Package 'r4pde'

May 9, 2026

Type Package

Title Companion to R for Plant Disease Epidemiology Book

Version 0.1.0

Description Datasets and utility functions to support the book
``R for Plant Disease Epidemiology'' (R4PDE). It includes functions for quantifying disease, assessing spatial patterns, and modeling plant disease epidemics based on weather predictors. These tools are intended for teaching and research in plant disease epidemiology. Several functions are based on classical and contemporary methods, including those discussed in Laurence V. Madden,
Gareth Hughes, and Frank van den Bosch (2007) <doi:10.1094/9780890545058>.

License MIT + file LICENSE

Depends R (>= 4.1.0)

Imports boot, car, cowplot, dplyr, ggplot2, igraph, interval,
lubridate, nasapower, progress, purrr, rlang, stats, survival,
tidyr

Suggests testthat, knitr, rmarkdown

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

URL https://github.com/emdelponte/r4pde

BugReports https://github.com/emdelponte/r4pde/issues

NeedsCompilation no

Author Emerson Del Ponte [aut, cre] (ORCID:
<https://orcid.org/0000-0003-4398-409X>)

Maintainer Emerson Del Ponte <delponte@ufv.br>

Repository CRAN

Date/Publication 2025-07-02 15:30:01 UTC

Contents

`AFSD` *Analysis of foci structure and dynamics (AFSD)*

Description

This function performs the analysis of a simple method introduced by Nelson (1996) and expanded by Laranjeira et al. (1998). The function assumes the dataframe supplied as input has columns 'x', 'y', and 'i', where 'x' and 'y' are spatial coordinates and 'i' is a disease indicator variable (1 if diseased, otherwise 0). The function performs several steps including filtering rows where 'i' is 1, converting to an adjacency matrix, and creating foci using igraph. It then calculates various statistics about the foci and returns these in a list.

Usage

```
AFSD(df)
```

Arguments

df	A dataframe containing at least three columns: 'x', 'y', and 'i'. 'x' and 'y' represent spatial coordinates and 'i' is a disease indicator (1 if diseased, otherwise 0).

Value

A list containing: cluster_summary2: a dataframe summarizing the number and size of foci, and proportions of diseased plants. cluster_df: a dataframe containing foci information, including size and number of rows and columns in each foci. df_clustered: the original dataframe with an added 'focus_id' column, showing which foci each row belongs to.

See Also

Other Spatial analysis: `BPL()`, `count_subareas()`, `count_subareas_random()`, `fit_gradients()`, `join_count()`, `oruns_test()`, `oruns_test_boustrophedon()`, `oruns_test_byrowcol()`, `plot_AFSD()`

Examples

```
# Generate a sample dataframe
set.seed(123)
df <- data.frame(x = sample(1:100, 500, replace = TRUE),
                 y = sample(1:100, 500, replace = TRUE),
                 i = sample(0:1, 500, replace = TRUE, prob = c(0.7, 0.3)))

# Perform the AFSD
result <- AFSD(df)
```

BlastWheat *BlastWheat dataset*

Description

Wheat blast dataset with severity and weather covariates.

Usage

```
BlastWheat
```

Format

A data frame with the following columns:

heading Date of heading

inc_mean Mean incidence

index_mean FHB index mean

latitude Latitude coordinate

location Experimental site name

longitude Longitude coordinate

state Brazilian state

study Study ID or code

year Crop year

yld_mean Mean yield

Source

Del Ponte Lab internal data

BPL *Binary Power Law Analysis for Spatial Disease Patterns*

Description

This function calculates the Binary Power Law (BPL) parameters for spatial disease patterns, fits a linear model, and performs a hypothesis test for the slope.

Usage

```
BPL(data)
```

Arguments

`data`	A data frame containing the following columns: • `field`: The field identifier. • `n`: The number of observations in each quadrat. • `i`: The incidence count in each quadrat.

Details

The function performs the following steps:

1. Summarizes the data by field to calculate the total number of observations (`n_total`), mean incidence (`incidence_mean`), observed variance (`V`), and binomial variance (`Vbin`).
2. Log-transforms the variances.
3. Fits a linear model to the log-transformed variances.
4. Tests the hypothesis that the slope of the linear model is equal to 1.

Value

A list containing the following elements:

- `summary`: A data frame summarizing the input data by field, including total observations (`n_total`), mean incidence (`incidence_mean`), observed variance (`V`), and binomial variance (`Vbin`).
- `model_summary`: A summary of the linear model fitted to the log-transformed variances.
- `hypothesis_test`: The result of the hypothesis test for the slope being equal to 1.
- `ln_Ap`: The intercept of the linear model, representing the natural logarithm of the parameter \(A_p \).
- `slope`: The slope of the linear model.

See Also

Other Spatial analysis: `AFSD()`, `count_subareas()`, `count_subareas_random()`, `fit_gradients()`, `join_count()`, `oruns_test()`, `oruns_test_boustrophedon()`, `oruns_test_byrowcol()`, `plot_AFSD()`

Examples

```
# Example usage with a sample data frame
result <- BPL(FHBWheat)
print(result$summary)
print(result$model_summary)
print(result$hypothesis_test)
print(paste("ln(Ap):", result$ln_Ap))
print(paste("Slope (b):", result$slope))
```

BudBlightSoybean *BudBlightSoybean dataset*

Description

Soybean bud blight incidence in experimental blocks.

Usage

```
BudBlightSoybean
```

Format

A data frame with the following columns:

block Block number

time Time point of assessment

treat Treatment name

y Incidence or severity value

Source

Del Ponte Lab internal data

`CompMuCens` *Survival analysis for quantitative ordinal scale data.*

Description

Survival analysis for quantitative ordinal scale data.

Usage

```
CompMuCens(dat, scale, grade = TRUE, ckData = FALSE)
```

Arguments

`dat`	Data frame containing the data to be processed.
`scale`	A numeric vector indicating the scale or order of classes.
`grade`	Logical. If TRUE, uses the class value. If FALSE, uses the NPE (Non-Parametric Estimate).
`ckData`	Logical. If TRUE, returns the input data along with the results. If FALSE, returns only the results.

Details

To assist plant pathologists in analyzing quantitative ordinal scale data and encourage the uptake of the interval-censored analysis method, Chiang and collaborators have developed this function and provided comprehensive explanation of the program code used to implement class ratings analyzed through this method in this repository: https://github.com/StatisticalMethodsinPlantProtection/CompMuCens According to results in the paper, the method can be applied to reduce the risk of type II errors when considering quantitative ordinal data, which are widely used in plant pathology and related disciplines.The function starts by converting the data into a censored data format and performs multiple pairwise comparisons to determine significance using the score statistic method.

Value

Returns a list containing the score statistic, hypothesis tests, adjusted significance level, and conclusion based on pairwise comparisons.

References

Chiang, K.S., Chang, Y.M., Liu, H.I., Lee, J.Y., El Jarroudi, M. and Bock, C., 2023. Survival Analysis as a Basis to Test Hypotheses When Using Quantitative Ordinal Scale Disease Severity Data. Phytopathology, in press. Available at: https://apsjournals.apsnet.org/doi/abs/10.1094/PHYTO-02-23-0055-R

See Also

Other Disease quantification: DSI(), DSI2()

Examples

```
# Entering your data as ordinal rating scores
trAs=c(5,4,2,5,5,4,4,2,5,2,2,3,4,3,2,2,6,2,2,4,2,4,2,4,5,3,4,2,2,3)
trBs=c(5,3,2,4,4,5,4,5,4,4,6,4,5,5,5,2,6,2,3,5,2,6,4,3,2,5,3,5,4,5)
trCs=c(2,3,1,4,1,1,4,1,1,3,2,1,4,1,1,2,5,2,1,3,1,4,2,2,2,4,2,3,2,2)
trDs=c(5,5,4,5,5,6,6,4,6,4,3,5,5,6,4,6,5,6,5,4,5,5,5,3,5,6,5,5,5,6)
# Data shaping into input format
inputData = data.frame(treatment=c(rep("A",30),rep("B",30),rep("C",30),
rep("D",30)), x=c(trAs, trBs, trCs, trDs))
# Perform analysis using CompMuCens() function
CompMuCens(dat=inputData, scale=c(0,3,6,12,25,50,75,88,94,97,100,100),ckData=TRUE)
```

---

count_subareas *Count the Number of Ones in Subareas of a Matrix*

Description

This function takes a binary matrix (0s and 1s) and divides it into rectangular subareas, counting the number of ones in each. Subareas are defined by the number of rows and columns specified by the user. If the matrix dimensions are not perfectly divisible by the subarea size, edge subareas may be smaller.

Usage

```
count_subareas(matrix_data, sub_rows, sub_cols)
```

Arguments

matrix_data	A matrix of 0s and 1s to analyze.
sub_rows	Number of rows in each subarea.
sub_cols	Number of columns in each subarea.

Value

A matrix where each cell corresponds to a subarea and contains the count of ones.

See Also

Other Spatial analysis: AFSD(), BPL(), count_subareas_random(), fit_gradients(), join_count(), oruns_test(), oruns_test_boustrophedon(), oruns_test_byrowcol(), plot_AFSD()

Examples

```
set.seed(123)
mat <- matrix(sample(c(0, 1), 12 * 16, replace = TRUE), nrow = 16, ncol = 12)
count_matrix <- count_subareas(mat, sub_rows = 3, sub_cols = 3)
print(count_matrix)
```

`count_subareas_random` *Random Subgrid Sampling of a Binary Matrix*

Description

Randomly samples submatrices (quadrats) of specified size from a binary matrix, and returns the positions, submatrices, and count of 1s in each sampled quadrat.

Usage

```
count_subareas_random(matrix_data, sub_rows = 3, sub_cols = 3, n_samples = 100)
```

Arguments

`matrix_data`	A binary matrix of 0s and 1s.
`sub_rows`	Number of rows in each subgrid sample.
`sub_cols`	Number of columns in each subgrid sample.
`n_samples`	Number of subgrid samples to draw.

Value

A list of sampled subgrids. Each element is a list with:

`position`	Row and column start position of the sample.
`submatrix`	The sampled subgrid matrix.
`count`	Number of 1s in the sampled submatrix.

See Also

Other Spatial analysis: `AFSD()`, `BPL()`, `count_subareas()`, `fit_gradients()`, `join_count()`, `oruns_test()`, `oruns_test_boustrophedon()`, `oruns_test_byrowcol()`, `plot_AFSD()`

DidymellaWatermelon *DidymellaWatermelon dataset*

Description

Assessment of Didymella symptoms in watermelon plots.

Usage

```
DidymellaWatermelon
```

Format

A data frame with:

EW_row Row position (east–west)

NS_col Column position (north–south)

dap Days after planting

severity Disease severity

Source

Del Ponte Lab internal data

DSI *Calculate the Disease Severity Index (DSI) (class for each unit)*

Description

This function calculates the Disease Severity Index (DSI) based on the provided unit, class, and maximum class value. The DSI is computed by aggregating the classes, calculating weights by multiplying the frequency of each class by the class itself, and then dividing the sum of these weights by the product of the total number of entries and the maximum class value, then multiplying by 100.

Usage

```
DSI(unit, class, max)
```

Arguments

`unit`	A vector representing the units.
`class`	A vector representing the classes corresponding to the units.
`max`	A numeric value representing the maximum possible class value.

Value

Returns a single numeric value representing the DSI.

See Also

Other Disease quantification: `CompMuCens()`, `DSI2()`

Examples

```
# Example usage:
unit <- c(1, 2, 3, 4, 5, 6)
class <- c(1, 2, 1, 2, 3, 1)
max <- 3
DSI(unit, class, max)
```

DSI2 *Calculate the Disease severity Index (DSI) (frequency of each class)*

Description

This function calculates the Disease Severity Index (DSI) given a vector of classes, a vector of frequencies, and a maximum possible class value. The DSI is calculated as a weighted sum of class values, where each class is multiplied by its corresponding frequency, then divided by the product of the total frequency and maximum class value, and finally multiplied by 100 to get a percentage.

Usage

```
DSI2(class, freq, max)
```

Arguments

`class`	A numeric vector representing the classes.
`freq`	A numeric vector representing the frequency of each class. Must be the same length as 'class'.
`max`	A numeric value representing the maximum possible class value.

Value

Returns a single numeric value representing the DSI.

See Also

Other Disease quantification: `CompMuCens()`, `DSI()`

Examples

```
DSI2(c(0, 1, 2, 3, 4), c(2, 0, 5, 0, 5), 4)
```

FHBWheat *FHBWheat dataset*

Description

Fusarium head blight quadrat assessments in wheat.

Usage

```
FHBWheat
```

Format

A data frame with:

field Field identifier

i Row position

n Column position

quadrat Quadrat ID

season Crop season

Source

Del Ponte Lab internal data

fit_gradients *Fit Gradient Models to Data*

Description

This function fits three gradient models (exponential, power, and modified power) to given data. It then ranks the models based on their R-squared values and returns diagnostic plots for each model.

Usage

```
fit_gradients(data, C = 1)
```

Arguments

data	A dataframe containing the data, with columns "x" representing distances and "Y" representing the corresponding measurements or counts.
C	A constant to be used in the modified power model. Defaults to 1.

Value

A list containing:

- `data` The input data, which will include an additional column 'mod_x'.
- `results_table` A table of the model parameters and R-squared values.
- `plot_exponential` Diagnostic plot for the exponential model.
- `plot_power` Diagnostic plot for the power model.
- `plot_modified_power` Diagnostic plot for the modified power model.
- `plot_exponential_original` Plot of the original data with the exponential model fit.
- `plot_power_original` Plot of the original data with the power model fit.
- `plot_modified_power_original` Plot of the original data with the modified power model fit.

See Also

Other Spatial analysis: `AFSD()`, `BPL()`, `count_subareas()`, `count_subareas_random()`, `join_count()`, `oruns_test()`, `oruns_test_boustrophedon()`, `oruns_test_byrowcol()`, `plot_AFSD()`

Examples

```
x <- c(0.8, 1.6, 2.4, 3.2, 4, 7.2, 12, 15.2, 21.6, 28.8)
Y <- c(184.9, 113.3, 113.3, 64.1, 25, 8, 4.3, 2.5, 1, 0.8)
grad1 <- data.frame(x = x, Y = Y)
library(ggplot2)
mg <- fit_gradients(grad1, C = 0.4)
mg$plot_power_original +
  labs(title = "", x = "Distance from focus (m)", y = "Count of lesions")
```

FusariumBanana *FusariumBanana dataset*

Description

Observations of Fusarium symptoms in banana fields.

Usage

```
FusariumBanana
```

Format

A data frame with:

field Field ID

lat Latitude

lon Longitude

marker Infection marker presence

Source

Del Ponte Lab internal data

---

get_nasapower *Fetch NASA POWER Data for Multiple Locations with a Progress Bar*

Description

This function downloads daily NASA POWER data for specified weather variables over a specified number of days around a given date column for multiple locations. It includes a progress bar to show the download progress.

Usage

```
get_nasapower(
  data,
  days_around,
  date_col,
  pars = c("T2M", "RH2M", "PRECTOTCORR", "T2M_MAX", "T2M_MIN", "T2MDEW")
)
```

Arguments

data	A data frame containing the input data, including columns for latitude, longitude, study identifier, and the date column.
days_around	An integer specifying the number of days before and after the date in the date column to download data.
date_col	A character string specifying the name of the date column in the data frame.
pars	A character vector specifying the weather variables to fetch from NASA POWER (default: c("T2M", "RH2M", "PRECTOTCORR", "T2M_MAX", "T2M_MIN", "T2MDEW")).

Details

The function uses the `get_power` function from the `nasapower` package to fetch weather data for a range of dates around the specified date column for each location. A progress bar is shown during the data download process, and the results are combined into a single data frame.

Value

A data frame with the downloaded weather data from NASA POWER, combined for all specified locations. Includes a new variable `study` indicating the study identifier from the input data. Returns an empty data frame if no data is retrieved.

See Also

Other Disease modeling: `windowpane()`

`join_count` *Test for Spatial Join Count Statistics*

Description

The function `join_count` calculates spatial join count statistics for a binary matrix, identifying patterns of aggregation or randomness.

Usage

```
join_count(matrix_data, verbose = TRUE)
```

Arguments

`matrix_data`	A binary matrix (with elements 0 and 1) representing the spatial distribution of two types of points: 0 for healthy plants (H) and 1 for diseased plants (D). This matrix reflects the geographical distribution or layout of plants in the studied area.
`verbose`	Logical. If TRUE (default), prints a formatted message to the console.

Details

The function conducts an analysis by first counting the occurrence of specific sequences ("01 or 10" and "11" - equivalent to HD and DD) in the binary matrix. It then calculates expected values, standard deviations, and Z-scores to determine the spatial randomness or aggregation. The analysis considers both horizontal and vertical adjacency (rook case) in the matrix.

Value

A comprehensive, rich-text formatted string of results that includes:

- Statistical counts of specific binary sequences (e.g., "01 or 10", "11")
- Expected counts under the assumption of Complete Spatial Randomness (CSR)
- Standard deviations and Z-scores (ZHD for "01 or 10" sequences, ZDD for "11" sequences)
- Interpretation of whether the spatial distribution for each sequence type is "Aggregated" or "Not Aggregated" based on Z-scores
- A summary explaining the implications of these statistics and patterns

The return value aims to provide a clear understanding of the spatial arrangement's characteristics, aiding in further spatial analysis or research.

References

Madden, L. V., Hughes, G., & van den Bosch, F. (2007). The Study of Plant Disease Epidemics. The American Phytopathological Society.

See Also

Other Spatial analysis: AFSD(), BPL(), count_subareas(), count_subareas_random(), fit_gradients(), oruns_test(), oruns_test_boustrophedon(), oruns_test_byrowcol(), plot_AFSD()

---

oruns_test *Runs Test*

Description

Perform a runs test on the input data to test for clustering or randomness.

Usage

```
oruns_test(x)
```

Arguments

x	A numeric vector representing the input data

Value

an r4pde.oruns object.

An r4pde.oruns object is a list containing:

- U, number of runs,
- EU, expected number of runs,
- sU, standard deviation of the expected number of runs
- Z, Z-score of the observed number of runs,
- pvalue, the p-value of the Z-score, and
- result, the test result of either "aggregation or clustering" or "randomness"

See Also

Other Spatial analysis: AFSD(), BPL(), count_subareas(), count_subareas_random(), fit_gradients(), join_count(), oruns_test_boustrophedon(), oruns_test_byrowcol(), plot_AFSD()

Examples

```
oruns_test(c(1, 0, 1, 1, 0, 1, 0, 0, 1, 1))
```

oruns_test_boustrophedon *Boustrophedon Run Test for Binary Matrix*

Description

Applies the ordinary runs test to a binary matrix using boustrophedon-style traversal. The function supports two modes: row-wise and column-wise boustrophedon. Each traversal flattens the matrix into a 1D sequence which is then tested using `oruns_test`.

Usage

```
oruns_test_boustrophedon(mat)
```

Arguments

`mat`	A binary matrix (containing 0s and 1s, and possibly NAs).

Value

A list with two elements:

`rowwise_boustrophedon`
: List containing the sequence and result of `oruns_test` for row-wise traversal.

`colwise_boustrophedon`
: List containing the sequence and result of `oruns_test` for column-wise traversal.

See Also

`oruns_test`

Other Spatial analysis: `AFSD()`, `BPL()`, `count_subareas()`, `count_subareas_random()`, `fit_gradients()`, `join_count()`, `oruns_test()`, `oruns_test_byrowcol()`, `plot_AFSD()`

---

oruns_test_byrowcol *Runs Test for Each Row and Column of a Binary Matrix*

Description

Applies the ordinary runs test to each row and column of a binary matrix individually.

Usage

```
oruns_test_byrowcol(mat)
```

Arguments

mat	A binary matrix (containing 0s and 1s, and possibly NAs).

Value

A list with four elements:

row_results	Data frame with test results for each row.
col_results	Data frame with test results for each column.
row_summary	Percentage summary of interpretation for rows.
col_summary	Percentage summary of interpretation for columns.

See Also

oruns_test

Other Spatial analysis: AFSD(), BPL(), count_subareas(), count_subareas_random(), fit_gradients(), join_count(), oruns_test(), oruns_test_boustrophedon(), plot_AFSD()

---

plot_AFSD *Plot ASFD*

Description

This function creates a tile plot of the foci (cluster) identified by the AFSD function. It colors each cell in a foci and labels the centroid of each cluster with the foci ID. The 'ggplot2' package is used for the plot, and will be automatically installed if not already present.

Usage

```
plot_AFSD(df)
```

Arguments

df	A dataframe containing at least three columns: 'x', 'y', and 'cluster_id'. 'x' and 'y' are spatial coordinates and 'cluster_id' is the cluster identifier to which each cell belongs.

Value

A ggplot object with the scatter plot of foci (clusters).

See Also

Other Spatial analysis: AFSD(), BPL(), count_subareas(), count_subareas_random(), fit_gradients(), join_count(), oruns_test(), oruns_test_boustrophedon(), oruns_test_byrowcol()

Examples

```
df <- data.frame(x = sample(1:100, 500, replace = TRUE),
                 y = sample(1:100, 500, replace = TRUE),
                 i = sample(0:1, 500, replace = TRUE, prob = c(0.7, 0.3)))

# Perform the AFSD
result <- AFSD(df)
# Plot the foci
plot_AFSD(result[[3]])
```

`RustSoybean` *RustSoybean dataset*

Description

Soybean rust severity and field metadata.

Usage

```
RustSoybean
```

Format

A data frame with:

detection Detection score or date

epidemia Epidemic phase

latitude Latitude

local Location name

longitude Longitude

planting Planting date or stage

severity Disease severity

Source

Del Ponte Lab internal data

SpatialAggregated *SpatialAggregated dataset*

Description

Simulated aggregated spatial binary disease pattern.

Usage

```
SpatialAggregated
```

Format

A data frame with:

x x-coordinate

y y-coordinate

Source

Simulated example

SpatialRandom *SpatialRandom dataset*

Description

Simulated random spatial binary disease pattern.

Usage

```
SpatialRandom
```

Format

A data frame with:

x x-coordinate

y y-coordinate

Source

Simulated example

theme_r4pde *Custom ggplot2 theme based on cowplot::theme_half_open*

Description

This function creates a new ggplot2 theme by modifying the cowplot::theme_half_open theme. It sets a custom font size and changes the panel background color to gray96.

Usage

```
theme_r4pde(font_size = 16)
```

Arguments

font_size	The base font size. Default is 16.

Value

A ggplot2 theme object.

WhiteMoldSoybean *WhiteMoldSoybean dataset*

Description

National dataset of white mold severity and yield.

Usage

```
WhiteMoldSoybean
```

Format

A data frame with:

country Country name

elevation Field elevation

elevation_class Elevation class

harvest_year Year of harvest

inc Incidence

inc_check Check plot incidence

inc_class Incidence class

location Location name

region Geographical region

scl Soybean canopy layer

season Crop season

state State name

study Study identifier

treat Treatment applied

yld Yield

yld_check Yield of untreated check

yld_class Yield class

Source

Del Ponte Lab internal data

`windowpane` *Window Pane for Epidemiological Analysis*

Description

This function calculates summary statistics within specified windows around a given end date in a dataset, facilitating epidemiological analysis. It allows backward, forward, or both directions of window calculations based on a user-defined variable and window lengths.

Usage

```
windowpane(
  data,
  end_date_col,
  date_col,
  variable,
  summary_type,
  threshold = NULL,
  window_lengths,
  direction = "backward",
  group_by_cols = NULL,
  date_format = "%Y-%m-%d"
)
```

Arguments

`data`	A data frame containing the input data.
`end_date_col`	A string specifying the name of the column representing the end date.
`date_col`	A string specifying the name of the column representing the date variable.
`variable`	A string specifying the name of the column for which summary statistics are calculated.

summary_type	A string specifying the type of summary to calculate. Options are "mean", "sum", "above_threshold", or "below_threshold".
threshold	Optional numeric value used when summary_type is "above_threshold" or "below_threshold".
window_lengths	A numeric vector specifying the window lengths (in days) for the calculations.
direction	A string specifying the direction of the window. Options are "backward" (default), "forward", or "both".
group_by_cols	Optional vector of strings specifying column names for grouping the data.
date_format	A string specifying the format of the date columns. Default is "%Y-%m-%d".

Value

A data frame with the calculated summary values for each window.

See Also

Other Disease modeling: `get_nasapower()`

`windowpane_tests` *Windowpane Tests for Correlation Analysis*

Description

This function performs bootstrapped correlation analysis for multiple predictors against a response variable. It applies the Simes method for global significance testing and calculates individual correlations, p-values, and bootstrap statistics.

Usage

```
windowpane_tests(
  data,
  response_var,
  corr_type = "spearman",
  R = 1000,
  global_alpha = 0.05,
  individual_alpha = 0.005
)
```

Arguments

data	A data frame containing the predictors and the response variable.
response_var	A string representing the name of the response variable in the data frame.
corr_type	A string specifying the correlation method to use; options are "spearman" (default), "pearson", or "kendall".
R	An integer indicating the number of bootstrap replications. Default is 1000.

`global_alpha` A numeric value representing the global alpha level for the Simes correction. Default is 0.05.

`individual_alpha`
A numeric value for the individual alpha threshold for testing individual predictors. Default is 0.005.

Details

The function calculates correlations between the response variable and each predictor in the data frame, using bootstrapping to generate mean, standard deviation, and median estimates of the correlation. The Simes correction is applied to control for multiple testing, providing a global p-value (Pg). The function also returns the maximum observed correlation.

Value

A list containing the following elements:

`results` A data frame with columns: `variable`, `correlation`, `p_value`, `mean_corr`, `sd_corr`, `median_corr`, `rank`, `simes_threshold`, `significant_simes`, and `individual_significant`.

`summary_table` A data frame summarizing the global p-value (Pg) and maximum correlation.

`global_significant`
A logical value indicating whether the global test is significant.

Index