Package ‘simhelpers’

February 29, 2024

Type Package

Title Helper Functions for Simulation Studies

Version 0.2.1

Maintainer Megha Joshi <megha.j456@gmail.com>

Description
Calculates performance criteria measures and associated Monte Carlo standard errors for simulation results. Includes functions to help run simulation studies. Our derivation and explanation of formulas and our general simulation workflow is closely aligned with the approach described by Morris, White, and Crowther (2019) <DOI:10.1002/sim.8086>.

URL https://meghapsimatrix.github.io/simhelpers/index.html

BugReports https://github.com/meghapsimatrix/simhelpers/issues

Depends R (>= 2.10)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

SystemRequirements RStudio

Imports stats, furrr, tidyr, tibble, rstudioapi, Rdpack

Suggests dplyr, plyr, purrr, future, knitr, rmarkdown, pkgdown, covr, testthat, kableExtra, ggplot2, broom

RdMacros Rdpack

VignetteBuilder knitr

NeedsCompilation no

Author Megha Joshi [aut, cre] (<https://orcid.org/0000-0001-7936-076X>), James Pustejovsky [aut] (<https://orcid.org/0000-0003-0591-9465>)

Repository CRAN

Date/Publication 2024-02-29 21:10:06 UTC

R topics documented:

`alpha_res` *Cronbach's alpha simulation results*

Description

A dataset containing simulation results from estimating Cronbach's alpha and its variance.

Usage

```
alpha_res
```

Format

A tibble with 1,000 rows and 3 variables:

A estimate of alpha.

Var_A estimate of the variance of alpha.

true_param true alpha used to generate the data.

`bundle_sim` *Bundle functions into a simulation driver function*

Description

Bundle a data-generation function, a data-analysis function, and (optionally) a performance summary function into a simulation driver.

Usage

```
bundle_sim(
  f_generate,
  f_analyze,
  f_summarize = NULL,
  reps_name = "reps",
  seed_name = "seed",
  summarize_opt_name = "summarize",
  row_bind_reps = TRUE
)
```

Arguments

`f_generate`	function for data-generation
`f_analyze`	function for data-analysis. The first argument must be the data, in the format generated by `f_analyze()`.
`f_summarize`	function for calculating performance summaries across replications. The first argument must be the replicated data analysis results. Default is `NULL`, so that no summary function is used.
`reps_name`	character string to set the name of the argument for the number of replications, with a default value of `"reps"`.
`seed_name`	character string to set the name of the argument for the seed option, with a default value of `"seed"`. Set to `NULL` to remove the argument from the simulation driver.
`summarize_opt_name`	character string to set the name of the argument for where to apply `f_summarize` to the simulation results, with a default value of `TRUE`. Ignored if no `f_summarize` function is specified. Set to `NULL` to remove the argument from the simulation driver.
`row_bind_reps`	logical indicating whether to combine the simulation results into a data frame using `rbind()`, with a default value of `TRUE`. If `FALSE`, then the function will return replications in a list and so `f_summarize` must be able to take a list as its first argument.

Value

A function to repeatedly run the ‘f_generate‘ and ‘f_analyze‘ functions and (optionally) apply ‘f_summarize‘ to the resulting replications.

Examples

```
f_G <- rnorm
f_A <- function(x, trim = 0) data.frame(y_bar = mean(x, trim = trim))
f_S <- function(x, calc_sd = FALSE) {
  if (calc_sd) {
    res_SD <- apply(x, 2, sd)
    res <- data.frame(M = colMeans(x), SD = res_SD)
  } else {
    res <- data.frame(M = colMeans(x))
  }
  res
}

# bundle data-generation and data-analysis functions
sim1 <- bundle_sim(f_generate = f_G, f_analyze = f_A)
args(sim1)
res1 <- sim1(4, n = 70, mean = 0.5, sd = 1, trim = 0.2)
res1

# bundle data-generation, data-analysis, and performance summary functions
sim2 <- bundle_sim(f_generate = f_G, f_analyze = f_A, f_summarize = f_S)
args(sim2)
res2 <- sim2(24, n = 7, mean = 0, sd = 1, trim = 0.2, calc_sd = TRUE)
res2

# bundle data-generation and data-analysis functions, returning results as a list
sim3 <- bundle_sim(f_generate = f_G, f_analyze = f_A, row_bind_reps = FALSE)
args(sim3)
res3 <- sim3(4, n = 70, mean = 0.5, sd = 3, trim = 0.2)
res3
```

---

calc_absolute *Calculate absolute performance criteria and MCSE*

Description

Calculates absolute bias, variance, mean squared error (mse) and root mean squared error (rmse). The function also calculates the associated Monte Carlo standard errors.

Usage

```
calc_absolute(
  data,
  estimates,
  true_param,
  criteria = c("bias", "variance", "mse", "rmse")
)
```

Arguments

`data`	data frame or tibble containing the simulation results.
`estimates`	Vector or name of column from `data` containing point estimates.
`true_param`	Vector or name of column from `data` containing corresponding true parameters.
`criteria`	character or character vector indicating the performance criteria to be calculated.

Value

A tibble containing the number of simulation iterations, performance criteria estimate(s) and the associated MCSE.

Examples

```
calc_absolute(data = t_res, estimates = est, true_param = true_param)
```

`calc_coverage` *Calculate confidence interval coverage, width and MCSE*

Description

Calculates confidence interval coverage and width. The function also calculates the associated Monte Carlo standard errors. The confidence interval percentage is based on how you calculated the lower and upper bounds.

Usage

```
calc_coverage(
  data,
  lower_bound,
  upper_bound,
  true_param,
  criteria = c("coverage", "width")
)
```

Arguments

`data`	data frame or tibble containing the simulation results.
`lower_bound`	Vector or name of column from `data` containing lower bounds of confidence intervals.
`upper_bound`	Vector or name of column from `data` containing upper bounds of confidence intervals.
`true_param`	Vector or name of column from `data` containing corresponding true parameters.
`criteria`	character or character vector indicating the performance criteria to be calculated.

Value

A tibble containing the number of simulation iterations, performance criteria estimate(s) and the associated MCSE.

Examples

```
calc_coverage(data = t_res, lower_bound = lower_bound,
              upper_bound = upper_bound, true_param = true_param)
```

`calc_rejection` *Calculate rejection rate and MCSE*

Description

Calculates rejection rate. The function also calculates the associated Monte Carlo standard error.

Usage

```
calc_rejection(data, p_values, alpha = 0.05, format = "wide")
```

Arguments

`data`	data frame or tibble containing the simulation results.
`p_values`	Vector or name of column from `data` containing p-values.
`alpha`	Scalar or vector indicating the nominal alpha level(s). Default value is set to the conventional .05.
`format`	Option `"wide"` (the default) will produce a tibble with one row, with separate variables for each specified `alpha`. Option `"long"` will produce a tibble with one row per specified `alpha`.

Value

A tibble containing the number of simulation iterations, performance criteria estimate and the associated MCSE.

Examples

```
calc_rejection(data = t_res, p_values = p_val)
```

`calc_relative` *Calculate relative performance criteria and MCSE*

Description

Calculates relative bias, mean squared error (relative mse), and root mean squared error (relative rmse). The function also calculates the associated Monte Carlo standard errors.

Usage

```
calc_relative(
  data,
  estimates,
  true_param,
  criteria = c("relative bias", "relative mse", "relative rmse")
)
```

Arguments

`data`	data frame or tibble containing the simulation results.
`estimates`	Vector or name of column from `data` containing point estimates.
`true_param`	Vector or name of column from `data` containing corresponding true parameters.
`criteria`	character or character vector indicating the performance criteria to be calculated.

Value

A tibble containing the number of simulation iterations, performance criteria estimate(s) and the associated MCSE.

Examples

```
calc_relative(data = t_res, estimates = est, true_param = true_param)
```

`calc_relative_var` *Calculate jack-knife Monte Carlo SE for variance estimators*

Description

Calculates relative bias, mean squared error (relative mse), and root mean squared error (relative rmse) of variance estimators. The function also calculates the associated jack-knife Monte Carlo standard errors.

Usage

```
calc_relative_var(
  data,
  estimates,
  var_estimates,
  criteria = c("relative bias", "relative mse", "relative rmse")
)
```

Arguments

`data`	data frame or tibble containing the simulation results.
`estimates`	Vector or name of column from `data` containing point estimates.
`var_estimates`	Vector or name of column from `data` containing variance estimates for point estimator in `estimates`.
`criteria`	character or character vector indicating the performance criteria to be calculated.

Value

A tibble containing the number of simulation iterations, performance criteria estimate(s) and the associated MCSE.

Examples

```
calc_relative_var(data = alpha_res, estimates = A, var_estimates = Var_A)
```

`create_skeleton` *Open a simulation skeleton*

Description

Creates and opens a .R file containing a skeleton for writing a Monte Carlo simulation study.

Usage

```
create_skeleton()
```

Examples

```
## Not run:
create_skeleton()

## End(Not run)
```

`evaluate_by_row` *Evaluate a simulation function on each row of a data frame or tibble*

Description

Evaluates a simulation function on each row of a data frame or tibble containing parameter values. Returns a single tibble with parameters and simulation results. The function uses `furrr::future_pmap`, which allows for easy parallelization.

Usage

```
evaluate_by_row(
  params,
  sim_function,
  ...,
  results_name = ".results",
  .progress = FALSE,
  .options = furrr::furrr_options(),
  system_time = TRUE
)
```

Arguments

`params`	data frame or tibble containing simulation parameter values. Each row should represent a separate set of parameter values.
`sim_function`	function to be evaluated, with argument names matching the variable names in `params`. The function must return a `data.frame`, `tibble`, or vector.
`...`	additional arguments passed to `sim_function`.
`results_name`	character string to set the name of the column storing the results of the simulation. Default is `".results"`.
`.progress`	A single logical. Should a progress bar be displayed? Only works with multisession, multicore, and multiprocess futures. Note that if a multicore/multisession future falls back to sequential, then a progress bar will not be displayed. **Warning:** The `.progress` argument will be deprecated and removed in a future version of furrr in favor of using the more robust progressr package.
`.options`	The `future` specific options to use with the workers. This must be the result from a call to `furrr_options()`.
`system_time`	logical indicating whether to print computation time. `TRUE` by default.

Value

A tibble containing parameter values and simulation results.

Examples

```
df <- data.frame(
  n = 3:5,
  lambda = seq(8, 16, 4)
)

evaluate_by_row(df, rpois)
```

`Tipton_Pusto` *Results for Figure 2 of Tipton & Pustejovsky (2015)*

Description

A dataset containing simulation results comparing small sample correction methods for cluster robust variance estimation in meta-analysis.

Usage

```
Tipton_Pusto
```

Format

A tibble with 15,300 rows and 8 variables:

num_studies the number of studies included in the meta-analysis.

r correlation between outcomes.

Isq measure of heterogeneity of true effects.

contrast type of contrast that was tested.

test small sample method used.

q the number of parameters in the hypothesis test.

rej_rate the Type 1 error rate.

mcse the Monte Carlo standard error for the estimate of the Type 1 error rate.

Source

Tipton E, Pustejovsky JE (2015). “Small-Sample Adjustments for Tests of Moderators and Model Fit Using Robust Variance Estimation in Meta-Regression.” *Journal of Educational and Behavioral Statistics*, **40**(6), 604–634. ISSN 1076-9986, 1935-1054, doi:10.3102/1076998615606099, `https://journals.sagepub.com/doi/10.3102/1076998615606099`.

`t_res` *t-test simulation results*

Description

A dataset containing simulation results from a study that just runs a t-test.

Usage

```
t_res
```

Format

A tibble with 1,000 rows and 5 variables:

est estimate of the mean difference.

p_val p-value from the t-test.

lower_bound lower bound of the confidence interval.

upper_bound upper bound of the confidence interval.

true_param true mean difference used to generate the data.

`welch_res` *Welch t-test simulation results*

Description

A dataset containing simulation results from a study comparing Welch t-test to the conventional t-test.

Usage

```
welch_res
```

Format

A tibble with 16,000 rows and 11 variables:

n1 sample size for Group 1.

n2 sample size for Group 2.

mean_diff true difference in means of two groups used to generate the data.

iterations number of iterations.

seed seed used to generate data.

method indicates whether Welch or conventional t-test was used.

est estimate of the mean difference.

var variance of the estimate.

p_val p-value from the t-test.

lower_bound lower bound of the confidence interval.

upper_bound upper bound of the confidence interval.

Index