTC

## Contents

aggr . . . . .	3
alphablend . . . . .	8
Animals_na . . . . .	8
barMiss . . . . .	9
bcancer . . . . .	12
bgmap . . . . .	13
brittleness . . . . .	14
chorizonDL . . . . .	14
colic . . . . .	18
collisions . . . . .	20
colormapMiss . . . . .	20
colSequence . . . . .	23
countInf . . . . .	25
diabetes . . . . .	25
evaluation . . . . .	27
food . . . . .	28
gapMiss . . . . .	29
gowerD . . . . .	30
growdotMiss . . . . .	31
histMiss . . . . .	34
hotdeck . . . . .	36
impPCA . . . . .	38
imputeRobust . . . . .	40
imputeRobustChain . . . . .	42

initialise . . . . .	43
irmi . . . . .	44
kNN . . . . .	47
kola.background . . . . .	49
mapMiss . . . . .	50
marginmatrix . . . . .	51
marginplot . . . . .	53
matchImpute . . . . .	56
matrixplot . . . . .	57
maxCat . . . . .	60
medianSamp . . . . .	60
mosaicMiss . . . . .	61
pairsVIM . . . . .	63
parcoordMiss . . . . .	65
pbox . . . . .	68
prepare . . . . .	70
pulplignin . . . . .	72
rangerImpute . . . . .	73
regressionImp . . . . .	74
rugNA . . . . .	75
sampleCat . . . . .	76
SBS5242 . . . . .	77
scattJitt . . . . .	78
scattmatrixMiss . . . . .	80
scattMiss . . . . .	82
sleep . . . . .	85
spineMiss . . . . .	86
tableMiss . . . . .	89
tao . . . . .	89
testdata . . . . .	90
toydataMiss . . . . .	91
vimpute . . . . .	92
wine . . . . .	93
xgboostImpute . . . . .	94
<b>Index</b>	<b>96</b>

aggr

*Aggregations for missing/imputed values***Description**

Calculate or plot the amount of missing/imputed values in each variable and the amount of missing/imputed values in certain combinations of variables.

Print method for objects of class "aggr".

Summary method for objects of class "aggr".

Print method for objects of class "summary.aggr".

## Usage

```
aggr(x, delimiter = NULL, plot = TRUE, ...)
```

```
## S3 method for class 'aggr'
plot(
  x,
  col = c("skyblue", "red", "orange"),
  bars = TRUE,
  numbers = FALSE,
  prop = TRUE,
  combined = FALSE,
  varheight = FALSE,
  only.miss = FALSE,
  border = par("fg"),
  sortVars = FALSE,
  sortCombs = TRUE,
  ylabs = NULL,
  axes = TRUE,
  labels = axes,
  cex.lab = 1.2,
  cex.axis = par("cex"),
  cex.numbers = par("cex"),
  gap = 4,
  ...
)
```

```
## S3 method for class 'aggr'
print(x, ..., digits = NULL)
```

```
## S3 method for class 'aggr'
summary(object, ...)
```

```
## S3 method for class 'summary.aggr'
print(x, ...)
```

## Arguments

<code>x</code>	an object of class "summary.aggr".
<code>delimiter</code>	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, <code>x</code> needs to have <code>colnames()</code> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see <code>col</code> ).
<code>plot</code>	a logical indicating whether the results should be plotted (the default is TRUE).
<code>...</code>	Further arguments, currently ignored.

col	a vector of length three giving the colors to be used for observed, missing and imputed data. If only one color is supplied, it is used for missing and imputed data and observed data is transparent. If only two colors are supplied, the first one is used for observed data and the second color is used for missing and imputed data.
bars	a logical indicating whether a small barplot for the frequencies of the different combinations should be drawn.
numbers	a logical indicating whether the proportion or frequencies of the different combinations should be represented by numbers.
prop	a logical indicating whether the proportion of missing/imputed values and combinations should be used rather than the total amount.
combined	a logical indicating whether the two plots should be combined. If FALSE, a separate barplot on the left hand side shows the amount of missing/imputed values in each variable. If TRUE, a small version of this barplot is drawn on top of the plot for the combinations of missing/imputed and non-missing values. See “Details” for more information.
varheight	a logical indicating whether the cell heights are given by the frequencies of occurrence of the corresponding combinations.
only.miss	a logical indicating whether the small barplot for the frequencies of the combinations should only be drawn for combinations including missing/imputed values (if bars is TRUE). This is useful if most observations are complete, in which case the corresponding bar would dominate the barplot such that the remaining bars are too compressed. The proportion or frequency of complete observations (as determined by prop) is then represented by a number instead of a bar.
border	the color to be used for the border of the bars and rectangles. Use border=NA to omit borders.
sortVars	a logical indicating whether the variables should be sorted by the number of missing/imputed values.
sortCombs	a logical indicating whether the combinations should be sorted by the frequency of occurrence.
ylabs	if combined is TRUE, a character string giving the y-axis label of the combined plot, otherwise a character vector of length two giving the y-axis labels for the two plots.
axes	a logical indicating whether axes should be drawn.
labels	either a logical indicating whether labels should be plotted on the x-axis, or a character vector giving the labels.
cex.lab	the character expansion factor to be used for the axis labels.
cex.axis	the character expansion factor to be used for the axis annotation.
cex.numbers	the character expansion factor to be used for the proportion or frequencies of the different combinations
gap	if combined is FALSE, a numeric value giving the distance between the two plots in margin lines.
digits	the minimum number of significant digits to be used (see <a href="#">print.default()</a> ).
object	an object of class “aggr”.

## Details

Often it is of interest how many missing/imputed values are contained in each variable. Even more interesting, there may be certain combinations of variables with a high number of missing/imputed values.

If `combined` is `FALSE`, two separate plots are drawn for the missing/imputed values in each variable and the combinations of missing/imputed and non-missing values. The barplot on the left hand side shows the amount of missing/imputed values in each variable. In the *aggregation plot* on the right hand side, all existing combinations of missing/imputed and non-missing values in the observations are visualized. Available, missing and imputed data are color coded as given by `col`. Additionally, there are two possibilities to represent the frequencies of occurrence of the different combinations. The first option is to visualize the proportions or frequencies by a small bar plot and/or numbers. The second option is to let the cell heights be given by the frequencies of the corresponding combinations. Furthermore, variables may be sorted by the number of missing/imputed values and combinations by the frequency of occurrence to give more power to finding the structure of missing/imputed values.

If `combined` is `TRUE`, a small version of the barplot showing the amount of missing/imputed values in each variable is drawn on top of the aggregation plot.

The graphical parameter `oma` will be set unless supplied as an argument.

## Value

for `aggr`, a list of class `"aggr"` containing the following components:

- `x` the data used.
- `combinations` a character vector representing the combinations of variables.
- `count` the frequencies of these combinations.
- `percent` the percentage of these combinations.
- `missings` a `data.frame` containing the amount of missing/imputed values in each variable.
- `tabcomb` the indicator matrix for the combinations of variables.

a list of class `"summary.aggr"` containing the following components:

- `missings` a `data.frame` containing the amount of missing or imputed values in each variable.
- `combinations` a `data.frame` containing a character vector representing the combinations of variables along with their frequencies and percentages.

## Note

Some of the argument names and positions have changed with version 1.3 due to extended functionality and for more consistency with other plot functions in VIM. For back compatibility, the arguments `labs` and `names.arg` can still be supplied to `...{}` and are handled correctly. Nevertheless, they are deprecated and no longer documented. Use `ylabs` and `labels` instead.

**Author(s)**

Andreas Alfons, Matthias Templ, modifications for displaying imputed values by Bernd Prantner

Matthias Templ, modifications by Andreas Alfons and Bernd Prantner

Matthias Templ, modifications by Andreas Alfons

Andreas Alfons, modifications by Bernd Prantner

**References**

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

**See Also**

`print.aggr()`, `summary.aggr()`

`aggr()`

`print.summary.aggr()`, `aggr()`

`summary.aggr()`, `aggr()`

Other plotting functions: `barMiss()`, `histMiss()`, `marginmatrix()`, `marginplot()`, `matrixplot()`, `mosaicMiss()`, `pairsVIM()`, `parcoordMiss()`, `pbox()`, `scattJitt()`, `scattMiss()`, `scattmatrixMiss()`, `spineMiss()`

**Examples**

```
data(sleep, package="VIM")
## for missing values
a <- aggr(sleep)
a
summary(a)

## for imputed values
sleep_IMPUTED <- kNN(sleep)
a <- aggr(sleep_IMPUTED, delimiter="_imp")
a
summary(a)
```

```
data(sleep, package = "VIM")
a <- aggr(sleep, plot=FALSE)
a
```

```
data(sleep, package = "VIM")
summary(aggr(sleep, plot=FALSE))
```

```
data(sleep, package = "VIM")
s <- summary(aggr(sleep, plot=FALSE))
```

s

---

alphablend	<i>Alphablending for colors</i>
------------	---------------------------------

---

**Description**

Convert colors to semitransparent colors.

**Usage**

```
alphablend(col, alpha = NULL, bg = NULL)
```

**Arguments**

- col                    a vector specifying colors.
- alpha                a numeric vector containing the alpha values (between 0 and 1).
- bg                    the background color to be used for alphablending. This can be used as a workaround for graphics devices that do not support semitransparent colors.

**Value**

a vector containing the semitransparent colors.

**Author(s)**

Andreas Alfons

**Examples**

```
alphablend("red", 0.6)
```

---

Animals_na	<i>Animals_na</i>
------------	-------------------

---

**Description**

Average log brain and log body weights for 28 Species

**Format**

A data frame with 28 observations on the following 2 variables.

- lbody** log body weight
- lbrain** log brain weight



## Details

The original data can be found in package MASS. 10 values on brain weight are set to be missing.

## Source

P. J. Rousseeuw and A. M. Leroy (1987) Robust Regression and Outlier Detection. Wiley, p. 57.

## References

Venables, W. N. and Ripley, B. D. (1999) Modern Applied Statistics with S-PLUS. Third Edition. Springer.

Templ, M. (2022) Visualization and Imputation of Missing Values. Springer Publishing. Upcoming book.

## Examples

```
data(Animals_na)
aggr(Animals_na)
```

---

barMiss

*Barplot with information about missing/imputed values*


---

## Description

Barplot with highlighting of missing/imputed values in other variables by splitting each bar into two parts. Additionally, information about missing/imputed values in the variable of interest is shown on the right hand side.

## Usage

```
barMiss(
  x,
  delimiter = NULL,
  pos = 1,
  selection = c("any", "all"),
  col = c("skyblue", "red", "skyblue4", "red4", "orange", "orange4"),
  border = NULL,
  main = NULL,
  sub = NULL,
  xlab = NULL,
  ylab = NULL,
  axes = TRUE,
  labels = axes,
  only.miss = TRUE,
  miss.labels = axes,
  interactive = TRUE,
  ...
)
```

## Arguments

<code>x</code>	a vector, matrix or <code>data.frame</code> .
<code>delimiter</code>	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, <code>x</code> needs to have <code>colnames()</code> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see <code>col</code> ).
<code>pos</code>	a numeric value giving the index of the variable of interest. Additional variables in <code>x</code> are used for highlighting.
<code>selection</code>	the selection method for highlighting missing/imputed values in multiple additional variables. Possible values are "any" (highlighting of missing/imputed values in <i>any</i> of the additional variables) and "all" (highlighting of missing/imputed values in <i>all</i> of the additional variables).
<code>col</code>	a vector of length six giving the colors to be used. If only one color is supplied, the bars are transparent and the supplied color is used for highlighting missing/imputed values. Else if two colors are supplied, they are recycled.
<code>border</code>	the color to be used for the border of the bars. Use <code>border=NA</code> to omit borders.
<code>main, sub</code>	main and sub title.
<code>xlab, ylab</code>	axis labels.
<code>axes</code>	a logical indicating whether axes should be drawn on the plot.
<code>labels</code>	either a logical indicating whether labels should be plotted below each bar, or a character vector giving the labels.
<code>only.miss</code>	logical; if TRUE, the missing/imputed values in the variable of interest are visualized by a single bar. Otherwise, a small barplot is drawn on the right hand side (see 'Details').
<code>miss.labels</code>	either a logical indicating whether label(s) should be plotted below the bar(s) on the right hand side, or a character string or vector giving the label(s) (see 'Details').
<code>interactive</code>	a logical indicating whether variables can be switched interactively (see 'Details').
<code>...</code>	further graphical parameters to be passed to <code>graphics::title()</code> and <code>graphics::axis()</code> .

## Details

If more than one variable is supplied, the bars for the variable of interest are split according to missingness/number of imputed missings in the additional variables.

If `only.miss=TRUE`, the missing/imputed values in the variable of interest are visualized by one bar on the right hand side. If additional variables are supplied, this bar is again split into two parts according to missingness/number of imputed missings in the additional variables.

Otherwise, a small barplot consisting of two bars is drawn on the right hand side. The first bar corresponds to observed values in the variable of interest and the second bar to missing/imputed values. Since these two bars are not on the same scale as the main barplot, a second y-axis is

plotted on the right (if `axes=TRUE`). Each of the two bars are again split into two parts according to missingness/number of imputed missings in the additional variables. Note that this display does not make sense if only one variable is supplied, therefore only `.miss` is ignored in that case.

If `interactive=TRUE`, clicking in the left margin of the plot results in switching to the previous variable and clicking in the right margin results in switching to the next variable. Clicking anywhere else on the graphics device quits the interactive session. When switching to a continuous variable, a histogram is plotted rather than a barplot.

## Value

a numeric vector giving the coordinates of the midpoints of the bars.

## Note

Some of the argument names and positions have changed with version 1.3 due to extended functionality and for more consistency with other plot functions in VIM. For back compatibility, the arguments `axisnames`, `names.arg` and `names.miss` can still be supplied to `...{}` and are handled correctly. Nevertheless, they are deprecated and no longer documented. Use `labels` and `miss.labels` instead.

## Author(s)

Andreas Alfons, modifications to show imputed values by Bernd Prantner

## References

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

## See Also

`spineMiss()`, `histMiss()`

Other plotting functions: `aggr()`, `histMiss()`, `marginmatrix()`, `marginplot()`, `matrixplot()`, `mosaicMiss()`, `pairsVIM()`, `parcoordMiss()`, `pbox()`, `scattJitt()`, `scattMiss()`, `scattmatrixMiss()`, `spineMiss()`

## Examples

```
data(sleep, package = "VIM")
## for missing values
x <- sleep[, c("Exp", "Sleep")]
barMiss(x)
barMiss(x, only.miss = FALSE)

## for imputed values
x_IMPUTED <- kNN(sleep[, c("Exp", "Sleep")])
barMiss(x_IMPUTED, delimiter = "_imp")
barMiss(x_IMPUTED, delimiter = "_imp", only.miss = FALSE)
```

bcancer

*Breast cancer Wisconsin data set***Description**

Dataset containing the original Wisconsin breast cancer data.

**Format**

A data frame with 699 observations on the following 11 variables.

**ID** Sample ID

**clump\_thickness** as integer from 1 - 10

**uniformity\_cellsize** as integer from 1 - 10

**uniformity\_cellshape** as integer from 1 - 10

**adhesion** as integer from 1 - 10

**epithelial\_cellsize** as integer from 1 - 10

**bare\_nuclei** as integer from 1 - 10, includes 16 missings

**chromatin** as integer from 1 - 10

**normal\_nucleoli** as integer from 1 - 10

**mitoses** as integer from 1 - 10

**class** benign or malignant

**References**

The data downloaded and conditioned for R from the UCI machine learning repository, see <https://archive.ics.uci.edu/ml/datas>. This breast cancer databases was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. If you publish results when using this database, then please include this information in your acknowledgements. Also, please cite one or more of: O. L. Mangasarian and W. H. Wolberg: "Cancer diagnosis via linear programming", SIAM News, Volume 23, Number 5, September 1990, pp 1 & 18. William H. Wolberg and O.L. Mangasarian: "Multisurface method of pattern separation for medical diagnosis applied to breast cytology", Proceedings of the National Academy of Sciences, U.S.A., Volume 87, December 1990, pp 9193-9196. O. L. Mangasarian, R. Setiono, and W.H. Wolberg: "Pattern recognition via linear programming: Theory and application to medical diagnosis", in: "Large-scale numerical optimization", Thomas F. Coleman and Yuying Li, editors, SIAM Publications, Philadelphia 1990, pp 22-30. K. P. Bennett & O. L. Mangasarian: "Robust linear programming discrimination of two linearly inseparable sets", Optimization Methods and Software 1, 1992, 23-34 (Gordon & Breach Science Publishers).

**Examples**

```
data(bcancer)
aggr(bcancer)
```

---

**bgmap***Background map*

---

**Description**

Plot a background map.

**Usage**

```
bgmap(map, add = FALSE, ...)
```

**Arguments**

map	either a matrix or data.frame with two columns, a list with components x and y, or an object of any class that can be used for maps and provides its own plot method (e.g., "SpatialPolygons" from package sp). A list of the previously mentioned types can also be provided.
add	a logical indicating whether map should be added to an already existing plot (the default is FALSE).
...	further arguments and graphical parameters to be passed to plot and/or <a href="#">graphics::lines()</a> .

**Author(s)**

Andreas Alfons

**References**

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

**See Also**

[growdotMiss\(\)](#), [mapMiss\(\)](#)

**Examples**

```
data(kola.background, package = "VIM")
bgmap(kola.background)
```

---

brittleness	<i>Brittleness index data set</i>
-------------	-----------------------------------

---

### Description

A plastic product is produced in three parallel reactors (TK104, TK105, or TK107). For each row in the dataset, we have the same batch of raw material that was split, and fed to the 3 reactors. These values are the brittleness index for the product produced in the reactor. A simulated data set.

### Format

A data frame with 23 observations on the following 3 variables.

**TK104** Brittleness for batches of raw material in reactor 104

**TK105** Brittleness for batches of raw material in reactor 105

**TK107** Brittleness for batches of raw material in reactor 107

### Source

<https://openmv.net/info/brittleness-index>

### Examples

```
data(brittleness)
aggr(brittleness)
```

---

chorizonDL	<i>C-horizon of the Kola data with missing values</i>
------------	---

---

### Description

This data set is the same as in package `mvoutlier`, except that values below the detection limit are coded as NA.

### Format

A data frame with 606 observations on the following 110 variables.

**\*ID** a numeric vector

**XCOO** a numeric vector

**YCOO** a numeric vector

**Ag** a numeric vector

**Ag\_INAA** a numeric vector

**Al** a numeric vector

**Al2O3** a numeric vector  
**As** a numeric vector  
**As\_INAA** a numeric vector  
**Au\_INAA** a numeric vector  
**B** a numeric vector  
**Ba** a numeric vector  
**Ba\_INAA** a numeric vector  
**Be** a numeric vector  
**Bi** a numeric vector  
**Br\_IC** a numeric vector  
**Br\_INAA** a numeric vector  
**Ca** a numeric vector  
**Ca\_INAA** a numeric vector  
**CaO** a numeric vector  
**Cd** a numeric vector  
**Ce\_INAA** a numeric vector  
**Cl\_IC** a numeric vector  
**Co** a numeric vector  
**Co\_INAA** a numeric vector  
**EC** a numeric vector  
**Cr** a numeric vector  
**Cr\_INAA** a numeric vector  
**Cs\_INAA** a numeric vector  
**Cu** a numeric vector  
**Eu\_INAA** a numeric vector  
**F\_IC** a numeric vector  
**Fe** a numeric vector  
**Fe\_INAA** a numeric vector  
**Fe2O3** a numeric vector  
**Hf\_INAA** a numeric vector  
**Hg** a numeric vector  
**Hg\_INAA** a numeric vector  
**Ir\_INAA** a numeric vector  
**K** a numeric vector  
**K2O** a numeric vector  
**La** a numeric vector  
**La\_INAA** a numeric vector

**Li** a numeric vector  
**LOI** a numeric vector  
**Lu\_INAA** a numeric vector  
**wt\_INAA** a numeric vector  
**Mg** a numeric vector  
**MgO** a numeric vector  
**Mn** a numeric vector  
**MnO** a numeric vector  
**Mo** a numeric vector  
**Mo\_INAA** a numeric vector  
**Na** a numeric vector  
**Na\_INAA** a numeric vector  
**Na2O** a numeric vector  
**Nd\_INAA** a numeric vector  
**Ni** a numeric vector  
**Ni\_INAA** a numeric vector  
**NO3\_IC** a numeric vector  
**P** a numeric vector  
**P2O5** a numeric vector  
**Pb** a numeric vector  
**pH** a numeric vector  
**PO4\_IC** a numeric vector  
**Rb** a numeric vector  
**S** a numeric vector  
**Sb** a numeric vector  
**Sb\_INAA** a numeric vector  
**Sc** a numeric vector  
**Sc\_INAA** a numeric vector  
**Se** a numeric vector  
**Se\_INAA** a numeric vector  
**Si** a numeric vector  
**SiO2** a numeric vector  
**Sm\_INAA** a numeric vector  
**Sn\_INAA** a numeric vector  
**SO4\_IC** a numeric vector  
**Sr** a numeric vector  
**Sr\_INAA** a numeric vector



**SUM\_XRF** a numeric vector  
**Ta\_INAA** a numeric vector  
**Tb\_INAA** a numeric vector  
**Te** a numeric vector  
**Th** a numeric vector  
**Th\_INAA** a numeric vector  
**Ti** a numeric vector  
**TiO2** a numeric vector  
**U\_INAA** a numeric vector  
**V** a numeric vector  
**W\_INAA** a numeric vector  
**Y** a numeric vector  
**Yb\_INAA** a numeric vector  
**Zn** a numeric vector  
**Zn\_INAA** a numeric vector  
**ELEV** a numeric vector  
**\*COUN** a numeric vector  
**\*ASP** a numeric vector  
**TOPC** a numeric vector  
**LITO** a numeric vector  
**Al\_XRF** a numeric vector  
**Ca\_XRF** a numeric vector  
**Fe\_XRF** a numeric vector  
**K\_XRF** a numeric vector  
**Mg\_XRF** a numeric vector  
**Mn\_XRF** a numeric vector  
**Na\_XRF** a numeric vector  
**P\_XRF** a numeric vector  
**Si\_XRF** a numeric vector  
**Ti\_XRF** a numeric vector

**Note**

For a more detailed description of this data set, see the help file chorizon in package mvoutlier.

**Source**

Kola Project (1993-1998)

## References

Reimann, C., Filzmoser, P., Garrett, R.G. and Dutter, R. (2008) *Statistical Data Analysis Explained: Applied Environmental Statistics with R*. Wiley.

## Examples

```
data(chorizonDL, package = "VIM")
summary(chorizonDL)
```

---

colic

*Colic horse data set*


---

## Description

This is a modified version of the original training data set taken from the UCI repository, see reference. The modifications are only related to having appropriate levels for factor variables. This data set is about horse diseases where the task is to determine, if the lesion of the horse was surgical or not.

## Format

A training data frame with 300 observations on the following 31 variables.

**surgery** yes or no

**age** 1 equals an adult horse, 2 is a horse younger than 6 months

**hospitalID** ID

**temp\_rectal** rectal temperature

**pulse** heart rate in beats per minute

**respiratory\_rate** a normal rate is between 8 and 10

**temp\_extreme** temperature of extremities

**pulse\_peripheral** factor with four categories

**capillayr\_refill\_time** a clinical judgement. The longer the refill, the poorer the circulation. Possible values are 1 = < 3 seconds and 2 = >= 3 seconds

**pain** a subjective judgement of the horse's pain level

**peristalsis** an indication of the activity in the horse's gut. As the gut becomes more distended or the horse becomes more toxic, the activity decreases

**abdominal\_distension** An animal with abdominal distension is likely to be painful and have reduced gut motility. A horse with severe abdominal distension is likely to require surgery just to relieve the pressure

**nasogastric\_tube** This refers to any gas coming out of the tube. A large gas cap in the stomach is likely to give the horse discomfort

**nasogastric\_reflux** possible values are 1 = none, 2 = > 1 liter, 3 = < 1 liter. The greater amount of reflux, the more likelihood that there is some serious obstruction to the fluid passage from the rest of the intestine

**nasogastric\_reflux\_PH** scale is from 0 to 14 with 7 being neutral. Normal values are in the 3 to 4 range

**rectal\_examination** Rectal examination. Absent feces probably indicates an obstruction

**abdomen** abdomen. possible values 1 = normal, 2 = other, 3 = firm feces in the large intestine, 4 = distended small intestine, 5 = distended large intestine

**cell\_volume** packed cell volume. normal range is 30 to 50. The level rises as the circulation becomes compromised or as the animal becomes dehydrated.

**protein** total protein. Normal values lie in the 6-7.5 (gms/dL) range. The higher the value the greater the dehydration

**abdominocentesis\_appearance** Abdominocentesis appearance. A needle is put in the horse's abdomen and fluid is obtained from the abdominal cavity

**abdomcentesis\_protein** abdomcentesis total protein. The higher the level of protein the more likely it is to have a compromised gut. Values are in gms/dL

**outcome** What eventually happened to the horse?

**surgical\_lesion** retrospectively, was the problem (lesion) surgical?

**lesion\_type1** type of lesion

**lesion\_type2** type of lesion

**lesion\_type3** type of lesion

**cp\_data**

**temp\_extreme\_ordered** temperature of extremities (ordered)

**mucous\_membranes\_col** mucous membranes. A subjective measurement of colour

**mucous\_membranes\_group** different recodings of mucous membranes

## Source

<https://archive.ics.uci.edu/ml/datasets/Horse+Colic> Creators: Mary McLeish & Matt Cecile, Department of Computer Science, University of Guelph, Guelph, Ontario, Canada N1G 2W1 Donor: Will Taylor

## Examples

```
data(colic)
aggr(colic)
```

---

collisions	<i>Subset of the collision data</i>
------------	-------------------------------------

---

### Description

Subset of the collision data from December 20. to December 31. 2018 from NYCD.

### Details

Each record represents a collision in NYC by city, borough, precinct and cross street.

### Source

<https://data.cityofnewyork.us/Public-Safety/NYPD-Motor-Vehicle-Collisions/h9gi-nx95>

### Examples

```
data(collisions)
aggr(collisions)
```

---

colormapMiss	<i>Colored map with information about missing/imputed values</i>
--------------	--

---

### Description

Colored map in which the proportion or amount of missing/imputed values in each region is coded according to a continuous or discrete color scheme. The sequential color palette may thereby be computed in the *HCL* or the *RGB* color space.

### Usage

```
colormapMiss(
  x,
  region,
  map,
  imp_index = NULL,
  prop = TRUE,
  polysRegion = 1:length(x),
  range = NULL,
  n = NULL,
  col = c("red", "orange"),
  gamma = 2.2,
  fixup = TRUE,
  coords = NULL,
  numbers = TRUE,
```

```

    digits = 2,
    cex.numbers = 0.8,
    col.numbers = par("fg"),
    legend = TRUE,
    interactive = TRUE,
    ...
)

colormapMissLegend(
  xleft,
  ybottom,
  xright,
  ytop,
  cmap,
  n = 1000,
  horizontal = TRUE,
  digits = 2,
  cex.numbers = 0.8,
  col.numbers = par("fg"),
  ...
)

```

### Arguments

<code>x</code>	a numeric vector.
<code>region</code>	a vector or factor of the same length as <code>x</code> giving the regions.
<code>map</code>	an object of any class that contains polygons and provides its own plot method (e.g., "SpatialPolygons" from package <code>sp</code> ).
<code>imp_index</code>	a logical-vector indicating which values of 'x' have been imputed. If given, it is used for highlighting and the colors are adjusted according to the given colors for imputed variables (see <code>col</code> ).
<code>prop</code>	a logical indicating whether the proportion of missing/imputed values should be used rather than the total amount.
<code>polysRegion</code>	a numeric vector specifying the region that each polygon belongs to.
<code>range</code>	a numeric vector of length two specifying the range (minimum and maximum) of the proportion or amount of missing/imputed values to be used for the color scheme.
<code>n</code>	for <code>colormapMiss</code> , the number of equally spaced cut-off points for a discretized color scheme. If this is not a positive integer, a continuous color scheme is used (the default). In the latter case, the number of rectangles to be drawn in the legend can be specified in <code>colormapMissLegend</code> . A reasonably large number makes it appear continuously.
<code>col</code>	the color range (start end end) to be used. RGB colors may be specified as character strings or as objects of class " <code>colorspace::RGB()</code> ". HCL colors need to be specified as objects of class " <code>colorspace::polarLUV()</code> ". If only one color is supplied, it is used as end color, while the start color is taken to be transparent for RGB or white for HCL.

<code>gamma</code>	numeric; the display <i>gamma</i> value (see <code>colorspace::hex()</code> ).
<code>fixup</code>	a logical indicating whether the colors should be corrected to valid RGB values (see <code>colorspace::hex()</code> ).
<code>coords</code>	a matrix or <code>data.frame</code> with two columns giving the coordinates for the labels.
<code>numbers</code>	a logical indicating whether the corresponding proportions or numbers of missing/imputed values should be used as labels for the regions.
<code>digits</code>	the number of digits to be used in the labels (in case of proportions).
<code>cex.numbers</code>	the character expansion factor to be used for the labels.
<code>col.numbers</code>	the color to be used for the labels.
<code>legend</code>	a logical indicating whether a legend should be plotted.
<code>interactive</code>	a logical indicating whether more detailed information about missing/imputed values should be displayed interactively (see ‘Details’).
<code>...</code>	further arguments to be passed to <code>plot</code> .
<code>xleft</code>	left <i>x</i> position of the legend.
<code>ybottom</code>	bottom <i>y</i> position of the legend.
<code>xright</code>	right <i>x</i> position of the legend.
<code>yttop</code>	top <i>y</i> position of the legend.
<code>cmap</code>	a list as returned by <code>colormapMiss</code> that contains the required information for the legend.
<code>horizontal</code>	a logical indicating whether the legend should be drawn horizontally or vertically.

## Details

The proportion or amount of missing/imputed values in *x* of each region is coded according to a continuous or discrete color scheme in the color range defined by `col`. In addition, the proportions or numbers can be shown as labels in the regions.

If `interactive` is `TRUE`, clicking in a region displays more detailed information about missing/imputed values on the console. Clicking outside the borders quits the interactive session.

## Value

`colormapMiss` returns a list with the following components:

- `nmiss` a numeric vector containing the number of missing/imputed values in each region.
- `nobs` a numeric vector containing the number of observations in each region.
- `pmiss` a numeric vector containing the proportion of missing values in each region.
- `prop` a logical indicating whether the proportion of missing/imputed values have been used rather than the total amount.
- `range` the range of the proportion or amount of missing/imputed values corresponding to the color range.
- `n` either a positive integer giving the number of equally spaced cut-off points for a discretized color scheme, or `NULL` for a continuous color scheme.

- start the start color of the color scheme.
- end the end color of the color scheme.
- space a character string giving the color space (either "rgb" for RGB colors or "hcl" for HCL colors).
- gamma numeric; the display *gamma* value (see `colorspace::hex()`).
- fixup a logical indicating whether the colors have been corrected to valid RGB values (see `colorspace::hex()`).

### Note

Some of the argument names and positions have changed with versions 1.3 and 1.4 due to extended functionality and for more consistency with other plot functions in VIM. For back compatibility, the arguments `cex.text` and `col.text` can still be supplied to `...{}` and are handled correctly. Nevertheless, they are deprecated and no longer documented. Use `cex.numbers` and `col.numbers` instead.

### Author(s)

Andreas Alfons, modifications to show imputed values by Bernd Prantner

### References

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

### See Also

`colSequence()`, `growdotMiss()`, `mapMiss()`

---

colSequence

*HCL and RGB color sequences*


---

### Description

Compute color sequences by linear interpolation based on a continuous color scheme between certain start and end colors. Color sequences may thereby be computed in the *HCL* or *RGB* color space.

### Usage

```
colSequence(p, start, end, space = c("hcl", "rgb"), ...)
```

```
colSequenceRGB(p, start, end, fixup = TRUE, ...)
```

```
colSequenceHCL(p, start, end, fixup = TRUE, ...)
```

**Arguments**

<code>p</code>	a numeric vector with values between 0 and 1 giving values to be used for interpolation between the start and end color (0 corresponds to the start color, 1 to the end color).
<code>start, end</code>	the start and end color, respectively. For HCL colors, each can be supplied as a vector of length three (hue, chroma, luminance) or an object of class " <code>colorspace::polarLUV()</code> ". For RGB colors, each can be supplied as a character string, a vector of length three (red, green, blue) or an object of class " <code>colorspace::RGB()</code> ".
<code>space</code>	character string; if <code>start</code> and <code>end</code> are both numeric, this determines whether they refer to HCL or RGB values. Possible values are " <code>hcl</code> " (for the HCL space) or " <code>rgb</code> " (for the RGB space).
<code>...</code>	for <code>colSequence</code> , additional arguments to be passed to <code>colSequenceHCL</code> or <code>colSequenceRGB</code> . For <code>colSequenceHCL</code> and <code>colSequenceRGB</code> , additional arguments to be passed to <code>colorspace::hex()</code> .
<code>fixup</code>	a logical indicating whether the colors should be corrected to valid RGB values (see <code>colorspace::hex()</code> ).

**Value**

A character vector containing hexadecimal strings of the form "`#RRGGBB`".

**Author(s)**

Andreas Alfons

**References**

Zeileis, A., Hornik, K., Murrell, P. (2009) Escaping RGBland: Selecting colors for statistical graphics. *Computational Statistics & Data Analysis*, **53** (9), 1259–1270.

**See Also**

`colorspace::hex()`, `colorspace::sequential_hcl()`

**Examples**

```
p <- c(0, 0.3, 0.55, 0.8, 1)

## HCL colors
colSequence(p, c(0, 0, 100), c(0, 100, 50))
colSequence(p, polarLUV(L=90, C=30, H=90), c(0, 100, 50))

## RGB colors
colSequence(p, c(1, 1, 1), c(1, 0, 0), space="rgb")
colSequence(p, RGB(1, 1, 0), "red")
```



---

countInf	<i>Count number of infinite or missing values</i>
----------	---

---

**Description**

Count the number of infinite or missing values in a vector.

**Usage**

```
countInf(x)
```

**Arguments**

x                      a vector.

**Value**

countInf returns the number of infinite values in x. countNA returns the number of missing values in x.

**Author(s)**

Andreas Alfons

**Examples**

```
data(sleep, package="VIM")
countInf(log(sleep$Dream))
countNA(sleep$Dream)
```

---

diabetes	<i>Indian Prime Diabetes Data</i>
----------	-----------------------------------

---

**Description**

The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

**Format**

A data frame with 768 observations on the following 9 variables.

**Pregnancies** Number of times pregnant

**Glucose** Plasma glucose concentration a 2 hours in an oral glucose tolerance test

**BloodPressure** Diastolic blood pressure (mm Hg)

**SkinThickness** Triceps skin fold thickness (mm)

**Insulin** 2-Hour serum insulin (mu U/ml)

**BMI** Body mass index (weight in kg/(height in m)^2)

**DiabetesPedigreeFunction** Diabetes pedigree function

**Age** Age in years

**Outcome** Diabetes (yes or no)

**Details**

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

**Source**

<https://www.kaggle.com/uciml/pima-indians-diabetes-database/data>

**References**

Smith, J.W., Everhart, J.E., Dickson, W.C., Knowler, W.C., & Johannes, R.S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In Proceedings of the Symposium on Computer Applications and Medical Care (pp. 261–265). IEEE Computer Society Press.

**Examples**

```
data(diabetes)
aggr(diabetes)
```

---

evaluation*Error performance measures*

---

**Description**

Various error measures evaluating the quality of imputations

**Usage**

```
evaluation(x, y, m, vartypes = "guess")
```

```
nrmse(x, y, m)
```

```
pfc(x, y, m)
```

```
msecov(x, y)
```

```
msecor(x, y)
```

**Arguments**

x	matrix or data frame
y	matrix or data frame of the same size as x
m	the indicator matrix for missing cells
vartypes	a vector of length ncol(x) specifying the variables types, like factor or numeric

**Details**

This function has been mainly written for procedures that evaluate imputation or replacement of rounded zeros. The ni parameter can thus, e.g. be used for expressing the number of rounded zeros.

**Value**

the error measures value

**Author(s)**

Matthias Templ

**References**

M. Templ, A. Kowarik, P. Filzmoser (2011) Iterative stepwise regression imputation using standard and robust methods. *Journal of Computational Statistics and Data Analysis*, Vol. 55, pp. 2793-2806.

**Examples**

```
data(iris)
iris_orig <- iris_imp <- iris
iris_imp$Sepal.Length[sample(1:nrow(iris), 10)] <- NA
iris_imp$Sepal.Width[sample(1:nrow(iris), 10)] <- NA
iris_imp$Species[sample(1:nrow(iris), 10)] <- NA
m <- is.na(iris_imp)
iris_imp <- kNN(iris_imp, imp_var = FALSE)
evaluation(iris_orig, iris_imp, m = m, vartypes = c(rep("numeric", 4), "factor"))
mse cov(iris_orig[, 1:4], iris_imp[, 1:4])
```

---

 food

*Food consumption*


---

**Description**

The relative consumption of certain food items in European and Scandinavian countries.

**Format**

A data frame with 16 observations on the following 21 variables.

**Details**

The numbers represent the percentage of the population consuming that food type.

**Source**

<https://openmv.net/info/food-consumption>

**Examples**

```
data(food)
str(food)
aggr(food)
```

---

`gapMiss`*Missing value gap statistics*

---

**Description**

Computes the average missing value gap of a vector.

**Usage**

```
gapMiss(x, what = mean)
```

**Arguments**

<code>x</code>	a numeric vector
<code>what</code>	default is the arithmetic mean. One can include an own function that returns a vector of length 1 (e.g. median)

**Details**

The length of each sequence of missing values (gap) in a vector is calculated and the mean gap is reported

**Value**

The gap statistics

**Author(s)**

Matthias Templ based on a suggestion and draft from Huang Tian Yuan.

**Examples**

```
v <- rnorm(20)
v[3] <- NA
v[6:9] <- NA
v[13:17] <- NA
v
gapMiss(v)
gapMiss(v, what = median)
gapMiss(v, what = function(x) mean(x, trim = 0.1))
gapMiss(v, what = var)
```

gowerD

*Computes the extended Gower distance of two data sets***Description**

The function gowerD is used by kNN to compute the distances for numerical, factor ordered and semi-continuous variables.

**Usage**

```
gowerD(
  data.x,
  data.y = data.x,
  weights = rep(1, ncol(data.x)),
  numerical = colnames(data.x),
  factors = vector(),
  orders = vector(),
  mixed = vector(),
  levOrders = vector(),
  mixed.constant = rep(0, length(mixed)),
  returnIndex = FALSE,
  nMin = 1L,
  returnMin = FALSE,
  methodStand = "range"
)
```

**Arguments**

data.x	data frame
data.y	data frame
weights	numeric vector providing weights for the observations in x
numerical	names of numerical variables
factors	names of factor variables
orders	names of ordered variables
mixed	names of mixed variables
levOrders	vector with number of levels for each orders variable
mixed.constant	vector with length equal to the number of semi-continuous variables specifying the point of the semi-continuous distribution with non-zero probability
returnIndex	logical if TRUE return the index of the minimum distance
nMin	integer number of values with smallest distance to be returned
returnMin	logical if the computed distances for the indices should be returned
methodStand	character either "range" or "iqr", iqr is more robust for outliers

## Details

returnIndex=FALSE: a numerical matrix n x m with the computed distances returnIndex=TRUE: a named list with "ind" containing the requested indices and "mins" the computed distances

## Examples

```
data(sleep)
# all variables used as numerical
gowerD(sleep)

# split in numerical and
gowerD(sleep, numerical = c("BodyWgt", "BrainWgt", "NonD", "Dream", "Sleep", "Span", "Gest"),
  orders = c("Pred", "Exp", "Danger"), levOrders = c(5,5,5))

# as before but only returning the index of the closest observation
gowerD(sleep, numerical = c("BodyWgt", "BrainWgt", "NonD", "Dream", "Sleep", "Span", "Gest"),
  orders = c("Pred", "Exp", "Danger"), levOrders = c(5,5,5), returnIndex = TRUE)
```

---

growdotMiss

---

*Growing dot map with information about missing/imputed values*


---

## Description

Map with dots whose sizes correspond to the values in a certain variable. Observations with missing/imputed values in additional variables are highlighted.

## Usage

```
growdotMiss(
  x,
  coords,
  map,
  pos = 1,
  delimiter = NULL,
  selection = c("any", "all"),
  log = FALSE,
  col = c("skyblue", "red", "skyblue4", "red4", "orange", "orange4"),
  border = par("bg"),
  alpha = NULL,
  scale = NULL,
  size = NULL,
  exp = c(0, 0.95, 0.05),
  col.map = grey(0.5),
  legend = TRUE,
  legtitle = "Legend",
  cex.legtitle = par("cex"),
  cex.legtext = par("cex"),
  ncircles = 6,
```

```

    ndigits = 1,
    interactive = TRUE,
    ...
)

```

## Arguments

<code>x</code>	a vector, matrix or <code>data.frame</code> .
<code>coords</code>	a matrix or <code>data.frame</code> with two columns giving the spatial coordinates of the observations.
<code>map</code>	a background map to be passed to <code>bgmap()</code> .
<code>pos</code>	a numeric value giving the index of the variable determining the dot sizes.
<code>delimiter</code>	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, <code>x</code> needs to have <code>colnames()</code> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see <code>col</code> ).
<code>selection</code>	the selection method for highlighting missing/imputed values in multiple additional variables. Possible values are "any" (highlighting of missing/imputed values in <i>any</i> of the additional variables) and "all" (highlighting of missing/imputed values in <i>all</i> of the additional variables).
<code>log</code>	a logical indicating whether the variable given by <code>pos</code> should be log-transformed.
<code>col</code>	a vector of length six giving the colors to be used in the plot. If only one color is supplied, it is used for the borders of non-highlighted dots and the surface area of highlighted dots. Else if two colors are supplied, they are recycled.
<code>border</code>	a vector of length four giving the colors to be used for the borders of the growing dots. Use NA to omit borders.
<code>alpha</code>	a numeric value between 0 and 1 giving the level of transparency of the colors, or NULL. This can be used to prevent overplotting.
<code>scale</code>	scaling factor of the map.
<code>size</code>	a vector of length two giving the sizes for the smallest and largest dots.
<code>exp</code>	a vector of length three giving the factors that define the shape of the exponential function (see 'Details').
<code>col.map</code>	the color to be used for the background map.
<code>legend</code>	a logical indicating whether a legend should be plotted.
<code>legtitle</code>	the title for the legend.
<code>cex.legtitle</code>	the character expansion factor to be used for the title of the legend.
<code>cex.legtext</code>	the character expansion factor to be used in the legend.
<code>ncircles</code>	the number of circles displayed in the legend.
<code>ndigits</code>	the number of digits displayed in the legend. Note that \ this is just a suggestion (see <code>format()</code> ).



`interactive` a logical indicating whether information about certain observations can be displayed interactively (see ‘Details’).

... for `growdotMiss`, further arguments and graphical parameters to be passed to `bgmap()`. For `bubbleMiss`, the arguments to be passed to `growdotMiss`.

## Details

The smallest dots correspond to the 10\ the 99\ defining the shape of the exponential function. Missings/imputed missings in the variable of interest will be drawn as rectangles.

If `interactive=TRUE`, detailed information for an observation can be printed on the console by clicking on the corresponding point. Clicking in a region that does not contain any points quits the interactive session.

## Note

The function was renamed to `growdotMiss` in version 1.3. `bubbleMiss` is a (deprecated) wrapper for `growdotMiss` for back compatibility with older versions. However, due to extended functionality, some of the argument positions have changed.

The code is based on (removed from CRAN) `bubbleFIN` from package `StatDA`.

## Author(s)

Andreas Alfons, Matthias Templ, Peter Filzmoser, Bernd Prantner

## References

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

## See Also

`bgmap()`, `mapMiss()`, `colormapMiss()`

## Examples

```
data(chorizonDL, package = "VIM")
data(kola.background, package = "VIM")
coo <- chorizonDL[, c("XC00", "YC00")]
## for missing values
x <- chorizonDL[, c("Ca", "As", "Bi")]
growdotMiss(x, coo, kola.background, border = "white")

## for imputed values
x_imp <- kNN(chorizonDL[,c("Ca", "As", "Bi" )])
growdotMiss(x_imp, coo, kola.background, delimiter = "_imp", border = "white")
```

histMiss

*Histogram with information about missing/imputed values***Description**

Histogram with highlighting of missing/imputed values in other variables by splitting each bin into two parts. Additionally, information about missing/imputed values in the variable of interest is shown on the right hand side.

**Usage**

```
histMiss(
  x,
  delimiter = NULL,
  pos = 1,
  selection = c("any", "all"),
  breaks = "Sturges",
  right = TRUE,
  col = c("skyblue", "red", "skyblue4", "red4", "orange", "orange4"),
  border = NULL,
  main = NULL,
  sub = NULL,
  xlab = NULL,
  ylab = NULL,
  axes = TRUE,
  only.miss = TRUE,
  miss.labels = axes,
  interactive = TRUE,
  ...
)
```

**Arguments**

<code>x</code>	a vector, matrix or <code>data.frame</code> .
<code>delimiter</code>	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, <code>x</code> needs to have <code>colnames()</code> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see <code>col</code> ).
<code>pos</code>	a numeric value giving the index of the variable of interest. Additional variables in <code>x</code> are used for highlighting.
<code>selection</code>	the selection method for highlighting missing/imputed values in multiple additional variables. Possible values are "any" (highlighting of missing/imputed values in <i>any</i> of the additional variables) and "all" (highlighting of missing/imputed values in <i>all</i> of the additional variables).

<code>breaks</code>	either a character string naming an algorithm to compute the breakpoints (see <a href="#">hist()</a> ), or a numeric value giving the number of cells.
<code>right</code>	logical; if TRUE, the histogram cells are right-closed (left-open) intervals.
<code>col</code>	a vector of length six giving the colors to be used. If only one color is supplied, the bars are transparent and the supplied color is used for highlighting missing/imputed values. Else if two colors are supplied, they are recycled.
<code>border</code>	the color to be used for the border of the cells. Use <code>border=NA</code> to omit borders.
<code>main, sub</code>	main and sub title.
<code>xlab, ylab</code>	axis labels.
<code>axes</code>	a logical indicating whether axes should be drawn on the plot.
<code>only.miss</code>	logical; if TRUE, the missing/imputed values in the first variable are visualized by a single bar. Otherwise, a small barplot is drawn on the right hand side (see ‘Details’).
<code>miss.labels</code>	either a logical indicating whether label(s) should be plotted below the bar(s) on the right hand side, or a character string or vector giving the label(s) (see ‘Details’).
<code>interactive</code>	a logical indicating whether the variables can be switched interactively (see ‘Details’).
<code>...</code>	further graphical parameters to be passed to <a href="#">graphics::title()</a> and <a href="#">graphics::axis()</a> .

## Details

If more than one variable is supplied, the bins for the variable of interest will be split according to missingness/number of imputed missings in the additional variables.

If `only.miss=TRUE`, the missing/imputed values in the variable of interest are visualized by one bar on the right hand side. If additional variables are supplied, this bar is again split into two parts according to missingness/number of imputed missings in the additional variables.

Otherwise, a small barplot consisting of two bars is drawn on the right hand side. The first bar corresponds to observed values in the variable of interest and the second bar to missing/imputed values. Since these two bars are not on the same scale as the main barplot, a second y-axis is plotted on the right (if `axes=TRUE`). Each of the two bars are again split into two parts according to missingness/number of imputed missings in the additional variables. Note that this display does not make sense if only one variable is supplied, therefore `only.miss` is ignored in that case.

If `interactive=TRUE`, clicking in the left margin of the plot results in switching to the previous variable and clicking in the right margin results in switching to the next variable. Clicking anywhere else on the graphics device quits the interactive session. When switching to a categorical variable, a barplot is produced rather than a histogram.

## Value

a list with the following components:

- `breaks` the breakpoints.
- `counts` the number of observations in each cell.
- `missings` the number of highlighted observations in each cell.
- `mids` the cell midpoints.

**Note**

Some of the argument names and positions have changed with version 1.3 due to extended functionality and for more consistency with other plot functions in VIM. For back compatibility, the arguments `axisnames` and `names.miss` can still be supplied to `...{}` and are handled correctly. Nevertheless, they are deprecated and no longer documented. Use `miss.labels` instead.

**Author(s)**

Andreas Alfons, Bernd Prantner

**References**

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

**See Also**

`spineMiss()`, `barMiss()`

Other plotting functions: `aggr()`, `barMiss()`, `marginmatrix()`, `marginplot()`, `matrixplot()`, `mosaicMiss()`, `pairsVIM()`, `parcoordMiss()`, `pbox()`, `scattJitt()`, `scattMiss()`, `scattmatrixMiss()`, `spineMiss()`

**Examples**

```
data(tao, package = "VIM")
## for missing values
x <- tao[, c("Air.Temp", "Humidity")]
histMiss(x)
histMiss(x, only.miss = FALSE)

## for imputed values
x_IMPUTED <- kNN(tao[, c("Air.Temp", "Humidity")])
histMiss(x_IMPUTED, delimiter = "_imp")
histMiss(x_IMPUTED, delimiter = "_imp", only.miss = FALSE)
```

---

hotdeck

*Hot-Deck Imputation*


---

**Description**

Implementation of the popular Sequential, Random (within a domain) hot-deck algorithm for imputation.

**Usage**

```
hotdeck(  
  data,  
  variable = NULL,  
  ord_var = NULL,  
  domain_var = NULL,  
  makeNA = NULL,  
  NAcond = NULL,  
  impNA = TRUE,  
  donorcond = NULL,  
  imp_var = TRUE,  
  imp_suffix = "imp"  
)
```

**Arguments**

data	data.frame or matrix
variable	variables where missing values should be imputed (not overlapping with ord_var)
ord_var	variables for sorting the data set before imputation (not overlapping with variable)
domain_var	variables for building domains and impute within these domains
makeNA	list of length equal to the number of variables, with values, that should be converted to NA for each variable
NAcond	list of length equal to the number of variables, with a condition for imputing a NA
impNA	TRUE/FALSE whether NA should be imputed
donorcond	list of length equal to the number of variables, with a donorcond condition as character string. e.g. ">5" or c(">5","<10"). If the list element for a variable is NULL no condition will be applied for this variable.
imp_var	TRUE/FALSE if a TRUE/FALSE variables for each imputed variable should be created show the imputation status
imp_suffix	suffix for the TRUE/FALSE variables showing the imputation status

**Value**

the imputed data set.

**Note**

If the sequential hotdeck does not lead to a suitable, a random donor in the group will be used.

**Author(s)**

Alexander Kowarik

## References

A. Kowarik, M. Templ (2016) Imputation with R package VIM. *Journal of Statistical Software*, 74(7), 1-16.

## See Also

Other imputation methods: `impPCA()`, `irmi()`, `kNN()`, `matchImpute()`, `medianSamp()`, `rangerImpute()`, `regressionImp()`, `sampleCat()`, `vimpute()`, `xgboostImpute()`

## Examples

```
data(sleep)
sleepI <- hotdeck(sleep)
sleepI2 <- hotdeck(sleep,ord_var="BodyWgt",domain_var="Pred")

# Usage of donorcond in a simple example
sleepI3 <- hotdeck(
  sleep,
  variable = c("NonD", "Dream", "Sleep", "Span", "Gest"),
  ord_var = "BodyWgt", domain_var = "Pred",
  donorcond = list(">4", "<17", ">1.5", "%between%c(8,13)", ">5")
)

set.seed(132)
nRows <- 1e3
# Generate a data set with nRows rows and several variables
x <- data.frame(
  x = rnorm(nRows), y = rnorm(nRows),
  z = sample(LETTERS, nRows, replace = TRUE),
  d1 = sample(LETTERS[1:3], nRows, replace = TRUE),
  d2 = sample(LETTERS[1:2], nRows, replace = TRUE),
  o1 = rnorm(nRows), o2 = rnorm(nRows), o3 = rnorm(100)
)
origX <- x
x[sample(1:nRows,nRows/10), 1] <- NA
x[sample(1:nRows,nRows/10), 2] <- NA
x[sample(1:nRows,nRows/10), 3] <- NA
x[sample(1:nRows,nRows/10), 4] <- NA
xImp <- hotdeck(x,ord_var = c("o1", "o2", "o3"), domain_var = "d2")
```

---

impPCA

---

*Iterative EM PCA imputation*


---

## Description

Greedy algorithm for EM-PCA including robust methods

**Usage**

```
impPCA(  
  x,  
  method = "classical",  
  m = 1,  
  eps = 0.5,  
  k = ncol(x) - 1,  
  maxit = 100,  
  boot = FALSE,  
  verbose = TRUE  
)
```

**Arguments**

x	data.frame or matrix
method	"classical" or "mcd" (robust estimation)
m	number of multiple imputations (only if parameter boot equals TRUE)
eps	threshold for convergence
k	number of principal components for reconstruction of x
maxit	maximum number of iterations
boot	residual bootstrap (if TRUE)
verbose	TRUE/FALSE if additional information about the imputation process should be printed

**Value**

the imputed data set. If boot = FALSE this is a data.frame. If boot = TRUE this is a list where each list element contains a data.frame.

**Author(s)**

Matthias Templ

**References**

Serneels, Sven and Verdonck, Tim (2008). Principal component analysis for data containing outliers and missing elements. Computational Statistics and Data Analysis, Elsevier, vol. 52(3), pages 1712-1727

**See Also**

Other imputation methods: [hotdeck\(\)](#), [irmi\(\)](#), [kNN\(\)](#), [matchImpute\(\)](#), [medianSamp\(\)](#), [rangerImpute\(\)](#), [regressionImp\(\)](#), [sampleCat\(\)](#), [vimpute\(\)](#), [xgboostImpute\(\)](#)

## Examples

```
data(Animals, package = "MASS")
Animals$brain[19] <- Animals$brain[19] + 0.01
Animals <- log(Animals)
colnames(Animals) <- c("log(body)", "log(brain)")
Animals_na <- Animals
probs <- abs(Animals$log(body)^2)
probs <- rep(0.5, nrow(Animals))
probs[c(6,16,26)] <- 0
set.seed(1234)
Animals_na[sample(1:nrow(Animals), 10, prob = probs), "log(brain)"] <- NA
w <- is.na(Animals_na$log(brain))
impPCA(Animals_na)
impPCA(Animals_na, method = "mcd")
impPCA(Animals_na, boot = TRUE, m = 10)
impPCA(Animals_na, method = "mcd", boot = TRUE)[[1]]
plot(log(brain) ~ log(body), data = Animals, type = "n", ylab = "", xlab="")
mtext(text = "impPCA robust", side = 3)
points(Animals$log(body)[!w], Animals$log(brain)[!w])
points(Animals$log(body)[w], Animals$log(brain)[w], col = "grey", pch = 17)
imputed <- impPCA(Animals_na, method = "mcd", boot = TRUE)[[1]]
colnames(imputed) <- c("log(body)", "log(brain)")
points(imputed$log(body)[w], imputed$log(brain)[w], col = "red", pch = 20, cex = 1.4)
segments(x0 = Animals$log(body)[w], x1 = imputed$log(body)[w], y0 = Animals$log(brain)[w],
y1 = imputed$log(brain)[w], lty = 2, col = "grey")
legend("topleft", legend = c("non-missings", "set to missing", "imputed values"),
pch = c(1,17,20), col = c("black","grey","red"), cex = 0.7)
mape <- round(100* 1/sum(is.na(Animals_na$log(brain))) * sum(abs((Animals$log(brain) -
imputed$log(brain)) / Animals$log(brain))), 2)
s2 <- var(Animals$log(brain))
nrmse <- round(sqrt(1/sum(is.na(Animals_na$log(brain))) * sum(abs((Animals$log(brain) -
imputed$log(brain)) / s2))), 2)
text(x = 8, y = 1.5, labels = paste("MAPE =", mape))
text(x = 8, y = 0.5, labels = paste("NRMSE =", nrmse))
```

---

imputeRobust

*Robust imputation*


---

## Description

Multiple imputation using classical and robust methods accounting for model and imputation uncertainty.

## Usage

```
imputeRobust(
  form,
  data,
```



```

boot = TRUE,
robustboot = "stratified",
method = "MM",
takeAll = TRUE,
alpha = 0.75,
uncert = "pmm",
family = "Gaussian",
value_back = "all"
)

```

### Arguments

form	Model formulas as a list.
data	Data set to impute
boot	Accounting for model uncertainty with a classical bootstrap, Default: TRUE
robustboot	Accounting for model uncertainty with robust bootstrap methods, Default: 'stratified'
method	Imputation method, Default: 'MM'
takeAll	Missing values are initialized when TRUE, Default: TRUE
alpha	Relative size of good data points. Used for the robust bootstrap methods, Default: 0.75
uncert	Imputation uncertainty method, Default: 'pmm'
family	Not supported and ignored. Foreseen for future versions, Default: 'Gaussian'
value_back	Only observations with imputed values as return object (ymiss), or the whole data set, Default: 'all'

### Details

Complex formulas can be provided for each variable in your data set.

### Value

Imputed data set.

### See Also

[initialise lmrob gam pdist](#)

### Examples

```

## Not run:
if(interactive()){
  #EXAMPLE1
}

## End(Not run)

```

---

imputeRobustChain	<i>FUNCTION_TITLE</i>
-------------------	-----------------------

---

**Description**

FUNCTION\_DESCRIPTION

**Usage**

```
imputeRobustChain(
  formulas = vector("list", ncol(data)),
  data,
  boot = TRUE,
  robustboot = TRUE,
  method = "lts",
  multinom.method = "multinom",
  takeAll = TRUE,
  eps = 0.5,
  maxit = 4,
  alpha = 0.5,
  uncert = "pmm",
  familiy = "Gaussian",
  value_back = "matrix",
  trace = FALSE
)
```

**Arguments**

formulas	PARAM_DESCRIPTION, Default: vector("list", ncol(data))
data	PARAM_DESCRIPTION
boot	PARAM_DESCRIPTION, Default: TRUE
robustboot	PARAM_DESCRIPTION, Default: TRUE
method	PARAM_DESCRIPTION, Default: 'lts'
multinom.method	PARAM_DESCRIPTION, Default: 'multinom'
takeAll	PARAM_DESCRIPTION, Default: TRUE
eps	PARAM_DESCRIPTION, Default: 0.5
maxit	PARAM_DESCRIPTION, Default: 4
alpha	PARAM_DESCRIPTION, Default: 0.5
uncert	PARAM_DESCRIPTION, Default: 'pmm'
familiy	PARAM_DESCRIPTION, Default: 'Gaussian'
value_back	PARAM_DESCRIPTION, Default: 'matrix'
trace	PARAM_DESCRIPTION, Default: FALSE

**Details**

DETAILS

**Value**

OUTPUT\_DESCRIPTION

**See Also**[initialise outCoDa](#)**Examples**

```
## Not run:
if(interactive()){
  #EXAMPLE1
}

## End(Not run)
```

initialise

*Initialization of missing values***Description**

Rough estimation of missing values in a vector according to its type.

**Usage**

```
initialise(x, mixed, method = "kNN", mixed.constant = NULL)
```

**Arguments**

<code>x</code>	a vector.
<code>mixed</code>	a character vector containing the names of variables of type mixed (semi-continous).
<code>method</code>	Method used for Initialization (median or kNN)
<code>mixed.constant</code>	vector with length equal to the number of semi-continuous variables specifying the point of the semi-continuous distribution with non-zero probability

**Details**

Missing values are imputed with the mean for vectors of class "numeric", with the median for vectors of class "integer", and with the mode for vectors of class "factor". Hence, x should be prepared in the following way: assign class "numeric" to numeric vectors, assign class "integer" to ordinal vectors, and assign class "factor" to nominal or binary vectors.

**Value**

the initialized vector.

**Note**

The function is used internally by some imputation algorithms.

**Author(s)**

Matthias Templ, modifications by Andreas Alfons

---

irmi

*Iterative robust model-based imputation (IRMI)*

---

**Description**

In each step of the iteration, one variable is used as a response variable and the remaining variables serve as the regressors.

**Usage**

```
irmi(  
  x,  
  eps = 5,  
  maxit = 100,  
  mixed = NULL,  
  mixed.constant = NULL,  
  count = NULL,  
  step = FALSE,  
  robust = FALSE,  
  takeAll = TRUE,  
  noise = TRUE,  
  noise.factor = 1,  
  force = FALSE,  
  robMethod = "lmrob",  
  force.mixed = TRUE,  
  mi = 1,  
  addMixedFactors = FALSE,  
  trace = FALSE,  
  init.method = "kNN",  
  modelFormulas = NULL,  
  multinom.method = "multinom",  
  imp_var = TRUE,  
  imp_suffix = "imp"  
)
```

## Arguments

<code>x</code>	data.frame or matrix
<code>eps</code>	threshold for convergency
<code>maxit</code>	maximum number of iterations
<code>mixed</code>	column index of the semi-continuous variables
<code>mixed.constant</code>	vector with length equal to the number of semi-continuous variables specifying the point of the semi-continuous distribution with non-zero probability
<code>count</code>	column index of count variables
<code>step</code>	a stepwise model selection is applied when the parameter is set to TRUE
<code>robust</code>	if TRUE, robust regression methods will be applied
<code>takeAll</code>	takes information of (initialised) missings in the response as well for regression imputation.
<code>noise</code>	irmi has the option to add a random error term to the imputed values, this creates the possibility for multiple imputation. The error term has mean 0 and variance corresponding to the variance of the regression residuals.
<code>noise.factor</code>	amount of noise.
<code>force</code>	if TRUE, the algorithm tries to find a solution in any case, possible by using different robust methods automatically.
<code>robMethod</code>	regression method when the response is continuous. Default is MM-regression with <code>lmrob</code> .
<code>force.mixed</code>	if TRUE, the algorithm tries to find a solution in any case, possible by using different robust methods automatically.
<code>mi</code>	number of multiple imputations.
<code>addMixedFactors</code>	if TRUE add additional factor variable for each mixed variable as X variable in the regression
<code>trace</code>	Additional information about the iterations when trace equals TRUE.
<code>init.method</code>	Method for initialization of missing values (kNN or median)
<code>modelFormulas</code>	a named list with the name of variables for the rhs of the formulas, which must contain a rhs formula for each variable with missing values, it should look like <code>'list(y1=c("x1","x2"),y2=c("x1","x3"))'</code> if factor variables for the mixed variables should be created for the regression models
<code>multinom.method</code>	Method for estimating the multinomial models (current default and only available method is <code>multinom</code> )
<code>imp_var</code>	TRUE/FALSE if a TRUE/FALSE variables for each imputed variable should be created show the imputation status
<code>imp_suffix</code>	suffix for the TRUE/FALSE variables showing the imputation status

## Details

The method works sequentially and iterative. The method can deal with a mixture of continuous, semi-continuous, ordinal and nominal variables including outliers.

A full description of the method can be found in the mentioned reference.

**Value**

the imputed data set.

**Author(s)**

Matthias Templ, Alexander Kowarik

**References**

M. Templ, A. Kowarik, P. Filzmoser (2011) Iterative stepwise regression imputation using standard and robust methods. *Journal of Computational Statistics and Data Analysis*, Vol. 55, pp. 2793-2806.

A. Kowarik, M. Templ (2016) Imputation with R package VIM. *Journal of Statistical Software*, 74(7), 1-16.

M. Templ (2023) *Visualization and Imputation of Missing Values*. Springer Publishing. Series in Computational Statistics. Cham. Switzerland. 463 pages. DOI: 10.1007/978-3-031-30073-8

**See Also**

Other imputation methods: [hotdeck\(\)](#), [impPCA\(\)](#), [kNN\(\)](#), [matchImpute\(\)](#), [medianSamp\(\)](#), [rangerImpute\(\)](#), [regressionImp\(\)](#), [sampleCat\(\)](#), [vimpute\(\)](#), [xgboostImpute\(\)](#)

**Examples**

```
data(sleep)
irmi(sleep)

data(testdata)
imp_testdata1 <- irmi(testdata$wna, mixed = testdata$mixed)

# mixed.constant != 0 (-10)
testdata$wna$m1[testdata$wna$m1 == 0] <- -10
testdata$wna$m2 <- log(testdata$wna$m2 + 0.001)
imp_testdata2 <- irmi(
  testdata$wna,
  mixed = testdata$mixed,
  mixed.constant = c(-10, log(0.001))
)
imp_testdata2$m2 <- exp(imp_testdata2$m2) - 0.001

#example with fixed formulas for the variables with missing
form = list(
  NonD = c("BodyWgt", "BrainWgt"),
  Dream = c("BodyWgt", "BrainWgt"),
  Sleep = c("BrainWgt"           ),
  Span  = c("BodyWgt"           ),
  Gest  = c("BodyWgt", "BrainWgt")
)
irmi(sleep, modelFormulas = form, trace = TRUE)
```

```
# Example with ordered variable
td <- testdata$wna
td$c1 <- as.ordered(td$c1)
irmi(td)
```

---

kNN

*k-Nearest Neighbour Imputation*


---

### Description

k-Nearest Neighbour Imputation based on a variation of the Gower Distance for numerical, categorical, ordered and semi-continuous variables.

### Usage

```
kNN(
  data,
  variable = colnames(data),
  k = 5,
  dist_var = colnames(data),
  weights = NULL,
  numFun = median,
  catFun = maxCat,
  makeNA = NULL,
  NAcond = NULL,
  impNA = TRUE,
  donorcond = NULL,
  mixed = vector(),
  mixed.constant = NULL,
  trace = FALSE,
  imp_var = TRUE,
  imp_suffix = "imp",
  addRF = FALSE,
  onlyRF = FALSE,
  addRandom = FALSE,
  useImputedDist = TRUE,
  weightDist = FALSE,
  methodStand = "range",
  ordFun = medianSamp
)
```

### Arguments

data	data.frame or matrix
variable	variables where missing values should be imputed
k	number of Nearest Neighbours used

<code>dist_var</code>	names or variables to be used for distance calculation
<code>weights</code>	weights for the variables for distance calculation. If <code>weights = "auto"</code> weights will be selected based on variable importance from random forest regression, using function <code>ranger::ranger()</code> . Weights are calculated for each variable separately.
<code>numFun</code>	function for aggregating the k Nearest Neighbours in the case of a numerical variable
<code>catFun</code>	function for aggregating the k Nearest Neighbours in the case of a categorical variable
<code>makeNA</code>	list of length equal to the number of variables, with values, that should be converted to NA for each variable
<code>NAcond</code>	list of length equal to the number of variables, with a condition for imputing a NA
<code>impNA</code>	TRUE/FALSE whether NA should be imputed
<code>donorcond</code>	list of length equal to the number of variables, with a donorcond condition as character string. e.g. a list element can be ">5" or <code>c("&gt;5", "&lt;10")</code> . If the list element for a variable is NULL no condition will be applied for this variable.
<code>mixed</code>	names of mixed variables
<code>mixed.constant</code>	vector with length equal to the number of semi-continuous variables specifying the point of the semi-continuous distribution with non-zero probability
<code>trace</code>	TRUE/FALSE if additional information about the imputation process should be printed
<code>imp_var</code>	TRUE/FALSE if a TRUE/FALSE variables for each imputed variable should be created show the imputation status
<code>imp_suffix</code>	suffix for the TRUE/FALSE variables showing the imputation status
<code>addRF</code>	TRUE/FALSE each variable will be modelled using random forest regression ( <code>ranger::ranger()</code> ) and used as additional distance variable.
<code>onlyRF</code>	TRUE/FALSE if TRUE only additional distance variables created from random forest regression will be used as distance variables.
<code>addRandom</code>	TRUE/FALSE if an additional random variable should be added for distance calculation
<code>useImputedDist</code>	TRUE/FALSE if an imputed value should be used for distance calculation for imputing another variable. Be aware that this results in a dependency on the ordering of the variables.
<code>weightDist</code>	TRUE/FALSE if the distances of the k nearest neighbours should be used as weights in the aggregation step
<code>methodStand</code>	either "range" or "iqr" to be used in the standardization of numeric variables in the gower distance
<code>ordFun</code>	function for aggregating the k Nearest Neighbours in the case of a ordered factor variable

### Value

the imputed data set.



**Author(s)**

Alexander Kowarik, Statistik Austria

**References**

A. Kowarik, M. Templ (2016) Imputation with R package VIM. *Journal of Statistical Software*, 74(7), 1-16.

**See Also**

Other imputation methods: [hotdeck\(\)](#), [impPCA\(\)](#), [irmi\(\)](#), [matchImpute\(\)](#), [medianSamp\(\)](#), [rangerImpute\(\)](#), [regressionImp\(\)](#), [sampleCat\(\)](#), [vimpute\(\)](#), [xgboostImpute\(\)](#)

**Examples**

```
data(sleep)
kNN(sleep)
library(laeken)
kNN(sleep, numFun = weightedMean, weightDist=TRUE)
```

---

kola.background	<i>Background map for the Kola project data</i>
-----------------	---

---

**Description**

Coordinates of the Kola background map.

**Source**

Kola Project (1993-1998)

**References**

Reimann, C., Filzmoser, P., Garrett, R.G. and Dutter, R. (2008) *Statistical Data Analysis Explained: Applied Environmental Statistics with R*. Wiley, 2008.

**Examples**

```
data(kola.background, package = "VIM")
bgmap(kola.background)
```

mapMiss

*Map with information about missing/imputed values***Description**

Map of observed and missing/imputed values.

**Usage**

```
mapMiss(
  x,
  coords,
  map,
  delimiter = NULL,
  selection = c("any", "all"),
  col = c("skyblue", "red", "orange"),
  alpha = NULL,
  pch = c(19, 15),
  col.map = grey(0.5),
  legend = TRUE,
  interactive = TRUE,
  ...
)
```

**Arguments**

<code>x</code>	a vector, matrix or data.frame.
<code>coords</code>	a data.frame or matrix with two columns giving the spatial coordinates of the observations.
<code>map</code>	a background map to be passed to <code>bgmap()</code> .
<code>delimiter</code>	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, <code>x</code> needs to have <code>colnames()</code> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see <code>col</code> ).
<code>selection</code>	the selection method for displaying missing/imputed values in the map. Possible values are "any" (display missing/imputed values in <i>any</i> variable) and "all" (display missing/imputed values in <i>all</i> variables).
<code>col</code>	a vector of length three giving the colors to be used for observed, missing and imputed values. If a single color is supplied, it is used for all values.
<code>alpha</code>	a numeric value between 0 and 1 giving the level of transparency of the colors, or NULL. This can be used to prevent overplotting.
<code>pch</code>	a vector of length two giving the plot characters to be used for observed and missing/imputed values. If a single plot character is supplied, it will be used for both.

<code>col.map</code>	the color to be used for the background map.
<code>legend</code>	a logical indicating whether a legend should be plotted.
<code>interactive</code>	a logical indicating whether information about selected observations can be displayed interactively (see ‘Details’).
<code>...</code>	further graphical parameters to be passed to <code>bgmap()</code> and <code>graphics::points()</code> .

### Details

If `interactive=TRUE`, detailed information for an observation can be printed on the console by clicking on the corresponding point. Clicking in a region that does not contain any points quits the interactive session.

### Author(s)

Matthias Templ, Andreas Alfons, modifications by Bernd Prantner

### References

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

### See Also

`bgmap()`, `bubbleMiss()`, `colormapMiss()`

### Examples

```
data(chorizonDL, package = "VIM")
data(kola.background, package = "VIM")
coo <- chorizonDL[, c("XC00", "YC00")]
## for missing values
x <- chorizonDL[, c("As", "Bi")]
mapMiss(x, coo, kola.background)

## for imputed values
x_imp <- kNN(chorizonDL[, c("As", "Bi")])
mapMiss(x_imp, coo, kola.background, delimiter = "_imp")
```

---

marginmatrix

---

*Marginplot Matrix*


---

### Description

Create a scatterplot matrix with information about missing/imputed values in the plot margins of each panel.

**Usage**

```
marginmatrix(
  x,
  delimiter = NULL,
  col = c("skyblue", "red", "red4", "orange", "orange4"),
  alpha = NULL,
  ...
)
```

**Arguments**

<code>x</code>	a matrix or <code>data.frame</code> .
<code>delimiter</code>	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, <code>x</code> needs to have <code>colnames()</code> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see <code>col</code> ).
<code>col</code>	a vector of length five giving the colors to be used in the marginplots in the off-diagonal panels. The first color is used for the scatterplot and the boxplots for the available data, the second/fourth color for the univariate scatterplots and boxplots for the missing/imputed values in one variable, and the third/fifth color for the frequency of missing/imputed values in both variables (see ‘Details’). If only one color is supplied, it is used for the bivariate and univariate scatterplots and the boxplots for missing/imputed values in one variable, whereas the boxplots for the available data are transparent. Else if two colors are supplied, the second one is recycled.
<code>alpha</code>	a numeric value between 0 and 1 giving the level of transparency of the colors, or <code>NULL</code> . This can be used to prevent overplotting.
<code>...</code>	further arguments and graphical parameters to be passed to <code>pairsVIM()</code> and <code>marginplot()</code> . <code>par("oma")</code> will be set appropriately unless supplied (see <code>graphics::par()</code> ).

**Details**

`marginmatrix` uses `pairsVIM()` with a panel function based on `marginplot()`.

The graphical parameter `oma` will be set unless supplied as an argument.

**Author(s)**

Andreas Alfons, modifications by Bernd Prantner

**References**

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

**See Also**

`marginplot()`, `pairsVIM()`, `scattmatrixMiss()`

Other plotting functions: `aggr()`, `barMiss()`, `histMiss()`, `marginplot()`, `matrixplot()`, `mosaicMiss()`, `pairsVIM()`, `parcoordMiss()`, `pbox()`, `scattJitt()`, `scattMiss()`, `scattmatrixMiss()`, `spineMiss()`

**Examples**

```
data(sleep, package = "VIM")
## for missing values
x <- sleep[, 1:5]
x[,c(1,2,4)] <- log10(x[,c(1,2,4)])
marginmatrix(x)

## for imputed values
x_imp <- kNN(sleep[, 1:5])
x_imp[,c(1,2,4)] <- log10(x_imp[,c(1,2,4)])
marginmatrix(x_imp, delimiter = "_imp")
```

---

marginplot

---

*Scatterplot with additional information in the margins*


---

**Description**

In addition to a standard scatterplot, information about missing/imputed values is shown in the plot margins. Furthermore, imputed values are highlighted in the scatterplot.

**Usage**

```
marginplot(
  x,
  delimiter = NULL,
  col = c("skyblue", "red", "red4", "orange", "orange4"),
  alpha = NULL,
  pch = c(1, 16),
  cex = par("cex"),
  numbers = TRUE,
  cex.numbers = par("cex"),
  zeros = FALSE,
  xlim = NULL,
  ylim = NULL,
  main = NULL,
  sub = NULL,
  xlab = NULL,
  ylab = NULL,
  ann = par("ann"),
  axes = TRUE,
```

```

    frame.plot = axes,
    ...
)

```

## Arguments

<code>x</code>	a matrix or data.frame with two columns.
<code>delimiter</code>	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, <code>x</code> needs to have <code>colnames()</code> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see <code>col</code> ).
<code>col</code>	a vector of length five giving the colors to be used in the plot. The first color is used for the scatterplot and the boxplots for the available data. In case of missing values, the second color is taken for the univariate scatterplots and boxplots for missing values in one variable and the third for the frequency of missing/imputed values in both variables (see ‘Details’). Otherwise, in case of imputed values, the fourth color is used for the highlighting, the frequency, the univariate scatterplot and the boxplots of mputed values in the first variable and the fifth color for the same applied to the second variable. A black color is used for the highlighting and the frequency of imputed values in both variables instead. If only one color is supplied, it is used for the bivariate and univariate scatterplots and the boxplots for missing/imputed values in one variable, whereas the boxplots for the available data are transparent. Else if two colors are supplied, the second one is recycled.
<code>alpha</code>	a numeric value between 0 and 1 giving the level of transparency of the colors, or NULL. This can be used to prevent overplotting.
<code>pch</code>	a vector of length two giving the plot symbols to be used for the scatterplot and the univariate scatterplots. If a single plot character is supplied, it is used for the scatterplot and the default value will be used for the univariate scatterplots (see ‘Details’).
<code>cex</code>	the character expansion factor to be used for the bivariate and univariate scatterplots.
<code>numbers</code>	a logical indicating whether the frequencies of missing/imputed values should be displayed in the lower left of the plot (see ‘Details’).
<code>cex.numbers</code>	the character expansion factor to be used for the frequencies of the missing/imputed values.
<code>zeros</code>	a logical vector of length two indicating whether the variables are semi-continuous, i.e., contain a considerable amount of zeros. If TRUE, only the non-zero observations are used for drawing the respective boxplot. If a single logical is supplied, it is recycled.
<code>xlim, ylim</code>	axis limits.
<code>main, sub</code>	main and sub title.
<code>xlab, ylab</code>	axis labels.

<code>ann</code>	a logical indicating whether plot annotation (main, sub, xlab, ylab) should be displayed.
<code>axes</code>	a logical indicating whether both axes should be drawn on the plot. Use graphical parameter "xaxt" or "yaxt" to suppress only one of the axes.
<code>frame.plot</code>	a logical indicating whether a box should be drawn around the plot.
<code>...</code>	further graphical parameters to be passed down (see <a href="#">graphics::par()</a> ).

## Details

Boxplots for available and missing/imputed data, as well as univariate scatterplots for missing/imputed values in one variable are shown in the plot margins.

Imputed values in either of the variables are highlighted in the scatterplot.

Furthermore, the frequencies of the missing/imputed values can be displayed by a number (lower left of the plot). The number in the lower left corner is the number of observations that are missing/imputed in both variables.

## Note

Some of the argument names and positions have changed with versions 1.3 and 1.4 due to extended functionality and for more consistency with other plot functions in VIM. For back compatibility, the argument `cex.text` can still be supplied to `...{}` and is handled correctly. Nevertheless, it is deprecated and no longer documented. Use `cex.numbers` instead.

## Author(s)

Andreas Alfons, Matthias Templ, modifications by Bernd Prantner

## References

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

## See Also

[scattMiss\(\)](#)

Other plotting functions: [aggr\(\)](#), [barMiss\(\)](#), [histMiss\(\)](#), [marginmatrix\(\)](#), [matrixplot\(\)](#), [mosaicMiss\(\)](#), [pairsVIM\(\)](#), [parcoordMiss\(\)](#), [pbox\(\)](#), [scattJitt\(\)](#), [scattMiss\(\)](#), [scattmatrixMiss\(\)](#), [spineMiss\(\)](#)

## Examples

```
data(tao, package = "VIM")
data(chorizonDL, package = "VIM")
## for missing values
marginplot(tao[,c("Air.Temp", "Humidity")])
marginplot(log10(chorizonDL[,c("Ca0", "Bi")]))
```

```
## for imputed values
marginplot(kNN(tao[,c("Air.Temp", "Humidity")]), delimiter = "_imp")
marginplot(kNN(log10(chorizonDL[,c("CaO", "Bi")])), delimiter = "_imp")
```

---

matchImpute

*Fast matching/imputation based on categorical variable*


---

## Description

Suitable donors are searched based on matching of the categorical variables. The variables are dropped in reversed order, so that the last element of 'match\_var' is dropped first and the first element of the vector is dropped last.

## Usage

```
matchImpute(
  data,
  variable = colnames(data)[!colnames(data) %in% match_var],
  match_var,
  imp_var = TRUE,
  imp_suffix = "imp"
)
```

## Arguments

data	data.frame, data.table or matrix
variable	variables to be imputed
match_var	variables used for matching
imp_var	TRUE/FALSE if a TRUE/FALSE variables for each imputed variable should be created show the imputation status
imp_suffix	suffix for the TRUE/FALSE variables showing the imputation status

## Details

The method works by sampling values from the suitable donors.

## Value

the imputed data set.

## Author(s)

Johannes Gussenbauer, Alexander Kowarik



**See Also**[hotdeck\(\)](#)

Other imputation methods: [hotdeck\(\)](#), [impPCA\(\)](#), [irmi\(\)](#), [kNN\(\)](#), [medianSamp\(\)](#), [rangerImpute\(\)](#), [regressionImp\(\)](#), [sampleCat\(\)](#), [vimpute\(\)](#), [xgboostImpute\(\)](#)

**Examples**

```
data(sleep, package="VIM")
imp_data <- matchImpute(sleep, variable=c("NonD", "Dream", "Sleep", "Span", "Gest"),
  match_var=c("Exp", "Danger"))

data(testdata, package="VIM")
imp_testdata1 <- matchImpute(testdata$wna, match_var=c("c1", "c2", "b1", "b2"))

dt <- data.table::data.table(testdata$wna)
imp_testdata2 <- matchImpute(dt, match_var=c("c1", "c2", "b1", "b2"))
```

matrixplot

*Matrix plot***Description**

Create a matrix plot, in which all cells of a data matrix are visualized by rectangles. Available data is coded according to a continuous color scheme, while missing/imputed data is visualized by a clearly distinguishable color.

**Usage**

```
matrixplot(
  x,
  delimiter = NULL,
  sortby = NULL,
  col = c("red", "orange"),
  fixup = TRUE,
  xlim = NULL,
  ylim = NULL,
  main = NULL,
  sub = NULL,
  xlab = NULL,
  ylab = NULL,
  axes = TRUE,
  labels = axes,
  xpd = NULL,
  interactive = TRUE,
  ...
)
```

## Arguments

<code>x</code>	a matrix or data.frame.
<code>delimiter</code>	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, <code>x</code> needs to have <code>colnames()</code> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see <code>col</code> ).
<code>sortby</code>	a numeric or character value specifying the variable to sort the data matrix by, or NULL to plot without sorting.
<code>col</code>	the colors to be used in the plot. RGB colors may be specified as character strings or as objects of class " <code>colorspace::RGB()</code> ". HCL colors need to be specified as objects of class " <code>colorspace::polarLUV()</code> ". If only one color is supplied, it is used for missing and imputed data and a greyscale is used for available data. If two colors are supplied, the first is used for missing and the second for imputed data and a greyscale for available data. If three colors are supplied, the first is used as end color for the available data, while the start color is taken to be transparent for RGB or white for HCL. Missing/imputed data is visualized by the second/third color in this case. If four colors are supplied, the first is used as start color and the second as end color for the available data, while the third/fourth color is used for missing/imputed data.
<code>fixup</code>	a logical indicating whether the colors should be corrected to valid RGB values (see <code>colorspace::hex()</code> ).
<code>xlim, ylim</code>	axis limits.
<code>main, sub</code>	main and sub title.
<code>xlab, ylab</code>	axis labels.
<code>axes</code>	a logical indicating whether axes should be drawn on the plot.
<code>labels</code>	either a logical indicating whether labels should be plotted below each column, or a character vector giving the labels.
<code>xpd</code>	a logical indicating whether the rectangles should be allowed to go outside the plot region. If NULL, it defaults to TRUE unless axis limits are specified.
<code>interactive</code>	a logical indicating whether a variable to be used for sorting can be selected interactively (see 'Details').
<code>...</code>	for <code>matrixplot</code> and <code>iimagMiss</code> , further graphical parameters to be passed to <code>graphics::plot.window()</code> , <code>graphics::title()</code> and <code>graphics::axis()</code> . For <code>TKRmatrixplot</code> , further arguments to be passed to <code>matrixplot</code> .

## Details

In a *matrix plot*, all cells of a data matrix are visualized by rectangles. Available data is coded according to a continuous color scheme. To compute the colors via interpolation, the variables are first scaled to the interval between 0 and 1. Missing/imputed values can then be visualized by a clearly distinguishable color. It is thereby possible to use colors in the *HCL* or *RGB* color space. A simple way of visualizing the magnitude of the available data is to apply a greyscale, which has the advantage that missing/imputed values can easily be distinguished by using a color such as

red/orange. Note that `-Inf` and `Inf` are always assigned the begin and end color, respectively, of the continuous color scheme.

Additionally, the observations can be sorted by the magnitude of a selected variable. If `interactive` is `TRUE`, clicking in a column redraws the plot with observations sorted by the corresponding variable. Clicking anywhere outside the plot region quits the interactive session.

## Note

This is a much more powerful extension to the function `imagmiss` in the former CRAN package `dprep`.

`iimagMiss` is deprecated and may be omitted in future versions of VIM. Use `matrixplot` instead.

## Author(s)

Andreas Alfons, Matthias Templ, modifications by Bernd Prantner

## References

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

## See Also

Other plotting functions: `aggr()`, `barMiss()`, `histMiss()`, `marginmatrix()`, `marginplot()`, `mosaicMiss()`, `pairsVIM()`, `parcoordMiss()`, `pbox()`, `scattJitt()`, `scattMiss()`, `scattmatrixMiss()`, `spineMiss()`

## Examples

```
data(sleep, package = "VIM")
## for missing values
x <- sleep[, -(8:10)]
x[, c(1,2,4,6,7)] <- log10(x[, c(1,2,4,6,7)])
matrixplot(x, sortby = "BrainWgt")

## for imputed values
x_imp <- kNN(sleep[, -(8:10)])
x_imp[, c(1,2,4,6,7)] <- log10(x_imp[, c(1,2,4,6,7)])
matrixplot(x_imp, delimiter = "_imp", sortby = "BrainWgt")
```

---

maxCat	<i>Aggregation function for a factor variable</i>
--------	---

---

**Description**

The function maxCat chooses the level with the most occurrences and random if the maximum is not unique.

**Usage**

```
maxCat(x, weights = NULL)
```

**Arguments**

x	factor vector
weights	numeric vector providing weights for the observations in x

---

medianSamp	<i>Aggregation function for a ordinal variable</i>
------------	--

---

**Description**

The function medianSamp chooses the level as the median or randomly between two levels.

**Usage**

```
medianSamp(x, weights = NULL)
```

**Arguments**

x	ordered factor vector
weights	numeric vector providing weights for the observations in x

**See Also**

Other imputation methods: [hotdeck\(\)](#), [impPCA\(\)](#), [irmi\(\)](#), [kNN\(\)](#), [matchImpute\(\)](#), [rangerImpute\(\)](#), [regressionImp\(\)](#), [sampleCat\(\)](#), [vimpute\(\)](#), [xgboostImpute\(\)](#)

mosaicMiss

*Mosaic plot with information about missing/imputed values***Description**

Create a mosaic plot with information about missing/imputed values.

**Usage**

```
mosaicMiss(
  x,
  delimiter = NULL,
  highlight = NULL,
  selection = c("any", "all"),
  plotvars = NULL,
  col = c("skyblue", "red", "orange"),
  labels = NULL,
  miss.labels = TRUE,
  ...
)
```

**Arguments**

<code>x</code>	a matrix or data.frame.
<code>delimiter</code>	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, <code>x</code> needs to have <code>colnames()</code> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see <code>col</code> ).
<code>highlight</code>	a vector giving the variables to be used for highlighting. If <code>NULL</code> (the default), all variables are used for highlighting.
<code>selection</code>	the selection method for highlighting missing/imputed values in multiple highlight variables. Possible values are "any" (highlighting of missing/imputed values in <i>any</i> of the highlight variables) and "all" (highlighting of missing/imputed values in <i>all</i> of the highlight variables).
<code>plotvars</code>	a vector giving the categorical variables to be plotted. If <code>NULL</code> (the default), all variables are plotted.
<code>col</code>	a vector of length three giving the colors to be used for observed, missing and imputed data. If only one color is supplied, the tiles corresponding to observed data are transparent and the supplied color is used for highlighting.
<code>labels</code>	a list of arguments for the labeling function <code>vcd::labeling_border()</code> .
<code>miss.labels</code>	either a logical indicating whether labels should be plotted for observed and missing/imputed (highlighted) data, or a character vector giving the labels.
<code>...</code>	additional arguments to be passed to <code>vcd::mosaic()</code> .

## Details

Mosaic plots are graphical representations of multi-way contingency tables. The frequencies of the different cells are visualized by area-proportional rectangles (tiles). Additional tiles are used to display the frequencies of missing/imputed values. Furthermore, missing/imputed values in a certain variable or combination of variables can be highlighted in order to explore their structure.

## Value

An object of class "structable" is returned invisibly.

## Note

This function uses the highly flexible strucplot framework of package vcd.

## Author(s)

Andreas Alfons, modifications by Bernd Prantner

## References

Meyer, D., Zeileis, A. and Hornik, K. (2006) The strucplot framework: Visualizing multi-way contingency tables with **vcd**. *Journal of Statistical Software*, **17** (3), 1–48.

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

## See Also

`spineMiss()`, `vcd::mosaic()`

Other plotting functions: `aggr()`, `barMiss()`, `histMiss()`, `marginmatrix()`, `marginplot()`, `matrixplot()`, `pairsVIM()`, `parcoordMiss()`, `pbox()`, `scattJitt()`, `scattMiss()`, `scattmatrixMiss()`, `spineMiss()`

## Examples

```
data(sleep, package = "VIM")
## for missing values
mosaicMiss(sleep, highlight = 4,
  plotvars = 8:10, miss.labels = FALSE)

## for imputed values
mosaicMiss(kNN(sleep), highlight = 4,
  plotvars = 8:10, delimiter = "_imp", miss.labels = FALSE)
```

**Description**

Create a scatterplot matrix.

**Usage**

```

pairsVIM(
  x,
  ...,
  delimiter = NULL,
  main = NULL,
  sub = NULL,
  panel = points,
  lower = panel,
  upper = panel,
  diagonal = NULL,
  labels = TRUE,
  pos.labels = NULL,
  cex.labels = NULL,
  font.labels = par("font"),
  layout = c("matrix", "graph"),
  gap = 1
)

```

**Arguments**

<code>x</code>	a matrix or data.frame.
<code>...</code>	further arguments and graphical parameters to be passed down. <code>par("oma")</code> will be set appropriately unless supplied (see <a href="#">graphics::par()</a> ).
<code>delimiter</code>	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, <code>x</code> needs to have <a href="#">colnames()</a> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see <code>col</code> ).
<code>main, sub</code>	main and sub title.
<code>panel</code>	a function( <code>x, y, ...{}</code> ), which is used to plot the contents of each off-diagonal panel of the display.
<code>lower, upper</code>	separate panel functions to be used below and above the diagonal, respectively.
<code>diagonal</code>	optional function( <code>x, ...{}</code> ) to be applied on the diagonal panels.
<code>labels</code>	either a logical indicating whether labels should be plotted in the diagonal panels, or a character vector giving the labels.

<code>pos.labels</code>	the vertical position of the labels in the diagonal panels.
<code>cex.labels</code>	the character expansion factor to be used for the labels.
<code>font.labels</code>	the font to be used for the labels.
<code>layout</code>	a character string giving the layout of the scatterplot matrix. Possible values are "matrix" (a matrix-like layout with the first row on top) and "graph" (a graph-like layout with the first row at the bottom).
<code>gap</code>	a numeric value giving the distance between the panels in margin lines.

### Details

This function is the workhorse for `marginmatrix()` and `scattmatrixMiss()`.

The graphical parameter `oma` will be set unless supplied as an argument.

A panel function should not attempt to start a new plot, since the coordinate system for each panel is set up by `pairsVIM`.

### Note

The code is based on `graphics::pairs()`. Starting with version 1.4, infinite values are no longer removed before passing the `x` and `y` vectors to the panel functions.

### Author(s)

Andreas Alfons, modifications by Bernd Prantner

### References

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

### See Also

`marginmatrix()`, `scattmatrixMiss()`

Other plotting functions: `aggr()`, `barMiss()`, `histMiss()`, `marginmatrix()`, `marginplot()`, `matrixplot()`, `mosaicMiss()`, `parcoordMiss()`, `pbox()`, `scattJitt()`, `scattMiss()`, `scattmatrixMiss()`, `spineMiss()`

### Examples

```
data(sleep, package = "VIM")
x <- sleep[, -(8:10)]
x[, c(1, 2, 4, 6, 7)] <- log10(x[, c(1, 2, 4, 6, 7)])
pairsVIM(x)
```



---

parcoordMiss	<i>Parallel coordinate plot with information about missing/imputed values</i>
--------------	---

---

## Description

Parallel coordinate plot with adjustments for missing/imputed values. Missing values in the plotted variables may be represented by a point above the corresponding coordinate axis to prevent disconnected lines. In addition, observations with missing/imputed values in selected variables may be highlighted.

## Usage

```
parcoordMiss(
  x,
  delimiter = NULL,
  highlight = NULL,
  selection = c("any", "all"),
  plotvars = NULL,
  plotNA = TRUE,
  col = c("skyblue", "red", "skyblue4", "red4", "orange", "orange4"),
  alpha = NULL,
  lty = par("lty"),
  xlim = NULL,
  ylim = NULL,
  main = NULL,
  sub = NULL,
  xlab = NULL,
  ylab = NULL,
  labels = TRUE,
  xpd = NULL,
  interactive = TRUE,
  ...
)
```

## Arguments

x	a matrix or data.frame.
delimiter	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, x needs to have <code>colnames()</code> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see col).
highlight	a vector giving the variables to be used for highlighting. If NULL (the default), all variables are used for highlighting.

selection	the selection method for highlighting missing/imputed values in multiple highlight variables. Possible values are "any" (highlighting of missing/imputed values in <i>any</i> of the highlight variables) and "all" (highlighting of missing/imputed values in <i>all</i> of the highlight variables).
plotvars	a vector giving the variables to be plotted. If NULL (the default), all variables are plotted.
plotNA	a logical indicating whether missing values in the plot variables should be represented by a point above the corresponding coordinate axis to prevent disconnected lines.
col	if plotNA is TRUE, a vector of length six giving the colors to be used for observations with different combinations of observed and missing/imputed values in the plot variables and highlight variables (vectors of length one or two are recycled). Otherwise, a vector of length two giving the colors for non-highlighted and highlighted observations (if a single color is supplied, it is used for both).
alpha	a numeric value between 0 and 1 giving the level of transparency of the colors, or NULL. This can be used to prevent overplotting.
lty	if plotNA is TRUE, a vector of length four giving the line types to be used for observations with different combinations of observed and missing/imputed values in the plot variables and highlight variables (vectors of length one or two are recycled). Otherwise, a vector of length two giving the line types for non-highlighted and highlighted observations (if a single line type is supplied, it is used for both).
xlim, ylim	axis limits.
main, sub	main and sub title.
xlab, ylab	axis labels.
labels	either a logical indicating whether labels should be plotted below each coordinate axis, or a character vector giving the labels.
xpd	a logical indicating whether the lines should be allowed to go outside the plot region. If NULL, it defaults to TRUE unless axis limits are specified.
interactive	a logical indicating whether interactive features should be enabled (see 'Details').
...	for parcoordMiss, further graphical parameters to be passed down (see <a href="#">graphics::par()</a> ). For TKRparcoordMiss, further arguments to be passed to parcoordMiss.

## Details

In parallel coordinate plots, the variables are represented by parallel axes. Each observation of the scaled data is shown as a line. Observations with missing/imputed values in selected variables may thereby be highlighted. However, plotting variables with missing values results in disconnected lines, making it impossible to trace the respective observations across the graph. As a remedy, missing values may be represented by a point above the corresponding coordinate axis, which is separated from the main plot by a small gap and a horizontal line, as determined by plotNA. Connected lines can then be drawn for all observations. Nevertheless, a caveat of this display is that it may draw attention away from the main relationships between the variables.

If `interactive` is `TRUE`, it is possible switch between this display and the standard display without the separate level for missing values by clicking in the top margin of the plot. In addition, the variables to be used for highlighting can be selected interactively. Observations with missing/imputed values in any or in all of the selected variables are highlighted (as determined by `selection`). A variable can be added to the selection by clicking on a coordinate axis. If a variable is already selected, clicking on its coordinate axis removes it from the selection. Clicking anywhere outside the plot region (except the top margin, if missing/imputed values exist) quits the interactive session.

### Note

Some of the argument names and positions have changed with versions 1.3 and 1.4 due to extended functionality and for more consistency with other plot functions in VIM. For back compatibility, the arguments `colcomb` and `xaxlabels` can still be supplied to `...{}` and are handled correctly. Nevertheless, they are deprecated and no longer documented. Use `highlight` and `labels` instead.

### Author(s)

Andreas Alfons, Matthias Templ, modifications by Bernd Prantner

### References

Wegman, E. J. (1990) Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association* **85** (411), 664–675.

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

### See Also

[pbox\(\)](#)

Other plotting functions: [aggr\(\)](#), [barMiss\(\)](#), [histMiss\(\)](#), [marginmatrix\(\)](#), [marginplot\(\)](#), [matrixplot\(\)](#), [mosaicMiss\(\)](#), [pairsVIM\(\)](#), [pbox\(\)](#), [scattJitt\(\)](#), [scattMiss\(\)](#), [scattmatrixMiss\(\)](#), [spineMiss\(\)](#)

### Examples

```
data(chorizonDL, package = "VIM")
## for missing values
parcoordMiss(chorizonDL[,c(15,101:110)],
  plotvars=2:11, interactive = FALSE)
legend("top", col = c("skyblue", "red"), lwd = c(1,1),
  legend = c("observed in Bi", "missing in Bi"))

## for imputed values
parcoordMiss(kNN(chorizonDL[,c(15,101:110)]), delimiter = "_imp" ,
  plotvars=2:11, interactive = FALSE)
legend("top", col = c("skyblue", "orange"), lwd = c(1,1),
  legend = c("observed in Bi", "imputed in Bi"))
```

---

pbox

---

*Parallel boxplots with information about missing/imputed values*


---

## Description

Boxplot of one variable of interest plus information about missing/imputed values in other variables.

## Usage

```
pbox(
  x,
  delimiter = NULL,
  pos = 1,
  selection = c("none", "any", "all"),
  col = c("skyblue", "red", "red4", "orange", "orange4"),
  numbers = TRUE,
  cex.numbers = par("cex"),
  xlim = NULL,
  ylim = NULL,
  main = NULL,
  sub = NULL,
  xlab = NULL,
  ylab = NULL,
  axes = TRUE,
  frame.plot = axes,
  labels = axes,
  interactive = TRUE,
  ...
)
```

## Arguments

x	a vector, matrix or data.frame.
delimiter	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, x needs to have <code>colnames()</code> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see col).
pos	a numeric value giving the index of the variable of interest. Additional variables in x are used for grouping according to missingness/number of imputed missings.
selection	the selection method for grouping according to missingness/number of imputed missings in multiple additional variables. Possible values are "none" (grouping according to missingness/number of imputed missings in every other variable

	that contains missing/imputed values), "any" (grouping according to missingness/number of imputed missings in <i>any</i> of the additional variables) and "all" (grouping according to missingness/number of imputed missings in <i>all</i> of the additional variables).
col	a vector of length five giving the colors to be used in the plot. The first color is used for the boxplots of the available data, the second/fourth are used for missing/imputed data, respectively, and the third/fifth color for the frequencies of missing/imputed values in both variables (see 'Details'). If only one color is supplied, it is used for the boxplots for missing/imputed data, whereas the boxplots for the available data are transparent. Else if two colors are supplied, the second one is recycled.
numbers	a logical indicating whether the frequencies of missing/imputed values should be displayed (see 'Details').
cex.numbers	the character expansion factor to be used for the frequencies of the missing/imputed values.
xlim, ylim	axis limits.
main, sub	main and sub title.
xlab, ylab	axis labels.
axes	a logical indicating whether axes should be drawn on the plot.
frame.plot	a logical indicating whether a box should be drawn around the plot.
labels	either a logical indicating whether labels should be plotted below each box, or a character vector giving the labels.
interactive	a logical indicating whether variables can be switched interactively (see 'Details').
...	for pbox, further arguments and graphical parameters to be passed to <code>graphics::boxplot()</code> and other functions. For TKRpbox, further arguments to be passed to pbox.

## Details

This plot consists of several boxplots. First, a standard boxplot of the variable of interest is produced. Second, boxplots grouped by observed and missing/imputed values according to `selection` are produced for the variable of interest.

Additionally, the frequencies of the missing/imputed values can be represented by numbers. If so, the first line corresponds to the observed values of the variable of interest and their distribution in the different groups, the second line to the missing/imputed values.

If `interactive=TRUE`, clicking in the left margin of the plot results in switching to the previous variable and clicking in the right margin results in switching to the next variable. Clicking anywhere else on the graphics device quits the interactive session.

## Value

a list as returned by `graphics::boxplot()`.

**Note**

Some of the argument names and positions have changed with version 1.3 due to extended functionality and for more consistency with other plot functions in VIM. For back compatibility, the arguments names and `cex.text` can still be supplied to `...{}` and are handled correctly. Nevertheless, they are deprecated and no longer documented. Use `labels` and `cex.numbers` instead.

**Author(s)**

Andreas Alfons, Matthias Templ, modifications by Bernd Prantner

**References**

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

**See Also**

`parcoordMiss()`

Other plotting functions: `aggr()`, `barMiss()`, `histMiss()`, `marginmatrix()`, `marginplot()`, `matrixplot()`, `mosaicMiss()`, `pairsVIM()`, `parcoordMiss()`, `scattJitt()`, `scattMiss()`, `scattmatrixMiss()`, `spineMiss()`

**Examples**

```
data(chorizonDL, package = "VIM")
## for missing values
pbox(log(chorizonDL[, c(4,5,8,10,11,16:17,19,25,29,37,38,40)]))

## for imputed values
pbox(kNN(log(chorizonDL[, c(4,8,10,11,17,19,25,29,37,38,40)])),
      delimiter = "_imp")
```

---

prepare

*Transformation and standardization*

---

**Description**

This function is used by the VIM GUI for transformation and standardization of the data.

**Usage**

```
prepare(
  x,
  scaling = c("none", "classical", "MCD", "robust", "onestep"),
  transformation = c("none", "minus", "reciprocal", "logarithm", "exponential", "boxcox",
    "clr", "ilr", "alr"),
```

```

    alpha = NULL,
    powers = NULL,
    start = 0,
    alrVar
  )

```

### Arguments

<code>x</code>	a vector, matrix or data.frame.
<code>scaling</code>	the scaling to be applied to the data. Possible values are "none", "classical", "MCD", "robust" and "onestep".
<code>transformation</code>	the transformation of the data. Possible values are "none", "minus", "reciprocal", "logarithm", "exponential", "boxcox", "clr", "ilr" and "alr".
<code>alpha</code>	a numeric parameter controlling the size of the subset for the <i>MCD</i> (if <code>scaling="MCD"</code> ). See <a href="#">robustbase::covMcd()</a> .
<code>powers</code>	a numeric vector giving the powers to be used in the Box-Cox transformation (if <code>transformation="boxcox"</code> ). If <code>NULL</code> , the powers are calculated with function <a href="#">car::powerTransform()</a> .
<code>start</code>	a constant to be added prior to Box-Cox transformation (if <code>transformation="boxcox"</code> ).
<code>alrVar</code>	variable to be used as denominator in the additive logratio transformation (if <code>transformation="alr"</code> ).

### Details

#### Transformation:

"none": no transformation is used.

"logarithm": compute the the logarithm (to the base 10).

"boxcox": apply a Box-Cox transformation. Powers may be specified or calculated with the function [car::powerTransform\(\)](#).

#### Standardization:

"none": no standardization is used.

"classical": apply a z-Transformation on each variable by using function [scale\(\)](#).

"robust": apply a robustified z-Transformation by using median and MAD.

### Value

Transformed and standardized data.

### Author(s)

Matthias Templ, modifications by Andreas Alfons

### See Also

[scale\(\)](#), [car::powerTransform\(\)](#)

**Examples**

```
data(sleep, package = "VIM")
x <- sleep[, c("BodyWgt", "BrainWgt")]
prepare(x, scaling = "robust", transformation = "logarithm")
```

---

pulplignin

*Pulp lignin content*

---

**Description**

Pulp quality by lignin content remaining

**Format**

A data frame with 301 observations on the following 23 variables.

**Details**

Pulp quality is measured by the lignin content remaining in the pulp: the Kappa number. This data set is used to understand which variables in the process influence the Kappa number, and if it can be predicted accurately enough for an inferential sensor application. Variables with a number at the end have been lagged by that number of hours to line up the data.

**Source**

<https://openmv.net/info/kamyr-digester>

**References**

K. Walkush and R.R. Gustafson. Application of feedforward neural networks and partial least squares regression for modelling Kappa number in a continuous Kamyr digester", Pulp and Paper Canada, 95, 1994, p T7-T13.

**Examples**

```
data(pulplignin)
str(pulplignin)
aggr(pulplignin)
```



---

rangerImpute	<i>Random Forest Imputation</i>
--------------	---------------------------------

---

## Description

Impute missing values based on a random forest model using `ranger::ranger()`

## Usage

```
rangerImpute(
  formula,
  data,
  imp_var = TRUE,
  imp_suffix = "imp",
  ...,
  verbose = FALSE,
  median = FALSE
)
```

## Arguments

formula	model formula for the imputation
data	A <code>data.frame</code> containing the data
imp_var	TRUE/FALSE if a TRUE/FALSE variables for each imputed variable should be created show the imputation status
imp_suffix	suffix used for TF imputation variables
...	Arguments passed to <code>ranger::ranger()</code>
verbose	Show the number of observations used for training and evaluating the RF-Model. This parameter is also passed down to <code>ranger::ranger()</code> to show computation status.
median	Use the median (rather than the arithmetic mean) to average the values of individual trees for a more robust estimate.

## Value

the imputed data set.

## See Also

Other imputation methods: `hotdeck()`, `impPCA()`, `irmi()`, `kNN()`, `matchImpute()`, `medianSamp()`, `regressionImp()`, `sampleCat()`, `vimpute()`, `xgboostImpute()`

## Examples

```
data(sleep)
rangerImpute(Dream+NonD~BodyWgt+BrainWgt,data=sleep)
```

---

regressionImp	<i>Regression Imputation</i>
---------------	------------------------------

---

**Description**

Impute missing values based on a regression model.

**Usage**

```
regressionImp(
  formula,
  data,
  family = "AUTO",
  robust = FALSE,
  imp_var = TRUE,
  imp_suffix = "imp",
  mod_cat = FALSE
)
```

**Arguments**

formula	model formula to impute one variable
data	A data.frame containing the data
family	family argument for <code>glm()</code> . "AUTO" (the default) tries to choose automatically and is the only really tested option!!!
robust	TRUE/FALSE if robust regression should be used. See details.
imp_var	TRUE/FALSE if a TRUE/FALSE variables for each imputed variable should be created show the imputation status
imp_suffix	suffix used for TF imputation variables
mod_cat	TRUE/FALSE if TRUE for categorical variables the level with the highest prediction probability is selected, otherwise it is sampled according to the probabilities.

**Details**

`lm()` is used for family "normal" and `glm()` for all other families. If `robust = TRUE`, `lmrob()` is used for family "normal" and `glmrob()` for all other families.

**Value**

the imputed data set.

**Author(s)**

Alexander Kowarik

## References

A. Kowarik, M. Templ (2016) Imputation with R package VIM. *Journal of Statistical Software*, 74(7), 1-16.

## See Also

Other imputation methods: `hotdeck()`, `impPCA()`, `irmi()`, `kNN()`, `matchImpute()`, `medianSamp()`, `rangerImpute()`, `sampleCat()`, `vimpute()`, `xgboostImpute()`

## Examples

```
data(sleep)
sleepImp1 <- regressionImp(Dream+NonD~BodyWgt+BrainWgt,data=sleep)
sleepImp2 <- regressionImp(Sleep+Gest+Span+Dream+NonD~BodyWgt+BrainWgt,data=sleep)

data(testdata)
imp_testdata1 <- regressionImp(b1+b2~x1+x2,data=testdata$wna)
imp_testdata3 <- regressionImp(x1~x2,data=testdata$wna,robust=TRUE)
```

---

`rugNA`

*Rug representation of missing/imputed values*

---

## Description

Add a rug representation of missing/imputed values in only one of the variables to scatterplots.

## Usage

```
rugNA(
  x,
  y,
  ticksize = NULL,
  side = 1,
  col = "red",
  alpha = NULL,
  miss = NULL,
  lwd = 0.5,
  ...
)
```

## Arguments

<code>x, y</code>	numeric vectors.
<code>ticksize</code>	the length of the ticks. Positive lengths give inward ticks.
<code>side</code>	an integer giving the side of the plot to draw the rug representation.
<code>col</code>	the color to be used for the ticks.

alpha	the alpha value (between 0 and 1).
miss	a data.frame or matrix with two columns and logical values. If NULL, x and y are searched for missing values, otherwise, the first column of miss is used to determine the imputed values in x and the second one for the imputed values in y.
lwd	the line width to be used for the ticks.
...	further arguments to be passed to <code>graphics::Axis()</code> .

### Details

If side is 1 or 3, the rug representation consists of values available in x but missing/imputed in y. Else if side is 2 or 4, it consists of values available in y but missing/imputed in x.

### Author(s)

Andreas Alfons, modifications by Bernd Prantner

### Examples

```
data(tao, package = "VIM")
## for missing values
x <- tao[, "Air.Temp"]
y <- tao[, "Humidity"]
plot(x, y)
rugNA(x, y, side = 1)
rugNA(x, y, side = 2)

## for imputed values
x_imp <- kNN(tao[, c("Air.Temp", "Humidity")])
x <- x_imp[, "Air.Temp"]
y <- x_imp[, "Humidity"]
miss <- x_imp[, c("Air.Temp_imp", "Humidity_imp")]
plot(x, y)
rugNA(x, y, side = 1, col = "orange", miss = miss)
rugNA(x, y, side = 2, col = "orange", miss = miss)
```

---

sampleCat

*Random aggregation function for a factor variable*

---

### Description

The function sampleCat samples with probabilities corresponding to the occurrence of the level in the NNs.

### Usage

```
sampleCat(x, weights = NULL)
```

**Arguments**

x	factor vector
weights	numeric vector providing weights for the observations in x

**See Also**

Other imputation methods: [hotdeck\(\)](#), [impPCA\(\)](#), [irmi\(\)](#), [kNN\(\)](#), [matchImpute\(\)](#), [medianSamp\(\)](#), [rangerImpute\(\)](#), [regressionImp\(\)](#), [vimpure\(\)](#), [xgboostImpute\(\)](#)

---

SBS5242

*Synthetic subset of the Austrian structural business statistics data*

---

**Description**

Synthetic subset of the Austrian structural business statistics (SBS) data, namely NACE code 52.42 (retail sale of clothing).

**Details**

The Austrian SBS data set consists of more than 320.000 enterprises. Available raw (unedited) data set: 21669 observations in 90 variables, structured according NACE revision 1.1 with 3891 missing values.

We investigate 9 variables of NACE 52.42 (retail sale of clothing).

From these confidential raw data set a non-confidential, close-to-reality, synthetic data set was generated.

**Source**

<http://www.statistik.at>

**Examples**

```
data(SBS5242)
aggr(SBS5242)
```

---

scattJitt

*Bivariate jitter plot*


---

## Description

Create a bivariate jitter plot.

## Usage

```
scattJitt(
  x,
  delimiter = NULL,
  col = c("skyblue", "red", "red4", "orange", "orange4"),
  alpha = NULL,
  cex = par("cex"),
  col.line = "lightgrey",
  lty = "dashed",
  lwd = par("lwd"),
  numbers = TRUE,
  cex.numbers = par("cex"),
  main = NULL,
  sub = NULL,
  xlab = NULL,
  ylab = NULL,
  axes = TRUE,
  frame.plot = axes,
  labels = c("observed", "missing", "imputed"),
  ...
)
```

## Arguments

<code>x</code>	a <code>data.frame</code> or matrix with two columns.
<code>delimiter</code>	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, <code>x</code> needs to have <code>colnames()</code> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see <code>col</code> ).
<code>col</code>	a vector of length five giving the colors to be used in the plot. The first color will be used for complete observations, the second/fourth color for missing/imputed values in only one variable, and the third/fifth color for missing/imputed values in both variables. If only one color is supplied, it is used for all. Else if two colors are supplied, the second one is recycled.
<code>alpha</code>	a numeric value between 0 and 1 giving the level of transparency of the colors, or <code>NULL</code> . This can be used to prevent overplotting.

<code>cex</code>	the character expansion factor for the plot characters.
<code>col.line</code>	the color for the lines dividing the plot region.
<code>lty</code>	the line type for the lines dividing the plot region (see <a href="#">graphics::par()</a> ).
<code>lwd</code>	the line width for the lines dividing the plot region.
<code>numbers</code>	a logical indicating whether the frequencies of observed and missing/imputed values should be displayed (see ‘Details’).
<code>cex.numbers</code>	the character expansion factor to be used for the frequencies of the observed and missing/imputed values.
<code>main, sub</code>	main and sub title.
<code>xlab, ylab</code>	axis labels.
<code>axes</code>	a logical indicating whether both axes should be drawn on the plot. Use graphical parameter <code>"xaxt"</code> or <code>"yaxt"</code> to suppress just one of the axes.
<code>frame.plot</code>	a logical indicating whether a box should be drawn around the plot.
<code>labels</code>	a vector of length three giving the axis labels for the regions for observed, missing and imputed values (see ‘Details’).
<code>...</code>	further graphical parameters to be passed down (see <a href="#">graphics::par()</a> ).

### Details

The amount of observed and missing/imputed values is visualized by jittered points. Thereby the plot region is divided into up to four regions according to the existence of missing/imputed values in one or both variables. In addition, the amount of observed and missing/imputed values can be represented by a number.

### Note

Some of the argument names and positions have changed with version 1.3 due to extended functionality and for more consistency with other plot functions in VIM. For back compatibility, the argument `cex.text` can still be supplied to `...{}` and is handled correctly. Nevertheless, it is deprecated and no longer documented. Use `cex.numbers` instead.

### Author(s)

Matthias Templ, modifications by Andreas Alfons and Bernd Prantner

### References

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

### See Also

Other plotting functions: [aggr\(\)](#), [barMiss\(\)](#), [histMiss\(\)](#), [marginmatrix\(\)](#), [marginplot\(\)](#), [matrixplot\(\)](#), [mosaicMiss\(\)](#), [pairsVIM\(\)](#), [parcoordMiss\(\)](#), [pbox\(\)](#), [scattMiss\(\)](#), [scattmatrixMiss\(\)](#), [spineMiss\(\)](#)

## Examples

```
data(tao, package = "VIM")
## for missing values
scattJitt(tao[, c("Air.Temp", "Humidity")])

## for imputed values
scattJitt(kNN(tao[, c("Air.Temp", "Humidity")]), delimiter = "_imp")
```

---

scattmatrixMiss

*Scatterplot matrix with information about missing/imputed values*


---

## Description

Scatterplot matrix in which observations with missing/imputed values in certain variables are highlighted.

## Usage

```
scattmatrixMiss(
  x,
  delimiter = NULL,
  highlight = NULL,
  selection = c("any", "all"),
  plotvars = NULL,
  col = c("skyblue", "red", "orange"),
  alpha = NULL,
  pch = c(1, 3),
  lty = par("lty"),
  diagonal = c("density", "none"),
  interactive = TRUE,
  ...
)
```

## Arguments

x	a matrix or data.frame.
delimiter	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, x needs to have <code>colnames()</code> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see col).
highlight	a vector giving the variables to be used for highlighting. If NULL (the default), all variables are used for highlighting.



selection	the selection method for highlighting missing/imputed values in multiple highlight variables. Possible values are "any" (highlighting of missing/imputed values in <i>any</i> of the highlight variables) and "all" (highlighting of missing/imputed values in <i>all</i> of the highlight variables).
plotvars	a vector giving the variables to be plotted. If NULL (the default), all variables are plotted.
col	a vector of length three giving the colors to be used in the plot. The second/third color will be used for highlighting missing/imputed values.
alpha	a numeric value between 0 and 1 giving the level of transparency of the colors, or NULL. This can be used to prevent overplotting.
pch	a vector of length two giving the plot characters. The second plot character will be used for the highlighted observations.
lty	a vector of length two giving the line types for the density plots in the diagonal panels (if diagonal="density"). The second line type is used for the highlighted observations. If a single value is supplied, it is used for both non-highlighted and highlighted observations.
diagonal	a character string specifying the plot to be drawn in the diagonal panels. Possible values are "density" (density plots for non-highlighted and highlighted observations) and "none".
interactive	a logical indicating whether the variables to be used for highlighting can be selected interactively (see 'Details').
...	for scattmatrixMiss, further arguments and graphical parameters to be passed to <code>pairsVIM()</code> . <code>par("oma")</code> will be set appropriately unless supplied (see <code>graphics::par()</code> ). For <code>TKRscattmatrixMiss</code> , further arguments to be passed to <code>scattmatrixMiss</code> .

## Details

`scattmatrixMiss` uses `pairsVIM()` with a panel function that allows highlighting of missing/imputed values.

If `interactive=TRUE`, the variables to be used for highlighting can be selected interactively. Observations with missing/imputed values in any or in all of the selected variables are highlighted (as determined by `selection`). A variable can be added to the selection by clicking in a diagonal panel. If a variable is already selected, clicking on the corresponding diagonal panel removes it from the selection. Clicking anywhere else quits the interactive session.

The graphical parameter `oma` will be set unless supplied as an argument.

`TKRscattmatrixMiss` behaves like `scattmatrixMiss`, but uses `tkrplot` to embed the plot in a *Tcl/Tk* window. This is useful if the number of variables is large, because scrollbars allow to move from one part of the plot to another.

## Note

Some of the argument names and positions have changed with version 1.3 due to a re-implementation and for more consistency with other plot functions in VIM. For back compatibility, the argument `colcomb` can still be supplied to `...{}` and is handled correctly. Nevertheless, it is deprecated and no longer documented. Use `highlight` instead. The arguments `smooth`, `reg.line` and `legend.plot` are no longer used and ignored if supplied.

**Author(s)**

Andreas Alfons, Matthias Templ, modifications by Bernd Prantner

**References**

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

**See Also**

`pairsVIM()`, `marginmatrix()`

Other plotting functions: `aggr()`, `barMiss()`, `histMiss()`, `marginmatrix()`, `marginplot()`, `matrixplot()`, `mosaicMiss()`, `pairsVIM()`, `parcoordMiss()`, `pbox()`, `scattJitt()`, `scattMiss()`, `spineMiss()`

**Examples**

```
data(sleep, package = "VIM")
## for missing values
x <- sleep[, 1:5]
x[,c(1,2,4)] <- log10(x[,c(1,2,4)])
scattmatrixMiss(x, highlight = "Dream")

## for imputed values
x_imp <- kNN(sleep[, 1:5])
x_imp[,c(1,2,4)] <- log10(x_imp[,c(1,2,4)])
scattmatrixMiss(x_imp, delimiter = "_imp", highlight = "Dream")
```

---

scattMiss

---

*Scatterplot with information about missing/imputed values*


---

**Description**

In addition to a standard scatterplot, lines are plotted for the missing values in one variable. If there are imputed values, they will be highlighted.

**Usage**

```
scattMiss(
  x,
  delimiter = NULL,
  side = 1,
  col = c("skyblue", "red", "orange", "lightgrey"),
  alpha = NULL,
  lty = c("dashed", "dotted"),
  lwd = par("lwd"),
```

```

    quantiles = c(0.5, 0.975),
    inEllipse = FALSE,
    zeros = FALSE,
    xlim = NULL,
    ylim = NULL,
    main = NULL,
    sub = NULL,
    xlab = NULL,
    ylab = NULL,
    interactive = TRUE,
    ...
)

```

### Arguments

<code>x</code>	a matrix or data.frame with two columns.
<code>delimiter</code>	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, <code>x</code> needs to have <code>colnames()</code> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see <code>col</code> ).
<code>side</code>	if <code>side=1</code> , a rug representation and vertical lines are plotted for the missing/imputed values in the second variable; if <code>side=2</code> , a rug representation and horizontal lines for the missing/imputed values in the first variable.
<code>col</code>	a vector of length four giving the colors to be used in the plot. The first color is used for the scatterplot, the second/third color for the rug representation for missing/imputed values. The second color is also used for the lines for missing values. Imputed values will be highlighted with the third color, and the fourth color is used for the ellipses (see 'Details'). If only one color is supplied, it is used for the scatterplot, the rug representation and the lines, whereas the default color is used for the ellipses. Else if a vector of length two is supplied, the default color is used for the ellipses as well.
<code>alpha</code>	a numeric value between 0 and 1 giving the level of transparency of the colors, or NULL. This can be used to prevent overplotting.
<code>lty</code>	a vector of length two giving the line types for the lines and ellipses. If a single value is supplied, it will be used for both.
<code>lwd</code>	a vector of length two giving the line widths for the lines and ellipses. If a single value is supplied, it will be used for both.
<code>quantiles</code>	a vector giving the quantiles of the chi-square distribution to be used for the tolerance ellipses, or NULL to suppress plotting ellipses (see 'Details').
<code>inEllipse</code>	plot lines only inside the largest ellipse. Ignored if <code>quantiles</code> is NULL or if there are imputed values.
<code>zeros</code>	a logical vector of length two indicating whether the variables are semi-continuous, i.e., contain a considerable amount of zeros. If TRUE, only the non-zero observations are used for computing the tolerance ellipses. If a single logical is supplied, it is recycled. Ignored if <code>quantiles</code> is NULL.

xlim, ylim	axis limits.
main, sub	main and sub title.
xlab, ylab	axis labels.
interactive	a logical indicating whether the side argument can be changed interactively (see ‘Details’).
...	further graphical parameters to be passed down (see <code>graphics::par()</code> ).

### Details

Information about missing values in one variable is included as vertical or horizontal lines, as determined by the `side` argument. The lines are thereby drawn at the observed x- or y-value. In case of imputed values, they will additionally be highlighted in the scatterplot. Supplementary, percentage coverage ellipses can be drawn to give a clue about the shape of the bivariate data distribution.

If `interactive` is TRUE, clicking in the bottom margin redraws the plot with information about missing/imputed values in the first variable and clicking in the left margin redraws the plot with information about missing/imputed values in the second variable. Clicking anywhere else in the plot quits the interactive session.

### Note

The argument `zeros` has been introduced in version 1.4. As a result, some of the argument positions have changed.

### Author(s)

Andreas Alfons, modifications by Bernd Prantner

### References

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

### See Also

`marginplot()`

Other plotting functions: `aggr()`, `barMiss()`, `histMiss()`, `marginmatrix()`, `marginplot()`, `matrixplot()`, `mosaicMiss()`, `pairsVIM()`, `parcoordMiss()`, `pbox()`, `scattJitt()`, `scattmatrixMiss()`, `spineMiss()`

### Examples

```
data(tao, package = "VIM")
## for missing values
scattMiss(tao[,c("Air.Temp", "Humidity")])

## for imputed values
scattMiss(kNN(tao[,c("Air.Temp", "Humidity")]), delimiter = "_imp")
```

---

sleep	<i>Mammal sleep data</i>
-------	--------------------------

---

**Description**

Sleep data with missing values.

**Format**

A data frame with 62 observations on the following 10 variables.

**BodyWgt** a numeric vector

**BrainWgt** a numeric vector

**NonD** a numeric vector

**Dream** a numeric vector

**Sleep** a numeric vector

**Span** a numeric vector

**Gest** a numeric vector

**Pred** a numeric vector

**Exp** a numeric vector

**Danger** a numeric vector

**Source**

Allison, T. and Chichetti, D. (1976) Sleep in mammals: ecological and constitutional correlates. *Science* **194** (4266), 732–734.

The data set was imported from GGobi.

**Examples**

```
data(sleep, package = "VIM")
summary(sleep)
aggr(sleep)
```

---

spineMiss

*Spineplot with information about missing/imputed values*


---

## Description

Spineplot or spinogram with highlighting of missing/imputed values in other variables by splitting each cell into two parts. Additionally, information about missing/imputed values in the variable of interest is shown on the right hand side.

## Usage

```
spineMiss(
  x,
  delimiter = NULL,
  pos = 1,
  selection = c("any", "all"),
  breaks = "Sturges",
  right = TRUE,
  col = c("skyblue", "red", "skyblue4", "red4", "orange", "orange4"),
  border = NULL,
  main = NULL,
  sub = NULL,
  xlab = NULL,
  ylab = NULL,
  axes = TRUE,
  labels = axes,
  only.miss = TRUE,
  miss.labels = axes,
  interactive = TRUE,
  ...
)
```

## Arguments

<code>x</code>	a vector, matrix or <code>data.frame</code> .
<code>delimiter</code>	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, <code>x</code> needs to have <code>colnames()</code> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see <code>col</code> ).
<code>pos</code>	a numeric value giving the index of the variable of interest. Additional variables in <code>x</code> are used for highlighting.
<code>selection</code>	the selection method for highlighting missing/imputed values in multiple additional variables. Possible values are "any" (highlighting of missing/imputed values in <i>any</i> of the additional variables) and "all" (highlighting of missing/imputed values in <i>all</i> of the additional variables).

<code>breaks</code>	if the variable of interest is numeric, <code>breaks</code> controls the breakpoints (see <code>graphics::hist()</code> for possible values).
<code>right</code>	logical; if TRUE and the variable of interest is numeric, the spinogram cells are right-closed (left-open) intervals.
<code>col</code>	a vector of length six giving the colors to be used. If only one color is supplied, the bars are transparent and the supplied color is used for highlighting missing/imputed values. Else if two colors are supplied, they are recycled.
<code>border</code>	the color to be used for the border of the cells. Use <code>border=NA</code> to omit borders.
<code>main, sub</code>	main and sub title.
<code>xlab, ylab</code>	axis labels.
<code>axes</code>	a logical indicating whether axes should be drawn on the plot.
<code>labels</code>	if the variable of interest is categorical, either a logical indicating whether labels should be plotted below each cell, or a character vector giving the labels. This is ignored if the variable of interest is numeric.
<code>only.miss</code>	logical; if TRUE, the missing/imputed values in the variable of interest are also visualized by a cell in the spineplot or spinogram. Otherwise, a small spineplot is drawn on the right hand side (see ‘Details’).
<code>miss.labels</code>	either a logical indicating whether label(s) should be plotted below the cell(s) on the right hand side, or a character string or vector giving the label(s) (see ‘Details’).
<code>interactive</code>	a logical indicating whether the variables can be switched interactively (see ‘Details’).
<code>...</code>	further graphical parameters to be passed to <code>graphics::title()</code> and <code>graphics::axis()</code> .

## Details

A spineplot is created if the variable of interest is categorical and a spinogram if it is numerical. The horizontal axis is scaled according to relative frequencies of the categories/classes. If more than one variable is supplied, the cells are split according to missingness/number of imputed values in the additional variables. Thus the proportion of highlighted observations in each category/class is displayed on the vertical axis. Since the height of each cell corresponds to the proportion of highlighted observations, it is now possible to compare the proportions of missing/imputed values among the different categories/classes.

If `only.miss=TRUE`, the missing/imputed values in the variable of interest are also visualized by a cell in the spine plot or spinogram. If additional variables are supplied, this cell is again split into two parts according to missingness/number of imputed values in the additional variables.

Otherwise, a small spineplot that visualizes missing/imputed values in the variable of interest is drawn on the right hand side. The first cell corresponds to observed values and the second cell to missing/imputed values. Each of the two cells is again split into two parts according to missingness/number of imputed values in the additional variables. Note that this display does not make sense if only one variable is supplied, therefore `only.miss` is ignored in that case.

If `interactive=TRUE`, clicking in the left margin of the plot results in switching to the previous variable and clicking in the right margin results in switching to the next variable. Clicking anywhere else on the graphics device quits the interactive session.

**Value**

a table containing the frequencies corresponding to the cells.

**Note**

Some of the argument names and positions have changed with version 1.3 due to extended functionality and for more consistency with other plot functions in VIM. For back compatibility, the arguments `xaxlabels` and `missaxlabels` can still be supplied to `...{}` and are handled correctly. Nevertheless, they are deprecated and no longer documented. Use `labels` and `miss.labels` instead.

The code is based on the function `graphics::spineplot()` by Achim Zeileis.

**Author(s)**

Andreas Alfons, Matthias Templ, modifications by Bernd Prantner

**References**

M. Templ, A. Alfons, P. Filzmoser (2012) Exploring incomplete data using visualization tools. *Journal of Advances in Data Analysis and Classification*, Online first. DOI: 10.1007/s11634-011-0102-y.

**See Also**

`histMiss()`, `barMiss()`, `mosaicMiss()`

Other plotting functions: `aggr()`, `barMiss()`, `histMiss()`, `marginmatrix()`, `marginplot()`, `matrixplot()`, `mosaicMiss()`, `pairsVIM()`, `parcoordMiss()`, `pbox()`, `scattJitt()`, `scattMiss()`, `scattmatrixMiss()`

**Examples**

```
data(tao, package = "VIM")
data(sleep, package = "VIM")
## for missing values
spineMiss(tao[, c("Air.Temp", "Humidity")])
spineMiss(sleep[, c("Exp", "Sleep")])

## for imputed values
spineMiss(kNN(tao[, c("Air.Temp", "Humidity")]), delimiter = "_imp")
spineMiss(kNN(sleep[, c("Exp", "Sleep")]), delimiter = "_imp")
```



---

tableMiss	<i>create table with highlighted missings/imputations</i>
-----------	---

---

### Description

Create a reactable table that highlights missing values and imputed values with the same colors as `histMiss()`

### Usage

```
tableMiss(x, delimiter = "_imp")
```

### Arguments

<code>x</code>	a vector, matrix or data.frame.
<code>delimiter</code>	a character-vector to distinguish between variables and imputation-indices for imputed variables (therefore, <code>x</code> needs to have <code>colnames()</code> ). If given, it is used to determine the corresponding imputation-index for any imputed variable (a logical-vector indicating which values of the variable have been imputed). If such imputation-indices are found, they are used for highlighting and the colors are adjusted according to the given colors for imputed variables (see <code>col</code> ).

### Examples

```
data(tao)
x_IMPUTED <- kNN(tao[, c("Air.Temp", "Humidity")])
tableMiss(x_IMPUTED[105:114, ])
x_IMPUTED[106, 2] <- NA
x_IMPUTED[105, 1] <- NA
x_IMPUTED[107, "Humidity_imp"] <- TRUE
tableMiss(x_IMPUTED[105:114, ])
```

---

tao	<i>Tropical Atmosphere Ocean (TAO) project data</i>
-----	---

---

### Description

A small subsample of the Tropical Atmosphere Ocean (TAO) project data, derived from the GGOBI project.

**Format**

A data frame with 736 observations on the following 8 variables.

**Year** a numeric vector

**Latitude** a numeric vector

**Longitude** a numeric vector

**Sea.Surface.Temp** a numeric vector

**Air.Temp** a numeric vector

**Humidity** a numeric vector

**UWind** zonal wind, i.e. latitude-parallel wind

**VWind** meridional wind, i.e. longitude-parallel wind

**Details**

All cases recorded for five locations and two time periods.

**Source**

<http://www.pmel.noaa.gov/tao/>

**Examples**

```
data(tao, package = "VIM")
summary(tao)
aggr(tao)
```

---

testdata

*Simulated data set for testing purpose*

---

**Description**

2 numeric, 2 binary, 2 nominal and 2 mixed (semi-continuous) variables

**Format**

The format is: List of 4

- \$wna : a data.frame with 500 obs. of 8 variables:
  - x1: numeric 10.87 9.53 7.83 8.53 8.67 ...
  - x2: numeric 10.9 9.32 7.68 8.2 8.41 ... ..
  - c1: Factor w/ 4 levels "a","b","c","d": 3 2 2 1 2 2 1 3 3 2 ...
  - c2: Factor w/ 4 levels "a","b","c","d": 2 3 2 2 2 2 2 4 2 2 ...
  - b1: Factor w/ 2 levels "0","1": 2 2 1 2 1 2 1 2 1 1 ...
  - b2: Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 2 2 2 ...

- m1: numeric 0 8.29 9.08 0 0 ...
- m2: numeric 10.66 9.39 7.8 8.11 7.33 ...
- \$wona : a 'data.frame' with 500 obs. of 8 variables:
  - x1: numeric 10.87 9.53 7.83 8.53 8.67 ...
  - x2: numeric 10.9 9.32 7.68 8.2 8.41 ...
  - c1: Factor w/ 4 levels "a","b","c","d": 3 2 2 1 2 2 1 3 3 2 ...
  - c2: Factor w/ 4 levels "a","b","c","d": 2 3 2 2 2 2 2 4 2 2 ...
  - b1: Factor w/ 2 levels "0","1": 2 2 1 2 1 2 1 2 1 1 ...
  - b2: Factor w/ 2 levels "0","1": 2 2 1 1 1 1 1 2 2 2 ...
  - m1: numeric 0 8.29 9.08 0 0 ...
  - m2: numeric 10.66 9.39 7.8 8.11 7.33 ...
- \$mixed: c("m1", "m2")
- \$outlierInd: 'NULL'

## Examples

```
data(testdata)
```

---

toydataMiss

*Simulated toy data set for examples*

---

## Description

A 2-dimensional data set with additional information.

## Format

data frame with 100 observations and 12 variables. The first two variables represent the fully observed data.

## Examples

```
data(toydataMiss)
```

---

vimpute	<i>Impute missing values with preferred Model, sequentially, with hyperparameter tuning and with PMM (if wanted) Need of 'helper_vimpute' script</i>
---------	--

---

## Description

Impute missing values with preferred Model, sequentially, with hyperparameter tuning and with PMM (if wanted) Need of 'helper\_vimpute' script

## Usage

```
vimpute(
  data,
  considered_variables = names(data),
  method = setNames(as.list(rep("ranger", length(considered_variables))),
    considered_variables),
  pmm = setNames(as.list(rep(TRUE, length(considered_variables))), considered_variables),
  formula = FALSE,
  sequential = TRUE,
  nseq = 10,
  eps = 0.005,
  imp_var = TRUE,
  pred_history = FALSE,
  tune = FALSE,
  verbose = FALSE
)
```

## Arguments

data	• Dataset with missing values. Can be provided as a data.table or data.frame.
considered_variables	• A character vector of variable names to be either imputed or used as predictors, excluding irrelevant columns from the imputation process.
method	• A named list specifying the imputation method for each variable:
pmm	• TRUE/FALSE indicating whether predictive mean matching is used. Provide as a list for each variable.
formula	• If not all variables are used as predictors, or if transformations or interactions are required (applies to all X, for Y only transformations are possible). Only applicable for the methods "robust" and "regularized". Provide as a list for each variable that requires specific conditions.
sequential	• If TRUE, all variables are imputed sequentially.
nseq	• Maximum number of iterations (if sequential is TRUE).
eps	• Threshold for convergence.
imp_var	• If TRUE, the imputed values are stored.

- `pred_history` • If TRUE, all predicted values across all iterations are stored.
- `tune` • Tunes hyperparameters halfway through iterations, TRUE or FALSE.
- `verbose` • If TRUE additional debugging output is provided

### Value

imputed data set or `c(imputed data set, prediction history)`

### See Also

Other imputation methods: `hotdeck()`, `impPCA()`, `irmi()`, `kNN()`, `matchImpute()`, `medianSamp()`, `rangerImpute()`, `regressionImp()`, `sampleCat()`, `xgboostImpute()`

### Examples

```
## Not run:
x <- vimpute(data = sleep, sequential = FALSE)
y <- vimpute(data = sleep, sequential = TRUE, nseq = 3)
z <- vimpute(data = sleep, considered_variables =
  c("Sleep", "Dream", "Span", "BodyWgt"), sequential = FALSE)

## End(Not run)
```

---

wine

*Wine tasting and price*

---

### Description

Wine reviews from France, Switzerland, Austria and Germany.

### Format

A data frame with 9627 observations on the following 9 variables.

**country** country of origin

**points** the number of points WineEnthusiast rated the wine on a scale of 1-100 (though they say they only post reviews for wines that score  $\geq 80$ )

**price** the cost for a bottle of the wine

**province** the province or state that the wine is from

**taster\_name** name of the person who tasted and reviewed the wine

**taster\_twitter\_handle** Twitter handle for the person who tasted and reviewed the wine

**variety** the type of grapes used to make the wine (ie pinot noir)

**winery** the winery that made the wine

**variety\_main** broader category as variety

## Details

The data was scraped from WineEnthusiast during the week of Nov 22th, 2017. The code for the scraper can be found at <https://github.com/zackthoutt/wine-deep-learning> This data set is slightly modified, i.e. only four countries are selected and broader categories on the variety have been added.

## Source

<https://www.kaggle.com/zynicide/wine-reviews>

## Examples

```
data(wine)
str(wine)
aggr(wine)
```

---

xgboostImpute

*Xgboost Imputation*

---

## Description

Impute missing values based on a random forest model using `xgboost::xgboost()`

## Usage

```
xgboostImpute(
  formula,
  data,
  imp_var = TRUE,
  imp_suffix = "imp",
  verbose = FALSE,
  nrounds = 100,
  objective = NULL,
  ...
)
```

## Arguments

formula	model formula for the imputation
data	A <code>data.frame</code> containing the data
imp_var	TRUE/FALSE if a TRUE/FALSE variables for each imputed variable should be created show the imputation status
imp_suffix	suffix used for TF imputation variables
verbose	Show the number of observations used for training and evaluating the RF-Model. This parameter is also passed down to <code>xgboost::xgboost()</code> to show computation status.

nrounds	max number of boosting iterations, argument passed to <code>xgboost::xgboost()</code>
objective	objective for xgboost, argument passed to <code>xgboost::xgboost()</code>
...	Arguments passed to <code>xgboost::xgboost()</code>

**Value**

the imputed data set.

**See Also**

Other imputation methods: `hotdeck()`, `impPCA()`, `irmi()`, `kNN()`, `matchImpute()`, `medianSamp()`, `rangerImpute()`, `regressionImp()`, `sampleCat()`, `vimpute()`

**Examples**

```
data(sleep)
xgboostImpute(Dream~BodyWgt+BrainWgt,data=sleep)
xgboostImpute(Dream+NonD~BodyWgt+BrainWgt,data=sleep)
xgboostImpute(Dream+NonD+Gest~BodyWgt+BrainWgt,data=sleep)

sleepx <- sleep
sleepx$Pred <- as.factor(LETTERS[sleepx$Pred])
sleepx$Pred[1] <- NA
xgboostImpute(Pred~BodyWgt+BrainWgt,data=sleepx)
```

# Index

## \* color

- alphablend, 8
- colSequence, 23
- rugNA, 75

## \* datasets

- Animals\_na, 8
- bcancer, 12
- brittleness, 14
- chorizonDL, 14
- colic, 18
- collisions, 20
- diabetes, 25
- food, 28
- kola.background, 49
- pulplignin, 72
- SBS5242, 77
- sleep, 85
- tao, 89
- testdata, 90
- toydataMiss, 91
- wine, 93

## \* hplot

- aggr, 3
- barMiss, 9
- bgmap, 13
- colormapMiss, 20
- growdotMiss, 31
- histMiss, 34
- mapMiss, 50
- marginmatrix, 51
- marginplot, 53
- matrixplot, 57
- mosaicMiss, 61
- pairsVIM, 63
- parcoordMiss, 65
- pbox, 68
- scattJitt, 78
- scattmatrixMiss, 80
- scattMiss, 82

- spineMiss, 86

## \* imputation methods

- hotdeck, 36
- impPCA, 38
- irmi, 44
- kNN, 47
- matchImpute, 56
- medianSamp, 60
- rangerImpute, 73
- regressionImp, 74
- sampleCat, 76
- vimpute, 92
- xgboostImpute, 94

## \* manip

- evaluation, 27
- gapMiss, 29
- hotdeck, 36
- impPCA, 38
- initialise, 43
- irmi, 44
- kNN, 47
- matchImpute, 56
- prepare, 70
- regressionImp, 74

## \* plotting functions

- aggr, 3
- barMiss, 9
- histMiss, 34
- marginmatrix, 51
- marginplot, 53
- matrixplot, 57
- mosaicMiss, 61
- pairsVIM, 63
- parcoordMiss, 65
- pbox, 68
- scattJitt, 78
- scattmatrixMiss, 80
- scattMiss, 82
- spineMiss, 86



- \* **print**
  - aggr, 3
- \* **utilities**
  - countInf, 25
- aggr, 3, 11, 36, 53, 55, 59, 62, 64, 67, 70, 79, 82, 84, 88
- aggr(), 7
- alphablend, 8
- Animals\_na, 8
- barMiss, 7, 9, 36, 53, 55, 59, 62, 64, 67, 70, 79, 82, 84, 88
- barMiss(), 36, 88
- bcancer, 12
- bgmap, 13
- bgmap(), 32, 33, 50, 51
- brittleness, 14
- bubbleMiss (growdotMiss), 31
- bubbleMiss(), 51
- car::powerTransform(), 71
- chorizonDL, 14
- colic, 18
- collisions, 20
- colnames(), 4, 10, 32, 34, 50, 52, 54, 58, 61, 63, 65, 68, 78, 80, 83, 86, 89
- colormapMiss, 20
- colormapMiss(), 33, 51
- colormapMissLegend (colormapMiss), 20
- colorspace::hex(), 22–24, 58
- colorspace::polarLUV(), 21, 24, 58
- colorspace::RGB(), 21, 24, 58
- colorspace::sequential\_hcl(), 24
- colSequence, 23
- colSequence(), 23
- colSequenceHCL (colSequence), 23
- colSequenceRGB (colSequence), 23
- countInf, 25
- countNA (countInf), 25
- diabetes, 25
- evaluation, 27
- food, 28
- format(), 32
- gam, 41
- gapMiss, 29
- glm(), 74
- glmrob(), 74
- gowerD, 30
- graphics::Axis(), 76
- graphics::axis(), 10, 35, 58, 87
- graphics::boxplot(), 69
- graphics::hist(), 87
- graphics::lines(), 13
- graphics::pairs(), 64
- graphics::par(), 52, 55, 63, 66, 79, 81, 84
- graphics::plot.window(), 58
- graphics::points(), 51
- graphics::spineplot(), 88
- graphics::title(), 10, 35, 58, 87
- growdotMiss, 31
- growdotMiss(), 13, 23
- hist(), 35
- histMiss, 7, 11, 34, 53, 55, 59, 62, 64, 67, 70, 79, 82, 84, 88
- histMiss(), 11, 88, 89
- hotdeck, 36, 39, 46, 49, 57, 60, 73, 75, 77, 93, 95
- hotdeck(), 57
- iimagMiss (matrixplot), 57
- impPCA, 38, 38, 46, 49, 57, 60, 73, 75, 77, 93, 95
- imputeRobust, 40
- imputeRobustChain, 42
- initialise, 41, 43, 43
- irmi, 38, 39, 44, 49, 57, 60, 73, 75, 77, 93, 95
- kNN, 38, 39, 46, 47, 57, 60, 73, 75, 77, 93, 95
- kola.background, 49
- lm(), 74
- lmrob, 41
- lmrob(), 74
- mapMiss, 50
- mapMiss(), 13, 23, 33
- marginmatrix, 7, 11, 36, 51, 55, 59, 62, 64, 67, 70, 79, 82, 84, 88
- marginmatrix(), 64, 82
- marginplot, 7, 11, 36, 53, 53, 59, 62, 64, 67, 70, 79, 82, 84, 88
- marginplot(), 52, 53, 84
- matchImpute, 38, 39, 46, 49, 56, 60, 73, 75, 77, 93, 95

matrixplot, [7](#), [11](#), [36](#), [53](#), [55](#), [57](#), [62](#), [64](#), [67](#),  
[70](#), [79](#), [82](#), [84](#), [88](#)  
 maxCat, [60](#)  
 medianSamp, [38](#), [39](#), [46](#), [49](#), [57](#), [60](#), [73](#), [75](#), [77](#),  
[93](#), [95](#)  
 mosaicMiss, [7](#), [11](#), [36](#), [53](#), [55](#), [59](#), [61](#), [64](#), [67](#),  
[70](#), [79](#), [82](#), [84](#), [88](#)  
 mosaicMiss(), [88](#)  
 msecor (evaluation), [27](#)  
 msecov (evaluation), [27](#)  
  
 nrmse (evaluation), [27](#)  
  
 outCoDa, [43](#)  
  
 pairsVIM, [7](#), [11](#), [36](#), [53](#), [55](#), [59](#), [62](#), [63](#), [67](#), [70](#),  
[79](#), [82](#), [84](#), [88](#)  
 pairsVIM(), [52](#), [53](#), [81](#), [82](#)  
 parcoordMiss, [7](#), [11](#), [36](#), [53](#), [55](#), [59](#), [62](#), [64](#),  
[65](#), [70](#), [79](#), [82](#), [84](#), [88](#)  
 parcoordMiss(), [70](#)  
 pbox, [7](#), [11](#), [36](#), [53](#), [55](#), [59](#), [62](#), [64](#), [67](#), [68](#), [79](#),  
[82](#), [84](#), [88](#)  
 pbox(), [67](#)  
 pdist, [41](#)  
 pfc (evaluation), [27](#)  
 plot.aggr (aggr), [3](#)  
 prepare, [70](#)  
 print.aggr (aggr), [3](#)  
 print.aggr(), [7](#)  
 print.default(), [5](#)  
 print.summary.aggr (aggr), [3](#)  
 print.summary.aggr(), [7](#)  
 pulplignin, [72](#)  
  
 ranger::ranger(), [48](#), [73](#)  
 rangerImpute, [38](#), [39](#), [46](#), [49](#), [57](#), [60](#), [73](#), [75](#),  
[77](#), [93](#), [95](#)  
 regressionImp, [38](#), [39](#), [46](#), [49](#), [57](#), [60](#), [73](#), [74](#),  
[77](#), [93](#), [95](#)  
 robustbase::covMcd(), [71](#)  
 rugNA, [75](#)  
  
 sampleCat, [38](#), [39](#), [46](#), [49](#), [57](#), [60](#), [73](#), [75](#), [76](#),  
[93](#), [95](#)  
 SBS5242, [77](#)  
 scale(), [71](#)  
 scattJitt, [7](#), [11](#), [36](#), [53](#), [55](#), [59](#), [62](#), [64](#), [67](#),  
[70](#), [78](#), [82](#), [84](#), [88](#)  
  
 scattmatrixMiss, [7](#), [11](#), [36](#), [53](#), [55](#), [59](#), [62](#),  
[64](#), [67](#), [70](#), [79](#), [80](#), [84](#), [88](#)  
 scattmatrixMiss(), [53](#), [64](#)  
 scattMiss, [7](#), [11](#), [36](#), [53](#), [55](#), [59](#), [62](#), [64](#), [67](#),  
[70](#), [79](#), [82](#), [82](#), [88](#)  
 scattMiss(), [55](#)  
 sleep, [85](#)  
 spineMiss, [7](#), [11](#), [36](#), [53](#), [55](#), [59](#), [62](#), [64](#), [67](#),  
[70](#), [79](#), [82](#), [84](#), [86](#)  
 spineMiss(), [11](#), [36](#), [62](#)  
 summary.aggr (aggr), [3](#)  
 summary.aggr(), [7](#)  
  
 tableMiss, [89](#)  
 tao, [89](#)  
 testdata, [90](#)  
 TKRmatrixplot (matrixplot), [57](#)  
 toydataMiss, [91](#)  
  
 vcd::labeling\_border(), [61](#)  
 vcd::mosaic(), [61](#), [62](#)  
 vimpure, [38](#), [39](#), [46](#), [49](#), [57](#), [60](#), [73](#), [75](#), [77](#), [92](#),  
[95](#)  
  
 wine, [93](#)  
  
 xgboost::xgboost(), [94](#), [95](#)  
 xgboostImpute, [38](#), [39](#), [46](#), [49](#), [57](#), [60](#), [73](#), [75](#),  
[77](#), [93](#), [94](#)