How to generate missing values?

Teresa Alves de Sousa, Imke Mayer

26 janvier 2021

Contents

Missing values occur in many domains and most datasets contain missing values (due to non-responses, lost records, machine failures, dataset fusions, etc.). These missing values have to be considered before or during analyses of these datasets.

Now, if you have a method that deals with missing values, for instance imputation or estimation with missing values, how can you assess the performance of your method on a given dataset? If the data already contains missing values, than this does not help you since you generally do not have a ground truth for these missing values. So you will have to simulate missing values, i.e. you remove values – which you therefore know to be the ground truth – to generate missing values.

The mechanisms generating missing values can be various but usually they are classified into three main categories defined by (Rubin 1976): *missing completely at random* (MCAR), *missing at random* (MAR) and *missing not at random* (MNAR). The first two are also qualified as *ignorable* missing values mechanisms, for instance in likelihood-based approaches to handle missing values, whereas the MNAR mechanism generates *nonignorable* missing values. In the following we will briefly introduce each mechanism (with the definitions used widely in the literature) and propose ways of simulations missing values under these three mechanism assumptions. For more precise definitions we refer to references in the bibliography on the R-miss-tastic website.

1 Introduction

1.1 Notations

Let's denote by $\mathbf{X} \in \mathcal{X}_\infty \times \cdots \times \mathcal{X}_{\surd}$ the complete observations. We assume that $\mathbf{X}$ is a concatenation of p columns $X_j \in \mathcal{X}_|$, $j \in \{1, \dots, p\}$, where $dim(\mathcal{X}_|) = n$ for all j.

The data can be composed of quantitative and/or qualitative values, hence $\mathcal{X}_|$ can be $\mathbb{R}^n$, $\mathbb{Z}^n$ or more generally $\mathcal{S}^n$ for any discrete set S.

Missing values are indicated as `NA` (not available) and we define an indicator matrix $\mathbf{R} \in \{0,1\}^{n \times p}$ such that $R_{ij} = 1$ if X_{ij} is observed and $R_{ij} = 0$ otherwise. We call this matrix $\mathbf{R}$ the response (or missingness) pattern of the observations $\mathbf{X}$. According to this pattern, we can partition the observations $\mathbf{X}$ into observed and missing: $\mathbf{X} = (\mathbf{X}^{obs}, \mathbf{X}^{mis})$.

1.2 Definition of the mechanisms

In order to define the different missing values mechanisms, both $\mathbf{X}$ and $\mathbf{R}$ are modeled as random variables with probability distributions $\mathbb{P}_X$ and $\mathbb{P}_R$ respectively. We parametrize the missingness distribution $\mathbb{P}_R$ by a parameter ϕ.

1.2.1 MCAR

The observations are said to be Missing Completely At Random (MCAR) if the probability that an observation is missing is independent of the variables and observations: the probability that an observation is missing does not depend on $(\mathbf{X}^{obs}, \mathbf{X}^{mis})$. Formally this is:

$$\mathbb{P}_R(R \,|\, X^{obs}, X^{mis}; \phi) = \mathbb{P}_R(R) \qquad \forall\, \phi.$$

1.2.2 MAR

The observations are said to be Missing At Random (MAR) if the probability that an observation is missing only depends on the observed data $\mathbf{X}^{obs}$. Formally,

$$\mathbb{P}_R(R \,|\, X^{obs}, X^{mis}; \phi) = \mathbb{P}_R(R \,|\, X^{obs}; \phi) \qquad \forall\, \phi,\, \forall\, X^{mis}.$$

1.2.3 MNAR

The observations are said to be Missing Not At Random (MNAR) in all other cases, i.e. the missingness depends on the missing values and potentially also on the observed values.

2 Use of `produce_NA` with default settings

With the main function `produce_NA` it is possible to generate missing values for quantitative, categorical or mixed data, provided that it is available in form of a `data.frame` or `matrix`.

Missing values can be generated following one or more of the three main missing values mechanisms (see below for details).

If the data is already incomplete, it is possible to add a specific amount of additional missing values, in the already incomplete features or other complete features.

Important: Currently there is no option available for the mains function `produce_NA` to specify that every observation must contain at least one value after amputation. Hence, in the data.frame output by `produce_NA` there might be empty observations.

Except for the MCAR mechanism, our function `produce_NA` internally calls the `ampute` function of the `mice` R-package. See (Schouten, Lugtig, and Vink 2018) for a detailed description of this latter function. A vignette for the function `ampute` is available here.

We generate a small example of observations $\mathbf{X}$:

```r
suppressPackageStartupMessages(require(MASS))
suppressPackageStartupMessages(require(norm))
suppressPackageStartupMessages(require(VIM))
suppressPackageStartupMessages(require(ggplot2))
suppressPackageStartupMessages(require(naniar))
library("devtools")
```

```
## Loading required package: usethis
```

```r
source_url('https://raw.githubusercontent.com/R-miss-tastic/website/master/static/how-to/generate/amput
```

```
## SHA-1 hash of file is a392b353c3ba88ecd276c2d94bd36009d5d40616
```

```r
set.seed(1)
```

```r
# Sample data generation ------------------------------------------------------
# Generate complete data
mu.X <- c(1, 1)
Sigma.X <- matrix(c(1, 1, 1, 4), nrow = 2)
n <- 100
X.complete.cont <- mvrnorm(n, mu.X, Sigma.X)

lambda <- 0.5
X.complete.discr <- rpois(n, lambda)

n.cat <- 5
X.complete.cat <- rbinom(n, size=5, prob = 0.5)

X.complete <- data.frame(cbind(X.complete.cont, X.complete.discr, X.complete.cat))
X.complete[,4] <- as.factor(X.complete[,4])
levels(X.complete[,4]) <- c("F", "E", "D", "C", "B", "A")
```

2.1 Minimal set of arguments

In order to generate missing values for given data, `produce_NA` requires the following arguments:

- `data`: the initial data (can be complete or incomplete) as a matrix or data.frame

- `mechanism`: one of “MCAR”, “MAR”, “MNAR” (default: “MCAR”)
- `perc.missing`: the proportion of new missing values among the initially observed values (default: 0.5)

2.2 Value

`produce_NA` returns a list containing three elements:

- `data.init`: the initial data
- `data.incomp`: the data with the newly generated missing values (and the initial missing values if applicable)
- `idx_newNA`: a matrix indexing only the newly generated missing values

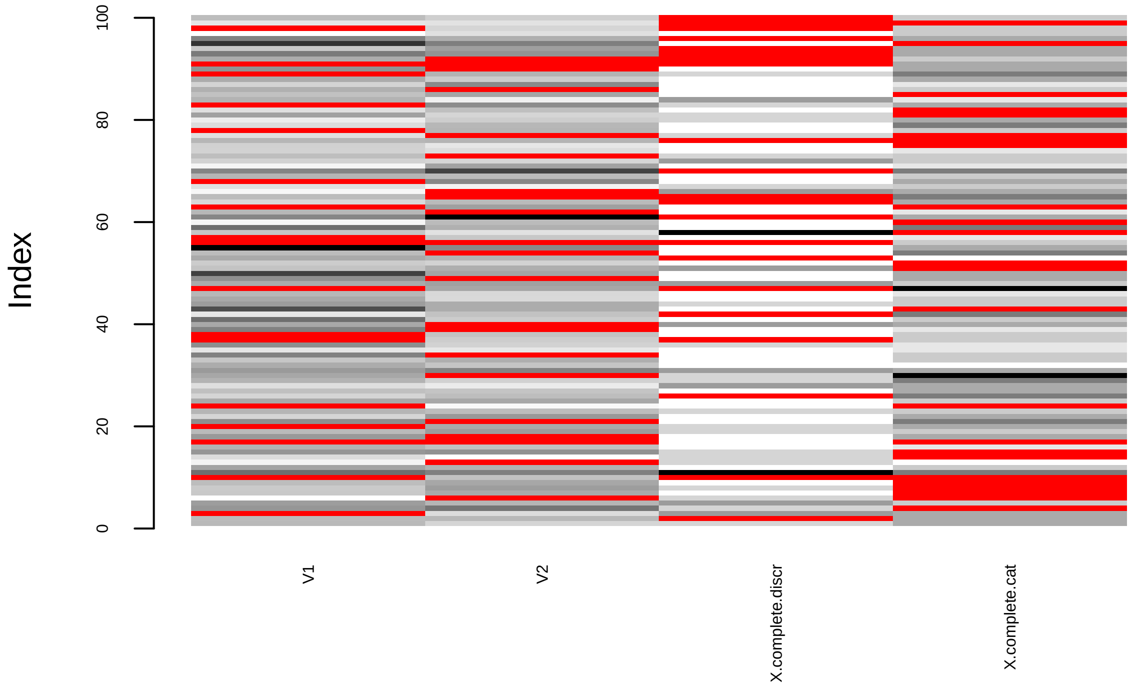
2.3 Example

On complete data

```
# Minimal example for generating missing data ------------------------
X.miss <- produce_NA(X.complete, mechanism="MCAR", perc.missing = 0.2)

X.mcar <- X.miss$data.incomp
R.mcar <- X.miss$idx_newNA

writeLines(paste0("Percentage of newly generated missing values: ", 100*sum(R.mcar)/prod(dim(R.mcar)),
```

```
## Percentage of newly generated missing values: 20.75 %
```

```
matrixplot(X.mcar, cex.axis = 0.5, interactive = F)
```



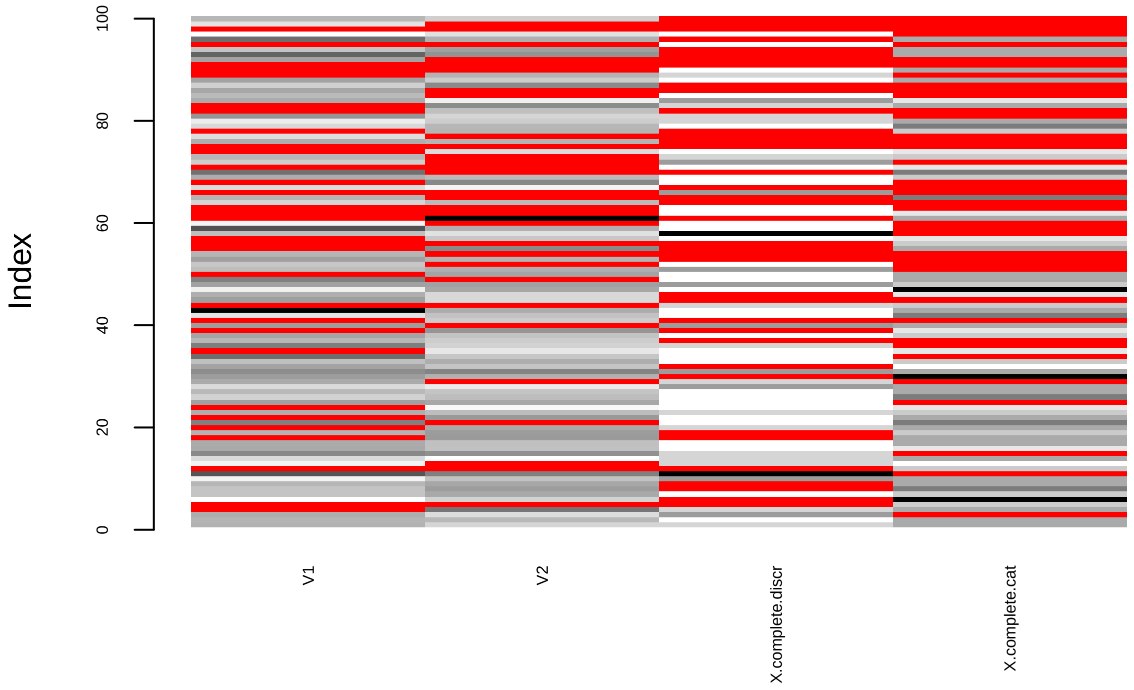
On incomplete data:

```
# Minimal example for generating missing data on an incomplete data set ------------------------
X.miss <- produce_NA(rbind(X.complete[1:50,], X.mcar[51:100,]), mechanism="MCAR", perc.missing = 0.2)
```

```
X.mcar <- X.miss$data.incomp
R.mcar <- X.miss$idx_newNA

writeLines(paste0("Percentage of newly generated missing values: ", 100*sum(R.mcar)/prod(dim(R.mcar)),
```

```
## Percentage of newly generated missing values: 22.5 %
```

```
matrixplot(X.mcar, cex.axis = 0.5, interactive = F)
```



3 Details on all available specifications

The main function `produce_NA` allows generating missing values in various ways. These can be specified through different arguments:

`produce_NA(data, mechanism = "MCAR", perc.missing = 0.5, self.mask=NULL, idx.incomplete = NULL, idx.covariates = NULL, weights.covariates = NULL, by.patterns = FALSE, patterns = NULL, freq.patterns = NULL, weights.patterns = NULL, use.all=FALSE, logit.model = "RIGHT", seed = NULL)`

3.1 Mechanisms

3.1.1 MCAR

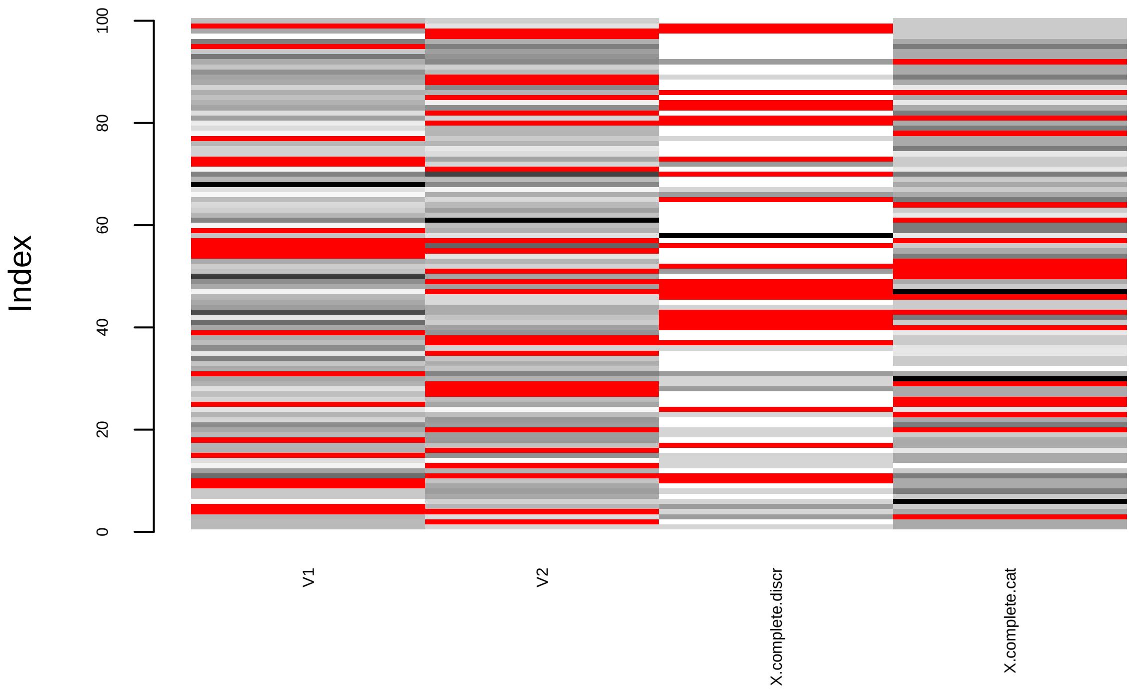
Missing Completely At Random values are generated using only the desired proportion of missing values `perc.missing`, i.e. each value have the same probability `perc.missings` of being missing. Therefore, we generate missing values using a Bernoulli distribution of parameter `perc.missing`.

```
# Sample mcar missing data ----------------------------------------
mcar <- produce_NA(X.complete, mechanism="MCAR", perc.missing = 0.2)

X.mcar <- mcar$data.incomp
```

```r
R.mcar <- mcar$idx_newNA

writeLines(paste0("Percentage of newly generated missing values: ", 100*sum(R.mcar)/prod(dim(R.mcar)),
```

```
## Percentage of newly generated missing values: 22.25 %
```

```r
matrixplot(X.mcar,  cex.axis = 0.5, interactive = F)
```



3.1.2 MAR

Missing At Random values are generated using a logistic model leading to `perc.missing` percent of missing values in each missing variable.

By default, all variables contain missing values (see Section 3.2 for changing it). More precisely, for

$$X = (X_1, X_2, X_3) = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix},$$

we generate missing values in X_1 using a logistic model depending on the variables (X_2, X_3) (thus the missingness depend on the observed values) where X_2 and X_3 have the same weights in the model (see Section 3.3.1 for changing it).

Then, there are two strategies:

- By default, we first generate missing values in X_1 and obtain a matrix containing missing values in the first column, which can be for example

$$X_1^{\text{NA}} = \begin{pmatrix} 1 \\ \text{NA} \\ 7 \\ 10 \end{pmatrix}.$$

We generate missing values in X_2 and X_3 in the same way to obtain X_2^{NA} and X_3^{NA}. The final matrix is formed by $X^{\text{NA}} = (X_1^{\text{NA}}, X_2^{\text{NA}}, X_3^{\text{NA}})$. The rows which only contain missing values are handled by replacing one of the missing values (chosen randomly) by its value given in X.

- We can also generate missing values by patterns setting `by.patterns=T`. For $X \in \mathbb{R}^{n\times 2}$, by default the patterns matrix is

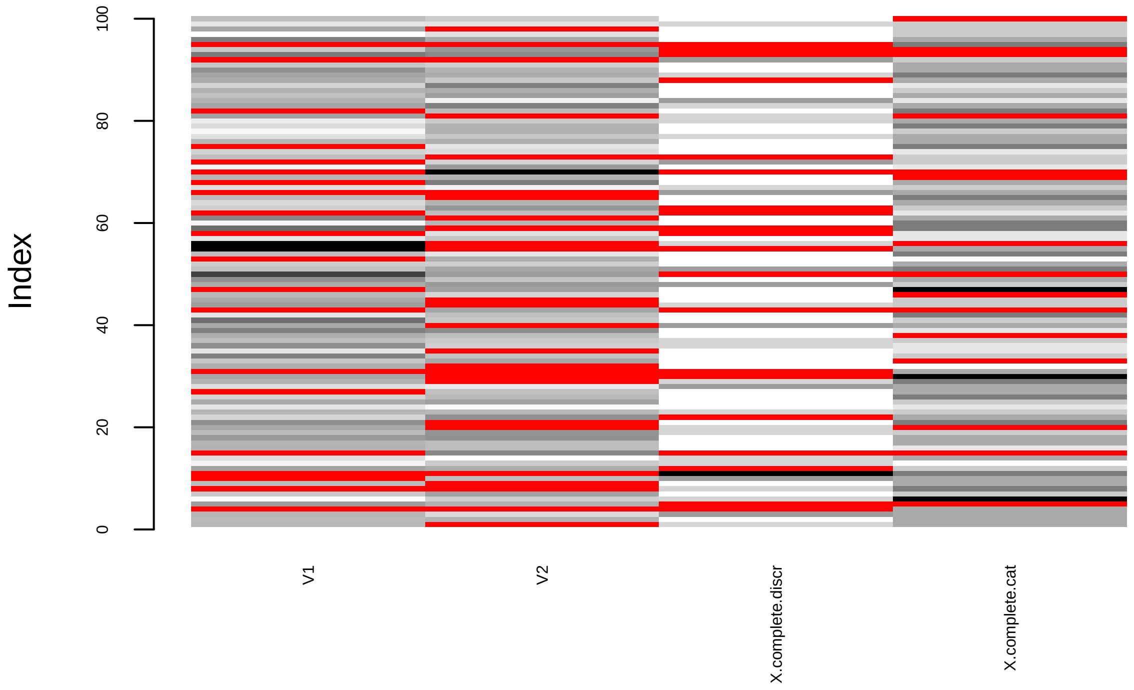
$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix},$$

  where `0` indicates that the variable should have missing values whereas `1` means that it should be oserved. In this case, there is a maximum of one missing data per row, since we generate missing values directly for the full matrix X using the patterns matrix repeatedly (the frequency of each pattern can be specified, see Section 3.3.2, by default they have the same frequency). We can also specify a patterns matrix (details are given in Section 3.3.2).

```r
# Sample mar missing data -----------------------------------------
mar <- produce_NA(X.complete, mechanism="MAR", perc.missing = 0.2, by.patterns= F)

X.mar <- mar$data.incomp
R.mar <- mar$idx_newNA

writeLines(paste0("Percentage of newly generated missing values: ", 100*sum(R.mar)/prod(dim(R.mar)), " 
```

```
## Percentage of newly generated missing values: 20.25 %
```

```r
matrixplot(X.mar,  cex.axis = 0.5, interactive = F)
```



Here, note that the generation of missing values relies on the first definition "by pattern" of MAR mechanism, which has been introduced in (Rubin 1976). There exists other ways to generate missing values as described in the Python Notebook How generate missing values?.

3.1.3 MNAR

3.1.3.1 Logistic model with missing values as predictors

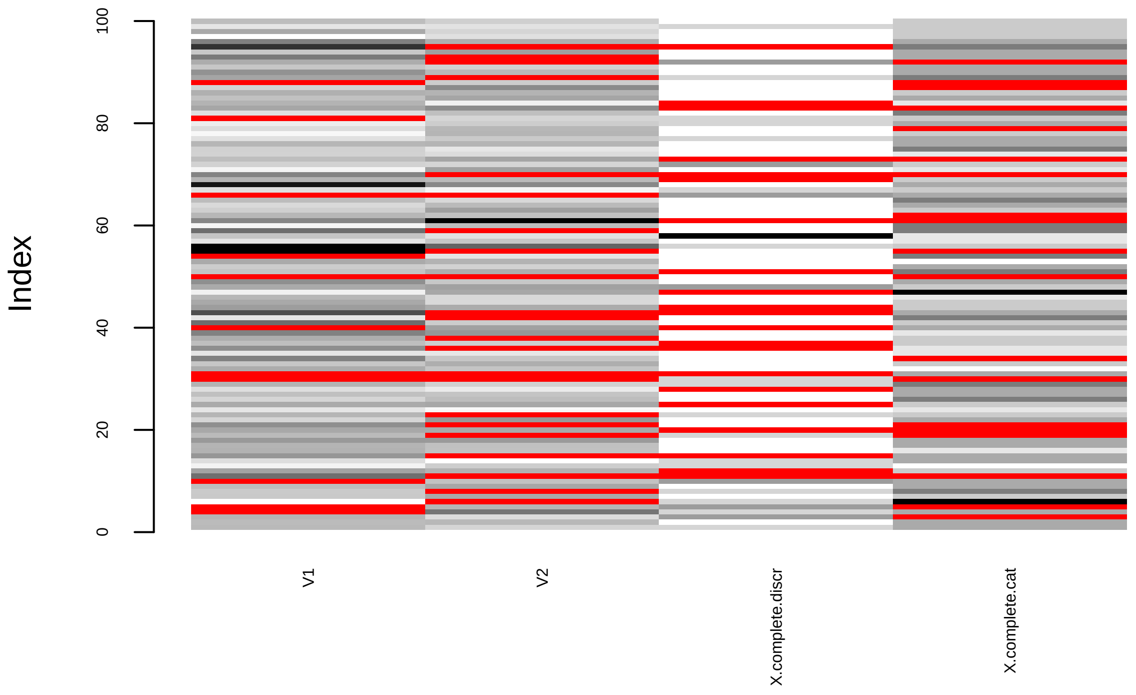
Missing Not At Random values are generated using a logistic model leading to `perc.missing` percent of missing values in each missing variable.

By default, all variables will contain missing values (see Section 3.2 for changing it). More precisely, for $X = (X_1, X_2, X_3)$ we will generate missing values in X_1 using a logistic model depending on the variables (X_1, X_2, X_3) (thus the missingness depends on the missing and observed values). Then, the same method as in Section 3.1.2 is used.

```
# Sample mnar missing data -----------------------------------------
mnar <- produce_NA(X.complete, mechanism="MNAR", perc.missing = 0.2, by.patterns= F)

X.mnar <- mnar$data.incomp
R.mnar <- mnar$idx_newNA

writeLines(paste0("Percentage of newly generated missing values: ", 100*sum(R.mnar)/prod(dim(R.mnar)),
```

```
## Percentage of newly generated missing values: 18.25 %
```

```
matrixplot(X.mnar,  cex.axis = 0.5, interactive = F)
```



3.1.3.2 Self-masked MNAR (for quantitative variables)

For self-masked MNAR values, the missingness of the variable X_j only depends on the values of X_j.

If the argument `self.mask` is filled in, self-masked missing Not At Random values are generated using a quantile censorship (three options: “sym”, “upper”, “lower”).

The variables for which missing values are generated can be given in the parameters `idx.incomplete`. Note that the proportion of missing values specified in the function call refers to the **proportion w.r.t. the incomplete variables**. Hence if you select half of your variables, to contain missing values and choose perc.missing=0.2, then the total proportion of missing values in the entire matrix/data.frame will be 0.2/2 = 0.1.

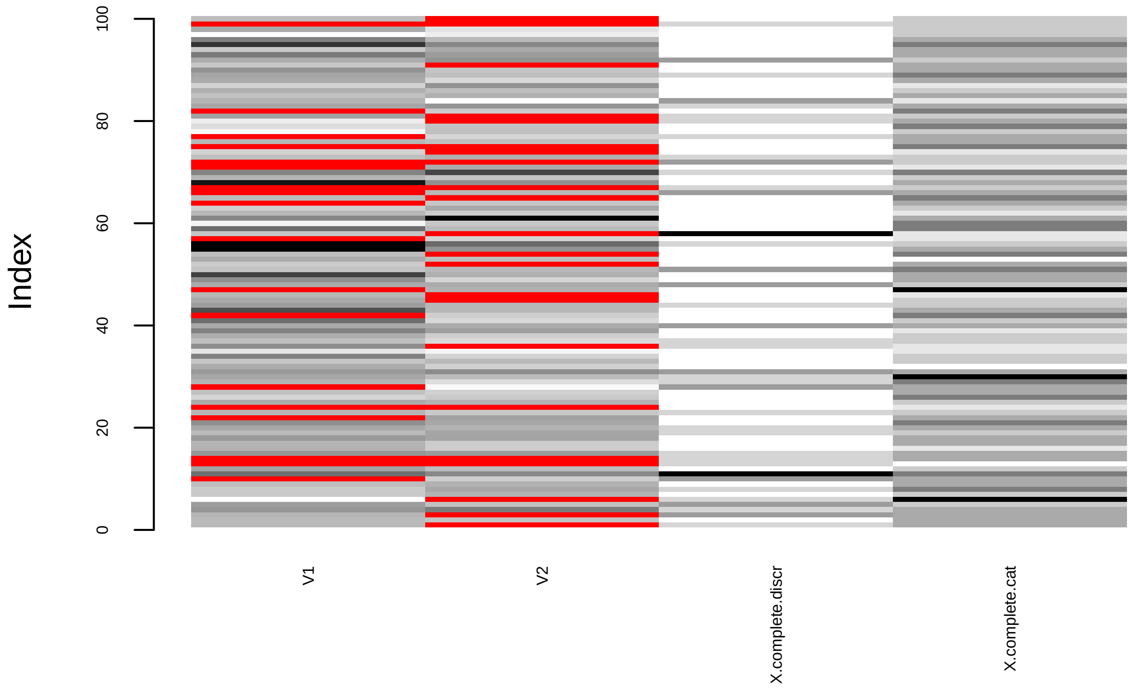
```
# Sample mnar missing data -----------------------------------------
mnar <- produce_NA(X.complete, mechanism="MNAR", perc.missing = 0.2, self.mask="lower", idx.incomplete =

X.mnar <- mnar$data.incomp
R.mnar <- mnar$idx_newNA

writeLines(paste0("Percentage of newly generated missing values: ", 100*sum(R.mnar)/prod(dim(R.mnar)),
```

```
## Percentage of newly generated missing values: 10 %
```

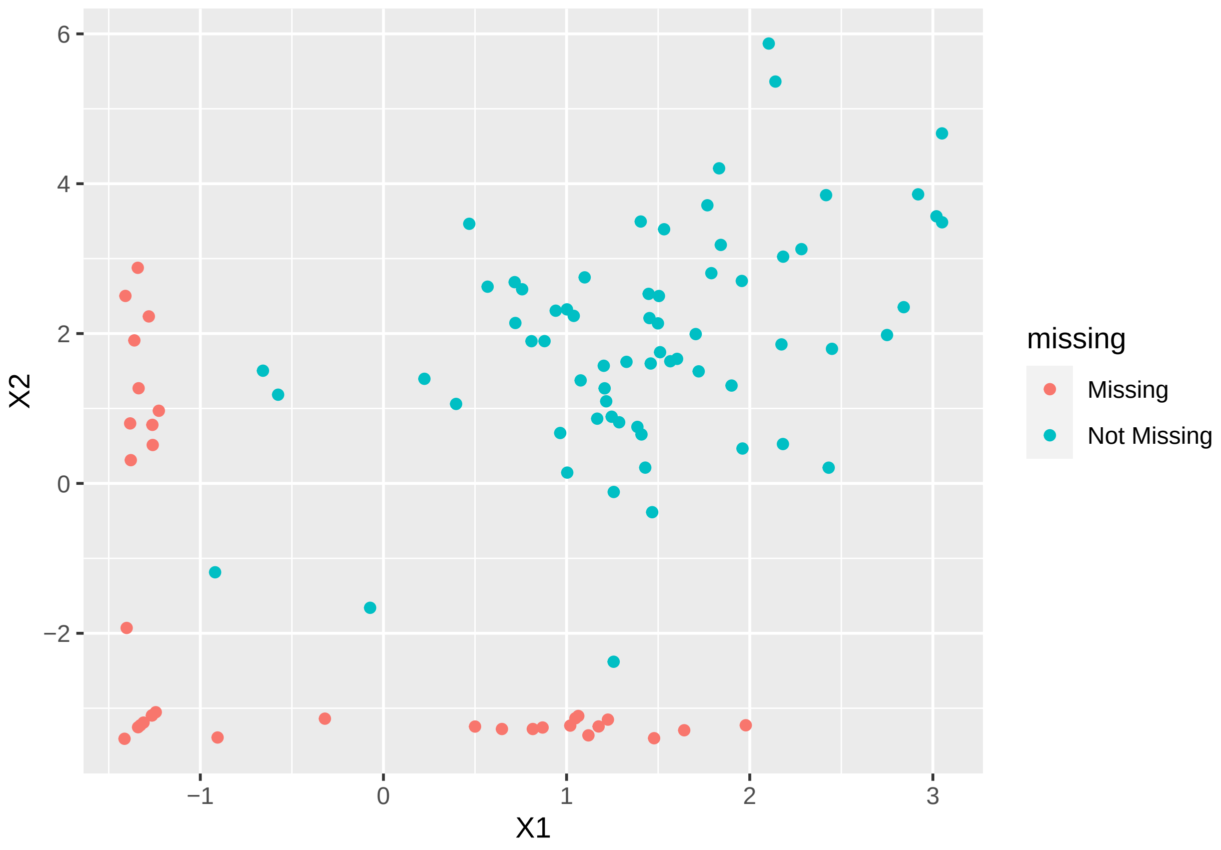
```
writeLines(paste0("Percentage of newly generated missing values (only w.r.t. to incomplete variables):
```

```
## Percentage of newly generated missing values (only w.r.t. to incomplete variables): 20 %
```

```
matrixplot(X.mnar,  cex.axis = 0.5, interactive = F)
```



```
ggplot(data=data.frame(X1=X.mnar[,1], X2=X.mnar[,2]),
       aes(x = X1,
           y = X2)) +
  geom_miss_point()
```



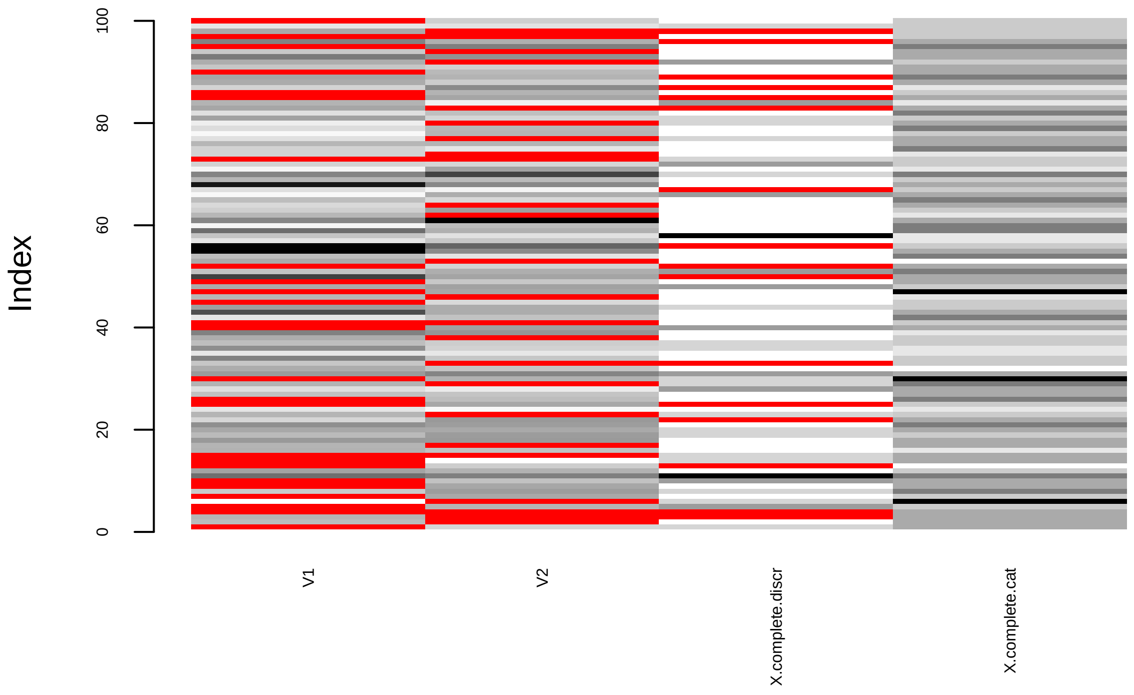
3.2 Specify incomplete variables

If you want to generate missing values only for a certain subset of variables, you can specify them by providing their position in the matrix/data.frame:

```
# Sample missing data for the first two variables in X ----------------------------------------
miss <- produce_NA(X.complete, mechanism="MCAR", perc.missing = 0.2, idx.incomplete = c(1, 1, 1, 0))

X.miss <- miss$data.incomp
R.miss <- miss$idx_newNA

writeLines(paste0("Percentage of newly generated missing values (only w.r.t. to incomplete variables):
```

```
## Percentage of newly generated missing values (only w.r.t. to incomplete variables): 21.6666666666667
```

```
matrixplot(X.miss,  cex.axis = 0.5, interactive = F)
```



3.3 Mice specific arguments

In the `mice` package there exists a function that allows already to generate missing values, `mice::ampute`. Our `produce_NA` function calls this function at some point but we chose to extend certain options, for instance with `mice::ampute` it currently is not possible to add new missing values to an already incomplete data.frame/matrix.

In order to stay close to this `ampute` function from `mice` we adopted (and adapted) some of its arguments.

3.3.1 Covariates and covariates weights

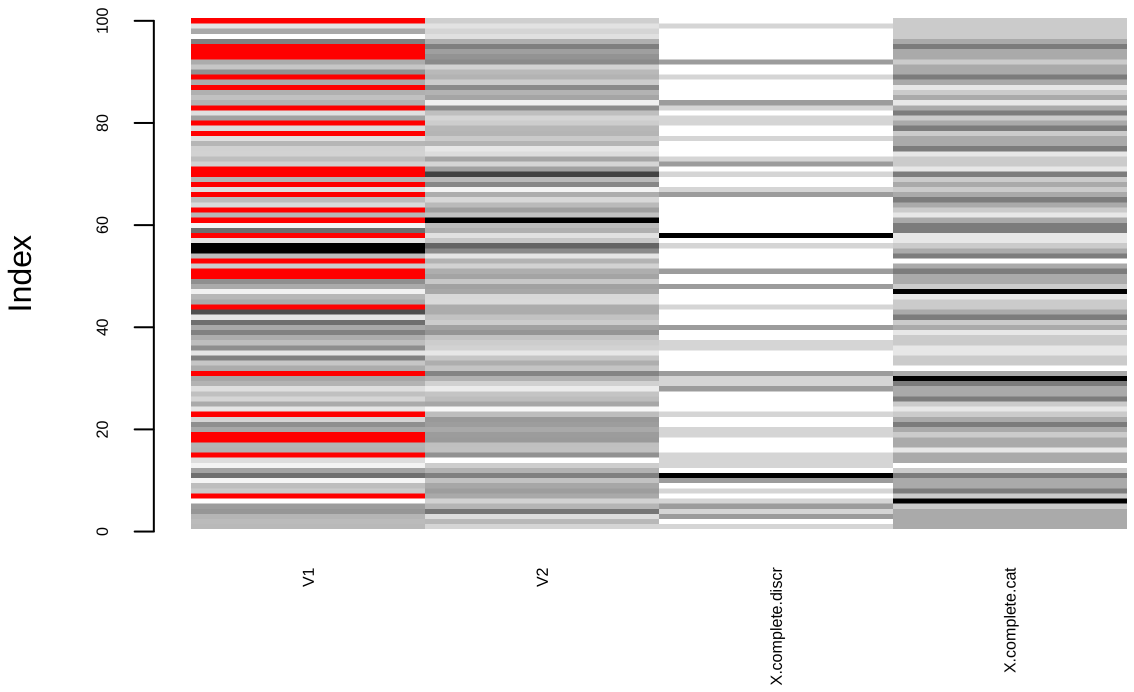
If you want to generate MAR or MNAR missing values, you can specify which variables will be used in the missingness model. You need to specify the variables that you want to use with a binary vector. For instance if you want to use variables 1 to 3 out of 7 variales, then you specify `idx.covariates = c(1,1,1,0,0,0,0)`. And you need to specify their weights as well, i.e. their contribution in the model. For instance `weights.covariates = c(1/3, 1/3, 1/3, 0, 0, 0, 0)`

Remark: if you choose `mechanism="MAR"` and `idx.incomplete = c(1,1,0,0,...,0)`, then `idx.covariates` must be of the form `c(0,0,*,*,...,*)` where `*` can be either 0 or a positive weight.

```
# Sample missing data for the first two variables in X --------------------------------------
miss <- produce_NA(X.complete, mechanism="MAR", perc.missing = 0.2, idx.incomplete = c(1, 0, 0, 0), idx

X.miss <- miss$data.incomp
R.miss <- miss$idx_newNA

writeLines(paste0("Percentage of newly generated missing values (only w.r.t. to incomplete variables):
```

```
## Percentage of newly generated missing values (only w.r.t. to incomplete variables): 26 %
```

```
matrixplot(X.miss,  cex.axis = 0.5, interactive = F)
```



3.3.2 Patterns

One might want to specify certain response/missingness patterns that are more relevant than others for a given application. This is possible by passing a matrix or data.frame whose rows contain the different patterns one wishes to generate. Additionally it is possible to specify the frequency of each pattern. We refer to the vignette of the mice::ampute function for more details on this and other related options.

This option is only implemented for the MAR and MNAR mechanisms.

3.3.2.1 Default patterns

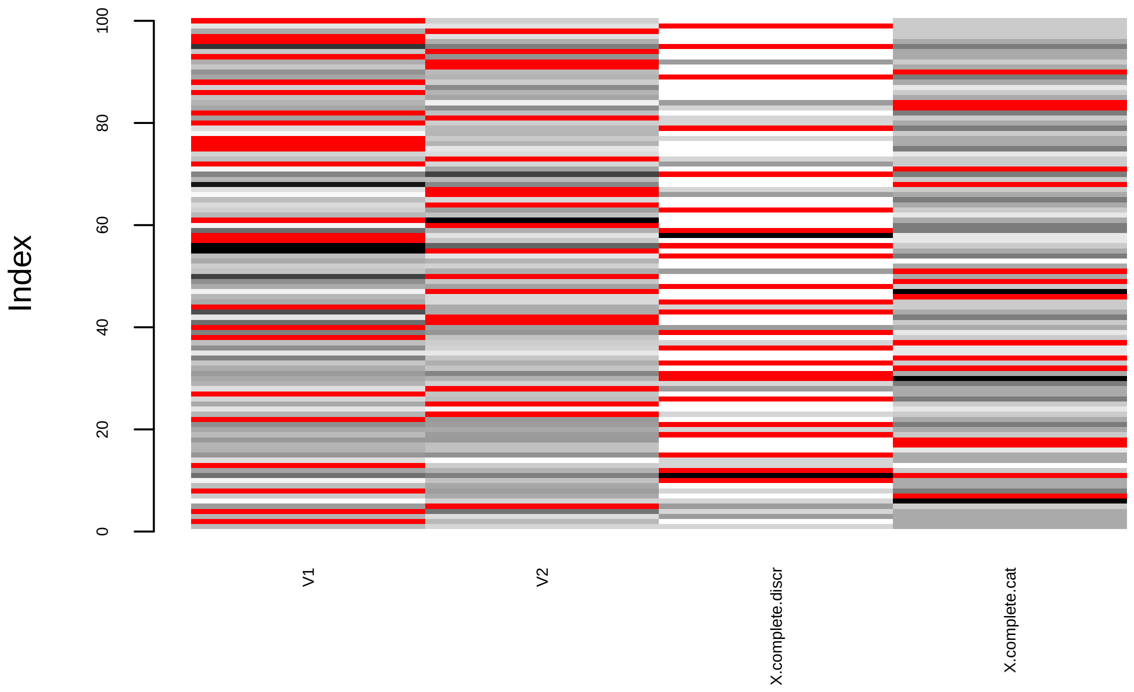
If you want to use patterns but do not wish to specify them manually, you can set `by.patterns=T` and the patterns will automatically be of the form:

$$\begin{array}{cccccc} 0 & 1 & 1 & 1 & \dots & 1 & 1 \\ 1 & 0 & 1 & 1 & \dots & 1 & 1 \\ & & & \dots & & & \\ & & & \dots & & & \\ 1 & 1 & 1 & 1 & \dots & 1 & 0 \end{array}$$

```
# Sample missing data by using the by.patterns option --------------------------------
miss <- produce_NA(X.complete, mechanism="MAR", perc.missing = 0.2, by.patterns = T)

X.miss <- miss$data.incomp
R.miss <- miss$idx_newNA

writeLines(paste0("Percentage of newly generated missing values (only w.r.t. to incomplete variables):
```

```
## Percentage of newly generated missing values (only w.r.t. to incomplete variables): 20.25 %
```

```
matrixplot(X.miss,  cex.axis = 0.5, interactive = F)
```



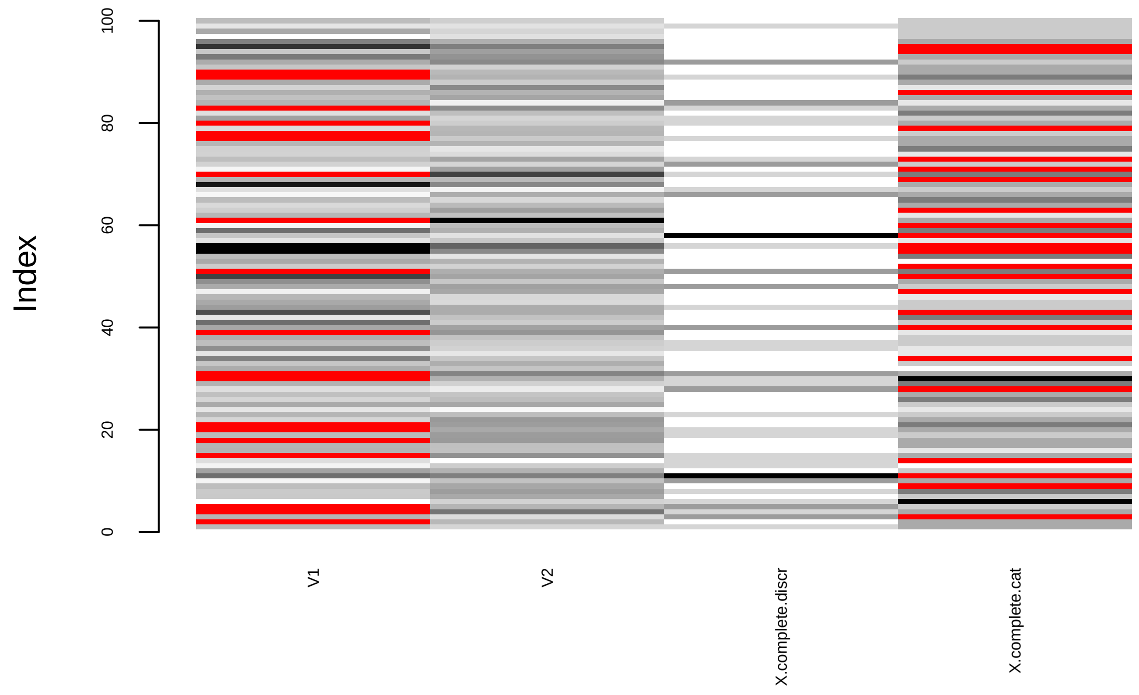
3.3.2.2 Specific patterns:

We can also specify different patterns as follows.

```
# Sample missing data by using the by.patterns option and user-specified patterns ----
miss <- produce_NA(X.complete, mechanism="MAR", perc.missing = 0.2, idx.incomplete = c(1,0,0,1), by.pat

X.miss <- miss$data.incomp
R.miss <- miss$idx_newNA

writeLines(paste0("Percentage of newly generated missing values (only w.r.t. to incomplete variables):
```

```
## Percentage of newly generated missing values (only w.r.t. to incomplete variables): 21 %
```

```
matrixplot(X.miss,  cex.axis = 0.5, interactive = F)
```

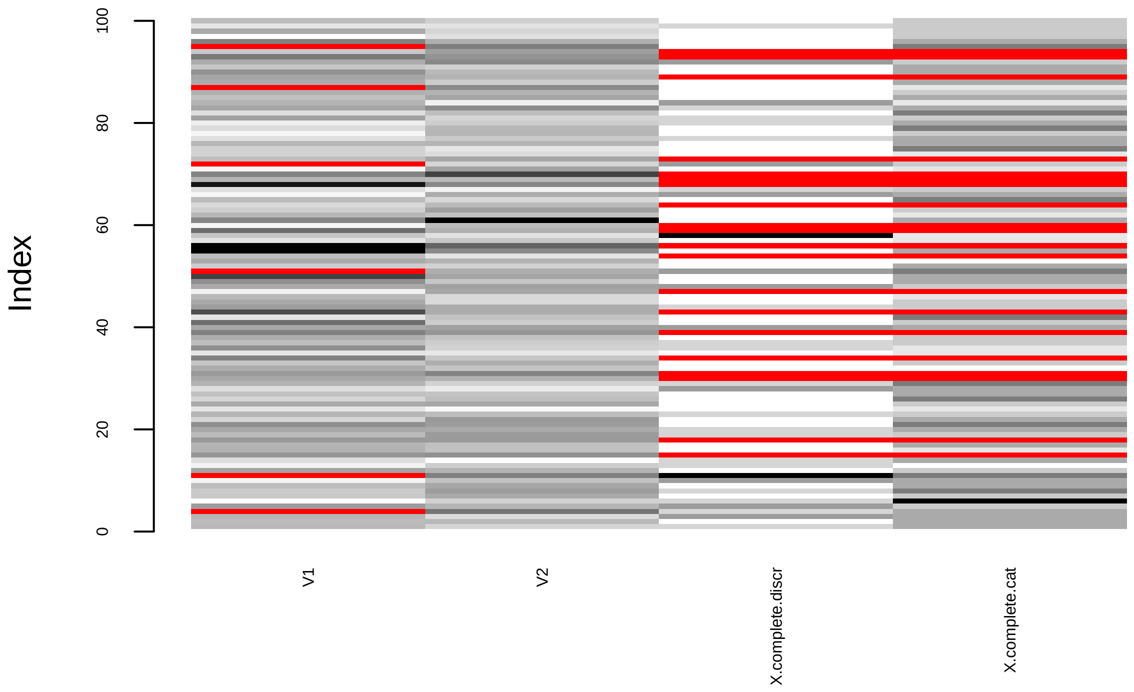


In addition, the frequency of each pattern can be chosen.

```
# Sample missing data by using the by.patterns option and user-specified patterns ----
miss <- produce_NA(X.complete, mechanism="MAR", perc.missing = 0.2, idx.incomplete = c(1,0,1,1), by.pat

X.miss <- miss$data.incomp
R.miss <- miss$idx_newNA

writeLines(paste0("Percentage of newly generated missing values (only w.r.t. to incomplete variables):
```

```
## Percentage of newly generated missing values (only w.r.t. to incomplete variables): 15.3333333333333
```

```
matrixplot(X.miss,  cex.axis = 0.5, interactive = F)
```



3.3.3 Logistic model

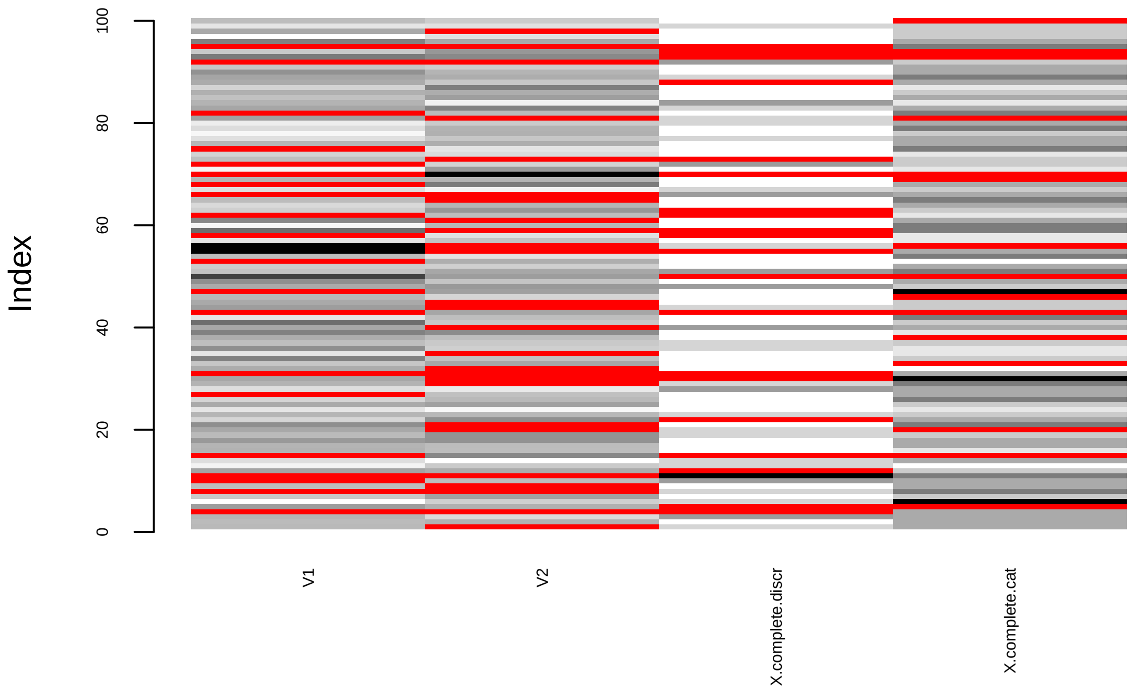
There are four possible logistic distribution functions implemented in the `mice::ampute` function: left-tailed (`"LEFT"`), right-tailed (`"RIGHT"`), centered (`"MID"`), both-tailed (`"TAIL"`).

From the mice vignette: "[These] functions are applied to the weighted sum scores. For instance, in the situation of RIGHT missingness, cases with high weighted sum scores will have a higher probability to have missing values, compared to cases with low weighted sum scores."

```
# Sample mar missing data with centered logistic distribution funciton ---------------
miss <- produce_NA(X.complete, mechanism="MAR", perc.missing = 0.2, logit.model = "MID")

X.miss <- miss$data.incomp
R.miss <- miss$idx_newNA

writeLines(paste0("Percentage of newly generated missing values: ", 100*sum(R.miss)/prod(dim(R.miss)),
```

```
## Percentage of newly generated missing values: 22 %
```

```
matrixplot(X.mar,  cex.axis = 0.5, interactive = F)
```



3.4 Other options

- `seed`: specify a seed for the random values generator, useful to obtain reproducible examples.

3.5 Full list of arguments

```
#' @param data [data.frame, matrix] (mixed) data table (n x p)
#' @param mechanism [string] either one of "MCAR", "MAR", "MNAR"; default is "MCAR"
#' @param self.mask [string] either NULL or one of "sym", "upper", "lower"; default is NULL
#' @param perc.missing [positive double] proportion of missing values, between 0 and 1; default is 0.5
#' @param idx.incomplete [array] indices of variables to generate missing values for; if NULL then miss
#' @param idx.covariates [matrix] binary matrix such that entries in row i that are equal to 1 indicate
#' @param weights.covariates [matrix] matrix of same size as idx.covariates with weights in row i for c
#' @param by.patterns [boolean] generate missing values according to (pre-specified) patterns; default
#' @param patterns [matrix] binary matrix with 1=observed, 0=missing (n_pattern x p); default is NULL
#' @param freq.patterns [array] array of size n_pattern containing desired proportion of each pattern;
#' @param weights.patterns [matrix] weights used to calculate weighted sum scores (n_pattern x p); if N
#' @param use.all [boolean] use all observations, including incomplete observations, for amputation whe
#' @param logit.model [string] either one of "RIGHT","LEFT","MID","TAIL"; default is "RIGHT"
#' @param seed [natural integer] seed for random numbers generator; default is NULL
#'
#' @return A list with the following elements
#' \item{data.init}{original data.frame}
#' \item{data.incomp}{data.frame with the newly generated missing values, observed values correspond to
#' \item{idx_newNA}{a boolean data.frame indicating the indices of the newly generated missing values}
```

4 Further resources

For more guidance on how to report results from simulations with and without missing values, we recomment the following resources:

- Using simulation studies to evaluate statistical methods (by Tim P. Morris, Ian R. White, and Michael J. Crowther)
- The Dance of the Mechanisms: How Observed Information Influences the Validity of Missingness Assumptions (by Rianne Margaretha Schouten and Gerko Vink)

5 Session info

```r
sessionInfo()
```

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Mojave 10.14.3
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
##
## locale:
## [1] fr_FR.UTF-8/fr_FR.UTF-8/fr_FR.UTF-8/C/fr_FR.UTF-8/fr_FR.UTF-8
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] mltools_0.3.5    mice_3.12.0      gdata_2.18.0     devtools_2.3.2
##  [5] usethis_2.0.0    naniar_0.6.0     ggplot2_3.3.3    VIM_6.1.0
##  [9] colorspace_2.0-0 norm_1.0-9.5     MASS_7.3-53
##
## loaded via a namespace (and not attached):
##  [1] fs_1.5.0          httr_1.4.2        rprojroot_2.0.2   tools_4.0.3
##  [5] backports_1.2.1   R6_2.5.0          nnet_7.3-14       withr_2.4.1
##  [9] sp_1.4-5          tidyselect_1.1.0  prettyunits_1.1.1 processx_3.4.5
## [13] curl_4.3          compiler_4.0.3    cli_2.2.0         desc_1.2.0
## [17] labeling_0.4.2    bookdown_0.21     scales_1.1.1      lmtest_0.9-38
## [21] DEoptimR_1.0-8    robustbase_0.93-7 callr_3.5.1       stringr_1.4.0
## [25] digest_0.6.27     foreign_0.8-80    rmarkdown_2.6     rio_0.5.16
## [29] pkgconfig_2.0.3   htmltools_0.5.1.1 sessioninfo_1.1.1 rlang_0.4.10
## [33] readxl_1.3.1      farver_2.0.3      generics_0.1.0    zoo_1.8-8
## [37] gtools_3.8.2      dplyr_1.0.3       zip_2.1.1         car_3.0-10
## [41] magrittr_2.0.1    Matrix_1.2-18     Rcpp_1.0.6        munsell_0.5.0
## [45] fansi_0.4.2       abind_1.4-5       lifecycle_0.2.0   visdat_0.5.3
## [49] stringi_1.5.3     yaml_2.2.1        carData_3.0-4     pkgbuild_1.2.0
## [53] forcats_0.5.0     crayon_1.3.4      lattice_0.20-41   haven_2.3.1
## [57] hms_1.0.0         knitr_1.30        ps_1.5.0          pillar_1.4.7
## [61] ranger_0.12.1     boot_1.3-25       pkgload_1.1.0     glue_1.4.2
## [65] evaluate_0.14     data.table_1.13.6 laeken_0.5.1      remotes_2.2.0
## [69] vcd_1.4-8         vctrs_0.3.6       testthat_3.0.1    cellranger_1.1.0
## [73] gtable_0.3.0      purrr_0.3.4       tidyr_1.1.2       assertthat_0.2.1
## [77] xfun_0.20         openxlsx_4.2.3    broom_0.7.3       e1071_1.7-4
## [81] class_7.3-17      tibble_3.0.5      memoise_1.1.0     ellipsis_0.3.1
```

References

Rubin, Donald B. 1976. "Inference and Missing Data." *Biometrika* 63 (3). [Oxford University Press, Biometrika Trust]: 581–92. http://www.jstor.org/stable/2335739.

Schouten, Rianne Margaretha, Peter Lugtig, and Gerko Vink. 2018. "Generating Missing Values for Simulation Purposes: A Multivariate Amputation Procedure." *Journal of Statistical Computation and Simulation* 88 (15). Taylor & Francis: 2909–30.