

# Package: simdata (via r-universe)

September 30, 2024

**Type** Package

**Title** Generate Simulated Datasets

**Version** 0.4.0

**Maintainer** Michael Kammer <michael.kammer@meduniwien.ac.at>

**Description** Generate simulated datasets from an initial underlying distribution and apply transformations to obtain realistic data. Implements the 'NORTA' (Normal-to-anything) approach from Cario and Nelson (1997) and other data generating mechanisms. Simple network visualization tools are provided to facilitate communicating the simulation setup.

**License** GPL-3

**Imports** Matrix, mvtnorm, igraph (>= 1.2.1)

**Suggests** doParallel, doRNG, dplyr, fitdistrplus, forcats, ggplot2, GGally, ggcorrplot, knitr, nhanesA, patchwork, purrr, reshape2, rmarkdown, testthat (>= 2.1.0)

**VignetteBuilder** knitr

**Encoding** UTF-8

**RoxygenNote** 7.3.1

**URL** <https://matherealize.github.io/simdata/>,  
<https://github.com/matherealize/simdata>

**BugReports** <https://github.com/matherealize/simdata/issues>

**NeedsCompilation** no

**Author** Michael Kammer [aut, cre]  
(<<https://orcid.org/0000-0003-4821-9928>>)

**Repository** CRAN

**Date/Publication** 2024-07-01 10:20:20 UTC

Contents

apply_array . . . . .	2
as_function_list . . . . .	3
colapply_functions . . . . .	3
contains_constant . . . . .	4
cor_from_upper . . . . .	5
cor_to_cov . . . . .	5
cor_to_upper . . . . .	6
do_processing . . . . .	6
estimate_final_correlation . . . . .	7
function_list . . . . .	8
get_from_function_list . . . . .	10
get_names_from_function_list . . . . .	10
is_collinear . . . . .	11
is_cor_matrix . . . . .	11
optimize_cor_for_pair . . . . .	12
optimize_cor_mat . . . . .	13
partial . . . . .	14
plot_cor_network . . . . .	15
plot_estimated_cor_network . . . . .	17
process_truncate_by_iqr . . . . .	18
process_truncate_by_threshold . . . . .	20
quantile_functions_from_data . . . . .	21
simdesign . . . . .	22
simdesign_discunif . . . . .	25
simdesign_mvtnorm . . . . .	26
simdesign_norta . . . . .	28
simulate_data . . . . .	30
simulate_data_conditional . . . . .	33
<b>Index</b>	<b>36</b>

---

apply_array	<i>Helper to apply functions</i>
-------------	----------------------------------

---

Description

Used to make use of apply-like operations, regardless of wether the input is a matrix or a data.frame

Usage

apply\_array(obj, dim, fun)

**Arguments**

obj	Matrix or data.frame.
dim	Dimension to apply function to.
fun	Function object to apply.

---

as_function_list	<i>Create function_list object from list of functions</i>
------------------	---

---

**Description**

Create a function\_list object from a list of functions. This is useful if such a list is created programmatically.

**Usage**

```
as_function_list(flist, ...)
```

**Arguments**

flist	List in which each entry is a function object. Can be named or unnamed.
...	Passed to <a href="#">function_list</a> .

**Value**

Function with a single input which outputs a data.frame. Has special 'flist' entry in its environment which stores individual functions as list.

**See Also**

[function\\_list](#)

---

colapply_functions	<i>Apply list of functions to column of object</i>
--------------------	--

---

**Description**

Helper function to simplify workflow with lists of functions.

**Usage**

```
colapply_functions(obj, flist)
```

**Arguments**

<code>obj</code>	2-dimensional array (matrix or data.frame).
<code>flist</code>	List of functions of length equal to the number of columns of <code>obj</code> . Each entry must be a function applicable to a single column of <code>obj</code> . The <i>i</i> -th entry of <code>flist</code> is applied to the <i>i</i> -th column of <code>obj</code> .

**Value**

Matrix or data.frame (same type as `obj`) with names taken from `obj`.

---

<code>contains_constant</code>	<i>Check if matrix contains constant column(s)</i>
--------------------------------	--

---

**Description**

Check if matrix contains constant column(s)

**Usage**

```
contains_constant(x, eps = .Machine$double.eps)
```

**Arguments**

<code>x</code>	Matrix or Data.frame.
<code>eps</code>	Threshold for standard deviation below which a column is considered to be constant.

**Value**

TRUE if one of the columns has standard deviation of below ‘eps’, else FALSE.

**Note**

Prints a warning if constant is found.

---

cor_from_upper	<i>Build correlation matrix</i>
----------------	---------------------------------

---

**Description**

Use to specify correlation matrix in convenient way by giving entries of the upper triangular part.

**Usage**

```
cor_from_upper(n_var, entries = NULL)
```

**Arguments**

n_var	Integer, number of variables (= rows = columns of matrix).
entries	Matrix of correlation entries. Consists of 3 columns (variable_1, variable_2, correlation) that specify both variables and corresponding correlation in the upper triangular part of the matrix (i.e. variable_1 < variable_2) .

**Value**

Matrix with user supplied entries.

**See Also**

[cor\\_to\\_upper](#)

**Examples**

```
cor_from_upper(2, rbind(c(1, 2, 0.8)))
```

---

cor_to_cov	<i>Convert correlation matrix to covariance matrix</i>
------------	--

---

**Description**

Rescale correlation matrix by variable standard deviations to yield a covariance matrix.

**Usage**

```
cor_to_cov(m, sds = NULL)
```

**Arguments**

m	Symmetric correlation matrix.
sds	Standard deviations of the variables. Set to 1 for all variables by default.

**Value**

Symmetric covariance matrix.

---

cor_to_upper	<i>Convert correlation matrix to specification used by cor_from_upper</i>
--------------	---

---

**Description**

Convert correlation matrix to specification used by cor\_from\_upper

**Usage**

```
cor_to_upper(m, remove_below = .Machine$double.eps)
```

**Arguments**

m	Symmetric correlation matrix.
remove_below	Threshold for absolute correlation values below which they are removed from the returned matrix. If NULL then no filtering is applied.

**Value**

Matrix with 3 columns (variable\_1, variable\_2, correlation), where correlation gives the entry at position (variable\_1, variable\_2) of the input correlation matrix. Note that variable\_1 < variable\_2 holds for all entries.

**See Also**

[cor\\_from\\_upper](#)

---

do_processing	<i>Post-processing of datamatrix</i>
---------------	--------------------------------------

---

**Description**

Applies functions to a matrix or data.frame.

**Usage**

```
do_processing(x, functions = list())
```

**Arguments**

x	Matrix or Data.frame.
functions	List of lists, specifying functions to be applied as well as their arguments. See details.

**Details**

Functions are passed into the post-processor as a named list. The name `f` of the list entry is the function to be applied via `base::do.call`. The list entry itself is another named list, specifying the arguments to the function `f` as named arguments.

The functions must take a matrix or data.frame as first argument and return another matrix or data.frame of the same dimensions as single output.

Examples of post-processing steps are truncation (`process_truncate_by_iqr`, `process_truncate_by_threshold`) or centering / standardizing data (via `scale`, see example section below).

Can be useful to apply on simulated datasets, even outside of the simulation function (e.g. when standardization is only required at the modeling step).

**Value**

Matrix or data.frame with post-processing applied.

**Note**

Use with caution - no error checking is done for now so the user has to take care of everything themselves! Furthermore, output of the functions is not checked either.

**Examples**

```
do_processing(diag(5),
  functions = list(scale = list(center = TRUE, scale = FALSE)))
```

---

```
estimate_final_correlation
```

*Estimate correlation matrix via simulation*

---

**Description**

Used to obtain an estimate of the correlation matrix after transforming the initial data.

**Usage**

```
estimate_final_correlation(
  obj,
  n_obs = 1e+05,
  cor_type = "pearson",
  seed = NULL,
  ...
)
```

**Arguments**

obj	S3 class object of type simdesign (or inheriting from it).
n_obs	Number of observations to simulate.
cor_type	Can be either a character (pearson, spearman, kendall) which is passed to <code>stats::cor</code> or a function, which is directly used to compute the correlation matrix on the simulated data. Such a function is expected to take a single input matrix (and possibly other arguments which can be set via <code>...</code> ) and output a single matrix.
seed	Random number seed. NULL does not change the current seed.
...	Further arguments are passed to the function that computes the correlation matrix (either <code>stats::cor</code> or the user provided function).

**Details**

This function is useful to estimate the final correlation of the data after transformation of the initial data. To provide a robust estimate it is advised to use a very large number of observations to compute the correlation matrix.

**Value**

A numeric matrix given by the pairwise correlation coefficients for each pair of variables defined by obj and computed according to cor\_type.

**See Also**

[simulate\\_data](#), [simdesign](#)

---

function_list	<i>Apply list of functions to input</i>
---------------	---

---

**Description**

Apply list of functions to input

**Usage**

```
function_list(..., stringsAsFactors = FALSE, check.names = TRUE)
```

**Arguments**

...	Named or unnamed arguments, each of which is a function taking exactly one input. See details.
stringsAsFactors, check.names	Arguments of <a href="#">data.frame</a> .



## Details

This is a convenience function which takes a number of functions and returns another function which applies all of the user specified functions to a new input, and collects the results as list or data.frame. This is useful to e.g. transform columns of a data.frame or check the validity of a matrix during simulations. See the example here and in [simulate\\_data\\_conditional](#).

The assumptions for the individual functions are:

- Each function is expected to take a single input.
- Each function is expected to output a result consistent with the other functions (i.e. same output length) to ensure that the results can be summarized as a data.frame.

## Value

Function with a single input which outputs a data.frame. Has special 'flist' entry in its environment which stores individual functions as list.

## Note

This function works fine without naming the input arguments, but the resulting data.frames have empty column names if that is the case. Thus, it is recommended to only pass named function arguments.

## See Also

[data.frame](#), [get\\_from\\_function\\_list](#), [get\\_names\\_from\\_function\\_list](#)

## Examples

```
f <- function_list(
  v1 = function(x) x[, 1] * 2,
  v2 = function(x) x[, 2] + 10)

f(diag(2))

# function_list can be used to add new columns
# naming of columns should be handled separately in such cases

f <- function_list(
  function(x) x, # return x as it is
  X1_X2 = function(x) x[, 2] + 10) # add new column

f(diag(2))
```

---

`get_from_function_list`*Extract individual functions from function\_list*

---

**Description**

Extract individual function objects from environment of a `function_list` object.

**Usage**

```
get_from_function_list(flist)
```

**Arguments**

`flist`                      `function_list` or function object.

**Value**

List with named or unnamed entries corresponding to individual function objects that were passed to the `function_list` object. If `flist` is a simple function, returns `NULL`.

**See Also**

[function\\_list](#)

---

`get_names_from_function_list`*Extract names of individual functions from function\_list*

---

**Description**

Extract names of individual function objects from environment of a `function_list` object.

**Usage**

```
get_names_from_function_list(flist)
```

**Arguments**

`flist`                      `function_list` or function object.

**Value**

Names of list corresponding to individual function objects that were passed to the `function_list` object. If `flist` is a simple function, returns `NULL`.

**See Also**[function\\_list](#)

---

is_collinear	<i>Check if matrix is collinear</i>
--------------	-------------------------------------

---

**Description**

Check if matrix is collinear

**Usage**

```
is_collinear(x)
```

**Arguments**

x	Matrix or Data.frame.
---	-----------------------

**Value**

TRUE if matrix is collinear, else FALSE.

**Note**

Prints a warning if collinear.

---

is_cor_matrix	<i>Check if matrix is a correlation matrix</i>
---------------	--

---

**Description**

Checks if matrix is numeric, symmetric, has diagonal elements of one, has only entries in  $[-1, 1]$ , and is positive definite. Prints a warning if a problem was found.

**Usage**

```
is_cor_matrix(m, tol = 1e-09)
```

**Arguments**

m	Matrix.
tol	Tolerance for checking diagonal elements.

**Value**

TRUE if matrix is a correlation matrix, else FALSE.

---

optimize\_cor\_for\_pair *Find pairwise initial correlation for NORTA from target correlation*

---

## Description

This function can be used to find a suitable initial correlation for use in the NORTA procedure for a pair of variables with given marginal distributions and target correlation.

## Usage

```
optimize_cor_for_pair(
  cor_target,
  dist1,
  dist2,
  n_obs = 1e+05,
  seed = NULL,
  tol = 0.01,
  ...
)
```

## Arguments

cor_target	Target correlation of variable pair.
dist1, dist2	Marginal distributions of variable pair, given as univariable quantile functions.
n_obs	Number of observations to be used in the numerical optimization procedure.
seed	Seed for generating standard normal random variables in the numerical optimization procedure.
tol, ...	Further parameters passed to <code>stats::uniroot</code> .

## Details

Uses `stats::uniroot` for actual optimization.

## Value

Output of `stats::uniroot` for the univariable optimization for find the initial correlation.

---

optimize_cor_mat	<i>Find initial correlation matrix for NORTA from target correlation</i>
------------------	--

---

## Description

This function can be used to find a suitable correlation matrix to be used for simulating initial multivariate normal data in a NORTA based simulation design (see [simdesign\\_norta](#)).

## Usage

```
optimize_cor_mat(
  cor_target,
  dist,
  ensure_cor_mat = TRUE,
  conv_norm_type = "0",
  return_diagnostics = FALSE,
  ...
)
```

## Arguments

cor_target	Target correlation matrix.
dist	List of functions of marginal distributions for simulated variables. Must have the same length as the specified correlation matrix (cor_target), and the order of the entries must correspond to the variables in the correlation matrix. See <a href="#">simdesign_norta</a> for details of the specification of the marginal distributions.
ensure_cor_mat	if TRUE, this function ensures that the optimized matrix is a proper correlation matrix by ensuring positive definiteness. If FALSE, the optimized matrix is returned as is.
conv_norm_type	Metric to be used to find closest positive definite matrix to optimal matrix, used if ensure_cor_mat is TRUE. Passed to <a href="#">Matrix::nearPD</a> .
return_diagnostics	TRUE to return additional diagnostics of the optimization procedure, see below.
...	Additional parameters passed to <a href="#">optimize_cor_for_pair</a> .

## Details

This function first finds a suitable correlation matrix for the underlying multivariate normal data used in the NORTA procedure. It does so by solving  $k*(k-1)$  univariable optimisation problems (where  $k$  is the number of variables). In case the result is not a positive-definite matrix, the nearest positive-definite matrix is found according to the user specified metric using [Matrix::nearPD](#). See e.g. Ghosh and Henderson (2003) for an overview of the procedure.

**Value**

If `return_diagnostics` is `FALSE`, a correlation matrix to be used in the definition of a `simdesign_norta` object. If `TRUE`, then a list with two entries: `cor_mat` containing the correlation matrix, and `convergence` containing a list of objects returned by the individual optimisation problems from `stats::uniroot`.

**References**

Ghosh, S. and Henderson, S. G. (2003) *Behavior of the NORTA method for correlated random vector generation as the dimension increases*. ACM Transactions on Modeling and Computer Simulation.

**See Also**

`simdesign_norta`

---

partial	<i>Define partial function</i>
---------	--------------------------------

---

**Description**

Partial functions are useful to define marginal distributions based on additional parameters.

**Usage**

```
partial(f, ...)
```

**Arguments**

<code>f</code>	Function in two or more parameters.
<code>...</code>	Parameters to be held fixed for function <code>f</code> .

**Details**

This helper function stores passed arguments in a list, and stores this list in the environment of the returned function. Thus, it remembers the arguments that should be held fixed, such that the returned partial function now is a function with fewer arguments.

**Value**

Function object.

**Examples**

```
marginal <- partial(function(x, meanx) qnorm(x, meanx), meanx = 2)
marginal(0.5)
```

---

plot_cor_network	<i>Visualize fixed correlation structure as a network</i>
------------------	---

---

### Description

Useful to visualize e.g. the associations of the initial multivariate gaussian distribution used by [simdesign\\_mvtnorm](#).

### Usage

```
plot_cor_network(obj, ...)

## Default S3 method:
plot_cor_network(
  obj,
  categorical_indices = NULL,
  decimals = 2,
  cor_cutoff = 0.1,
  vertex_labels = NULL,
  vertex_label_prefix = "z",
  edge_width_function = function(x) x * 10,
  edge_label_function = function(x) round(x, decimals),
  use_edge_weights = FALSE,
  edge_weight_function = base::identity,
  seed = NULL,
  return_network = FALSE,
  mar = c(0, 0, 0, 0),
  vertex.size = 12,
  margin = 0,
  asp = 0,
  vertex.color = "#ecec",
  vertex.frame.color = "#979797",
  vertex.label.color = "black",
  edge.color = "ramp",
  edge.label.color = "black",
  edge.label.cex = 0.8,
  ...
)

## S3 method for class 'simdesign_mvtnorm'
plot_cor_network(obj, ...)
```

### Arguments

obj	Correlation matrix or S3 class object which has a class method available (see below).
-----	---

...	Passed to <code>igraph::plot</code> , with a complete list of arguments and details given in <a href="#">igraph.plotting</a> .
categorical_indices	Vector of indices of variables which should be drawn as rectangles (i.e. represent categorical data).
decimals	Number of decimals, used for default labeling of the network edges.
cor_cutoff	Threshold of absolute correlation below which nodes are not considered as connected. Useful to control complexity of drawn network. Set to NULL to disable.
vertex_labels	Character vector of length <code>nrow(obj)</code> of labels for vertices. If not NULL, overrides the <code>vertex_label_prefix</code> argument. If set to NA omits all or some vertex labels.
vertex_label_prefix	String which is added as prefix to node labels.
edge_width_function	Function which takes one vector input (absolute correlation values) and outputs transformation of this vector (must be $\geq 0$ ). Defines edge widths.
edge_label_function	Function which takes on vector input (absolute correlation values) and outputs labels for these values as character vector. Defines edges labels. If set to NULL, then no edge labels will be displayed.
use_edge_weights	Logical, if TRUE then the layout will be influenced by the absolute correlations (i.e. edge weights) such that highly correlated variables will be put closer together. If FALSE, then the layout is independent of the correlation structure.
edge_weight_function	Function which takes one vector input (absolute correlation values) and outputs transformation of this vector (must be $\geq 0$ ). Defines edge weights. Only relevant if <code>use_edge_weights</code> is TRUE.
seed	Set random seed to ensure reproducibility of results. Can be fixed to obtain same layout but vary edge widths, correlation functions etc. Can also be used to obtain nicer looking graph layouts.
return_network	If TRUE, the <code>igraph</code> network object is returned and can be plotted by the user using e.g. the interactive <code>igraph::tkplot</code> function.
mar	<code>mar</code> argument to the <code>par</code> function to set margins of the plot (often required when the axes should be drawn). A numerical vector of the form <code>c(bottom, left, top, right)</code> which gives the number of lines of margin to be specified on the four sides of the plot. The default is <code>c(5, 4, 4, 2) + 0.1</code> . Note that this is not the same argument as the <code>margin</code> argument for the <code>igraph::plot.igraph</code> function.
vertex.size, margin, asp, vertex.frame.color, vertex.label.color, edge.label.color, edge.label.cex	Arguments to <code>igraph::plot</code> , with sensible defaults for this package's usage.
vertex.color	Argument passed to <code>igraph::plot</code> . Usually a character vector with a hex color specification for vertex color. Alternatively a function that takes as input a <code>data.frame</code> with a column "id" that gives the column number of the simulated data, and outputs a valid color specification for the corresponding vertices (i.e. a single character hex color or a vector of such hex colors of appropriate length).



`edge.color` Argument passed to [igraph::plot](#). This package implements some special functionality: if `edge.color = "ramp"` then a `colormap` from red (-1) via white (0) to blue (1) is mapped to the correlations and the edges colored accordingly. If `edge.color = "clipped-ramp"` then the ramp is restricted to the correlation values observed, which may be useful if they are low to increase visibility. If `edge.color = "red-blue"` then all edges with positive correlation values are colored uniformly red, and all edges with negative correlations are colored uniformly blue. Alternatively, may be a function that takes as input the edge correlation values and outputs valid color specifications (i.e. a single hex color or a vector of hex colors of appropriate length).

### Details

For an explanation of all parameters not listed here, please refer to [igraph::plot](#).

### Value

If `return_network` is `TRUE`, then an `igraph` network object is returned that can be plotted by the user using e.g. the interactive [igraph::tkplot](#) function. Otherwise, the network object is plotted directly and no output is returned.

### Methods (by class)

- `plot_cor_network(default)`: Function to be used for correlation matrix.
- `plot_cor_network(simdesign_mvtnorm)`: Function to be used with [simdesign\\_mvtnorm](#) S3 class object to visualize initial correlation network of the underlying multivariate normal distribution.

### See Also

[plot\\_cor\\_network.simdesign\\_mvtnorm](#), [plot\\_estimated\\_cor\\_network](#)

---

`plot_estimated_cor_network`

*Visualize estimated correlation matrix as a network*

---

### Description

Based on approximation via simulation specified by given simulation design. Convenience wrapper for combining [estimate\\_final\\_correlation](#) and [plot\\_cor\\_network](#).

### Usage

```
plot_estimated_cor_network(
  obj,
  n_obs = 1e+05,
  cor_type = "pearson",
  seed = NULL,
```

```

    show_categorical = TRUE,
    return_network = FALSE,
    ...
  )

```

### Arguments

<code>obj</code>	S3 class object of type <code>simdesign</code> (or inheriting from it).
<code>n_obs</code>	Number of observations to simulate.
<code>cor_type</code>	Can be either a character ( <code>pearson</code> , <code>spearman</code> , <code>kendall</code> ) which is passed to <code>stats::cor</code> or a function, which is directly used to compute the correlation matrix on the simulated data. Such a function is expected to take a single input matrix (and possibly other arguments which can be set via <code>...</code> ) and output a single matrix.
<code>seed</code>	Random number seed. <code>NULL</code> does not change the current seed.
<code>show_categorical</code>	If <code>TRUE</code> , marks categorical variables differently from numeric ones. Determined by the <code>types_final</code> slot of the <code>obj</code> argument.
<code>return_network</code>	If <code>TRUE</code> , the <code>igraph</code> network object is returned and can be plotted by the user using e.g. the interactive <code>igraph::tkplot</code> function.
<code>...</code>	Passed to <code>plot_cor_network</code> .

### Details

This function is useful to estimate the correlation network of a simulation setup after the initial underlying distribution  $Z$  has been transformed to the final dataset  $X$ .

### Value

If `return_network` is `TRUE`, then an `igraph` network object is returned that can be plotted by the user using e.g. the interactive `igraph::tkplot` function. Otherwise, the network object is plotted directly and no output is returned.

### See Also

`plot_cor_network`, `estimate_final_correlation`

---

`process_truncate_by_iqr`

*Truncate columns of datamatrix at datamatrix specific thresholds*

---

### Description

Truncation based on the interquartile range to be applied to a dataset.

**Usage**

```
process_truncate_by_iqr(x, truncate_multipliers = NA, only_numeric = TRUE)
```

**Arguments**

<code>x</code>	Matrix or Data.frame.
<code>truncate_multipliers</code>	Vector of truncation parameters. Either a single value which is replicated as necessary or of same dimension as <code>ncol(x)</code> . If any vector entry is NA, the corresponding column will not be truncated. If named, then the names must correspond to columnnames in <code>x</code> , and only specified columns will be processed. See details.
<code>only_numeric</code>	If TRUE and if <code>x</code> is a data.frame, then only columns of type numeric will be processed. Otherwise all columns will be processed (e.g. also in the case that <code>x</code> is a matrix).

**Details**

Truncation is processed as follows:

1. Compute the 1st and 3rd quartile  $q1$  /  $q3$  of variables in `x`.
2. Multiply these quantities by values in `truncate_multipliers` to obtain  $L$  and  $U$ . If a value is NA, the corresponding variable is not truncated.
3. Set any value smaller / larger than  $L$  /  $U$  to  $L$  /  $U$ .

Truncation multipliers can be specified in three ways (note that whenever `only_numeric` is set to TRUE, then only numeric columns are affected):

- A single numeric - then all columns will be processed in the same way
- A numeric vector without names - it is assumed that the length can be replicated to the number of columns in `x`, each column is processed by the corresponding value in the vector
- A numeric vector with names - length can differ from the columns in `x` and only the columns for which the names occur in the vector are processed

**Value**

Matrix or data.frame of same dimensions as input.

---

process\_truncate\_by\_threshold

*Truncate columns of datamatrix at specified thresholds*


---

## Description

Truncation based on fixed thresholds to be applied to a dataset. Allows to implement truncation by measures derived from the overall data generating mechanism.

## Usage

```
process_truncate_by_threshold(
  x,
  truncate_lower = NA,
  truncate_upper = NA,
  only_numeric = TRUE
)
```

## Arguments

x	Matrix or Data.frame.
truncate_lower, truncate_upper	Vectors of truncation parameters, i.e. lower and upper thresholds for truncation. Either a single value which is replicated as necessary or of same dimension as ncol(x). If any vector entry is NA, the corresponding column will not be truncated. Truncation at lower and upper thresholds is treated independently. If named, then the names must correspond to columnnames in x, and only specified columns will be processed. See details.
only_numeric	If TRUE and if x is a data.frame, then only columns of type numeric will be processed. Otherwise all columns will be processed (e.g. also in the case that x is a matrix).

## Details

Truncation is defined by setting all values below or above the truncation threshold to the truncation threshold.

Truncation parameters can be specified in three ways (note that whenever only\_numeric is set to TRUE, then only numeric columns are affected):

- A single numeric - then all columns will be processed in the same way
- A numeric vector without names - it is assumed that the length can be replicated to the number of columns in x, each column is processed by the corresponding value in the vector
- A numeric vector with names - length can differ from the columns in x and only the columns for which the names occur in the vector are processed

**Value**

Matrix or data.frame of same dimensions as input.

---

quantile\_functions\_from\_data

*Helper to estimate quantile functions from data for NORTA*

---

**Description**

Helper to estimate quantile functions from data for NORTA

**Usage**

```
quantile_functions_from_data(
  data,
  method_density = "linear",
  n_density = 200,
  method_quantile = "constant",
  probs_quantile = seq(0, 1, 0.01),
  n_small = 10,
  use_quantile = c(),
  ...
)

quantile_function_from_density(
  x,
  method_density = "linear",
  n_density = 200,
  ...
)

quantile_function_from_quantiles(
  x,
  method_quantile = "constant",
  probs_quantile = seq(0, 1, 0.01)
)
```

**Arguments**

data	A matrix or data.frame for which quantile function should be estimated.
method_density	Interpolation method used for density based quantile functions, passed to <a href="#">stats::approxfun</a> . See Details.
n_density	Number of points at which the density is estimated for density based quantile functions, passed to <a href="#">stats::density</a> .

method_quantile	Interpolation method used for quantile based quantile functions, passed to <a href="#">stats::approxfun</a> . See Details.
probs_quantile	Specification of quantiles to be estimated from data for quantile based quantile functions, passed to <a href="#">stats::quantile</a> . See Details.
n_small	An integer giving the number of distinct values below which quantile functions are estimated using <code>quantile_function_from_quantiles()</code> (more suited for categorical data), rather than <code>quantile_function_from_density()</code> .
use_quantile	A vector of names indicating columns for which the quantile function should be estimated using <code>quantile_function_from_quantiles()</code> . Overrides <code>n_small</code> .
...	Passed to <code>quantile_function_from_density()</code> .
x	Single vector representing variable input to <code>quantile_function_from_density()</code> or <code>quantile_function_from_quantiles()</code> .

## Details

The NORTA approach requires the specification of the marginals by quantile functions. This helper estimates those given a dataset automatically and non-parametrically. There are two ways implemented to estimate quantile functions from data.

1. Estimate the quantile function by interpolating the observed quantiles from the data. This is most useful for categorical data, when the interpolation is using a step-function (default). Implemented in `quantile_function_from_quantiles()`.
2. Estimate the quantile function via the the empirical cumulative density function derived from the density of the data. Since the density is only estimated at specific points, any values in between are interpolated linearly (default, other options are possible). This is most useful for continuous data. Implemented in `quantile_function_from_density()`.

## Value

A named list of functions with length `ncol(data)` giving the quantile functions of the input data. Each entry is a function returned from [stats::approxfun](#).

## See Also

[simdesign\\_norta](#)

---

simdesign

*Design specification for simulating datasets*

---

## Description

Stores information necessary to simulate and visualize datasets based on underlying distribution  $Z$ .

**Usage**

```

simdesign(
  generator,
  transform_initial = base::identity,
  n_var_final = -1,
  types_final = NULL,
  names_final = NULL,
  prefix_final = "v",
  process_final = list(),
  name = "Simulation design",
  check_and_infer = TRUE,
  ...
)

```

**Arguments**

<code>generator</code>	Function which generates data from the underlying base distribution. It is assumed it takes the number of simulated observations <code>n_obs</code> as first argument, as all random generation functions in the <b>stats</b> and <b>extraDistr</b> do. Furthermore, it is expected to return a two-dimensional array as output (matrix or data.frame). Alternatively an R object derived from the <code>simdata::simdesign</code> class. See details.
<code>transform_initial</code>	Function which specifies the transformation of the underlying dataset Z to final dataset X. See details.
<code>n_var_final</code>	Integer, number of columns in final datamatrix X. Can be inferred when <code>check_and_infer</code> is TRUE.
<code>types_final</code>	Optional vector of length equal to <code>n_var_final</code> (set by the user or inferred) and hence number of columns of final dataset X. Allowed entries are "logical", "factor" and "numeric". Stores the type of the columns of X. If not specified by, inferred if <code>check_and_infer</code> is set to TRUE.
<code>names_final</code>	NULL or character vector with variable names for final dataset X. Length needs to equal the number of columns of X. Overrides other naming options. See details.
<code>prefix_final</code>	NULL or prefix attached to variables in final dataset X. Overridden by <code>names_final</code> argument. Set to NULL if no prefixes should be added. See details.
<code>process_final</code>	List of lists specifying post-processing functions applied to final datamatrix X before returning it. See <a href="#">do_processing</a> .
<code>name</code>	Character, optional name of the simulation design.
<code>check_and_infer</code>	If TRUE, then the simulation design is tested by simulating 5 observations using <a href="#">simulate_data</a> . If everything works without error, the variables <code>n_var_final</code> and <code>types_final</code> will be inferred from the results if not already set correctly by the user.
<code>...</code>	Further arguments are directly stored in the list object to be passed to <a href="#">simulate_data</a> .

## Details

The `simdesign` class should be used in the following workflow:

1. Specify a design template which will be used in subsequent data generating / visualization steps.
2. Sample / visualize datamatrix following template (possibly multiple times) using [simulate\\_data](#).
3. Use sampled datamatrix for simulation study.

For more details on generators and transformations, please see the documentation of [simulate\\_data](#).

For details on post-processing, please see the documentation of [do\\_processing](#).

## Value

List object with class attribute "simdesign" (S3 class) containing the following entries (if no further information given, entries are directly saved from user input):

```
generator
name
transform_initial
n_var_final
types_final
names_final
process_final
entries for further information as passed by the user
```

## Naming of variables

If `check_and_infer` is set to `TRUE`, the following procedure determines the names of the variables:

1. use `names_final` if specified and of correct length
2. otherwise, use the names of `transform_initial` if present and of correct length
3. otherwise, use `prefix_final` to prefix the variable number if not `NULL`
4. otherwise, use names from dataset as generated by the generator function

## Simulation Templates

This class is intended to be used as a template for simulation designs which are based on specific underlying distributions. All such a template needs to define is the `generator` function and its construction and pass it to this function along with the other arguments. See [simdesign\\_mvtnorm](#) for an example.

## See Also

[simdesign\\_mvtnorm](#), [simulate\\_data](#), [simulate\\_data\\_conditional](#)



## Examples

```
generator <- function(n) mvtnorm::rmvnorm(n, mean = 0)
sim_design <- simdesign(generator)
simulate_data(sim_design, 10, seed = 19)
```

---

simdesign_discunif	<i>Uniform disc sampling design specification</i>
--------------------	---

---

## Description

Provides 2-dimensional points, spread uniformly over disc, or partial disc segment (i.e. a circle, or ring, or ring segment). Useful for e.g. building up clustering exercises.

## Usage

```
simdesign_discunif(
  r_min = 0,
  r_max = 1,
  angle_min = 0,
  angle_max = 2 * pi,
  name = "Uniform circle simulation design",
  ...
)
```

## Arguments

<code>r_min</code>	Minimum radius of points.
<code>r_max</code>	Maximum radius of points.
<code>angle_min</code>	Minimum angle of points (between 0 and 2pi).
<code>angle_max</code>	Maximum angle of points (between 0 and 2pi).
<code>name</code>	Character, optional name of the simulation design.
<code>...</code>	Further arguments are passed to the <a href="#">simdesign</a> constructor.

## Details

The distribution of points on a disk depends on the radius - the farther out, the more area the points need to cover. Thus, simply sampling two uniform values for radius and angle will not work. See references.

**Value**

List object with class attribute "simdesign\_discunif" (S3 class), inheriting from "simdesign". It contains the same entries as a [simdesign](#) object but in addition the following entries:

```
r_min
r_max
angle_min
angle_max
```

**References**

<https://mathworld.wolfram.com/DiskPointPicking.html>

**Examples**

```
disc_sampler <- simdesign_discunif()
plot(simulate_data(disc_sampler, 1000, seed = 19))

ring_segment_sampler <- simdesign_discunif(r_min = 0.5, angle_min = 0.5 * pi)
plot(simulate_data(ring_segment_sampler, 1000, seed = 19))

circle_sampler <- simdesign_discunif(r_min = 1)
plot(simulate_data(circle_sampler, 1000, seed = 19))
```

---

simdesign_mvtnorm	<i>Multivariate normal design specification</i>
-------------------	---

---

**Description**

Stores information necessary to simulate and visualize datasets based on underlying distribution multivariate normal distribution Z.

**Usage**

```
simdesign_mvtnorm(
  relations_initial,
  mean_initial = 0,
  sd_initial = 1,
  is_correlation = TRUE,
  method = "svd",
  name = "Multivariate-normal based simulation design",
  ...
)
```

**Arguments**

<code>relations_initial</code>	Correlation / Covariance matrix of the initial multivariate Normal distribution Z.
<code>mean_initial</code>	Vector of mean values of the initial multivariate Normal distribution Z. Dimension needs to correspond to dimension of relations.
<code>sd_initial</code>	Vector of standard deviations of the initial multivariate Normal distribution Z. Dimension needs to correspond to dimension of relations. Overridden by square root of diagonal elements of relations if <code>is_correlation</code> is FALSE.
<code>is_correlation</code>	If TRUE, then relations specifies a correlation matrix (default, this type of specification is usually more natural than specifying a covariance matrix). Otherwise, relations specifies a covariance matrix whose square root diagonal elements override <code>sd_initial</code> .
<code>method</code>	method argument of <code>mvtnorm::rmvnorm</code> .
<code>name</code>	Character, optional name of the simulation design.
<code>...</code>	Further arguments are passed to the <code>simdesign</code> constructor.

**Details**

This S3 class implements a simulation design based on an underlying multivariate normal distribution by creating a generator function based on `mvtnorm::rmvnorm`.

**Value**

List object with class attribute "simdesign\_mvtnorm" (S3 class), inheriting from "simdesign". It contains the same entries as a `simdesign` object but in addition the following entries:

```
mean_initial
sd_initial
cor_initial Initial correlation matrix of multivariate normal distribution
```

**Data Generation**

Data will be generated by `simulate_data` using the following procedure:

1. The underlying data matrix Z is sampled from a multivariate Normal distribution (number of dimensions specified by dimensions of relations).
2. Z is then transformed into the final dataset X by applying the `transform_initial` function to Z.
3. X is post-processed if specified.

**Note**

Note that relations specifies the correlation / covariance of the underlying Normal data Z and thus does not directly translate into correlations between the variables of the final datamatrix X.

**See Also**

`simdesign`, `simulate_data`, `simulate_data_conditional`, `plot_cor_network.simdesign_mvtnorm`

---

simdesign_norta	<i>NORTA-based design specification</i>
-----------------	---

---

## Description

Stores information necessary to simulate datasets based on the NORTA procedure (Cario and Nelson 1997).

## Usage

```
simdesign_norta(
  cor_target_final = NULL,
  cor_initial = NULL,
  dist = list(),
  tol_initial = 0.001,
  n_obs_initial = 10000,
  seed_initial = 1,
  conv_norm_type = "0",
  method = "svd",
  name = "NORTA based simulation design",
  ...
)
```

## Arguments

<code>cor_target_final</code>	Target correlation matrix for simulated datasets. At least one of <code>cor_target_final</code> or <code>cor_initial</code> must be specified.
<code>cor_initial</code>	Correlation matrix for underlying multivariate standard normal distribution on which the final data is based on. At least one of <code>cor_target_final</code> or <code>cor_initial</code> must be specified. If <code>NULL</code> , then <code>cor_initial</code> will be numerically optimized by simulation for the NORTA procedure using <code>cor_target_final</code> .
<code>dist</code>	List of functions of marginal distributions for simulated variables. Must have the same length as the specified correlation matrix ( <code>cor_target_final</code> and / or <code>cor_initial</code> ), and the order of the entries must correspond to the variables in the correlation matrix. See details for the specification of the marginal distributions.
<code>tol_initial</code>	If <code>cor_initial</code> is numerically optimized, specifies the tolerance for the difference to the target correlation <code>cor_target_final</code> . Parameter passed to <a href="#">optimize_cor_for_pair</a> .
<code>n_obs_initial</code>	If <code>cor_initial</code> is numerically optimized, specifies the number of draws in simulation during optimization used to estimate correlations. Parameter passed to <a href="#">optimize_cor_for_pair</a> .
<code>seed_initial</code>	Seed used for draws of the initial distribution used during optimization to estimate correlations.

conv_norm_type	If <code>cor_initial</code> is numerically optimized and found not to be a proper correlation matrix (i.e. not positive-definite), specifies the metric used to find the nearest positive-definite correlation matrix. Parameter passed to <code>Matrix::nearPD</code> ( <code>conv.norm.type</code> ), see there for details.
method	method argument of <code>mvtnorm::rmvnorm</code> .
name	Character, optional name of the simulation design.
...	Further arguments are passed to the <code>simdesign</code> constructor.

### Details

This S3 class implements a simulation design based on the NORmal-To-Anything (NORTA) procedure by Cario and Nelson (1997). See the corresponding NORTA vignette for usage examples how to approximate real datasets.

### Value

List object with class attribute "simdesign\_norta" (S3 class), inheriting from "simdesign". It contains the same entries as a `simdesign` object but in addition the following entries:

```
cor_target_final
cor_initial  Initial correlation matrix of multivariate normal distribution
dist
tol_initial
n_obs_initial
conv_norm_type
method
```

### Data Generation

Data will be generated using the following procedure:

1. An underlying data matrix  $Z$  is sampled from a multivariate standard Normal distribution with correlation structure given by `cor_initial`.
2.  $Z$  is then transformed into a dataset  $X$  by applying the functions given in `dist` to the columns of  $Z$ . The resulting dataset  $X$  will then have the desired marginal distributions, and approximate the target correlation `cor_target_final`, if specified.
3.  $X$  is further transformed by the transformation `transform_initial` (note that this may affect the correlation of the final dataset and is not respected by the optimization procedure), and post-processed if specified.

### Marginal distributions

A list of functions `dist` is used to define the marginal distributions of the variables. Each entry must be a quantile function, i.e. a function that maps  $[0, 1]$  to the domain of a probability distribution. Each entry must take a single input vector, and return a single numeric vector. Examples for acceptable entries include all standard quantile functions implemented in R (e.g. `qnorm`, `qbinom`,

...), user defined functions wrapping these (e.g. `function(x) = qnorm(x, mean = 10, sd = 4)`), or empirical quantile functions. The helper function [quantile\\_functions\\_from\\_data](#) can be used to automatically estimate empirical quantile functions from a given data to reproduce it using the NORTA approach. See the example in the NORTA vignette of this package for workflow details.

### Target correlations

Not every valid correlation matrix (i.e. symmetric, positive-definite matrix with elements in  $[-1, 1]$  and unity diagonal) for a number of variables is feasible for given desired marginal distributions (see e.g. Ghosh and Henderson 2003). Therefore, if `cor_target_final` is specified as target correlation, this class optimises `cor_initial` in such a way, that the final simulated dataset has a correlation which approximates `cor_target_final`. However, the actual correlation in the end may differ if `cor_target_final` is infeasible for the given specification, or the NORTA procedure cannot exactly reproduce the target correlation. In general, however, approximations should be acceptable if target correlations and marginal structures are derived from real datasets. See e.g. Ghosh and Henderson 2003 for the motivation why this works.

### References

Cario, M. C. and Nelson, B. L. (1997) *Modeling and generating random vectors with arbitrary marginal distributions and correlation matrix*. Technical Report, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, Illinois.

Ghosh, S. and Henderson, S. G. (2003) *Behavior of the NORTA method for correlated random vector generation as the dimension increases*. ACM Transactions on Modeling and Computer Simulation.

### See Also

[simdesign](#), [simulate\\_data](#), [simulate\\_data\\_conditional](#), [quantile\\_functions\\_from\\_data](#)

---

simulate_data	<i>Simulate design matrix</i>
---------------	-------------------------------

---

### Description

Generate simulated dataset based on transformation of an underlying base distribution.

### Usage

```
simulate_data(generator, ...)

## Default S3 method:
simulate_data(
  generator = function(n) matrix(rnorm(n)),
  n_obs = 1,
  transform_initial = base::identity,
  names_final = NULL,
```

```

    prefix_final = NULL,
    process_final = list(),
    seed = NULL,
    ...
)

## S3 method for class 'simdesign'
simulate_data(
  generator,
  n_obs = 1,
  seed = NULL,
  apply_transformation = TRUE,
  apply_processing = TRUE,
  ...
)
```

### Arguments

generator	Function which generates data from the underlying base distribution. It is assumed it takes the number of simulated observations <code>n_obs</code> as first argument, as all random generation functions in the <b>stats</b> and <b>extraDistr</b> do. Furthermore, it is expected to return a two-dimensional array as output (matrix or data.frame). Alternatively an R object derived from the <code>simdata::simdesign</code> class. See details.
...	Further arguments passed to generator function.
n_obs	Number of simulated observations.
transform_initial	Function which specifies the transformation of the underlying dataset Z to final dataset X. See details.
names_final	NULL or character vector with variable names for final dataset X. Length needs to equal the number of columns of X. Overrides other naming options. See details.
prefix_final	NULL or prefix attached to variables in final dataset X. Overridden by <code>names_final</code> argument. Set to NULL if no prefixes should be added. See details.
process_final	List of lists specifying post-processing functions applied to final datamatrix X before returning it. See <a href="#">do_processing</a> .
seed	Set random seed to ensure reproducibility of results.
apply_transformation	This argument can be set to FALSE to override the information stored in the passed <code>simdesign</code> object and not transform and process data. Thus, the raw data from the design generator is returned. This can be useful for debugging purposes.
apply_processing	This argument can be set to FALSE to override the information stored in the passed <code>simdesign</code> object and not transform and process data after the initial data is transformed. This can be useful for debugging purposes.

## Details

Data is generated using the following procedure:

1. An underlying dataset  $Z$  is sampled from some distribution. This is done by a call to the generator function.
2.  $Z$  is then transformed into the final dataset  $X$  by applying the transform function to  $Z$ .
3.  $X$  is post-processed if specified (e.g. truncation to avoid outliers).

## Value

Data.frame or matrix with `n_obs` rows for simulated dataset  $X$ .

## Methods (by class)

- `simulate_data(default)`: Function to be used if no `simdesign` S3 class is used.
- `simulate_data(simdesign)`: Function to be used with `simdesign` S3 class.

## Generators

The generator function which is either passed directly, or via a `simdata::simdesign` object, is assumed to provide the same interface as the random generation functions in the **R** `stats` and **extraDistr** packages. Specifically, that means it takes the number of observations as first argument. All further arguments can be set via passing them as named argument to this function. It is expected to return a two-dimensional array (matrix or data.frame) for which the number of columns can be determined. Otherwise the `check_and_infer` step will fail.

## Transformations

Transformations should be applicable to the output of the generator function (i.e. take a data.frame or matrix as input) and output another data.frame or matrix. A convenience function `function_list` is provided by this package to specify transformations as a list of functions, which take the whole datamatrix  $Z$  as single argument and can be used to apply specific transformations to the columns of that matrix. See the documentation for `function_list` for details.

## Post-processing

Post-processing the datamatrix is based on `do_processing`.

## Naming of variables

Variables are named by `names_final` if not NULL and of correct length. Otherwise, if `prefix_final` is not NULL, it is used as prefix for variable numbers. Otherwise, variables names remain as returned by the generator function.

## Note

This function is best used in conjunction with the `simdesign` S3 class or any template based upon it, which facilitates further data visualization and conveniently stores information as a template for simulation tasks.



**See Also**

[simdesign](#), [simdesign\\_mvtnorm](#), [simulate\\_data\\_conditional](#), [do\\_processing](#)

**Examples**

```
generator <- function(n) mvtnorm::rmvnorm(n, mean = 0)
simulate_data(generator, 10, seed = 24)
```

---

`simulate_data_conditional`

*Simulate data which satisfies certain conditions*

---

**Description**

Generate simulated dataset based on transformation of an underlying base distribution while checking that certain conditions are met.

**Usage**

```
simulate_data_conditional(
  generator,
  n_obs = 1,
  reject = function(x) TRUE,
  reject_max_iter = 10,
  on_reject = "ignore",
  return_tries = FALSE,
  seed = NULL,
  ...
)
```

**Arguments**

<code>generator</code>	Function which generates data from the underlying base distribution. It is assumed it takes the number of simulated observations <code>n_obs</code> as first argument, as all random generation functions in the <b>stats</b> and <b>extraDistr</b> do. Furthermore, it is expected to return a two-dimensional array as output (matrix or data.frame). See details.
<code>n_obs</code>	Number of simulated observations.
<code>reject</code>	Function which takes a matrix or data.frame <code>X</code> as single input and outputs TRUE or FALSE. Specifies when a simulated final datamatrix <code>X</code> should be rejected. Functions must output TRUE if condition IS NOT met / FALSE if condition IS met and matrix can be accepted. Intended to be used with <a href="#">function_list</a> . See details.
<code>reject_max_iter</code>	Integer > 0. In case of rejection, how many times should a new datamatrix be simulated until the conditions in <code>reject</code> are met?

on_reject	If "stop", an error is returned if after reject_max_iter times no suitable data-matrix X could be found. If "current", the current datamatrix is returned, regardless of the conditions in reject. Otherwise, NULL is returned. In each case a warning is reported.
return_tries	If TRUE, then the function also outputs the number of tries necessary to find a dataset fulfilling the condition. Useful to record to assess the possible bias of the simulated datasets. See Value.
seed	Set random seed to ensure reproducibility of results. See Note below.
...	All further parameters are passed to <a href="#">simulate_data</a> .

### Details

For details on generating, transforming and post-processing datasets, see [simulate\\_data](#). This function simulates data conditional on certain requirements that must be met by the final datamatrix X. This checking is conducted on the output of `simulate_data` (i.e. also includes possible post-processing steps).

### Value

Data.frame or matrix with n\_obs rows for simulated dataset X if all conditions are met within the iteration limit. Otherwise NULL.

If return\_tries is TRUE, then the output is a list with the first entry being the data.frame or matrix as described above, and the second entry (n\_tries) giving a numeric with the number of tries necessary to find the returned dataset.

### Rejecting Datasets

Examples for restrictions include variance restrictions (e.g. no constant columns which could happen due to extreme transformations of the initial gaussian distribution Z), ensuring a sufficient number of observations in a given class (e.g. certain binary variables should have at least x\ multicollinearity (e.g. X must have full column rank). If reject evaluates to FALSE, the current data-matrix X is rejected. In case of rejection, new datasets can be simulated until the conditions are met or a given maximum iteration limit is hit (reject\_max\_iter), after which the latest datamatrix is returned or an error is reported.

### Rejection Function

The reject function should take a single input (a data.frame or matrix) and output TRUE if the dataset is to be rejected or FALSE if it is to be accepted. This package provides the [function\\_list](#) convenience function which allows to easily create a rejection function which assesses several conditions on the input dataset by simply passing individual test functions to `function_list`. Such test function templates are found in [is\\_collinear](#) and [contains\\_constant](#). See the example below.

### Note

Seeding the random number generator is tricky in this case. The seed can not be passed to `simulate_data` but is set before calling it, otherwise the random number generation is the same for each of the tries. This means that the seed used to call this function might not be the seed corresponding to the returned dataset.

**See Also**

[simdesign](#), [simulate\\_data](#), [function\\_list](#), [is\\_collinear](#), [contains\\_constant](#)

**Examples**

```
dsgn <- simdesign_mvtnorm(diag(5))
simulate_data_conditional(dsgn, 10,
  reject = function_list(is_collinear, contains_constant),
  seed = 18)
```

# Index

`apply_array`, [2](#)  
`as_function_list`, [3](#)  
  
`base::do.call`, [7](#)  
  
`colapply_functions`, [3](#)  
`contains_constant`, [4](#), [34](#), [35](#)  
`cor_from_upper`, [5](#), [6](#)  
`cor_to_cov`, [5](#)  
`cor_to_upper`, [5](#), [6](#)  
  
`data.frame`, [8](#), [9](#)  
`do_processing`, [6](#), [23](#), [24](#), [31–33](#)  
  
`estimate_final_correlation`, [7](#), [17](#), [18](#)  
  
`function_list`, [3](#), [8](#), [10](#), [11](#), [32–35](#)  
  
`get_from_function_list`, [9](#), [10](#)  
`get_names_from_function_list`, [9](#), [10](#)  
  
`igraph.plotting`, [16](#)  
`igraph::plot`, [16](#), [17](#)  
`igraph::tkplot`, [16–18](#)  
`is_collinear`, [11](#), [34](#), [35](#)  
`is_cor_matrix`, [11](#)  
  
`Matrix::nearPD`, [13](#), [29](#)  
`mvtnorm::rmvnorm`, [27](#), [29](#)  
  
`optimize_cor_for_pair`, [12](#), [13](#), [28](#)  
`optimize_cor_mat`, [13](#)  
  
`par`, [16](#)  
`partial`, [14](#)  
`plot_cor_network`, [15](#), [17](#), [18](#)  
`plot_cor_network.simdesign_mvtnorm`, [17](#),  
[27](#)  
`plot_estimated_cor_network`, [17](#), [17](#)  
`process_truncate_by_iqr`, [7](#), [18](#)  
`process_truncate_by_threshold`, [7](#), [20](#)  
  
`quantile_function_from_density`  
    (`quantile_functions_from_data`),  
    [21](#)  
`quantile_function_from_quantiles`  
    (`quantile_functions_from_data`),  
    [21](#)  
`quantile_functions_from_data`, [21](#), [30](#)  
  
`scale`, [7](#)  
`simdesign`, [8](#), [22](#), [25–27](#), [29](#), [30](#), [32](#), [33](#), [35](#)  
`simdesign_discunif`, [25](#)  
`simdesign_mvtnorm`, [15](#), [17](#), [24](#), [26](#), [33](#)  
`simdesign_norta`, [13](#), [14](#), [22](#), [28](#)  
`simulate_data`, [8](#), [23](#), [24](#), [27](#), [30](#), [30](#), [34](#), [35](#)  
`simulate_data_conditional`, [9](#), [24](#), [27](#), [30](#),  
    [33](#), [33](#)  
`stats::approxfun`, [21](#), [22](#)  
`stats::cor`, [8](#), [18](#)  
`stats::density`, [21](#)  
`stats::quantile`, [22](#)  
`stats::uniroot`, [12](#), [14](#)