Package 'mcstatsim'

July 29, 2024

Type Package

Title Monte Carlo Statistical Simulation Tools Using a Functional Approach

Version 0.5.0

Description A lightweight package designed to facilitate statistical simulations through functional programming. It centralizes the simulation process into a single higher-order function, enhancing manageability and usability without adding overhead from external dependencies. The package includes ready-to-use functions for common simulation targets. A detailed example can be found on <https://github.com/ielbadisy/mcstatsim>.

License AGPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.1

Imports pbapply

Suggests testthat (>= 3.0.0)

URL https://github.com/ielbadisy/mcstatsim

Config/testthat/edition 3

NeedsCompilation no

Author Imad EL BADISY [aut, cre, cph]

Maintainer Imad EL BADISY <elbadisyimad@gmail.com>

Repository CRAN

Date/Publication 2024-07-29 12:30:02 UTC

Contents

`calc_bias` *Calculate Bias and Bias Monte Carlo Standard Error*

Description

This function computes the bias and the Monte Carlo Standard Error (MCSE) of the bias for a set of estimates relative to a true parameter value. The bias is the difference between the mean of the estimates and the true parameter. The MCSE of the bias is calculated as the square root of the variance of the estimates divided by the number of estimates, providing a measure of the precision of the bias estimate.

Usage

```
calc_bias(estimates, true_param)
```

Arguments

`estimates`	A numeric vector of estimates from the simulation or sampling process.
`true_param`	The true parameter value that the estimates are intended to approximate.

Value

A list with two components: ‘bias‘, the calculated bias of the estimates, and ‘bias_mcse‘, the Monte Carlo Standard Error of the bias, indicating the uncertainty associated with the bias estimate.

Examples

```
estimates <- rnorm(100, mean = 50, sd = 10)
true_param <- 50
bias_info <- calc_bias(estimates, true_param)
print(bias_info)
```

`calc_coverage` *Calculate Coverage Probability and its Monte Carlo Standard Error*

Description

Computes the coverage probability of a confidence interval, defined as the proportion of times the true parameter value falls within the calculated lower and upper bounds across a set of simulations. Additionally, calculates the Monte Carlo Standard Error (MCSE) of the coverage probability to assess the uncertainty associated with this coverage estimate. This function is useful for evaluating the accuracy and reliability of confidence intervals generated by statistical models or estimation procedures.

Usage

```
calc_coverage(lower_bound, upper_bound, true_param)
```

Arguments

`lower_bound`	A numeric vector of lower bounds of confidence intervals.
`upper_bound`	A numeric vector of upper bounds of confidence intervals, corresponding to 'lower_bound'.
`true_param`	The true parameter value that the confidence intervals are intended to estimate.

Value

A list with two components: 'coverage', the calculated coverage probability of the confidence intervals, and 'coverage_mcse', the Monte Carlo Standard Error of the coverage. This MCSE provides a measure of the precision of the coverage probability estimate.

Examples

```
set.seed(123) # For reproducibility
estimates <- rnorm(100, mean = 50, sd = 10)
ci_lower <- estimates - 1.96 * 10
ci_upper <- estimates + 1.96 * 10
true_param <- 50
coverage_info <- calc_coverage(ci_lower, ci_upper, true_param)
print(coverage_info)
```

calc_mse *Calculate Mean Squared Error and its Monte Carlo Standard Error*

Description

Computes the Mean Squared Error (MSE) of a set of estimates relative to a true parameter value, along with the Monte Carlo Standard Error (MCSE) for the MSE. The MCSE takes into account the variance, skewness, and kurtosis of the estimates to provide a more accurate measure of uncertainty. This function is useful for assessing the accuracy of simulation or estimation methods by comparing the squared deviations of estimated values from a known parameter.

Usage

```
calc_mse(estimates, true_param)
```

Arguments

estimates	A numeric vector of estimates from a simulation or sampling process.
true_param	The true parameter value that the estimates are intended to approximate.

Value

A list with two components: ‘mse‘, the calculated Mean Squared Error of the estimates, and ‘mse_mcse‘, the Monte Carlo Standard Error of the MSE, offering insight into the reliability of the MSE calculation.

Examples

```
estimates <- rnorm(100, mean = 50, sd = 10)
true_param <- 50
mse_info <- calc_mse(estimates, true_param)
print(mse_info)
```

calc_rejection_rate *Calculate Rejection Rate and its Monte Carlo Standard Error*

Description

Computes the rejection rate of hypotheses tests based on a vector of p-values and a specified significance level (alpha). The rejection rate is the proportion of p-values that are lower than alpha, indicating significant results. Additionally, the function calculates the Monte Carlo Standard Error (MCSE) for the rejection rate, which quantifies the uncertainty associated with the estimated rejection rate. This function is useful for assessing the overall type I error rate or the power of a statistical test across multiple simulations or experimental replications.

Usage

```
calc_rejection_rate(p_values, alpha = 0.05)
```

Arguments

p_values	A numeric vector of p-values from multiple hypothesis tests.
alpha	The significance level used to determine if a p-value indicates a significant result. Default is 0.05.

Value

A list with two components: 'rejection_rate', the proportion of tests that resulted in rejection of the null hypothesis, and 'rejection_rate_mcse', the Monte Carlo Standard Error of the rejection rate, providing an estimate of its variability.

Examples

```
set.seed(123) # For reproducibility
p_values <- runif(100, min = 0, max = 1) # Simulated p-values
rejection_info <- calc_rejection_rate(p_values)
print(rejection_info)
```

---

calc_relative_bias *Calculate Relative Bias and its Monte Carlo Standard Error*

Description

Computes the relative bias of a set of estimates with respect to a true parameter value, along with the Monte Carlo Standard Error (MCSE) of the relative bias. Relative bias is the ratio of the mean of the estimates to the true parameter, providing a scale-independent measure of bias. This function is particularly useful for evaluating the accuracy of estimates in situations where the magnitude of the true parameter is crucial to the interpretation of bias. The function gracefully handles cases where the true parameter is zero by returning 'NA' for both relative bias and its MCSE, avoiding division by zero errors.

Usage

```
calc_relative_bias(estimates, true_param)
```

Arguments

estimates	A numeric vector of estimates from a simulation or sampling process.
true_param	The true parameter value that the estimates are intended to approximate. Note that 'true_param' must not be zero, as relative bias calculation involves division by the true parameter value.

Value

A list with two components: ‘rel_bias‘, the calculated relative bias of the estimates, and ‘rel_bias_mcse‘, the Monte Carlo Standard Error of the relative bias. If ‘true_param‘ is zero, both ‘rel_bias‘ and ‘rel_bias_mcse‘ will be ‘NA‘.

Examples

```
estimates <- rnorm(100, mean = 50, sd = 10)
true_param <- 50 # Non-zero true parameter
relative_bias_info <- calc_relative_bias(estimates, true_param)
print(relative_bias_info)
```

---

calc_relative_mse *Calculate Relative Mean Squared Error and its Monte Carlo Standard Error*

Description

Computes the Relative Mean Squared Error (Relative MSE) of a set of estimates with respect to a true parameter value, along with the Monte Carlo Standard Error (MCSE) of the Relative MSE. The Relative MSE is a normalized measure of error that scales the Mean Squared Error (MSE) by the square of the true parameter value, making it particularly useful for comparing the accuracy of estimates across different scales. The function also calculates the MCSE for the Relative MSE, taking into account the variance, skewness, and kurtosis of the estimates to provide a measure of uncertainty. The function returns ‘NA‘ for both Relative MSE and its MCSE if the true parameter is zero, to avoid division by zero.

Usage

```
calc_relative_mse(estimates, true_param)
```

Arguments

estimates	A numeric vector of estimates from a simulation or sampling process.
true_param	The true parameter value that the estimates are intended to approximate. Note that ‘true_param‘ must not be zero, as the calculation involves division by the true parameter value.

Value

A list with two components: ‘rel_mse‘, the calculated Relative Mean Squared Error of the estimates, and ‘rel_mse_mcse‘, the Monte Carlo Standard Error of the Relative MSE. If ‘true_param‘ is zero, both ‘rel_mse‘ and ‘rel_mse_mcse‘ will be ‘NA‘.

Examples

```
estimates <- rnorm(100, mean = 50, sd = 10)
true_param <- 50 # Non-zero true parameter
relative_mse_info <- calc_relative_mse(estimates, true_param)
print(relative_mse_info)
```

`calc_relative_rmse` *Calculate Relative Root Mean Squared Error and its Monte Carlo Standard Error*

Description

Computes the Relative Root Mean Squared Error (Relative RMSE) of a set of estimates with respect to a true parameter value. The Relative RMSE is derived from the Relative Mean Squared Error (MSE), providing a scale-independent measure of error that facilitates comparisons across different scales of the true parameter. This function is especially useful for evaluating the accuracy of estimates when the magnitude of the true parameter varies significantly across different scenarios. The function gracefully handles cases where the true parameter is zero by returning 'NA' for both Relative RMSE and its MCSE, to avoid division by zero. The MCSE for the Relative RMSE is not directly computed in this function and is marked as a placeholder for future implementation.

Usage

```
calc_relative_rmse(estimates, true_param)
```

Arguments

`estimates`	A numeric vector of estimates from a simulation or sampling process.
`true_param`	The true parameter value that the estimates are intended to approximate. Note that 'true_param' must not be zero, as the calculation involves division by the true parameter value.

Value

A list with two components: 'rel_rmse', the calculated Relative Root Mean Squared Error of the estimates, and 'rel_rmse_mcse', the Monte Carlo Standard Error of the Relative RMSE. The MCSE is currently not calculated and returned as 'NA'. This is a placeholder for future implementation.

Examples

```
estimates <- rnorm(100, mean = 50, sd = 10)
true_param <- 50 # Non-zero true parameter
relative_rmse_info <- calc_relative_rmse(estimates, true_param)
print(relative_rmse_info)
```

calc_rmse — *Calculate Root Mean Squared Error and its Monte Carlo Standard Error*

Description

Computes the Root Mean Squared Error (RMSE) of a set of estimates relative to a true parameter value, along with the Monte Carlo Standard Error (MCSE) for the RMSE. The RMSE is a measure of the accuracy of the estimates, representing the square root of the average squared differences between the estimated values and the true parameter. The MCSE for the RMSE is calculated using jackknife estimates, providing an assessment of the uncertainty associated with the RMSE value.

Usage

```
calc_rmse(estimates, true_param)
```

Arguments

estimates	A numeric vector of estimates from a simulation or sampling process.
true_param	The true parameter value that the estimates are intended to approximate.

Value

A list with two components: ‘rmse‘, the calculated Root Mean Squared Error of the estimates, and ‘rmse_mcse‘, the Monte Carlo Standard Error of the RMSE. This MCSE is derived from jackknife estimates, offering insight into the reliability of the RMSE calculation.

Examples

```
estimates <- rnorm(100, mean = 50, sd = 10)
true_param <- 50
rmse_info <- calc_rmse(estimates, true_param)
print(rmse_info)
```

calc_variance — *Calculate Variance and its Monte Carlo Standard Error*

Description

Computes the sample variance of a set of estimates and the Monte Carlo Standard Error (MCSE) for the variance. The MCSE is adjusted by the sample kurtosis to account for the shape of the distribution of the estimates. This function is particularly useful in simulation studies where understanding the variability of an estimator and the precision of this variability estimate is crucial.

Usage

```
calc_variance(estimates)
```

Arguments

estimates	A numeric vector of estimates from a simulation or sampling process.

Value

A list containing two elements: ‘variance‘, the calculated sample variance of the estimates, and ‘variance_mcse‘, the Monte Carlo Standard Error of the variance. The MCSE provides a measure of the uncertainty associated with the variance estimate, adjusted for kurtosis.

Examples

```
estimates <- rnorm(100, mean = 50, sd = 10)
variance_info <- calc_variance(estimates)
print(variance_info)
```

---

calc_width *Calculate Average Width of Confidence Intervals and its Monte Carlo Standard Error*

Description

This function computes the average width of a set of confidence intervals, represented by their lower and upper bounds, along with the Monte Carlo Standard Error (MCSE) of this average width. The average width provides insight into the precision of the estimation process, with narrower intervals typically indicating more precise estimates. The MCSE of the width offers a measure of the uncertainty associated with the average width calculation, useful for assessing the variability of interval estimates across simulations or bootstrap samples.

Usage

```
calc_width(lower_bound, upper_bound)
```

Arguments

lower_bound	A numeric vector of lower bounds of confidence intervals.
upper_bound	A numeric vector of upper bounds of confidence intervals, corresponding to ‘lower_bound‘.

Value

A list with two components: ‘width‘, the average width of the confidence intervals, and ‘width_mcse‘, the Monte Carlo Standard Error of the average width. This MCSE provides a quantification of the uncertainty in the average width estimate.

Examples

```
set.seed(123) # For reproducibility
estimates <- rnorm(100, mean = 50, sd = 10)
ci_lower <- estimates - 1.96 * 10
ci_upper <- estimates + 1.96 * 10
width_info <- calc_width(ci_lower, ci_upper)
print(width_info)
```

---

combine_df *Combine Dataframes in a Nested List*

Description

This function combines dataframes that are nested within a list of lists into a single dataframe.

Usage

```
combine_df(nested_list)
```

Arguments

nested_list	A list of lists, where each sublist contains dataframes to be combined.

Details

The function first checks if the input is a non-empty list of lists containing dataframes. It then iterates through each sublist, combining the dataframes and adding an ID column to indicate their source. Finally, all combined dataframes are row-bound into a single dataframe.

Value

A combined dataframe where each original dataframe is augmented with an ID column indicating its source list.

Examples

```
df1 <- data.frame(a = 1:3, b = 4:6)
df2 <- data.frame(a = 7:9, b = 10:12)
nested_list <- list(list(df1, df2), list(df1, df2))
combine_df(nested_list)
```

mcpmap *Parallel Map Function using pbapply::pbmapply*

Description

This function applies a given function over a list of parameters in parallel using multiple cores.

Usage

```
mcpmap(
  lists,
  func,
  num_cores = parallel::detectCores() - 1,
  show_progress = TRUE
)
```

Arguments

lists	A list of lists containing the parameters for the function.
func	The function to be applied.
num_cores	The number of cores to use for parallel execution. Default is one less than the total number of available cores.
show_progress	Logical indicating whether to display the progress bar. Default is TRUE.

Details

The function ensures that all elements in the list have the same length and uses ‘pbapply::pbmapply‘ for parallel processing. It sets the number of cores based on the operating system and then applies the function in parallel.

Value

A list of results from applying the function over the parameters.

Examples

```
params <- list(a = 1:3, b = 4:6)
mcpmap(params, function(a, b) a + b, num_cores = 2)
```

runsim *Run Monte Carlo Simulations in Parallel*

Description

This function executes a series of Monte Carlo simulations in parallel, providing detailed progress updates.

Usage

```
runsim(
  n,
  grid_params,
  sim_func,
  show_progress = TRUE,
  num_cores = parallel::detectCores() - 1
)
```

Arguments

n	The number of times the simulation function should be executed for each set of parameters. Must be a positive integer.
grid_params	A dataframe where each row corresponds to a unique combination of parameters for the simulation. Typically generated using ‘expand.grid‘.
sim_func	The simulation function to be applied. This function should accept parameters corresponding to a row in ‘grid_params‘ and return a dataframe or a list that can be row-bound.
show_progress	Logical indicating whether to display progress messages during the execution of the simulations.
num_cores	The number of cores to use for parallel execution. The default is one less than the total number of cores available on the system.

Details

The function first validates the input parameters. It then uses parallel processing to apply ‘sim_func‘ to each combination of parameters specified in ‘grid_params‘, repeating each simulation ‘n‘ times. The results are combined into a single dataframe.

Value

A combined dataframe of all simulation results.

Examples

```
## Not run:
library(mcstatsim)

# Define a simple simulation function
sim_function <- function(a, b) {
  Sys.sleep(0.1)  # Simulate a time-consuming process
  return(data.frame(result = a + b))
}

# Generate a grid of parameters
params <- expand.grid(a = 1:3, b = 4:6)

# Run simulations
results <- runsim(n = 1, grid_params = params, sim_func = sim_function)
print(results)

## End(Not run)
```

Index