Package 'mcmodule'

May 25, 2026

Title Modular Monte Carlo Risk Analysis

Version 1.3.0

Description Framework for building modular Monte Carlo risk analysis models. It extends the capabilities of 'mc2d' to facilitate working with multiple risk pathways, variates and scenarios. It provides tools to organize risk analysis in independent flexible modules, align multivariate mcnodes, automate the creation of mcnodes, visualise model structure, assess convergence, and perform sensitivity analysis. For more details see Ciria (2026) <https://nataliaciria.com/mcmodule/>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

URL https://nataliaciria.com/mcmodule/, https://github.com/NataliaCiria/mcmodule

BugReports https://github.com/NataliaCiria/mcmodule/issues

Suggests knitr, rmarkdown, testthat (>= 3.0.0), visNetwork, igraph, ggplot2, sensitivity, sensobol

Config/testthat/edition 3

Imports dplyr, stats, utils

Depends mc2d, R (>= 3.5)

LazyData true

VignetteBuilder knitr

NeedsCompilation no

Author Natalia Ciria [aut, cre, cph] (ORCID: <https://orcid.org/0009-0006-2669-6634>), Alberto Allepuz [ths] (ORCID: <https://orcid.org/0000-0003-3518-1991>), Giovanna Ciaravino [ths] (ORCID: <https://orcid.org/0000-0002-5796-8093>)

Maintainer Natalia Ciria <nataliaciria@hotmail.com>

Repository CRAN

Date/Publication 2026-05-25 15:00:15 UTC

Contents

`add_prefix` *Add Prefix to mcnode Names*

Description

Adds a prefix to node names and their input references. Existing prefixes are preserved to avoid breaking references.

Usage

```
add_prefix(mcmodule, prefix = NULL, rewrite_module = NULL)
```

Arguments

`mcmodule`	(mcmodule or list). mcmodule object or node_list to prefix.
`prefix`	(character, optional). Prefix to add to node names; defaults to mcmodule name. Default: NULL.
`rewrite_module`	(character, optional). Module name to rewrite prefixes for. Default: NULL.

Value

The mcmodule with prefixed node names.

Examples

```
print(names(imports_mcmodule$node_list))
imports_mcmodule_prefix<-purchase <- add_prefix(imports_mcmodule)
print(names(imports_mcmodule_prefix$node_list))
```

agg_totals *Aggregate mcnode Values Across Groups*

Description

Aggregates node values across grouping variables using various methods (combined probability, sum, mean, or automatic selection). Returns an updated mcmodule with new aggregated node.

Usage

```
agg_totals(
  mcmodule,
  mc_name,
  agg_keys = NULL,
  agg_suffix = NULL,
  prefix = NULL,
  name = NULL,
  summary = TRUE,
  keep_variates = FALSE,
  agg_func = NULL
)
```

Arguments

mcmodule	(mcmodule object). Module containing node list and data.
mc_name	(character). Name of node to aggregate.
agg_keys	(character vector, optional). Column names for grouping. If NULL, defaults to "scenario_id". Default: NULL.
agg_suffix	(character, optional). Suffix for aggregated node name. Default: "agg".
prefix	(character, optional). Prefix for output node name. Default: NULL.
name	(character, optional). Custom name for output node. Default: NULL.
summary	(logical). If TRUE, include summary statistics. Default: TRUE.
keep_variates	(logical). If TRUE, preserve individual variate values. Default: FALSE.
agg_func	(character, optional). Aggregation method: "prob" (combined probability), "sum", "avg", or NULL (automatic). Default: NULL.

Details

If sample-design nodes are aggregated, the resulting node will be equal to the original node, but with the "agg_total" type and summary statistics added.

Value

mcmodule with new aggregated node added

Examples

```
imports_mcmodule <- agg_totals(
  imports_mcmodule, "no_detect",
  agg_keys = c("scenario_id", "pathogen")
)
print(imports_mcmodule$node_list$no_detect_agg$summary)
```

`animal_imports` *Example Animal Import Data*

Description

A dataset containing information about animal imports from three different regions.

Usage

```
animal_imports
```

Format

animal_imports:

A data frame with 3 rows and 4 columns:

origin Region of origin (nord, south, east)

farms_n Number of farms exporting animals

animals_n_mean Mean number of animals exported per farm

animals_n_sd Standard deviation of animals exported per farm

Source

Simulated data for demonstration purposes

`at_least_one` *Combine Probabilities Assuming Independence*

Description

Combines probabilities of multiple independent events using the formula: P(at least one) = 1 - (1-P(A)) * (1-P(B)) * ... Automatically matches dimensions and keys.

Usage

```
at_least_one(
  mcmodule,
  mc_names,
  name = NULL,
  all_suffix = NULL,
  prefix = NULL,
  summary = TRUE
)
```

Arguments

mcmodule	(mcmodule object). Module containing node list and data frames.
mc_names	(character vector). Node names to combine.
name	(character, optional). Custom name for combined node. If NULL, auto-generated. Default: NULL.
all_suffix	(character). Suffix for auto-generated node name. Default: "all".
prefix	(character, optional). Prefix for output node name. Default: NULL.
summary	(logical). If TRUE, calculate summary statistics. Default: TRUE.

Value

Updated mcmodule with new combined probability node.

Examples

```
module <- list(
  node_list = list(
    p1 = list(
      mcnode = mcstoc(runif,
        min = mcdata(c(0.1, 0.2, 0.3), type = "0", nvariates = 3),
        max = mcdata(c(0.2, 0.3, 0.4), type = "0", nvariates = 3),
        nvariates = 3
      ),
      data_name = "data_x",
      keys = c("category")
    ),
    p2 = list(
      mcnode = mcstoc(runif,
        min = mcdata(c(0.5, 0.6, 0.7), type = "0", nvariates = 3),
        max = mcdata(c(0.6, 0.7, 0.8), type = "0", nvariates = 3),
        nvariates = 3
      ),
      data_name = "data_y",
      keys = c("category")
    )
  ),
  data = list(
    data_x = data.frame(
```

```
      category = c("A", "B", "C"),
      scenario_id = c("0", "0", "0")
    ),
    data_y = data.frame(
      category = c("B", "B", "B"),
      scenario_id = c("0", "1", "2")
    )
  )
)

module <- at_least_one(module, c("p1", "p2"), name = "p_combined")
print(module$node_list$p_combined$summary)
```

`check_mctable` *Validate and Prepare mctable Data Frame*

Description

Validates that an mctable contains required columns (`mcnode`, `mc_func`), issues warnings for missing columns, and auto-fills missing optional columns with `NA`.

Usage

```
check_mctable(data)
```

Arguments

`data`	(data frame). mctable with `mcnode` column (required) and optionally `mc_func`, `description`, `from_variable`, `sample_space`, `transformation`, and `sensi_variation`. Default: required.

Details

If `mc_func` is missing, all nodes are treated as deterministic (no uncertainty). Optional columns are auto-filled with `NA` if absent. When `sample_space` values are provided, they must use supported formats (`c(...)` or named assignments such as `min = X, max = Y`).

Value

The validated `data` frame with all standard mctable columns present, with missing optional columns filled as `NA`.

`combine_modules` *Combine Two mcmodule Objects*

Description

Combines two mcmodule objects into a single mcmodule by merging their data and node lists.

Usage

```
combine_modules(mcmodule_x, mcmodule_y)
```

Arguments

`mcmodule_x`	(mcmodule object). First module.
`mcmodule_y`	(mcmodule object). Second module.

Value

An mcmodule object with combined data and node lists.

Examples

```
module_x <- list(
  data = list(data_x = data.frame(x = 1:3)),
  node_list = list(
    node1 = list(type = "in_node"),
    node2 = list(type = "out_node")
  ),
  modules = c("module_x"),
  exp = quote({node2 <- node1 * 2})
)

module_y <- list(
  data = list(data_y = data.frame(y = 4:6)),
  node_list = list(node3 = list(type = "out_node")),
  modules = c("module_y"),
  exp = quote({node3 <- node1 + node2})
)

module_xy <- combine_modules(module_x, module_y)
```

`create_mcnodes` *Create mcnodes from Data and Configuration Table*

Description

Creates mcnodes based on mctable specifications and input data. Applies transformations and generates mcnodes in the calling environment.

Usage

```
create_mcnodes(data, mctable = set_mctable(), envir = parent.frame())
```

Arguments

`data`	(data frame). Input data containing variables for mcnode creation.
`mctable`	(data frame). Configuration table with columns: mcnode, mc_func, transformation, from_variable.
`envir`	(environment, optional). Environment where nodes are created. Default: parent.frame().

Value

NULL (invisibly). mcnodes created in `envir`.

Examples

```
create_mcnodes(
  data = imports_data,
  mctable = imports_mctable
)
```

`eval_module` *Evaluate Monte Carlo Model Expressions*

Description

Evaluates a model expression or list of expressions to produce an mcmodule object containing simulation results and metadata. Expression may use `mc2d::mcstoc()` and `mc2d::mcdata()` to create nodes inline; nvariates is automatically inferred from the data unless `sample_design` is provided.

Usage

```
eval_module(
  exp,
  data = NULL,
  param_names = NULL,
  prev_mcmodule = NULL,
  summary = FALSE,
  mctable = set_mctable(),
  data_keys = set_data_keys(),
  match_keys = NULL,
  keys = NULL,
  overwrite_keys = NULL,
  sample_design = set_sample_design(),
  if_not_sampled = c("median", "mean", "max", "min"),
  use_variation = NULL
)
```

Arguments

exp	(language or list). Model expression or list of expressions to evaluate.
data	(data frame). Input data; number of rows determines nvariates for mc2d::mcstoc()/mc2d::mcdata() in expressions when sample_design is not provided. With sample_design, inline nvariates is removed and defaults to 1. Default: NULL (can only be NULL if sample_design with all inputs is provided).
param_names	(named character vector, optional). Names to rename parameters. Default: NULL.
prev_mcmodule	(mcmodule or list, optional). Previous module(s) for dependent calculations. Default: NULL.
summary	(logical). If TRUE, calculate summary statistics for output nodes. Default: FALSE.
mctable	(data frame). Reference table for mcnodes with mcnode and mc_func columns. If NULL or not provided, nodes matching data column names are automatically created. If sample_design is provided, required inputs present in sample_design are created from it even when absent from mctable. Default set_mctable().
data_keys	(list). Data structure and keys for input data. Default: set_data_keys().
match_keys	(character vector, optional). Keys to match prev_mcmodule mcnodes with current data. Default: NULL.
keys	(character vector, optional). Explicit keys for input data. Default: NULL.
overwrite_keys	(logical or NULL). If NULL (default), becomes TRUE when data_keys is NULL or empty; otherwise FALSE.
sample_design	(matrix, data frame, or list, optional). Sampling design used to create input nodes via matrix_to_mcnodes(). Accepts a matrix/data frame or a list with element X (typically output of sensitivity::sensitivity functions). Columns matching expression input nodes are created from this matrix. Defaults to set_sample_design().

`if_not_sampled` (character). How to fill input nodes that are required by the expression but do not appear as columns in `sample_design`. A fixed value is computed from `mctable$sample_space` and replicated across all samples. Options are `"median"` (default), `"mean"`, `"max"`, and `"min"`.

`use_variation` (character vector, optional). mcnode names to apply `sensi_variation` expression from `mctable` before node creation. Default: NULL.

Details

- `mc2d::mcstoc()` and `mc2d::mcdata()` may be used directly inside model expressions. When these are used you should NOT explicitly supply nvariates, nvariates will be inferred automatically as the number of rows in the input `data`. If `sample_design` is provided, any inline `nvariates` argument is removed and the default `nvariates = 1` is used for inline nodes. Other arguments are preserved, for example specify `type = "0"` when providing data without variability/uncertainty (see `mc2d::mcdata()` and `mc2d::mcstoc()`).
- By design, mcmodule supports type = "V" (the default, with variability) and type = "0" (no variability) nodes. Expressions that specify other node types ("U" or "VU") are not fully supported and downstream compatibility is not guaranteed.
- An explicit `mctable` is optional but highly recommended. If no mctable is provided, any model nodes that match column names in `data` will be built from the data. If a `mctable` is provided and a node is not found there but exists as a data column, a warning will be issued and the node will be created from the data column. When `sample_design` is provided, required inputs that match `sample_design` column names are also created from `sample_design` even if they are not listed in `mctable`.
- Within expressions reference input mcnodes by their bare names (e.g. column1). Do not use `data$column1` or `data["column1"]`.

Value

An mcmodule object (list) with elements:

- data: List containing input data frames.
- exp: List of evaluated expressions.
- node_list: Named list of mcnode objects with metadata.

Examples

```
# Basic usage with single expression
# Build a quoted expression using mcnodes defined in mctable or built with
# mcstoc()/mcdata within the expression (do NOT set nvariates, it is
# inferred from nrow(data) when evaluated by eval_module() unless
# sample_design is provided, in which case inline nvariates defaults to 1).
expr_example <- quote({
  # Within-herd prevalence (assigned from a pre-built mcnode w_prev)
  infected <- w_prev

  # Estimate of clinic sensitivity
  clinic_sensi <- mcstoc(runif, min = 0.6, max = 0.8)
```

```
  # Probability an infected animal is tested in origin and not detected
  false_neg <- infected * test_origin * (1 - test_sensi) * (1 - clinic_sensi)

  # Probability an infected animal is not tested and not detected
  no_test <- infected * (1 - test_origin) * (1 - clinic_sensi)

  # no_detect: total probability an infected animal is not detected
  no_detect <- false_neg + no_test
})

# Evaluate
eval_module(
  exp = expr_example,
  data = imports_data,
  mctable = imports_mctable,
  data_keys = imports_data_keys
)
```

`get_edge_table` *Generate Edge Table for Network Visualisation*

Description

Creates a data frame containing edge relationships between nodes in a Monte Carlo module network. Each row represents a directed edge from one node to another.

Usage

```
get_edge_table(mcmodule, inputs = FALSE)
```

Arguments

`mcmodule`	(mcmodule object). Module containing node relationships.
`inputs`	(logical). If TRUE, include non-node inputs (datasets, dataframes, and columns). Default: FALSE.

Value

A data frame with columns `node_from` and `node_to` representing network edges.

Examples

```
edge_table <- get_edge_table(imports_mcmodule)
```

get_mcmodule_nodes *Get Nodes from Monte Carlo Module*

Description

Retrieves nodes from a Monte Carlo module and assigns them to the parent environment

Usage

```
get_mcmodule_nodes(mcmodule, mc_names = NULL, envir = parent.frame())
```

Arguments

mcmodule	An mcmodule or mcnode_list object
mc_names	Optional vector of node names to retrieve
envir	Environment where MC nodes will be created (default: parent.frame())

Value

A subset of the node list containing requested nodes

get_node_list *Create Node List from Model Expression*

Description

Creates a list of nodes based on a given model expression, handling input, output, and previous nodes with their properties and relationships.

Usage

```
get_node_list(
  exp,
  param_names = NULL,
  mctable = set_mctable(),
  data_keys = set_data_keys(),
  keys = NULL
)
```

Arguments

exp	An R expression containing model calculations
param_names	Optional named vector for parameter renaming
mctable	Reference table for mcnodes, defaults to set_mctable()
data_keys	Data structure and keys, defaults to set_data_keys()
keys	Optional explicit keys for the input data (character vector)

Value

A list of class "mcnode_list" containing node information

---

get_node_table *Generate Node Table for Network Visualisation*

Description

Creates a data frame containing node information from a Monte Carlo module network. Includes node attributes, values, and relationships.

Usage

```
get_node_table(mcmodule, variate = 1, inputs = FALSE)
```

Arguments

mcmodule	(mcmodule object). Module containing node information.
variate	(integer). Which variate to extract. Default: 1.
inputs	(logical). If TRUE, include non-node inputs (datasets, dataframes, and columns). Default: FALSE.

Value

A data frame containing node information and attributes.

Examples

```
node_table <- get_node_table(imports_mcmodule)
```

---

imports_data *Merged Import Data for Risk Assessment*

Description

A dataset combining information about animal imports, pathogen prevalence, and test sensitivity across regions.

Usage

```
imports_data
```

Format

imports_data:
A data frame with 6 rows and 12 columns:

pathogen Pathogen identifier (a or b)
origin Region of origin (nord, south, east)
h_prev_min Minimum herd prevalence value
h_prev_max Maximum herd prevalence value
w_prev_min Minimum within-herd prevalence value
w_prev_max Maximum within-herd prevalence value
farms_n Number of farms exporting animals
animals_n_mean Mean number of animals exported per farm
animals_n_sd Standard deviation of animals exported per farm
test_origin Test used to detect infected animals at origin
test_sensi_min Minimum test sensitivity value
test_sensi_mode Most likely test sensitivity value
test_sensi_max Maximum test sensitivity value

Source

Simulated data for demonstration purposes

`imports_data_keys` *Example Data Keys for Animal Imports Risk Assessment*

Description

A hierarchical data structure containing test sensitivity, animal import, and regional prevalence information, each with defined columns and keys.

Usage

```
imports_data_keys
```

Format

A list with three components:

test_sensitivity List containing column names for test sensitivity data and "pathogen" as key

animal_imports List containing column names for animal import data and "origin" as key

prevalence_region List containing column names for prevalence data with "pathogen" and "origin" as keys

Source

Simulated data for demonstration purposes

imports_exp *Expression for Calculating Import Infection Probability*

Description

A quoted R expression that calculates the probability of importing an infected animal from an infected herd, taking into account testing procedures and accuracy.

Usage

```
imports_exp
```

Format

A quoted R expression containing the following variables:

w_prev Within-herd prevalence

test_origin Probability of testing at origin

test_sensi Test sensitivity

infected Probability of animal being infected

false_neg Probability of false negative test result

no_test Probability of no testing

no_detect Overall probability of non-detection

imports_mcmodule *Example Monte Carlo Module for Animal Imports Risk Assessment*

Description

A list containing simulation results for pathogen testing of animal imports from different origins, including:

- Within-herd prevalence (`w_prev`)
- Test sensitivity (`test_sensi`)
- Test origin probability (`test_origin`)
- Infection probability (`infected`)
- False negative probability (`false_neg`)
- No test probability (`no_test`)
- Non-detection probability (`no_detect`)

Usage

```
imports_mcmodule
```

Format

An mcmodule object with the following components:

data Input data frame with 6 rows and 13 variables

exp Model expressions for calculating probabilities

node_list List of Monte Carlo nodes with simulation results

modules Character vector of module names

Source

Simulated data for demonstration purposes

---

imports_mctable *Example Monte Carlo Input Table for Import Risk Assessment*

Description

A configured table of Monte Carlo nodes used for modeling import risk scenarios, particularly focused on animal disease transmission pathways.

Usage

```
imports_mctable
```

Format

imports_mctable:

A data frame with 7 rows and 7 columns:

mcnode Node identifier used in Monte Carlo simulations

description Human-readable description of what the node represents

mc_func R function used for random number generation (e.g., runif, rnorm, rpert)

from_variable Dependency reference to other variables if applicable

transformation Mathematical transformations applied to the node values

sample_space Allowed values for the input. Use min/max for numeric ranges (e.g., `min=0, max=1`) and explicit vectors for factors/logicals (e.g., `c("a", "b", "c")` or `c(TRUE, FALSE)`)

sensi_variation OAT variation expression using 'value' placeholder

Source

Simulated data for demonstration purposes

keys_match *Match and Align Keys Between Datasets*

Description

Matches and aligns keys between two datasets for downstream operations.

Usage

```
keys_match(x, y, keys_names = NULL, match_scenario = TRUE)
```

Arguments

x	First dataset containing keys to match
y	Second dataset containing keys to match
keys_names	Names of columns to use as matching keys. If NULL, uses common columns
match_scenario	(logical). If TRUE, exclude scenario_id from matching keys. If FALSE, include scenario_id. Default: TRUE.

Value

A list containing:

x	First dataset with group IDs
y	Second dataset with group IDs
xy	Matched datasets with aligned group and scenario IDs

matrix_to_mcnodes *Create mcnodes from Matrix/Data Frame*

Description

Creates one mcdata mcnode per column in X, using ndvar = nrow(X). This is useful when X is a design matrix generated by sensitivity::sensitivity functions or sensobol::sobol_matrices().

Usage

```
matrix_to_mcnodes(X, envir = parent.frame())
```

Arguments

X	(matrix or data frame). Input values with one mcnode per column.
envir	(environment, optional). Environment where nodes are created. Default: parent.frame().

Value

NULL (invisibly). mcnodes created in `envir`.

Examples

```
X <- matrix(c(0.1, 0.2, 0.3, 10, 11, 12), ncol = 2)
colnames(X) <- c("a", "b")
matrix_to_mcnodes(X)
```

---

mcmodule_converg *Analyse Monte Carlo Simulation Convergence*

Description

[Experimental]

Analyses convergence in Monte Carlo simulations by computing standardised and raw differences between consecutive iterations to evaluate stability and convergence of statistical measures.

Usage

```
mcmodule_converg(
  mcmodule,
  from_quantile = 0.95,
  to_quantile = 1,
  conv_threshold = NULL,
  tiny_threshold = NULL,
  print_summary = TRUE,
  progress = FALSE,
  mc_names = NULL
)
```

Arguments

`mcmodule`	(mcmodule object). Module containing simulation results.
`from_quantile`	(numeric). Lower bound quantile for analysis. Default: 0.95.
`to_quantile`	(numeric). Upper bound quantile for analysis. Default: 1.
`conv_threshold`	(numeric, optional). Custom convergence threshold for standardised differences. Default: NULL.
`tiny_threshold`	(numeric). Threshold for identifying negligible differences, even if they don't meet the convergence threshold. Default: 0.0001.
`print_summary`	(logical). If TRUE, print convergence analysis summary. Default: TRUE.
`progress`	(logical). If TRUE, print progress information. Default: FALSE.
`mc_names`	(character vector, optional). Node names to include in analysis. If NULL (default), includes all nodes in the module. Default: NULL.

Details

The function performs the following:

- Calculates convergence statistics for specified quantile range
- Generates diagnostic plots for standardized and raw differences
- Provides detailed convergence summary including:
  - Total nodes analyzed
  - Number and percentage of nodes converged at different thresholds
  - Maximum/minimum deviations
  - List of non-converged nodes (if any)

Value

A data frame with convergence statistics. Each row represents one node. Key columns:

- expression: Expression identifier.
- variate: Variate (data row) identifier.
- node: Node name.
- max_dif_scaled: Maximum standardised difference.
- max_dif: Maximum raw difference.
- conv_01, conv_025, conv_05: Convergence at 1%, 2.5%, 5% thresholds.

Examples

```
## Not run:
results <- mcmodule_converg(mc_results)
results <- mcmodule_converg(mc_results, from_quantile = 0.90, conv_threshold = 0.01)

## End(Not run)
```

---

mcmodule_corr *Calculate Correlation Coefficients Between Inputs and Outputs*

Description

[Experimental]

Computes correlation coefficients between mcmodule inputs and outputs using tornado analysis (from the `mc2d` package). Supports multiple correlation methods and captures warnings generated during calculation.

Usage

```
mcmodule_corr(
  mcmodule,
  output = NULL,
  by_exp = FALSE,
  match_variates = TRUE,
  variates_as_nsv = FALSE,
  mc_names = NULL,
  print_summary = TRUE,
  progress = FALSE,
  method = c("spearman", "kendall", "pearson"),
  use = "all.obs",
  lim = c(0.025, 0.975),
  plot = FALSE
)
```

Arguments

mcmodule	(mcmodule object). Module containing simulation results.
output	(character, optional). Output node name. If NULL (default), uses the last node in mcmodule$node_list. If by_exp = TRUE, uses the last output node per expression. Default: NULL.
by_exp	(logical). If TRUE, calculate correlations by expression output; if FALSE, use global output (last node). Default: FALSE.
match_variates	(logical). If TRUE, match input nodes to output variates when data dimensions differ. Default: TRUE.
variates_as_nsv	(logical). If TRUE, combine all variates into one mc object; if FALSE, analyse each variate separately. See mcmodule_to_mc(). Default: FALSE.
mc_names	(character vector, optional). Node names to include in analysis. If NULL (default), includes all nodes in the module. Default: NULL.
print_summary	(logical). If TRUE, print correlation analysis summary. Default: TRUE.
progress	(logical). If TRUE, print progress information while running. Default: FALSE.
method	(character). Correlation coefficient type: "spearman" (default), "kendall", or "pearson". See stats::cor(). Default: "spearman".
use	(character). Method for handling missing values: "all.obs", "complete.obs", or "pairwise.complete.obs". See stats::cor(). Default: "all.obs".
lim	(numeric vector). Quantiles for credible interval computation (reserved for two-dimensional models). Default: c(0.025, 0.975).
plot	(logical). If TRUE, plots a tornado plot generated from the computed correlation table using mcmodule_tornado(). Default: FALSE.

Value

A data frame with correlation coefficients and metadata. Columns include:

- exp: Expression name
- exp_n: Expression number
- variate: Variate number
- output: Output node names
- input: Input node name
- value: Correlation coefficient value
- strength: Qualitative strength of association (Very strong, Strong, Moderate, Weak, Very weak/None)
- method: Correlation method used (spearman, kendall, or pearson)
- use: Method for handling missing values (passed to the correlation function)
- warnings: Any warnings generated during correlation calculation (if present)
- Additional columns for global keys (e.g., pathogen, origin)

Examples

```
mcmodule <- agg_totals(
  mcmodule = imports_mcmodule,
  mc_name = "no_detect",
  agg_keys = "pathogen"
)
cor_results <- mcmodule_corr(mcmodule)

# Use single method
cor_results_spearman <- mcmodule_corr(mcmodule, method = "spearman")
```

---

mcmodule_dim_check *Check Dimension Compatibility of Monte Carlo Nodes*

Description

[Experimental]

Validates that all mcnodes in a module have compatible dimensions for sensitivity analysis by checking uncertainty and variate dimensions.

Usage

```
mcmodule_dim_check(mcmodule, mc_names = NULL)
```

Arguments

mcmodule	(mcmodule object). Module containing nodes.
mc_names	(character vector, optional). Node names to check. If NULL, checks all nodes. Default: NULL.

Value

A list with: `n_mcnodes` (count), `n_variate` (variate count), `n_uncertainty` (uncertainty simulation count).

`mcmodule_info` *Get Comprehensive Monte Carlo Module Information*

Description

Extracts comprehensive metadata about a Monte Carlo module, including:

- Module composition (raw vs combined modules)
- Input data per expression
- Keys for each variate (data row)
- Global data keys

Usage

```
mcmodule_info(mcmodule)
```

Arguments

`mcmodule`	A Monte Carlo module object

Details

A raw module has a single expression in `mcmodule$exp`. A combined module has multiple expressions in `mcmodule$exp`, each representing a component module that was combined via `combine_modules()`.

For combined modules, module names are recursively extracted up to one level deep. This allows identifying all base modules even in deeply nested combinations.

Value

A list with six elements:

`is_combined`	Logical. TRUE if module is combined, FALSE if raw
`n_modules`	Integer. Number of component modules (1 for raw, >1 for combined)
`module_names`	Character vector. Names of all component modules (recursive)
`module_exp_data`	Data frame with module and expression information, including data_name
`data_keys`	Data frame with keys for each variate, including variate number and data_name
`global_keys`	Character vector of global key names used across the module

Examples

```
# Get comprehensive module information
info <- mcmodule_info(imports_mcmodule)
str(info)

# Access composition information
info$is_combined
info$n_modules
info$module_names

# Access index information
head(info$module_exp_data)
head(info$data_keys)
info$global_keys
```

`mcmodule_tornado` *Plot Tornado-Style Correlation Results Across Variates*

Description

[Experimental]

Creates a tornado-style plot from `mcmodule_corr()` results. For each input node, the plot shows all variate-level correlations as small vertical ticks, a black horizontal range line (min to max), a median marker, and a larger marker at the maximum absolute correlation.

Usage

```
mcmodule_tornado(
  mcmodule = NULL,
  corr_results = NULL,
  output = NULL,
  by_exp = FALSE,
  match_variates = TRUE,
  variates_as_nsv = FALSE,
  print_summary = TRUE,
  progress = FALSE,
  method = c("spearman", "kendall", "pearson"),
  use = "all.obs",
  lim = c(0.025, 0.975),
  colour = "strength"
)
```

Arguments

`mcmodule`	(mcmodule object, optional). Module used to compute correlations when `corr_results` is NULL.

corr_results	(data frame, optional). Output table from mcmodule_corr(). If provided, no new correlation analysis is run. Default: NULL.
output	(character, optional). Output node name. Passed to mcmodule_corr() when corr_results is NULL.
by_exp	(logical). Passed to mcmodule_corr(). Default: FALSE.
match_variates	(logical). Passed to mcmodule_corr(). Default: TRUE.
variates_as_nsv	(logical). Passed to mcmodule_corr(). Default: FALSE.
print_summary	(logical). Passed to mcmodule_corr(). Default: TRUE.
progress	(logical). Passed to mcmodule_corr(). Default: FALSE.
method	(character). Passed to mcmodule_corr(). Default: c("spearman", "kendall", "pearson").
use	(character). Passed to mcmodule_corr(). Default: "all.obs".
lim	(numeric vector). Passed to mcmodule_corr(). Default: c(0.025, 0.975).
colour	(character or logical). Colouring for max absolute points. Default: "strength". If TRUE or "strength", points are coloured by qualitative correlation strength.

Details

mcmodule_tornado() returns a ggplot object. Use mcmodule_corr() when you need the correlation table; use mcmodule_tornado() when you need a plot object that can be further customised. **Interpretation:** In the tornado plot each point (one per variate) shows the correlation between that input and the chosen output across the model variates. The coloured point highlights the variate with the maximum absolute correlation for each input and is used to rank inputs. The black point is the median correlation across variates and the black horizontal line shows the range (minimum to maximum) of correlations for that input. The grey horizontal line connects the maximum-absolute point to the zero-correlation vertical line to facilitate interpretation. Use mcmodule_corr() to inspect the numeric per-variate correlations, the plot is designed to give a compact visual summary.

Value

A ggplot2 object.

---

mcmodule_to_matrices *Convert Monte Carlo Module to Matrices*

Description

Transforms an mcmodule into a list of matrices, with one matrix per variate. Each matrix has uncertainty simulations as rows and mcnodes as columns.

Usage

```
mcmodule_to_matrices(mcmodule, mc_names = NULL)
```

Arguments

mcmodule	(mcmodule object). Module to convert.
mc_names	(character vector, optional). Node names to include. If NULL, includes all nodes. Default: NULL.

Value

A list of matrices (one per variate). Each matrix has uncertainty simulations as rows and mcnodes as columns.

---

mcmodule_to_mc *Convert Monte Carlo Module to* mc2d *Objects*

Description

Converts an mcmodule into one or more mc objects (from the mc2d package). Returns either one mc object per variate or a single mc object with all variates combined into the variability dimension.

Usage

```
mcmodule_to_mc(
  mcmodule,
  mc_names = NULL,
  match = FALSE,
  variates_as_nsv = FALSE
)
```

Arguments

mcmodule	(mcmodule object). Module to convert.
mc_names	(character vector, optional). Node names to include. If NULL, includes all nodes. Default: NULL.
match	(logical, unused). Reserved for future functionality. Default: FALSE.
variates_as_nsv	(logical). If TRUE, combine all variates into a single mc object by multiplying variates by uncertainty simulations in the variability dimension. If FALSE, return one mc object per variate. Default: FALSE.

Value

If variates_as_nsv = FALSE, a list of mc objects (one per variate). If variates_as_nsv = TRUE, a single mc object with all variates combined into the variability dimension. Each mc object is compatible with mc2d package functions.

mcnode_na_rm *Replace NA and Infinite Values in mcnode Objects*

Description

Replaces NA and infinite values in mcnode objects with a specified value.

Usage

```
mcnode_na_rm(mcnode, na_value = 0)
```

Arguments

mcnode	An mcnode object containing NA or infinite values
na_value	Numeric value to replace NA and infinite values (default = 0)

Value

An mcnode object with NA and infinite values replaced by na_value

Examples

```
sample_mcnode <- mcstoc(runif,
               min = mcdata(c(NA, 0.2, -Inf), type = "0", nvariates = 3),
               max = mcdata(c(NA, 0.3, Inf), type = "0", nvariates = 3),
               nvariates = 3
)
# Replace NA and Inf with 0
clean_mcnode <- mcnode_na_rm(sample_mcnode)
```

mcnode_null_rm *Replace NULL mcnode object*

Description

Replaces an mcnode that is not found in the data or in the previous module with a specified value.

Usage

```
mcnode_null_rm(mcnode, null_value = 0)
```

Arguments

mcnode	An mcnode object containing NA or infinite values
null_value	Numeric value to replace NA and infinite values (default = 0)

Value

The mcnode if is found, otherwise the null_value

Examples

```
mcnode_null_rm(unexisting_mcnode)
```

`mctable_bounds` *Extract Morris Bounds From mctable*

Description

Extract the bounds required by `sensitivity::morris()` from an `mctable`.

Usage

```
mctable_bounds(
  mctable = set_mctable(),
  mc_names = NULL,
  if_not_sampled = c("exclude", "median", "mean", "max", "min"),
  transformation = TRUE,
  n_probe = 1000
)
```

Arguments

`mctable`	(data frame). Table containing at least `mcnode` and `sample_space`; may also contain `transformation`. Default: `set_mctable()`.
`mc_names`	(character vector, optional). Node names to include. If `NULL`, all nodes in `mctable$mcnode` are used.
`if_not_sampled`	(character). How to handle nodes not listed in `mc_names` (and nodes with missing or empty `sample_space`): `"exclude"`, `"median"`, `"mean"`, `"max"`, or `"min"`. Default: `"exclude"`.
`transformation`	(logical). Whether to apply `transformation` rules. Default: `TRUE`.
`n_probe`	(integer). Number of probe draws used to approximate bounds when `transformation = TRUE`. Default: 1000.

Details

Supports sampling only a subset of nodes via `mc_names` and controls how non-sampled nodes are handled via `if_not_sampled`. If `transformation` is enabled and `mctable` includes a `transformation` column, the function computes bounds on the transformed values.

Value

A list bounds with:

- binf numeric vector of lower bounds (same order as factors).
- bsup numeric vector of upper bounds (same order as factors).
- factors character vector of factor names.
- fixed named numeric vector with fixed values for non-sampled factors when if_not_sampled != "exclude".

---

mctable_sobol_matrices

Sobol sampling matrices from an mctable

Description

Create Sobol sampling matrices using sensobol::sobol_matrices() and an mctable definition. The function generates quasi-random draws in [0, 1] and then maps them to the target distributions defined in mctable$mc_func (or mctable$func) and mctable$sample_space.

Usage

```
mctable_sobol_matrices(
  mctable = set_mctable(),
  N,
  matrices = c("A", "B", "AB"),
  order = c("first", "second", "third", "fourth"),
  type = c("QRN", "LHS", "R"),
  mc_names = NULL,
  ...
)
```

Arguments

mctable	(data frame). Table containing at least mcnode and sample_space; may also contain mc_func / func.
N	(integer). Base sample size (see sensobol::sobol_matrices()).
matrices	(character). Which Sobol matrices to create (see sensobol::sobol_matrices()). Default: c("A", "B", "AB").
order	(character). Either "first", "second", "third", or "fourth" (see sensobol::sobol_matrices()).
type	(character). Sampling design used by sensobol::sobol_matrices(). In sensobol 1.1.6, options include "QRN" (default), "LHS", and "R".
mc_names	(character vector, optional). Node names to include. If NULL, all nodes in mctable$mcnode are used.
...	Additional arguments passed to sensobol::sobol_matrices() (and potentially to randtoolbox::sobol() when type = "QRN").

Details

If the distribution function is missing but numeric bounds are available in `sample_space` (e.g. `min = 0, max = 1` or `c(0, 1)`), the function assumes a uniform distribution (`stats::runif`).

Value

A numeric matrix where each column is a model input distributed in (0, 1) **after mapping to the distributions defined in the** `mctable`, and each row is a sampling point. The matrix has the same layout/row binding as `sensobol::sobol_matrices()`.

`mc_compare` *Compare Monte Carlo Node Against Baseline Scenario*

Description

Compares an mcnode's what-if scenarios against a baseline scenario (default "0") using various comparison metrics. Returns an mcmodule with a new comparison node.

Usage

```
mc_compare(
  mcmodule,
  mc_name,
  baseline = "0",
  type = "difference",
  keys_names = NULL,
  name = NULL,
  prefix = NULL,
  suffix = "compared",
  summary = TRUE,
  align_uncertainty = TRUE
)
```

Arguments

`mcmodule`	(mcmodule object). Module containing the node.
`mc_name`	(character). Name of the mcnode to compare.
`baseline`	(character). Baseline scenario ID to compare against. Default: "0".
`type`	(character). Type of comparison. One of: • "difference" (default): whatif - baseline (absolute change) • "relative_difference": (whatif - baseline) / baseline (proportional change) • "reduction": baseline - whatif (absolute reduction) • "relative_reduction": (baseline - whatif) / baseline (proportional reduction)
`keys_names`	(character vector, optional). Column names for grouping. If NULL, uses keys from the node. Default: NULL.

`name` (character, optional). Name for the new comparison node. If NULL, auto-generated from `mc_name` and `suffix`. Default: NULL.

`prefix` (character, optional). Prefix for the auto-generated node name. Default: NULL.

`suffix` (character). Suffix appended to auto-generated name. Default: "compared".

`summary` (logical). If TRUE, compute summary statistics for the new node. Default: TRUE.

`align_uncertainty` (logical). If TRUE, align uncertainty iterations between baseline and what-if nodes using rank correlation (Spearman). This ensures that the same uncertainty iteration in both nodes represents similar uncertainty realizations, making comparisons more meaningful when nodes have multivariate dimensions. Default: TRUE.

Details

This function compares what-if scenarios against a baseline by:

1. Filtering the baseline scenario (scenario_id == baseline)
2. Filtering what-if scenarios (scenario_id != baseline)
3. Matching them across scenarios using keys
4. Optionally aligning uncertainty iterations using rank correlation
5. Applying the selected comparison formula
6. Creating a new comparison node in the mcmodule

When `align_uncertainty` = `TRUE`, the function uses `mc2d::cornode()` to align the uncertainty iterations between matched baseline and what-if nodes. For multivariate nodes, correlation is applied independently to each variate.

For derived nodes with pre-computed summaries (types `"filter"`, `"compare"`, or `"agg_total"`), scenario filtering and key alignment use the node's `summary` by default as the source data.

The baseline scenario must contain all key combinations present in what-if scenarios. If what-if scenarios are missing key combinations present in baseline, those are interpreted as having baseline values (no change).

Value

Updated mcmodule with a new comparison node containing:

- mcnode: Comparison values as mcnode object
- type: "compare"
- baseline: Baseline scenario ID
- compare_type: Type of comparison performed
- param: Original node name
- inputs: Original node name
- keys: Same keys as original node
- summary: Summary statistics (if summary = TRUE)

Examples

```
# Create example data with baseline and what-if scenarios
example_data <- data.frame(
  origin = c("A", "B", "A", "B"),
  scenario_id = c("0", "0", "1", "1")
)

# Create mcnodes for each scenario
example_mcnode <- mc2d::mcstoc(
  runif,
  min = mc2d::mcdata(c(0.1, 0.2, 0.15, 0.25), type = "0", nvariates = 4),
  max = mc2d::mcdata(c(0.2, 0.3, 0.25, 0.35), type = "0", nvariates = 4),
  nvariates = 4
)

# Create mcmodule
example_module <- list(
  data = list(example_data = example_data),
  node_list = list(
    risk = list(
      mcnode = example_mcnode,
      data_name = "example_data",
      keys = c("origin")
    )
  )
)

# Compare what-if scenario "1" against baseline "0"
result <- mc_compare(
  example_module,
  "risk",
  baseline = "0",
  type = "relative_reduction"
)

# View comparison results
result$node_list$risk_compared$summary
```

`mc_filter` *Filter mcnode Variates by Condition*

Description

Filters variates (data rows) from an mcnode based on logical conditions, similar to `dplyr::filter()`. Can return a new node in the mcmodule or return a filtered mcnode directly.

Usage

```
mc_filter(
```

```
  mcmodule = NULL,
  mc_name = NULL,
  ...,
  data = NULL,
  mcnode = NULL,
  name = NULL,
  prefix = NULL,
  suffix = "filtered",
  summary = TRUE
)
```

Arguments

mcmodule	(mcmodule object, optional). Module containing the node. Default: NULL.
mc_name	(character, optional). Name of the mcnode in the module.
...	(expression). Logical conditions to filter by; evaluated in context of the data associated with the mcnode.
data	(data frame, optional). Input data frame. Default: NULL.
mcnode	(mcnode object, optional). mcnode to filter directly. Default: NULL.
name	(character, optional). Name for the new filtered node when adding to mcmodule. If NULL, auto-generated from mc_name and suffix. Default: NULL.
prefix	(character, optional). Prefix for the auto-generated node name. Default: NULL.
suffix	(character). Suffix appended to auto-generated name. Default: "filtered".
summary	(logical). If TRUE, compute summary statistics for the new node. Default: TRUE.

Details

Call signatures:

- To add filtered node to mcmodule: `mc_filter(mcmodule, "node", conditions, name = "new_name")`
- To return filtered mcnode only: `mc_filter(conditions, data = data, mcnode = mcnode)`

Filter conditions work on variates (data rows); only rows meeting all conditions are retained in the resulting mcnode.

For derived nodes with pre-computed summaries (types `"filter"`, `"compare"`, or `"agg_total"`), filtering uses the node's `summary` by default as the data source (instead of `mcmodule$data[[data_name]]`) so variate alignment is preserved.

Value

Either:

- Updated mcmodule with new filtered node (when mcmodule and `name` provided).
- Filtered mcnode object (when only `data` and `mcnode` provided).

Either:

- An updated mcmodule with a new filtered node (when mcmodule and name are provided)
- A raw filtered mcnode object (when only data and mcnode are provided)

Examples

```
# Filter within an mcmodule and create new node
imports_mcmodule <- mc_filter(
  imports_mcmodule,
  "w_prev",
  origin == "nord",
  name = "w_prev_countryA"
)

# Filter and return raw mcnode (note: conditions before named args)
w_prev <- imports_mcmodule$node_list$w_prev$mcnode
w_prev_filtered <- mc_filter(
  origin == "nord",
  data = imports_data,
  mcnode = w_prev
)

# Multiple filter conditions
imports_mcmodule <- mc_filter(
  imports_mcmodule,
  "w_prev",
  origin == "nord",
  pathogen == "virus",
  name = "w_prev_countryA_virus"
)
```

---

mc_keys *Extract Key Columns from Monte Carlo Nodes*

Description

Extracts key columns from a mcnode's associated data.

Usage

```
mc_keys(mcmodule, mc_name, keys_names = NULL)
```

Arguments

mcmodule	(mcmodule object). Module containing node.
mc_name	(character). Node name to extract keys from.
keys_names	(character vector, optional). Column names to extract. If NULL, uses all keys for the node. Default: NULL.

Details

Sample-design nodes are treated as row-aligned inputs: they have no data name or key columns, and key extraction returns only `scenario_id`.

Value

A data frame with `scenario_id` and requested key columns.

Examples

```
keys_df <- mc_keys(imports_mcmodule, "w_prev")
```

`mc_match` *Match Two Monte Carlo Nodes*

Description

Matches two mcnodes by aligning groups, scenarios, or adding missing groups across different scenarios.

Usage

```
mc_match(
  mcmodule,
  mc_name_x,
  mc_name_y,
  keys_names = NULL,
  match_scenario = TRUE
)
```

Arguments

`mcmodule`	(mcmodule object). Module containing nodes.
`mc_name_x`	(character). First mcnode name.
`mc_name_y`	(character). Second mcnode name.
`keys_names`	(character vector, optional). Column names for matching. Default: NULL.
`match_scenario`	(logical). If TRUE, scenario_id is used for alignment (default behavior). If FALSE, scenario_id is treated as a regular key, enabling cross-scenario matching. Default: TRUE.

Details

Matching proceeds in order:

1. Group matching — align nodes with same scenarios but different group order
2. Scenario matching — align nodes with same groups but different scenarios
3. Null matching — add missing groups across different scenarios

Sample-design nodes behave as 1-variate that can be matched directly.

Value

A list containing matched nodes and combined keys (`keys_xy`).

Examples

```
test_module <- list(
  node_list = list(
    node_x = list(
      mcnode = mcstoc(runif,
        min = mcdata(c(1, 2, 3), type = "0", nvariates = 3),
        max = mcdata(c(2, 3, 4), type = "0", nvariates = 3),
        nvariates = 3
      ),
      data_name = "data_x",
      keys = c("category")
    ),
    node_y = list(
      mcnode = mcstoc(runif,
        min = mcdata(c(5, 6, 7), type = "0", nvariates = 3),
        max = mcdata(c(6, 7, 8), type = "0", nvariates = 3),
        nvariates = 3
      ),
      data_name = "data_y",
      keys = c("category")
    )
  ),
  data = list(
    data_x = data.frame(
      category = c("A", "B", "C"),
      scenario_id = c("0", "0", "0")
    ),
    data_y = data.frame(
      category = c("B", "B", "B"),
      scenario_id = c("0", "1", "2")
    )
  )
)

result <- mc_match(test_module, "node_x", "node_y")
```

mc_match_data *Match Monte Carlo Node with Data Frame*

Description

Matches an mcnode with a data frame by aligning groups, scenarios, or adding missing groups across different scenarios.

Usage

```
mc_match_data(
  mcmodule,
  mc_name,
  data,
  keys_names = NULL,
  match_scenario = TRUE
)
```

Arguments

mcmodule	(mcmodule object). Module containing node.
mc_name	(character). Node name.
data	(data frame). Data to match with mcnode.
keys_names	(character vector, optional). Column names for matching. Default: NULL.
match_scenario	(logical). If TRUE, scenario_id is used for alignment (default behavior). If FALSE, scenario_id is treated as a regular key, enabling cross-scenario matching. Default: TRUE.

Details

Matching proceeds in order:

1. Group matching — same scenarios but different group order
2. Scenario matching — same groups but different scenarios
3. Null matching — add missing groups across different scenarios

Sample-design nodes behave as 1-variate that can be matched directly.

Value

A list containing matched mcnode, matched data, and combined keys (keys_xy).

Examples

```
test_data  <- data.frame(pathogen=c("a","b"),
                         inf_dc_min=c(0.05,0.3),
                         inf_dc_max=c(0.08,0.4))
result<-mc_match_data(imports_mcmodule,"no_detect", test_data)
```

`mc_network` *Create Interactive Network Visualisation*

Description

[Experimental]

Generates an interactive network visualisation using visNetwork library. The visualisation includes interactive features for exploring model structure and relationships.

By default, nodes are colored as:

- **inputs** (light blue, #B0DFF9): Input datasets, data frames, files, and columns
- **in_node** (blue, #6ABDEB): Input nodes and scalar values
- **out_node** (green, #A4CF96): Output nodes
- **filter** (light purple, #E8A5E5): Filtered nodes created with `mc_filter()`
- **compare** (medium purple, #D88FD5): Comparison nodes created with `mc_compare()`
- **trials_info** (light orange, #FAE4CB): Trial, subset, and related information nodes
- **total** (orange, #F39200): Total nodes created with `at_least_one()`
- **agg_total** (dark orange, #C17816): Aggregated total nodes created with `agg_totals()`

Usage

```
mc_network(
  mcmodule,
  variate = 1,
  color_pal = NULL,
  color_by = NULL,
  legend = FALSE,
  inputs = FALSE
)
```

Arguments

`mcmodule`	(mcmodule object). Module containing network to visualise.
`variate`	(integer). Which variate to visualise. Default: 1.
`color_pal`	(character vector, optional). Custom colour palette for nodes. Default: NULL.
`color_by`	(character, optional). Column name to determine node colours. Default: NULL.
`legend`	(logical). If TRUE, show colours legend. Default: FALSE.
`inputs`	(logical). If TRUE, show non-node inputs. Default: FALSE.

Value

An interactive visNetwork object with highlighting of connected nodes, node selection and filtering by module, directional arrows, hierarchical layout, and draggable nodes.

Examples

```
network <- mc_network(mcmodule=imports_mcmodule)
```

`mc_plot` *Plot Monte Carlo Node Distribution with Boxplot and Scatter Points*

Description

[Experimental]

Creates a ggplot2 visualisation of Monte Carlo node data showing distributions as semi-transparent boxplots overlaid with scatter points representing individual uncertainty iterations.

Usage

```
mc_plot(
  mcmodule = NULL,
  mc_name = NULL,
  mcnode = NULL,
  data = NULL,
  keys_names = NULL,
  color_by = NULL,
  order_by = NULL,
  group_by = NULL,
  filter = NULL,
  threshold = NULL,
  scale = NULL,
  max_dots = 300,
  point_alpha = 0.4,
  boxplot_alpha = 0.3,
  color_pal = NULL
)
```

Arguments

`mcmodule`	(mcmodule object, optional). Module containing the node.
`mc_name`	(character, optional). Name of the mcnode in the module.
`mcnode`	(mcnode object, optional). mcnode to plot directly.
`data`	(data frame, optional). Input data. If NULL, extracted from `mcmodule`. Default: NULL.
`keys_names`	(character vector, optional). Column names for grouping variates. If NULL, uses node keys from module or row indices. Default: NULL.
`color_by`	(character, optional). Column name to colour points and boxplot. Must be in `keys_names` or `data`. Default: NULL.

order_by	(character, optional). Column name or "median" to reorder y-axis groups. If "median", groups ordered by median value. Default: NULL.
group_by	(character, optional). Column name to group variates (e.g., "commodity"). Variates organised so all scenarios per group appear together. Default: NULL.
filter	(expression, optional). Unquoted expression to filter variates (e.g., pathogen == "a" or origin == "nord"). Passed to tidy_mcnode(). Default: NULL.
threshold	(numeric, optional). Reference value for vertical dashed line. Default: NULL.
scale	(character, optional). Transformation for x-axis: "identity" (default), "log10", "log", "sqrt", or "asinh". Default: NULL.
max_dots	(integer). Maximum dots per variate; exceeding this triggers representative sampling. Boxplots always use all simulations. Default: 300.
point_alpha	(numeric). Transparency for points (0–1). Default: 0.4.
boxplot_alpha	(numeric). Transparency for boxplots (0–1). Default: 0.3.
color_pal	(character vector, optional). Named vector of colours for color_by categories. Default: NULL.

Details

When color_by is NULL, scenarios are coloured by default: — baseline scenario (scenario_id == "0"): blue (#6ABDEB); — alternative scenarios: green (#A4CF96). Boxplots show all uncertainty iterations for statistical accuracy; scatter points are sampled to improve readability with many variates.

Value

A ggplot2 object for further customisation and display.

Examples

```
# Basic plot using mcmodule and mc_name
mc_plot(imports_mcmodule, "w_prev")

# Plot with custom coloring and ordering
mc_plot(imports_mcmodule, "w_prev",
  color_by = "origin",
  order_by = "median"
)

# Plot with threshold and scale transformation
mc_plot(imports_mcmodule, "no_detect",
  threshold = 0.5,
  scale = "log10"
)
```

`mc_summary` *Summarise Monte Carlo Node Values*

Description

Computes summary statistics for an mcnode object, including mean, standard deviation, and quantiles. Can be called with an mcmodule and node name, or directly with an mcnode and data frame.

Usage

```
mc_summary(
  mcmodule = NULL,
  mc_name = NULL,
  keys_names = NULL,
  data = NULL,
  mcnode = NULL,
  sep_keys = TRUE,
  digits = NULL
)
```

Arguments

`mcmodule`	(mcmodule object, optional). Module containing the node. Default: NULL.
`mc_name`	(character, optional). Name of the mcnode in the module.
`keys_names`	(character vector, optional). Column names for grouping. Default: NULL.
`data`	(data frame, optional). Input data frame. Default: NULL.
`mcnode`	(mcnode object, optional). mcnode to summarise directly. Default: NULL.
`sep_keys`	(logical). If TRUE, keep keys in separate columns; if FALSE, combine into single column. Default: TRUE.
`digits`	(integer, optional). Number of significant digits for rounding. Default: NULL.

Details

This function can be called in two ways:

1. By providing an mcmodule and mc_name
2. By providing data and mcnode directly

For filtered nodes (type = "filter"), compared nodes (type = "compare"), and aggregated nodes (type = "agg_total"), this function returns the pre-calculated summary statistics that were computed when the node was created, rather than recalculating from the original data.

Value

A data frame with summary statistics for each mcnode variate. Columns include:

- mc_name: Node name.
- Key columns (if sep_keys = TRUE) or single keys column (if FALSE).
- mean: Average value.
- sd: Standard deviation.
- Quantile columns (2.5%, 25%, 50%, 75%, 97.5%).

Examples

```
# Use with mcmodule
summary_basic <- mc_summary(imports_mcmodule, "w_prev")

# Using custom keys and rounding
summary_custom <- mc_summary(imports_mcmodule, "w_prev",
  keys_names = c("origin"),
  digits = 3
)

# Use with data and mcnode
w_prev <- imports_mcmodule$node_list$w_prev$mcnode
summary_direct <- mc_summary(
  data = imports_data,
  mcnode = w_prev,
  sep_keys = FALSE
)
```

---

optim_ndvar *Optimize Number of Variability Iterations Based on Convergence*

Description

[Experimental]

Automatically determines the minimum number of variability iterations (ndvar) required for all input nodes in a Monte Carlo model to converge at the 5% threshold. Uses an iterative algorithm starting with 1,001 variates and adjusting up or down based on observed convergence.

Usage

```
optim_ndvar(
  mctable = set_mctable(),
  exp = NULL,
  mc_names = NULL,
  min_ndvar = 100,
  max_ndvar = 50000,
  start_ndvar = 1001,
```

```
  conv_threshold = 0.05,
  print_summary = TRUE,
  progress = FALSE
)
```

Arguments

mctable	(data frame). Table with mcnode and sample_space columns defining the sampling distribution for each input. Default: set_mctable().
exp	(language or list). Optional model expression(s) to evaluate. Default: NULL, will create an expression multipling all input nodes.
mc_names	(character vector, optional). Specific node names to analyze. If NULL, analyzes all nodes. Default: NULL.
min_ndvar	(integer). Minimum allowed ndvar. Default: 100.
max_ndvar	(integer). Maximum allowed ndvar. Default: 50000.
start_ndvar	(integer). Initial ndvar to test. Default: 1001.
conv_threshold	(numeric). Convergence threshold at 5%. Default: 0.05.
print_summary	(logical). If TRUE, print optimization summary. Default: TRUE.
progress	(logical). If TRUE, print progress for each iteration. Default: FALSE.

Details

The optimization algorithm:

- Starts with start_ndvar (default 1,001)
- If convergence achieved: tries n/2 (lower bound search)
- If convergence not achieved: tries 2n (upper bound search)
- Continues until minimum converging ndvar is found
- Warns if limits (min_ndvar, max_ndvar) are reached

Value

A list containing:

- optimal_ndvar: The minimum ndvar where all nodes converge.
- converged: Logical indicating if convergence was achieved.
- iterations: Data frame with each iteration's details (ndvar, converged, reason).
- convergence_results: Convergence analysis results from mcmodule_converg().

Examples

```
# Define mctable
mctable <- data.frame(
  mcnode = c("input_a", "input_b"),
  sample_space = c("min = 0, max = 1", "min = 10, max = 20")
)

# Optimize ndvar
result <- optim_ndvar(
  exp = quote({result <- input_a * input_b}),
  mctable = mctable
)

result$optimal_ndvar
```

`prevalence_region` *Regional Pathogen Prevalence Data*

Description

A dataset containing prevalence information for two pathogens across three regions.

Usage

```
prevalence_region
```

Format

prevalence_region:

A data frame with 6 rows and 4 columns:

pathogen Pathogen identifier (a or b)

origin Region of origin (nord, south, east)

h_prev_min Minimum herd prevalence value

h_prev_max Maximum herd prevalence value

w_prev_min Minimum within-herd prevalence value

w_prev_max Maximum within-herd prevalence value

test_origin Test used to detect infected animals at origin

Source

Simulated data for demonstration purposes

reset_data_keys *Reset Data Keys*

Description

Reset Global Data Keys

Clears and resets the global data keys to an empty state.

Usage

```
reset_data_keys()
```

Value

NULL (invisibly). Clears global data_keys.

Examples

```
reset_data_keys()
```

reset_mctable *Reset Monte Carlo Inputs Table*

Description

Clears and resets the global mctable to an empty state with standard columns.

Usage

```
reset_mctable()
```

Value

Empty data frame with standard mctable columns.

reset_sample_design *Reset Global Sample Design*

Description

Clears and resets the global sample design to NULL.

Usage

```
reset_sample_design()
```

Value

NULL (invisibly). Clears global sample design.

Examples

```
reset_sample_design()
```

set_data_keys *Set or Get Global Data Keys*

Description

Manages a global data model by setting or retrieving data keys. The data model defines column names and their associated grouping keys for each data frame.

Usage

```
set_data_keys(data_keys = NULL)
```

Arguments

data_keys	(list, optional). List of data specifications. Each element is a named list with: • cols: Character vector of column names. • keys: Character vector of key columns (subset of cols). If NULL, returns the current global data model. Default: NULL.

Value

Current or newly set global data model (invisibly).

Examples

```
print(imports_data_keys)
set_data_keys(imports_data_keys)
```

set_mctable *Set or Get Monte Carlo Inputs Table*

Description

Manages the global mctable (Monte Carlo nodes reference table) by setting or retrieving its value. The table stores metadata about mcnodes including descriptions, functions, and sensitivity analysis parameters.

Usage

```
set_mctable(data = NULL)
```

Arguments

data	(data frame, optional). mctable with at minimum an mcnode column. Other columns auto-filled if absent. If NULL, returns the current table. Default: NULL.

Details

mctable columns are interpreted as follows:

- mcnode: Name of the Monte Carlo node (required).
- description: Human-readable description of the node.
- mc_func: Distribution function used to create stochastic nodes (for example runif, rpert). If missing/NA, node is deterministic.
- from_variable: Source column name in data when different from mcnode.
- sample_space: Sampling definition used by mctable_bounds() and mctable_sobol_matrices(). Supported formats include c(...) and named bounds such as min = X, max = Y.
- transformation: R expression applied using value as placeholder before node creation.
- sensi_variation: OAT variation expression using value placeholder in eval_module().

Value

Current or newly set mctable. Columns include: mcnode (required), description, mc_func, from_variable, sample_space, transformation, sensi_variation.

Examples

```
# Get current MC table
current_table <- set_mctable()

# Set new MC table
mct <- data.frame(
  mcnode = c("h_prev", "w_prev"),
  description = c("Herd prevalence", "Within herd prevalence"),
```

```
  mc_func = c("runif", "runif")
)
set_mctable(mct)
```

`set_sample_design` *Set or Get Global Sample Design*

Description

Manages a global sample design matrix/data frame by setting or retrieving it. This object is typically a matrix with one column per input parameter and one row per sample, or the output of sensitivity::sensitivity functions. It can be used as default input in `eval_module()`.

Usage

```
set_sample_design(data = NULL)
```

Arguments

`data`	(matrix, data frame, or list, optional). Sample design to store globally. Accepts a matrix/data frame or a list with a matrix in element `X` (typically output of sensitivity::sensitivity functions). If `NULL`, returns the current global sample design. Default: `NULL`.

Value

Current or newly set sample design (`list` with elements `sa` and `X`) or `NULL` if no sample design has been set.

Examples

```
# Get current sample design (NULL if not set)
current_sample_design <- set_sample_design()

# Set sample design
X <- data.frame(a = c(0.1, 0.2), b = c(1, 2))
set_sample_design(X)

# Reset sample design
reset_sample_design()
```

test_sensitivity *Test Sensitivity Data for Pathogens*

Description

A dataset containing test sensitivity values for two pathogens.

Usage

```
test_sensitivity
```

Format

test_sensitivity:

A data frame with 2 rows and 4 columns:

pathogen Pathogen identifier (a or b)

test_sensi_min Minimum test sensitivity value

test_sensi_mode Most likely test sensitivity value

test_sensi_max Maximum test sensitivity value

tidy_mcnode *Convert mcnode to Long Format for Plotting*

Description

[Experimental]

Converts an mcnode to long format suitable for ggplot2 and tidyverse analysis. Each row represents one uncertainty iteration for one variate.

Usage

```
tidy_mcnode(
  mcmodule = NULL,
  mc_name = NULL,
  mcnode = NULL,
  data = NULL,
  keys_names = NULL,
  filter = NULL
)
```

Arguments

mcmodule	(mcmodule object, optional). Module containing the node.
mc_name	(character, optional). Name of the mcnode in the module.
mcnode	(mcnode object, optional). mcnode to convert directly.
data	(data frame, optional). Input data; extracted from mcmodule if NULL. Default: NULL.
keys_names	(character vector, optional). Column names for grouping variates. If NULL, uses node keys from module or all available keys. Default: NULL.
filter	(expression, optional). Unquoted expression to filter variates (e.g., pathogen == "a" or origin == "nord"). Evaluated in context of keys data frame. Default: NULL.

Details

Call signatures:

- tidy_mcnode(mcmodule, \"node_name\")
- tidy_mcnode(mcnode = mcnode, data = data)
- tidy_mcnode(mcmodule, mcnode = mcnode)

Value

A long data frame with columns:

- All key columns from keys_names.
- variate: Variate index (data row number).
- simulation: Uncertainty iteration index.
- value: mcnode value for that combination.

Examples

```
# Using mcmodule and node name
long_data <- tidy_mcnode(imports_mcmodule, "w_prev")

# Using with specific keys
long_data <- tidy_mcnode(imports_mcmodule, "w_prev",
  keys_names = "origin"
)

# Using mcnode and data directly
w_prev <- imports_mcmodule$node_list$w_prev$mcnode
long_data <- tidy_mcnode(mcnode = w_prev, data = imports_data)

# Filter variates
long_data <- tidy_mcnode(imports_mcmodule, "w_prev",
  filter = pathogen == "a"
)
```

`trial_totals` *Trial Probability and Expected Counts*

Description

Calculates probabilities and expected counts across hierarchical levels (trial, subset, set) in a structured population. Uses trial probabilities and handles nested sampling with conditional probabilities.

Usage

```
trial_totals(
  mcmodule,
  mc_names,
  trials_n,
  subsets_n = NULL,
  subsets_p = NULL,
  name = NULL,
  prefix = NULL,
  combine_prob = TRUE,
  all_suffix = NULL,
  level_suffix = c(trial = "trial", subset = "subset", set = "set"),
  mctable = set_mctable(),
  sample_design = set_sample_design(),
  agg_keys = NULL,
  agg_suffix = NULL,
  keep_variates = FALSE,
  summary = TRUE,
  data_name = NULL
)
```

Arguments

`mcmodule`	(mcmodule object). Module containing input data and node structure.
`mc_names`	(character vector). Node names to process.
`trials_n`	(character). Trial count column name.
`subsets_n`	(character, optional). Subset count column name. Default: NULL.
`subsets_p`	(character, optional). Subset prevalence column name. Default: NULL.
`name`	(character, optional). Custom name for output nodes. Default: NULL.
`prefix`	(character, optional). Prefix for output node names. Default: NULL.
`combine_prob`	(logical). If TRUE, combine probability of all nodes assuming independence. Default: TRUE.
`all_suffix`	(character). Suffix for combined node name. Default: "all".
`level_suffix`	(list, optional). Suffixes for each hierarchical level. Default: c(trial="trial", subset="subset", set="set").

mctable	(data frame, optional). Monte Carlo nodes definitions. Default: set_mctable().
sample_design	(matrix, data frame, or list, optional). Sampling design used to create missing input nodes via matrix_to_mcnodes(). Accepts a matrix/data frame (for example from sensobol::sobol_matrices()) or a list with element X (typically output of sensitivity::sensitivity functions such as sensitivity::morris()). Defaults to set_sample_design().
agg_keys	(character vector, optional). Column names for aggregation. Default: NULL.
agg_suffix	(character). Suffix for aggregated node names. Default: "hag".
keep_variates	(logical). If TRUE, preserve individual variate values. Default: FALSE.
summary	(logical). If TRUE, include summary statistics. Default: TRUE.
data_name	(character, optional). Data name used to create trials_n, subsets_n and subsets_p nodes if they don't exist in mcmodule. Default: NULL.

Value

Updated mcmodule object containing combined node probabilities and probabilities/counts at trial, subset, and set levels.

Examples

```
imports_mcmodule <- trial_totals(
  mcmodule = imports_mcmodule,
  mc_names = "no_detect",
  trials_n = "animals_n",
  subsets_n = "farms_n",
  subsets_p = "h_prev",
  mctable = imports_mctable
)
print(imports_mcmodule$node_list$no_detect_set$summary)
```

---

visNetwork_edges *Generate Formatted visNetwork Edge Table*

Description

Creates a formatted edge table suitable for visualisation with visNetwork.

Usage

```
visNetwork_edges(mcmodule, inputs = FALSE)
```

Arguments

mcmodule	(mcmodule object). Module containing node relationships.
inputs	(logical). If TRUE, include non-node inputs. Default: FALSE.

Value

A data frame containing edge information for visNetwork with columns: from, to, and id.

`visNetwork_nodes` *Generate Formatted Network Node Table for Visualisation*

Description

Creates a formatted node table for visualisation with visNetwork. Includes styling and formatting for interactive network display.

Usage

```
visNetwork_nodes(
  mcmodule,
  variate = 1,
  color_pal = NULL,
  color_by = NULL,
  inputs = FALSE
)
```

Arguments

`mcmodule`	(mcmodule object). Module containing network structure.
`variate`	(integer). Which variate to extract. Default: 1.
`color_pal`	(character vector, optional). Custom colour palette for nodes. Default: NULL.
`color_by`	(character, optional). Column name to determine node colours. Default: NULL.
`inputs`	(logical). If TRUE, include non-node inputs. Default: FALSE.

Value

A data frame formatted for visNetwork with columns: id, label, color, grouping, expression, and title (hover text).

which_mcnode *Find mcnodes Matching a Condition*

Description

Applies a test function to each mcnode in an mcmodule and returns the names of nodes where the test returns TRUE. Useful for identifying nodes with specific properties (e.g., NA values, negative values).

Usage

```
which_mcnode(mcmodule, test_func)
```

Arguments

mcmodule	(mcmodule object). Module containing node_list with mcnodes.
test_func	(function). Function that takes an mcnode and returns logical; TRUE if the condition is met.

Value

Character vector of mcnode names where test_func returns TRUE. Empty vector if no nodes meet the condition.

See Also

which_mcnode_na(), which_mcnode_inf()

Examples

```
# Find nodes with negative values
which_mcnode(imports_mcmodule, function(x) any(x < 0, na.rm = TRUE))

# Find nodes with values greater than 1
which_mcnode(imports_mcmodule, function(x) any(x > 1, na.rm = TRUE))
```

which_mcnode_inf *Find mcnodes with Infinite Values*

Description

Identifies which mcnodes within an mcmodule contain infinite values (Inf or -Inf). Useful for troubleshooting and debugging Monte Carlo models.

Usage

```
which_mcnode_inf(mcmodule)
```

Arguments

mcmodule	(mcmodule object). Module containing node_list.

Value

Character vector of mcnode names containing infinite values. Returns empty vector if no infinite values found.

See Also

which_mcnode(), which_mcnode_na(), mcnode_na_rm()

Examples

```
# Find nodes with infinite values in the imports_mcmodule
which_mcnode_inf(imports_mcmodule)

# Create a test mcmodule with Inf values
test_mcnode_inf <- mcdata(c(0.1, Inf, 0.3), type = "0", nvariates = 3)
test_mcnode_clean <- mcdata(c(0.1, 0.2, 0.3), type = "0", nvariates = 3)
test_mcmodule <- list(
  node_list = list(
    node_a = list(mcnode = test_mcnode_inf),
    node_b = list(mcnode = test_mcnode_clean)
  )
)
which_mcnode_inf(test_mcmodule)
```

---

which_mcnode_na *Find* mcnode*s with Missing Values*

Description

Find mcnodes with Missing Values

Usage

```
which_mcnode_na(mcmodule)
```

Arguments

mcmodule	(mcmodule object). Module containing node_list.

Details

Identifies which mcnodes within an mcmodule contain NA values. Useful for troubleshooting and debugging Monte Carlo models.

Value

Character vector of mcnode names containing NA values. Returns empty vector if no NAs found.

See Also

`which_mcnode()`, `which_mcnode_inf()`, `mcnode_na_rm()`

Examples

```
# Find nodes with NAs in the imports_mcmodule
which_mcnode_na(imports_mcmodule)

# Create a test mcmodule with NAs
test_mcnode_na <- mcdata(c(0.1, NA, 0.3), type = "0", nvariates = 3)
test_mcnode_clean <- mcdata(c(0.1, 0.2, 0.3), type = "0", nvariates = 3)
test_mcmodule <- list(
  node_list = list(
    node_a = list(mcnode = test_mcnode_na),
    node_b = list(mcnode = test_mcnode_clean)
  )
)
which_mcnode_na(test_mcmodule)
```

---

wif_match *Match Datasets with Differing Scenarios*

Description

Matches datasets by group and preserves baseline scenarios (scenario_id = 0) when scenarios differ between them.

Usage

```
wif_match(x, y, by = NULL)
```

Arguments

x	(data frame). First dataset to match.
y	(data frame). Second dataset to match.
by	(character vector, optional). Grouping column name(s) to match on. If NULL, auto-detected from column names. Default: NULL.

Value

A list containing matched datasets with aligned scenario IDs. Element 1: matched version of x. Element 2: matched version of y.

Examples

```
x <- data.frame(
  category = c("a", "b", "a", "b"),
  scenario_id = c(0, 0, 1, 1),
  value = 1:4
)

y <- data.frame(
  category = c("a", "b", "a", "b"),
  scenario_id = c(0, 0, 2, 2),
  value = 5:8
)

# Automatic matching
result <- wif_match(x, y)
```

Index