Package 'tidyMC'

May 8, 2026

Title Monte Carlo Simulations Made Easy and Tidy

Version 1.0.1

Description Framework to run Monte Carlo simulations over a parameter grid.
Allows to parallelize the simulations.
Generates plots and 'LaTeX' tables
summarizing the results from the simulation.

License MIT + file LICENSE

Imports checkmate, dplyr, furrr, future, ggplot2, hms, kableExtra,
magrittr, methods, purrr, rlang, stringr, tibble, tidyr, utils

Suggests rmarkdown, knitr, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

VignetteBuilder knitr

URL https://github.com/stefanlinner/tidyMC,
https://stefanlinner.github.io/tidyMC/

BugReports https://github.com/stefanlinner/tidyMC/issues

RoxygenNote 7.3.2.9000

NeedsCompilation no

Author Stefan Linner [aut, cre, cph],
Ignacio Moreira Lara [aut],
Konstantin Lehmann [aut]

Maintainer Stefan Linner <stefan.linner97@gmail.com>

Repository CRAN

Date/Publication 2025-07-24 08:20:02 UTC

Contents

`future_mc` *Run a Parallelized Monte Carlo Simulation*

Description

`future_mc` runs a Monte Carlo simulation study for a user-specified function and the desired parameter grids.

Usage

```
future_mc(
  fun,
  repetitions,
  param_list = NULL,
  param_table = NULL,
  parallelisation_plan = list(strategy = future::multisession),
  parallelisation_options = list(),
  check = TRUE,
  parallel = TRUE,
  ...
)
```

Arguments

`fun`	The function to be evaluated. See details.
`repetitions`	An integer that specifies the number of Monte Carlo iterations
`param_list`	A list whose components are named after the parameters of `fun` which should vary for the different Monte Carlo Simulations. Each component is a vector containing the desired grid values for that parameter. The Monte Carlo Simulation is run for all possible combinations of that parameter list.
`param_table`	Alternative to `param_list`. A `data.frame` or `data.table` containing a prebuilt grid of values, where the columns are the parameters of `fun` which should vary for the different Monte Carlo Simulations. This is useful if you only want to run a Monte Carlo Simulation for a subset of all possible combinations.
`parallelisation_plan`	A list whose components are named after possible parameters of `future::plan()` specifying the parallelisation plan which should be used in the Monte Carlo Simulation. Default is `strategy = multisession`.

parallelisation_options
: A list whose components are named after possible parameters of furrr::furrr_options() for fine tuning functions, such as furrr::future_map(). Default is seed = TRUE as long as not specified differently in order to assure reproducibility.

check
: Boolean that specifies whether a single test-iteration should be run for each parameter combination in order to check for possible occuring errors in fun. Default is TRUE.

parallel
: Boolean that specifies whether the Monte Carlo simulation should be run in parallel. Default is TRUE.

...
: Additional parameters that are passed on to fun and which are not part of the parameter grid.

Details

The user defined function fun handles (if specified) the generation of data, the application of the method of interest and the evaluation of the result for a single repetition and parameter combination. future_mc handles the generation of loops over the desired parameter grids and the repetition of the Monte Carlo experiment for each of the parameter constellations.

There are four formal requirements that fun has to fulfill:

- The arguments of fun which are present in param_list need to be scalar values.
- The value returned by fun has to be a named list and must have the same components for each iteration and parameter combination.
- The names of the returned values and those of the arguments contained in param_list need to be different. Moreover, they cannot be "params", "repetitions" or "setup"
- Every variable used inside fun has either to be defined inside fun or given as an argument through the ... argument. In particular, fun cannot use variables which are only defined in the global environment.

In order to use the comfort functions plot.mc(), summary.mc(), plot.summary.mc(), and tidy_mc_latex() the value returned by fun has to be a named list of scalars.

Value

A list of type mc containing the following objects:

- output: A tibble containing the return value of fun for each iteration and parameter combination
- parameter: A tibble which shows the different parameter combinations
- simple_output: A boolean value indicating whether the return value of fun is a named list of scalars or not
- nice_names: A character vector containing "nice names" for the different parameter setups
- calculation_time: The calculation time needed to run the whole Monte Carlo Simulation
- n_results: A numeric value indicating the number of results
- seed: The value which is used for the parameter seed in furrr::furrr_options()
- fun: The user-defined function fun

- repetitions: The number of repetitions run for each parameter setup
- parallel: Boolean whether the Monte Carlo Simulation was run in parallel or not
- plan: A list that specified the parallelisation plan via `future::plan()`

Examples

```
test_func <- function(param = 0.1, n = 100, x1 = 1, x2 = 2) {
  data <- rnorm(n, mean = param) + x1 + x2
  stat <- mean(data)
  stat_2 <- var(data)

  if (x2 == 5) {
    stop("x2 can't be 5!")
  }

  return(list(mean = stat, var = stat_2))
}

param_list <- list(
  param = seq(from = 0, to = 1, by = 0.5),
  x1 = 1:2
)

set.seed(101)
test_mc <- future_mc(
  fun = test_func,
  repetitions = 1000,
  param_list = param_list,
  n = 10,
  x2 = 2
)
```

`plot.mc` *Plot the results of a Monte Carlo Simulation*

Description

Plot density plots for numeric results and bar plots for non-numeric results of a Monte Carlo Simulation run by `future_mc()`.

Usage

```
## S3 method for class 'mc'
plot(
  x,
  join = NULL,
  which_setup = NULL,
  parameter_comb = NULL,
```

```
  plot = TRUE,
  ...
)
```

Arguments

x	An object of class mc, for which holds simple_output = TRUE. See value of future_mc().
join	A character vector containing the nice_names for the different parameter combinations (returned by future_mc()), which should be plotted together. Default: Each parameter combination is plotted distinctly.
which_setup	A character vector containing the nice_names for the different parameter combinations (returned by future_mc()), which should be plotted. Default: All parameter combinations are plotted.
parameter_comb	Alternative to which_setup. A named list whose components are named after (some of) the parameters in param_list in future_mc() and each component is a vector containing the values for the parameters to filter by. Default: All parameter combinations are plotted.
plot	Boolean that specifies whether the plots should be printed while calling the function or not. Default: TRUE
...	ignored

Details

Only one of the arguments join, which_setup, and paramter_comb can be specified at one time.

Value

A list whose components are named after the outputs of fun and each component contains an object of class ggplot and gg which can be plotted and modified with the ggplot2::ggplot2 functions.

Examples

```
test_func <- function(param = 0.1, n = 100, x1 = 1, x2 = 2){

  data <- rnorm(n, mean = param) + x1 + x2
  stat <- mean(data)
  stat_2 <- var(data)

  if (x2 == 5){
    stop("x2 can't be 5!")
  }

  return(list(mean = stat, var = stat_2))
}

param_list <- list(param = seq(from = 0, to = 1, by = 0.5),
                   x1 = 1:2)
```

```
set.seed(101)
test_mc <- future_mc(
  fun = test_func,
  repetitions = 1000,
  param_list = param_list,
  n = 10,
  x2 = 2
)

returned_plot1 <- plot(test_mc)

returned_plot1$mean +
 ggplot2::theme_minimal() +
 ggplot2::geom_vline(xintercept = 3)

returned_plot2 <- plot(test_mc,
which_setup = test_mc$nice_names[1:2], plot = FALSE)
returned_plot2$mean

returned_plot3 <- plot(test_mc,
join = test_mc$nice_names[1:2], plot = FALSE)
returned_plot3$mean
```

`plot.summary.mc` *Plot the summarized results of a Monte Carlo Simulation*

Description

Plot line plots of the path of the summarized output over all simulation repetitions of a Monte Carlo simulation run by `future_mc()` and summarized by `summary.mc()`

Usage

```
## S3 method for class 'summary.mc'
plot(
  x,
  join = NULL,
  which_setup = NULL,
  parameter_comb = NULL,
  plot = TRUE,
  ...
)
```

Arguments

x	An object of class `summary.mc`. For restrictions see details.

join	A character vector containing the nice_names for the different parameter combinations (returned by future_mc()), which should be plotted together. Default: Each parameter combination is plotted distinct.
which_setup	A character vector containing the nice_names for the different parameter combinations (returned by future_mc()), which should be plotted. Default: All parameter combinations are plotted.
parameter_comb	Alternative to which_setup. A named list whose components are named after (some of) the parameters in param_list in future_mc() and each component is a vector containing the values for the parameters to filter by. Default: All parameter combinations are plotted.
plot	Boolean that specifies whether the plots should be printed while calling the function or not. Default: TRUE
...	additional arguments passed to callies.

Details

Only one of the arguments join, which_setup, and paramter_comb can be specified at a time.

A plot is only created for (output - parameter combination)-pairs for which in summary.mc() a function is provided in sum_funs which returns a single numeric value and if the output is included in which_path.

Value

A list whose components are named after the outputs of fun and each component contains an object of class ggplot and gg which can be plotted and modified with the ggplot2::ggplot2 functions.

Examples

```
test_func <- function(param = 0.1, n = 100, x1 = 1, x2 = 2){

  data <- rnorm(n, mean = param) + x1 + x2
  stat <- mean(data)
  stat_2 <- var(data)

  if (x2 == 5){
    stop("x2 can't be 5!")
  }

  return(list(mean = stat, var = stat_2))
}

param_list <- list(param = seq(from = 0, to = 1, by = 0.5),
                   x1 = 1:2)

set.seed(101)
test_mc <- future_mc(
  fun = test_func,
  repetitions = 1000,
  param_list = param_list,
```

```
  n = 10,
  x2 = 2
)

returned_plot1 <- plot(summary(test_mc))

returned_plot1$mean +
 ggplot2::theme_minimal()

returned_plot2 <- plot(summary(test_mc),
which_setup = test_mc$nice_names[1:2], plot = FALSE)
returned_plot2$mean

returned_plot3 <- plot(summary(test_mc),
join = test_mc$nice_names[1:2], plot = FALSE)
returned_plot3$mean
```

`print.mc` *Print the results of a Monte Carlo Simulation*

Description

Print the results of a Monte Carlo Simulation run by `future_mc()`

Usage

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

Arguments

`x`	An object of class `mc`.
`...`	ignored

Value

print shows a complete representation of the run Monte Carlo Simulation

Examples

```
test_func <- function(param = 0.1, n = 100, x1 = 1, x2 = 2){

  data <- rnorm(n, mean = param) + x1 + x2
  stat <- mean(data)
  stat_2 <- var(data)

  if (x2 == 5){
    stop("x2 can't be 5!")
```

```
  }

  return(list(mean = stat, var = stat_2))
}

param_list <- list(param = seq(from = 0, to = 1, by = 0.5),
                   x1 = 1:2)

set.seed(101)
test_mc <- future_mc(
  fun = test_func,
  repetitions = 1000,
  param_list = param_list,
  n = 10,
  x2 = 2
)

test_mc
```

`print.summary.mc` *Print the summarized results of a Monte Carlo Simulation*

Description

Print the summarized results of a Monte Carlo Simulation run by `future_mc()` and summarized by `summary.mc()`

Usage

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

Arguments

`x`	An object of class `summary.mc`
`...`	ignored

Value

print shows a nice representation of the summarized results of a Monte Carlo Simulation

Examples

```
test_func <- function(param = 0.1, n = 100, x1 = 1, x2 = 2){

  data <- rnorm(n, mean = param) + x1 + x2
  stat <- mean(data)
  stat_2 <- var(data)
```

```
  if (x2 == 5){
    stop("x2 can't be 5!")
  }

  return(list(mean = stat, var = stat_2))
}

param_list <- list(param = seq(from = 0, to = 1, by = 0.5),
                   x1 = 1:2)

set.seed(101)
test_mc <- future_mc(
  fun = test_func,
  repetitions = 1000,
  param_list = param_list,
  n = 10,
  x2 = 2
)

summary(test_mc)
```

---

summary.mc *Summarize the Results of a Monte Carlo Simulation*

Description

Summarize the results of a Monte Carlo Simulation run by future_mc() with (optionally) user-defined summary functions.

Usage

```
## S3 method for class 'mc'
summary(object, sum_funs = NULL, which_path = "all", ...)
```

Arguments

object	An object of class mc, for which holds simple_output = TRUE. See value of future_mc().
sum_funs	A named (nested) list containing summary functions. See details.
which_path	A character vector containing the names of (some of) the named outputs (the names of the returned list of fun in future_mc()), for which to return a "path" of the stepwise calculation of the result of the summary function. Alternatively, "all" or "none" can be used to return either the path for all or none of the numeric outputs. Default: "all".
...	Ignored

Details

In order to use summary(), the output of future_mc() has to be "simple", which is the case if the return value of fun is a named list of scalars. If the returned value of fun is a named list of more complex data structures, summary() cannot be used.

With sum_funs the user can define (different) functions which summarize the simulation results for each output (return values of fun in future_mc()) and each parameter combination. Thus, the functions inside sum_funs only take one argument, which is the output vector (with length repetitions) of one output of one specific parameter combination.

The default summary functions are base::mean() for numeric outputs and base::summary() for outputs with non-numeric data types.

The user can define summary functions by supplying a named (nested) list to sum_funs. When the functions provided for each output return only one numeric value the results are twofold: first, a single scalar result of the function evaluating the whole output vector. Second, a "path" with length repetitions of the stepwise calculation of the function's result across the output vector (assumed that the output is contained in which_path).

If the user wants to summarize the simulation results of a respective output in the same way for each parameter combination, a list whose components are named after the outputs (the names of the returned list of fun in future_mc()) is supplied and each component is a function which only takes the vector of results of one output as the main argument.

If the user wants to summarize the simulation results of a respective output differently for different parameter combinations, a nested list has to be supplied. The components of the outer list must be equal in length and naming to the nice_names of the parameter combinations (see value of future_mc()) and each component is another list (inner list). The components of the inner list are then defined the same way as above (components named after the outputs and each component is a function).

The provided summary functions are not restricted regarding the complexity of their return value. However, the path of the summarized output over all simulation repetitions is only returned if the provided summary functions return a single numeric value (and the output is contained in which_path). Thus, plot.summary.mc() will only work in this specific case.

Value

A list of type summary.mc containing the result of the summary functions of the simulation results of a respective output and parameter combination.

If the provided summary functions return a single numeric value, the path of the summarized output (which are contained in which_path) over all simulation repetitions is also returned.

Examples

```
test_func <- function(param = 0.1, n = 100, x1 = 1, x2 = 2){

  data <- rnorm(n, mean = param) + x1 + x2
  stat <- mean(data)
  stat_2 <- var(data)

  if (x2 == 5){
    stop("x2 can't be 5!")
```

```
  }

  return(list(mean = stat, var = stat_2))
}

param_list <- list(param = seq(from = 0, to = 1, by = 0.5),
                   x1 = 1:2)

set.seed(101)
test_mc <- future_mc(
  fun = test_func,
  repetitions = 1000,
  param_list = param_list,
  n = 10,
  x2 = 2
)

summary(test_mc)
summary(test_mc, sum_funs = list(mean = mean, var = sd))

sum_funcs <- list(
  list(
    mean = mean, var = sd
  ),
  list(
    mean = mean, var = summary
  ),
  list(
    mean = max, var = min
  ),
  list(
    mean = mean, var = sd
  ),
  list(
    mean = mean, var = summary
  ),
  list(
    mean = max, var = min
  )
)

names(sum_funcs) <- test_mc$nice_names

summary(test_mc, sum_funs = sum_funcs)
```

`tidy_mc_latex` *Create a 'LaTeX' table with the summarized results of a Monte Carlo Simulation*

Description

Create a 'LaTeX' table containing the summarized results of a Monte Carlo simulation run by `future_mc()` and summarized by `summary.mc()`.

Usage

```
tidy_mc_latex(
  x,
  repetitions_set = NULL,
  which_setup = NULL,
  parameter_comb = NULL,
  which_out = NULL,
  kable_options = NULL
)
```

Arguments

`x`	An object of class `summary.mc`. For restrictions see details.
`repetitions_set`	A vector of integers specifying at which repetitions the summarized results should be displayed in the table. Default: The argument `repetitions` in `future_mc()`, which means that the summarized results after the last repetition are displayed in the table.
`which_setup`	A character vector containing the `nice_names` for the different parameter combinations (returned by `future_mc()`), which should be presented in the table. Default: All parameter combinations are presented.
`parameter_comb`	Alternative to `which_setup`. A named list whose components are named after (some of) the parameters in `param_list` in `future_mc()`. Each component is a vector containing the values for the parameters to be filtered by. Default: All parameter combinations are presented.
`which_out`	A character vector containing the names of (some of) the named outputs (the names of the returned list of `fun` in `future_mc()`), which should be displayed in the table. Default: All outputs are displayed.
`kable_options`	A list whose components are named after possible parameters of `kableExtra::kbl()` customizing the generated table.

Details

Only one of the arguments `which_setup` and `parameter_comb` can be specified at one time.

Only (output - parameter combination)-pairs for which the summary function specified in the `sum_funs` argument of `summary.mc()` returns a single scalar value appear as non-`NA` values in the 'LaTeX' table. If a specific output is summarized with functions that do not return a single numeric value over all parameter combinations, then this output is discarded from the table. Similarly, if for a specific parameter combination all `fun` outputs are summarized with functions which do not return a single numeric value, then this parameter combination is discarded as well. In summary, all outputs must be summarized with functions which return just one numeric value.

Value

An object of class `knitr_kable` which can be modified by the functions in the kableExtra::kableExtra package is returned.

Examples

```
test_func <- function(param = 0.1, n = 100, x1 = 1, x2 = 2) {
  data <- rnorm(n, mean = param) + x1 + x2
  stat <- mean(data)
  stat_2 <- var(data)

  if (x2 == 5) {
    stop("x2 can't be 5!")
  }

  return(list(mean = stat, var = stat_2))
}

param_list <- list(
  param = seq(from = 0, to = 1, by = 0.5),
  x1 = 1:2
)

test_mc <- future_mc(
  fun = test_func,
  repetitions = 1000,
  param_list = param_list,
  n = 10,
  x2 = 2
)

tidy_mc_latex(summary(test_mc))

set.seed(101)
tidy_mc_latex(
  summary(test_mc),
  repetitions_set = c(10, 1000),
  which_out = "mean",
  kable_options = list(caption = "Mean MCS results")
)
```

Index