

Package ‘SimBaRepro’

May 7, 2026

Title Simulation-Based, Finite-Sample Inference via Repro Samples

Version 0.1.0

Description Functions for obtaining p-values (for hypothesis tests), confidence intervals, and multi-variate confidence sets. In particular, the method is compatible with differentially private dataset, as long as the privacy mechanism is known. For more details, see Awan and Wang (2024), ‘‘Simulation-based, Finite-sample Inference for Privatized Data’’, <[doi:10.48550/arXiv.2303.05328](https://doi.org/10.48550/arXiv.2303.05328)>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.2

Imports ddalpaha, ggplot2

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Author Xinlong Du [aut, cre],
Zhanyu Wang [aut],
Jordan Awan [aut]

Maintainer Xinlong Du <du339@purdue.edu>

Repository CRAN

Date/Publication 2025-06-29 10:20:06 UTC

Contents

confidence_grid	2
get_CI	3
grid_projection	5
plot_grid	6
p_value	7

Index	9
--------------	----------

confidence_grid *confidence_grid*

Description

returns the indicator array

Usage

```
confidence_grid(
  alpha,
  lower_bds,
  upper_bds,
  seeds,
  generating_fun,
  s_obs,
  tol,
  resolution,
  theta_init = NULL,
  T_stat = ma_depth
)
```

Arguments

alpha	A numeric representing the significance level of the test.
lower_bds	A vector containing the lower bounds for the parameter search space.
upper_bds	A vector containing the upper bounds for the parameter search space.
seeds	A matrix (or array) of seeds for generating artificial statistics.
generating_fun	A function that takes the random seeds above and a parameter in the search space as inputs to generate artificial statistics.
s_obs	A vector representing the observed statistic.
tol	A numeric specifying the tolerance of the confidence interval.
resolution	An integer specifying the mesh number of the search space.
theta_init	A vector specifying the starting point for the initial optim search.
T_stat	Default to the Mahalanobis distance. See Vignette for detailed explanation.

Value

A list containing an indicator array (`ind_array`) representing the confidence set, the confidence set lower bounds (`updated_lower_bds`), and the confidence set upper bounds (`updated_upper_bds`).

Examples

```

### Note that the examples may take a few seconds to run.
### Regular normal
set.seed(123)
n <- 50 # sample size
R <- 50 # Repro sample size (should be at least 200 for accuracy in practice)
alpha <- .05 # significance level
tol <- 0.01 # tolerance for the confidence set (use smaller tolerance in practice)
s_obs <- c(1.12, 0.67) # the observed sample mean
seeds <- matrix(rnorm(R * (n + 2)), nrow = R, ncol = n + 2) # pre-generated seeds

# this function computes the repro statistics given the seeds and the parameter
s_sample <- function(seeds, theta) {
  # generate the raw data points
  raw_data <- theta[1] + sqrt(theta[2]) * seeds[, 1:n]

  # compute the regular statistics
  s_mean <- apply(raw_data, 1, mean)
  s_var <- apply(raw_data, 1, var)

  return(cbind(s_mean, s_var))
}

lower_bds <- c(0.5, 0.3) # lower bounds for the parameter search region
upper_bds <- c(1.5, 1.3) # upper bounds for the parameter search region

resolution = 10 # resolution of the grid
result <- confidence_grid(alpha, lower_bds, upper_bds, seeds, s_sample, s_obs, tol, resolution)
print(result$ind_array)
print(result$search_lower_bds)
print(result$search_upper_bds)

```

get_CI

get_CI

Description

Given the observed statistic, this function computes a confidence interval given that the data generating process is known using the supplied random seeds.

Usage

```

get_CI(
  alpha,
  lower_bds,
  upper_bds,
  parameter_index,
  seeds,

```

```

    generating_fun,
    s_obs,
    tol,
    theta_init = NULL,
    T_stat = ma_depth,
    verbose = FALSE,
    check_input = TRUE
  )

```

Arguments

alpha	A numeric representing the significance level of the test.
lower_bds	A vector containing the lower bounds for the parameter search space.
upper_bds	A vector containing the upper bounds for the parameter search space.
parameter_index	An integer indicating the parameter of interest.
seeds	A matrix (or array) of seeds for generating artificial statistics.
generating_fun	A function that takes the random seeds above and a parameter in the search space as inputs to generate artificial statistics.
s_obs	A vector representing the observed statistic.
tol	A numeric specifying the tolerance of the confidence interval.
theta_init	A vector specifying the starting point for the initial optim search.
T_stat	Default to the Mahalanobis distance. See Vignette for detailed explanation.
verbose	A Boolean variable indicating whether or not to print out the optim messages.
check_input	A Boolean variable indicating whether or not to run checks on the function inputs.

Value

A length-2 vector representing the obtained confidence interval. In the case when no point is accepted in the search space, return NULL.

Examples

```

### Note that the examples may take a few seconds to run.
### Regular Normal
set.seed(123)
n <- 30 # sample size
R <- 50 # Repro sample size (should be at least 200 for accuracy in practice)
alpha <- .05 # significance level
tol <- 0.01 # tolerance for the confidence set (use smaller tolerance in practice)
s_obs <- c(1.12, 0.67) # the observed sample mean and variance
seeds <- matrix(rnorm(R * (n + 2)), nrow = R, ncol = n + 2) # pre-generated seeds

# this function computes the repro statistics given the seeds and the parameter
s_sample <- function(seeds, theta) {
  # generate the raw data points

```

```

raw_data <- theta[1] + sqrt(theta[2]) * seeds[, 1:n]

# compute the regular statistics
s_mean <- apply(raw_data, 1, mean)
s_var <- apply(raw_data, 1, var)

return(cbind(s_mean, s_var))
}

lower_bds <- c(-5, 0.01) # lower bounds for the parameter search region
upper_bds <- c(5, 5) # upper bounds for the parameter search region

# choose parameter_index = 1 to get the confidence interval for the mean
mean_CI <- get_CI(alpha, lower_bds, upper_bds, 1, seeds, s_sample, s_obs, tol)
print(mean_CI) # estimated confidence interval for mean
var_CI <- get_CI(alpha, lower_bds, upper_bds, 2, seeds, s_sample, s_obs, tol)
print(var_CI) # estimated confidence interval for variance

```

grid_projection	<i>grid_projection</i>
-----------------	------------------------

Description

Projects the multidimensional indicator array generated by `confidence_grid` down to 2d for visualization

Usage

```
grid_projection(indicator_array, index_set)
```

Arguments

`indicator_array` An indicator array generated using the `confidence_grid` function.

`index_set` A vector containing the indices representing the dimensions to keep.

Value

A two-dimensional indicator array ready for visualization (`indicator_array` projected onto the subspace specified by `index_set`).

Examples

```

### simple projection
ind_arr <- array(c(1, 0, 0, 0, 0, 1, 0, 1), dim = rep(2, 3))
print(ind_arr)
# project this indicator array onto a 2d subspace by first and second dimension
ind_arr_12 <- grid_projection(ind_arr, c(1,2))

```

```

print(ind_arr_12)
ind_arr_13 <- grid_projection(ind_arr, c(1,3))
print(ind_arr_13)
ind_arr_23 <- grid_projection(ind_arr, c(2,3))
print(ind_arr_23)

```

plot_grid

plot_grid

Description

projects the indicator array generated by confidence_grid down to 2d for visualization

Usage

```
plot_grid(indicator_array, lower_bds, upper_bds, parameter_names = NULL)
```

Arguments

indicator_array An 2-dimensional indicator array generated by confidence_grid or grid_projection.

lower_bds A vector containing the lower bounds for the parameter search space.

upper_bds A vector containing the upper bounds for the parameter search space.

parameter_names An optional vector argument specifying the names of each parameter

Value

A grid plot showing the confidence regions.

Examples

```

### Note that the examples may take a few seconds to run.
### Regular normal
set.seed(123)
n <- 50 # sample size
R <- 50 # Repro sample size (should be at least 200 for accuracy in practice)
alpha <- .05 # significance level
tol <- 1e-2 # tolerance for the confidence set
s_obs <- c(1.12, 0.67) # the observed sample mean
seeds <- matrix(rnorm(R * (n + 2)), nrow = R, ncol = n + 2) # pre-generated seeds

# this function computes the repro statistics given the seeds and the parameter
s_sample <- function(seeds, theta) {
  # generate the raw data points
  raw_data <- theta[1] + sqrt(theta[2]) * seeds[, 1:n]

  # compute the regular statistics
  s_mean <- apply(raw_data, 1, mean)

```

```

    s_var <- apply(raw_data, 1, var)

    return(cbind(s_mean, s_var))
}

lower_bds <- c(0.5, 0.4) # lower bounds for the parameter search region
upper_bds <- c(1.5, 1.4) # upper bounds for the parameter search region

resolution = 10 # resolution of the grid

result <- confidence_grid(alpha, lower_bds, upper_bds, seeds, s_sample, s_obs, tol, resolution)
ind_arr <- result$ind_array
parameter_names <- c("mean", "variance") # specifying the names of each parameter
plot_grid(ind_arr, lower_bds, upper_bds, parameter_names)

```

p_value

p_value

Description

Given the observed statistic and the given seeds, this function finds the p-value. The method uses simulation-based inference, where having fixed seeds, the parameter is searched which makes the observed statistics most "plausible". In particular, the `T_stat` function measures the "plausibility" of any data point and the procedure maximizes the rank of the observed `T_stat` value relative to the "repro" `T_stat` values. The p-value is determined from the maximum rank and the corresponding parameter is returned.

Usage

```

p_value(
  lower_bds,
  upper_bds,
  seeds,
  generating_fun,
  s_obs,
  theta_init = NULL,
  T_stat = ma_depth,
  verbose = FALSE,
  check_input = TRUE
)

```

Arguments

lower_bds	A vector containing the lower bounds for the parameter search space.
upper_bds	A vector containing the upper bounds for the parameter search space.
seeds	A matrix (or array) of seeds for generating artificial statistics.

<code>generating_fun</code>	A function that takes the random seeds above and a parameter in the search space as inputs to generate artificial statistics.
<code>s_obs</code>	A vector representing the observed statistic.
<code>theta_init</code>	A vector specifying the starting point for the initial <code>optim</code> search.
<code>T_stat</code>	See Vignette for detailed explanation.
<code>verbose</code>	A Boolean variable indicating whether or not to print out the <code>optim</code> messages.
<code>check_input</code>	A Boolean variable indicating whether or not to run checks on the function inputs.

Value

A list containing the most likely parameter in the search region (`theta_hat`) and its corresponding p-value (`p_val`).

Examples

```
### Regular Normal
set.seed(123)
n <- 50 # sample size
R <- 50 # Repro sample size (should be at least 200 for accuracy in practice)
s_obs <- c(1.12, 0.67) # the observed sample mean and variance
seeds <- matrix(rnorm(R * (n + 2)), nrow = R, ncol = n + 2) # pre-generated seeds

# this function computes the repro statistics given the seeds and the parameter
s_sample <- function(seeds, theta) {
  # generate the raw data points
  raw_data <- theta[1] + sqrt(theta[2]) * seeds[, 1:n]

  # compute the regular statistics
  s_mean <- apply(raw_data, 1, mean)
  s_var <- apply(raw_data, 1, var)

  return(cbind(s_mean, s_var))
}

lower_bds <- c(-5, 0.01) # lower bounds for null hypothesis region
upper_bds <- c(5, 5) # upper bounds for null hypothesis region

result <- p_value(lower_bds, upper_bds, seeds, s_sample, s_obs)
print(result$p_val) # the largest p_value found
print(result$theta_hat) # the parameter corresponding to the largest p value
```

Index

confidence_grid, 2

get_CI, 3

grid_projection, 5

p_value, 7

plot_grid, 6