Package 'mcmodule'

October 22, 2025

Title Modular Monte Carlo Risk Analysis

Version 1.1.0

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, perform multivariate Monte Carlo node operations, automate the creation of Monte Carlo nodes and visualize risk analysis models. For more details see Ciria (2025) <https://nataliaciria.github.io/mcmodule/articles/mcmodule>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

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

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

Suggests knitr, rmarkdown, testthat (>= 3.0.0), visNetwork, igraph, bookdown

Config/testthat/edition 3

Imports dplyr

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 2025-10-22 18:50:02 UTC

Contents

add_group_id *Add Group IDs to Data Frames*

Description

Add Group IDs to Data Frames

Usage

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

Arguments

x	First dataset
y	Optional second dataset
by	Grouping variables

Value

Dataframe or list of dataframes with added group IDs

add_prefix *Add Prefix to Node Names*

Description

Adds a prefix to node_list names and all input nodes. Preserves previous prefixes unless rewrite_module argument is specified.

Usage

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

Arguments

mcmodule	An mcmodule or a node_list object
prefix	String to add as prefix of the new mcmodule mcnodes, defaults to mcmodule name
rewrite_module	Name of a module to rewrite prefixes

Value

A mcmodule with new prefixes in node_list 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 Across Groups*

Description

Combines node values across specified grouping variables using different aggregation methods. The aggregation method can be specified via agg_func parameter:

- "prob": Combined probability assuming independence
- "sum": Sum of values
- "avg": Average of values
- NULL: defaults to "sum" if mc_name ends in "_n", else defaults to "prob"

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 containing nodes and data
`mc_name`	name of node to aggregate
`agg_keys`	grouping variables for aggregation
`agg_suffix`	Suffix for aggregated node name (default: "agg")
`prefix`	Optional prefix for output node name - includes metadata as add_prefix() (default: NULL)
`name`	Custom name for output node (optional)
`summary`	whether to include summary statistics (default: TRUE)
`keep_variates`	whether to preserve individual values (default: FALSE)
`agg_func`	aggregation method ("prob", "sum", "avg", or NULL)

Value

mcmodule with new aggregated node added

Examples

```
imports_mcmodule <- agg_totals(
  imports_mcmodule, "no_detect_a",
  agg_keys = c("scenario_id", "pathogen")
)
print(imports_mcmodule$node_list$no_detect_a_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 *Combined Probability of Events (At least one)*

Description

Combines probabilities of multiple events assuming independence, using the formula P(A or B) = 1 - (1-P(A))*(1-P(B)). It matches dimensions automatically.

Usage

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

Arguments

mcmodule	Module containing node list and input data frames
mc_names	Vector of node names to combine
name	Optional custom name for combined node (default: NULL)
all_suffix	Suffix for combined node name (default: "all")
prefix	Optional prefix for output node name (default: NULL)
summary	Whether to 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` *Checks mctable data*

Description

Checks mctable data

Usage

```
check_mctable(data)
```

Arguments

`data`	A data frame containing MC table information. Must contain an 'mcnode' column.

Value

A data frame with the standard mctable structure

`combine_modules` *Combine Two Modules*

Description

Combines two mcmodules into a single mcmodule by merging their data and components.

Usage

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

Arguments

`mcmodule_x`	First module to combine
`mcmodule_y`	Second module to combine

Value

A combined mcmodule object

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 Monte Carlo Nodes from Data and Configuration Table*

Description

Creates Monte Carlo nodes (mcnodes) based on instructions provided in a configuration table (mctable) and input variables from a dataframe.

Usage

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

Arguments

data	A data frame containing the input variables for creating Monte Carlo nodes
mctable	A configuration table specifying MC node definitions. Must contain columns: • mcnode: Name of the Monte Carlo node • mc_func: Distribution function to use (if applicable) • transformation: Optional transformation to apply to input data • from_variable: Optional source variable name for transformation
envir	Environment where MC nodes will be created (default: parent.frame())

Value

No return value, creates MC nodes in the specified environment

Examples

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

eval_module *Evaluate a Monte Carlo Model Expression and create an mcmcmodule*

Description

Takes a set of Monte Carlo model expressions and evaluates them and creates an mcmcmodule containing results and metadata.

Usage

```
eval_module(
  exp,
  data,
  param_names = NULL,
  prev_mcmodule = NULL,
  summary = FALSE,
  mctable = set_mctable(),
  data_keys = set_data_keys(),
  match_keys = NULL
)
```

Arguments

exp	Model expression or list of expressions to evaluate
data	Input data frame containing model parameters
param_names	Named vector for parameter renaming (optional)
prev_mcmodule	Previous module(s) for dependent calculations
summary	Logical; whether to calculate summary statistics
mctable	Monte Carlo configuration table
data_keys	List of key columns for each dataset
match_keys	Keys to match prev_mcmodule mcnodes and data by

Value

An mcmodule object containing data, expressions, and nodes

Examples

```
# Basic usage with single expression
eval_module(
  exp = imports_exp,
  data = imports_data,
  mctable = imports_mctable,
  data_keys = imports_data_keys
)
```

---

get_edge_table *Generate Edge Table for Network Construction*

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	An mcmodule object containing node relationships
inputs	Include non-node inputs: data-sets, data-frames and columns (optional)

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()
)
```

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()

Value

A list of class "mcnode_list" containing node information

get_node_table *Generate Node Table for Network Construction*

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	An mcmodule object containing node information
variate	Integer indicating which variate to extract (default: 1)
inputs	Include non-node inputs: data-sets, data-frames and columns (optional)

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

inf_a Probability of animal being infected

false_neg_a Probability of false negative test result

no_test_a Probability of no testing

no_detect_a Overall probability of non-detection

`imports_mcmodule` *Example mcmodule object containing Monte Carlo simulation results 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 (inf_a)
- False negative probability (false_neg_a)
- No test probability (no_test_a)
- Non-detection probability (no_detect_a)

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 6 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

sensi_analysis Logical flag indicating if node is included in sensitivity analysis

Source

Simulated data for demonstration purposes

keys_match *Match and align keys between two datasets*

Description

Match and align keys between two datasets

Usage

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

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

Value

List containing:

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

---

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

See Also

is.na.mcnode

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)
```

mc_keys *Get Monte Carlo Node Keys*

Description

Extracts key columns from Monte Carlo node's associated data.

Usage

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

Arguments

mcmodule	Monte Carlo module containing nodes and data
mc_name	Name of the node to extract keys from
keys_names	Vector of column names to extract (optional)

Value

Dataframe with scenario_id and requested key columns

Examples

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

mc_match *Match Monte Carlo Nodes*

Description

Matches two mcnodes by:

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

Usage

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

Arguments

mcmodule	Monte Carlo module
mc_name_x	First node name
mc_name_y	Second node name
keys_names	Names of key columns

Value

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 other data frame*

Description

Matches an mcnode with a data frame by:

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

Usage

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

Arguments

mcmodule	Monte Carlo module
mc_name	Node name
data	Data frame containing keys to match with
keys_names	Names of key columns

Value

List containing matched node, 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_a", test_data)
```

---

mc_network *Create Interactive Network Visualization*

Description

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

Usage

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

Arguments

mcmodule	An mcmodule object
variate	Integer specifying which variate to visualize (default: 1)
color_pal	Custom color palette for nodes (optional)
color_by	Column name to determine node colors (optional)
legend	Show colors legend (optional)
inputs	Show non-node inputs: data-sets, data-frames and columns (optional)

Value

An interactive visNetwork object with features:

- Highlighting of connected nodes
- Node selection and filtering by module
- Directional arrows showing relationships
- Hierarchical layout
- Draggable nodes

Examples

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

`mc_summary` *Compute summary statistics for an mcnode object*

Description

Compute summary statistics for an mcnode object

Usage

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

Arguments

`mcmodule`	An mcmodule object containing the node to summarize
`mc_name`	Character string specifying the name of the mcnode in the module
`keys_names`	Vector of column names to use as keys for grouping (default: NULL)
`data`	Optional data frame containing the input data (default: NULL)
`mcnode`	Optional mcnode object to summarize directly (default: NULL)
`sep_keys`	Logical; if TRUE, keeps keys in separate columns (default: TRUE)
`digits`	Integer indicating 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

Value

A data frame containing summary statistics with columns:

- mc_name: Name of the mcnode
- keys: Grouping variables (if sep_keys=FALSE) or individual key columns (if sep_keys=TRUE)
- Summary statistics including:
  - mean: Average value
  - sd: Standard deviation
  - Various quantiles (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
)
```

---

mc_summary_keys *Get mcnode summary keys*

Description

Get mcnode summary keys

Usage

```
mc_summary_keys(mcsummary)
```

Arguments

mcsummary	data frame from mc_summary()

Value

vector of key names

---

node_list_summary *Include summary and keys in node_list*

Description

Include summary and keys in node_list

Usage

```
node_list_summary(mcmodule = NULL, data = NULL, node_list = NULL)
```

Arguments

mcmodule	mc module object
data	data frame with mc inputs
node_list	list of nodes

Value

updated node_list

---

prevalence_region *Regional 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

Resets the data model to an empty list

Usage

```
reset_data_keys()
```

Value

No return value, resets data keys

Examples

```
reset_data_keys()
```

reset_mctable *Resets the Monte Carlo inputs table*

Description

Resets the Monte Carlo inputs table

Usage

```
reset_mctable()
```

Value

An empty data frame with the standard mctable structure

set_data_keys *Set or Get Global Data Keys*

Description

Manages a global data model by either setting new data keys or retrieving the current ones. The data model consists of named lists containing column names and their associated key columns.

Usage

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

Arguments

data_keys	Optional list of lists. Each inner list must contain: • cols: A vector containing the column names for the data • keys: A vector specifying the key columns If NULL, returns the current data model.

Value

- If data_keys = NULL: Returns the current global data model
- If data_keys provided: Sets the new data model and returns invisibly

Examples

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

`set_mctable` *Set or Get Monte Carlo Inputs Table*

Description

Manages a Monte Carlo inputs table in the global package environment by either setting new data or retrieving the current table. The table stores information about Monte Carlo nodes including their descriptions, functions, dependencies, and sensitivity analysis settings.

Usage

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

Arguments

`data`	Optional data frame containing MC table information. Must contain an 'mcnode' column. Other columns will be auto-filled if missing. If NULL, returns the current MC table.

Value

- If data = NULL: Returns the current MC table
- If data provided: Sets the new MC table and returns invisibly

The table contains the following columns:

- mcnode - Character. Name of the Monte Carlo node (required)
- description - Character. Description of the parameter
- mc_func - Character. Probability distribution
- from_variable - Character. Variable name in the data table, if it is in a column with a name different from the mcnode
- transformation - Character. Transformation to be applied to the original column values
- sensi_analysis - Logical. Whether to include in sensitivity analysis

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"),
  sensi_analysis = c(TRUE, TRUE)
)
set_mctable(mct)
```

test_sensitivity *Test Sensitivity Data*

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

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(),
  agg_keys = NULL,
```

```
  agg_suffix = NULL,
  keep_variates = FALSE,
  summary = TRUE
)
```

Arguments

mcmodule	mcmodule object containing input data and node structure
mc_names	Vector of node names to process
trials_n	Trial count column name
subsets_n	Subset count column name (optional)
subsets_p	Subset prevalence column name (optional)
name	Custom name for output nodes (optional)
prefix	Prefix for output node names (optional)
combine_prob	Combine probability of all nodes assuming independence (default: TRUE)
all_suffix	Suffix for combined node name (default: "all")
level_suffix	A list of suffixes for each hierarchical level (default: c(trial="trial",subset="subset",set="set"))
mctable	Data frame containing Monte Carlo nodes definitions (default: set_mctable())
agg_keys	Column names for aggregation (optional)
agg_suffix	Suffix for aggregated node names (default: "hag")
keep_variates	whether to preserve individual values (default: FALSE)
summary	Include summary statistics if TRUE (default: TRUE)

Value

Updated mcmodule object containing:

- Combined node probabilities
- Probabilities and counts at trial level
- Probabilities and counts at subset level
- Probabilities and counts at set level

Examples

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

visNetwork_edges *Generate visNetwork Edge Table*

Description

Creates a formatted edge table suitable for visualization with visNetwork.

Usage

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

Arguments

mcmodule	An mcmodule object
inputs	Include non-node inputs: data-sets, data-frames and columns (optional)

Value

A data frame containing edge information for visNetwork

visNetwork_nodes *Generate Network Node Table for Visualization*

Description

Creates a formatted node table for visualization with visNetwork. Includes styling and formatting for network visualization.

Usage

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

Arguments

mcmodule	An mcmodule object containing the network structure
variate	Integer specifying which variate to extract (default: 1)
color_pal	Custom color palette for nodes (optional)
color_by	Column name to determine node colors (optional)
inputs	Include non-node inputs: data-sets, data-frames and columns (optional)

Value

A data frame formatted for visNetwork with columns:

- id: Unique node identifier
- color: Node color based on type/category
- grouping: Module association
- expression: Node expression or type
- title: Hover text containing node details

`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`	First dataset to match
`y`	Second dataset to match
`by`	Grouping variable(s) to match on, defaults to NULL

Value

List containing matched datasets with aligned scenario IDs:

- First element: matched version of dataset x
- Second element: matched version of dataset 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