Package ‘simTool’

May 17, 2020

Type Package

Title Conduct Simulation Studies with a Minimal Amount of Source Code

Version 1.1.6

Date 2020-05-16

Maintainer Marsel Scheer <scheer@freescience.de>

Description Tool for statistical simulations that have two components.
One component generates the data and the other one analyzes the data. The main aims of the package are the reduction of the administrative source code (mainly loops and management code for the results) and a simple applicability of the package that allows the user to quickly learn how to work with it. Parallel computing is also supported. Finally, convenient functions are provided to summarize the simulation results.

Depends R (>= 2.14.0)

Imports dplyr (>= 0.7.2), purrr (>= 0.2.3), tidyr (>= 1.0.0), tibble (>= 2.0.0), vctrs (>= 0.3.0), parallel, methods

Suggests ggplot2, knitr, boot, broom, testthat, rmarkdown

License GPL-3

VignetteBuilder knitr

RoxygenNote 7.1.0

URL https://github.com/MarselScheer/simTool

BugReports https://github.com/MarselScheer/simTool/issues

NeedsCompilation no

Author Marsel Scheer [aut, cre]

Repository CRAN

Date/Publication 2020-05-17 14:20:03 UTC
R topics documented:

- eval_tibbles ......................................................... 2
- expand_tibble ....................................................... 5
- print.eval_tibbles ................................................ 6

Index

| eval_tibbles | Workhorse for simulation studies |

Description

Generates data according to all provided constellations in data_tibble and applies all provided constellations in proc_tibble to them.

Usage

```r
eval_tibbles(
  data_grid,
  proc_grid = expand_tibble(proc = "length"),
  replications = 1,
  discard_generated_data = FALSE,
  post_analyze = identity,
  summary_fun = NULL,
  group_for_summary = NULL,
  ncpus = 1L,
  cluster = NULL,
  cluster_seed = rep(12345, 6),
  cluster_libraries = NULL,
  cluster_global_objects = NULL,
  envir = globalenv(),
  simplify = TRUE
)
```

Arguments

- `data_grid` a data.frame or tibble where the first column is a character vector with function names. The other columns contain parameters for the functions specified in the first column. Parameters with NA are ignored. If a column with name .truth exist, then the corresponding entry is passed to functions generated from proc_grid and the function specified in post_analyze.

- `proc_grid` similar as data_grid the first column must contain function names. The other columns contain parameters for the functions specified in the first column. The data generated according to data_grid will always be passed to the first unspecified argument of the functions specified in the first column of proc_grid. If a function specified in proc_grid has an argument .truth, then the corresponding entry in the .truth column from data_grid is passed to the .truth
parameter or if no column .truth exist in data_grid, then all parameters used for the data generation are passed to the .truth parameter.

replications number of replications for the simulation

discard_generated_data if TRUE the generated data is deleted after all function constellations in proc_grid have been applied. Otherwise, ALL generated data sets will be part of the returned object.

post_analyze this is a convenience function, that is applied directly after the data analyzing function. If this function has an argument .truth, then the corresponding entry in the .truth column from data_grid is passed to the .truth parameter or if no column .truth exist in data_grid, then all parameters used for the data generation are passed to the .truth parameter.

summary_fun named list of univariate function to summarize the results (numeric or logical) over the replications, e.g. list(mean = mean, sd = sd).

summary_fun this is a convenience function, that is applied directly after the data analyzing function. If this function has an argument .truth, then the corresponding entry in the .truth column from data_grid is passed to the .truth parameter or if no column .truth exist in data_grid, then all parameters used for the data generation are passed to the .truth parameter.

summary_fun this is a convenience function, that is applied directly after the data analyzing function. If this function has an argument .truth, then the corresponding entry in the .truth column from data_grid is passed to the .truth parameter or if no column .truth exist in data_grid, then all parameters used for the data generation are passed to the .truth parameter.

Value The returned object list of the class eval_tibbles, where the element simulations contain the results of the simulation.
Note

If cluster is provided by the user the function eval_tibbles will NOT stop the cluster. This has
to be done by the user. Conducting parallel simulations by specifying ncpus will internally create a
cluster and stop it after the simulation is done.

Author(s)

Marsel Scheer

Examples

rng <- function(data, ...) {
  ret <- range(data)
  names(ret) <- c("min", "max")
  ret
}

### The following line is only necessary
### if the examples are not executed in the global
### environment, which for instance is the case when
### the oneline-documentation
### is build. In such case eval_tibble() would search the
### above defined function rng() in the global environment where
### it does not exist!
eval_tibbles <- purrr::partial(eval_tibbles, envir = environment())

dg <- expand_tibble(fun = "rnorm", n = c(5L, 10L))
pg <- expand_tibble(proc = c("rng", "median", "length"))

eval_tibbles(dg, pg, rep = 2, simplify = FALSE)
eval_tibbles(dg, pg, rep = 2)
eval_tibbles(dg, pg,
  rep = 2,
  post_analyze = purrr::compose(as.data.frame, t)
)
eval_tibbles(dg, pg, rep = 2, summary_fun = list(mean = mean, sd = sd))

regData <- function(n, SD) {
  data.frame(
    x = seq(0, 1, length = n),
    y = rnorm(n, sd = SD)
  )
}

eg <- eval_tibbles(
  expand_tibble(fun = "regData", n = 5L, SD = 1:2),
  expand_tibble(proc = "lm", formula = c("y~x", "y-I(x^2)")),
  replications = 3
)

eg
presever_rownames <- function(mat) {
  rn <- rownames(mat)
  ret <- tibble::as_tibble(mat)
  ret$term <- rn
  ret
}

eg <- eval_tibbles(
  expand_tibble(fun = "regData", n = 5L, SD = 1:2),
  expand_tibble(proc = "lm", formula = c("y~x", "y~I(x^2)")),
  post_analyze = purrr::compose(presever_rownames, coef, summary),
  # post_analyze = broom::tidy, # is a nice out of the box alternative
  summary.fun = list(mean = mean, sd = sd),
  group_for_summary = "term",
  replications = 3
)

eg$simulation

dg <- expand_tibble(fun = "rexp", rate = c(10, 100), n = c(50L, 100L))
dg <- dplyr::bind_rows(
  expand_tibble(fun = "rexp", rate = 10, .truth = 1 / 10, n = c(50L, 100L)),
  expand_tibble(fun = "rnorm", .truth = 0, n = c(50L, 100L))
)

pg <- expand_tibble(proc = c("t.test"), conf.level = c(0.8, 0.9, 0.95))

et <- eval_tibbles(dg, pg,
  ncpus = 1,
  replications = 10^1,
  post_analyze = function(ttest, .truth) {
    mu <- 1 / .truth$rate
  },
  summary.fun = list(mean = mean, sd = sd)
)

et

dg <- expand_tibble(fun = "rexp", rate = c(10, 100), n = c(50L, 100L))

pg <- expand_tibble(proc = c("t.test"), conf.level = c(0.8, 0.9, 0.95))

et <- eval_tibbles(dg, pg,
  ncpus = 1,
  replications = 10^1,
  post_analyze = function(ttest, .truth) {
  },
  summary.fun = list(mean = mean, sd = sd)
)

et

### need to remove the locally adapted eval_tibbles()
### otherwise executing the examples would mask
### eval_tibbles from simTool-namespace.
rm(eval_tibbles)

---

**expand_tibble**

*Creates a tibble from All Combinations*
**Description**

Actually a wrapper for `expand.grid`, but character vectors will stay as characters.

**Usage**

`expand_tibble(...)`

**Arguments**

... vectors, factors or a list containing these.

**Value**

See `expand.grid` but instead of a `data.frame` a `tibble` is returned.

**Author(s)**

Marsel Scheer

**See Also**

`expand.grid`

**Examples**

```r
expand_tibble(fun = "rnorm", mean = 1:4, sd = 2:5)
```

---

**Description**

Prints objects created by `eval_tibbles()`

**Usage**

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

**Arguments**

- `x` object of class `eval_tibbles`
- `...` not used. only necessary to define the function consistently with respect to `print(x,...)`

**Author(s)**

Marsel Scheer
Index

data.frame, 6

eval_tibbles, 2
expand.grid, 6
expand_tibble, 5
print.eval_tibbles, 6
	ibble, 6