Package 'phase.portrait'

November 14, 2024

Title Phase Portrait Analysis

Version 0.0.0.1000

Description Tools for generating tables and plots needed for phase portrait analysis.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports data.table,
metR,
dplyr,
tidyr,
viridis,
ggplot2,
parallel,
gridExtra,
deSolve,
Cairo

R topics documented:

calculate_derivatives *Calculate Rate of Change (Delta) in Values Over Time*

Description

This function calculates the rate of change (delta) in the `value` column over `pseudotime`. The calculation method can be specified as lead, lag, or lead-lag, allowing flexibility for datasets with varying temporal structures.

Arguments

`df`	A data frame with `value`, `pseudotime`, `sample`, `condition`, and `variable` columns.
`derivative`	A character string specifying the derivative method to calculate the rate of change. Options are: • `"lead"` (default): Uses the next timepoint to calculate the change. • `"lag"`: Uses the previous timepoint to calculate the change. • `"lead-lag"`: Calculates the change over two surrounding timepoints, useful for datasets with more than two timepoints.
`delta_name`	A character string specifying the name of the new column for the rate of change. Default is `"delta"`.
`save`	Logical, if TRUE, saves the resulting data frame to a file.

Value

A data frame with an added `delta` column (or specified name) representing the rate of change in `value` over `pseudotime`.

Examples

```
df <- data.frame(
  value = c(1, 2, 3, 4, 5),
  pseudotime = c(1, 2, 3, 4, 5),
  sample = "sample1",
  condition = "condition1",
  variable = "variable1"
)
calculate_derivatives(df, derivative = "lead-lag", delta_name = "rate_change")
```

iterate_plotting *Generate Multiple Plots with Fixed or Dynamic Axes This function iterates over a dataset and generates multiple plots using a specified plotting function. It supports fixed or dynamic axis limits based on the overall dataset or additional arguments provided. Each subset of data is passed to the plotting function along with optional parameters.*

Description

Generate Multiple Plots with Fixed or Dynamic Axes This function iterates over a dataset and generates multiple plots using a specified plotting function. It supports fixed or dynamic axis limits based on the overall dataset or additional arguments provided. Each subset of data is passed to the plotting function along with optional parameters.

Usage

```
iterate_plotting(
  df,
  fx,
  input = "standard",
  fixed_axes = TRUE,
  save = FALSE,
  n_cores = NULL,
  ...
)
```

Arguments

df	A data frame containing the dataset with x, y, x_label, and y_label columns.
fx	A function to generate individual plots. It should accept arguments df_split, input, save, and optionally min_x, max_x, min_y, max_y.
input	A character string specifying the input type, either "standard" or "fraction". Default is "standard".
fixed_axes	Logical. If TRUE, uses global axis limits for all plots. If FALSE, each plot can have dynamic axis limits. Default is TRUE.
save	Logical. If TRUE, saves each plot generated by the fx function. Default is FALSE.
n_cores	Integer. Specifies the number of cores to use for parallel processing. Default is NULL (automatically detects available cores).
...	Additional arguments passed to fx, allowing custom axis limits (min_x, max_x, min_y, max_y).

Value

A list of plots generated by the fx function, each created based on a subset of df.

Examples

```
# Example usage:
# plots <- iterate_plotting(df, fx = plot_phase_portrait, input = "standard", fixed_axes = TRUE)
```

lotka_volterra *Lotka-Volterra Model Simulation This function simulates the Lotka-Volterra predator-prey model over a specified time range and initial conditions for the populations of prey and predator species, generating a table of population dynamics for various starting conditions and parameter values.*

Description

This function generates an example dataset for this package, simulating several trajectories using predator-prey equations.

Usage

```
lotka_volterra(
  alpha = 0.5,
  beta = 0.8,
  gamma = 1,
  delta = 0.8,
  min_x = 1,
  max_x = 10,
  bin_width_x = 1,
  min_y = 1,
  max_y = 5,
  bin_width_y = 1,
  min_time = 0,
  max_time = 25,
  time_interval = 0.1
)
```

Arguments

alpha	Numeric. The growth rate of the prey population. Default is 1.1.
beta	Numeric. The rate of predation, describing how frequently predators consume prey. Default is 0.4.
gamma	Numeric. The mortality rate of the predator population. Default is 0.1.
delta	Numeric. The reproduction rate of predators per prey consumed. Default is 0.01.
min_x	Numeric. Minimum initial population of prey. Default is 10.
max_x	Numeric. Maximum initial population of prey. Default is 50.
bin_width_x	Numeric. Interval width for the prey population bins. Default is 10.
min_y	Numeric. Minimum initial population of predators. Default is 5.
max_y	Numeric. Maximum initial population of predators. Default is 25.
bin_width_y	Numeric. Interval width for the predator population bins. Default is 5.
min_time	Numeric. Starting time for the simulation. Default is 0.
max_time	Numeric. Ending time for the simulation. Default is 50.
time_interval	Numeric. Time increment for each simulation step. Default is 0.1.

Details

The Lotka-Volterra model describes the dynamics of biological systems in which two species interact, one as a predator and the other as prey. This function allows for exploring various initial conditions and parameter values to study the system's behavior under different scenarios.

Value

A data frame containing the time series of prey and predator populations for each initial condition and parameter set. The output includes columns for time, prey and predator populations, and initial conditions.

Examples

```
# Run the Lotka-Volterra simulation with default parameters
results <- lotka_volterra()

# Run with custom parameters
results <- lotka_volterra(alpha = 1.2, beta = 0.5, gamma = 0.15, delta = 0.02)
```

`normalize_values` *Normalize Values*

Description

This function normalizes the `value` column in a data frame using either max or min-max normalization. It scales the values based on the specified method and optionally saves the output as a compressed CSV file.

Usage

```
normalize_values(df, normalization = "max", save = FALSE)
```

Arguments

`df`	A data frame with a `value` column to be normalized.
`normalization`	A character string specifying the normalization method. Options are: • `"max"` (default): Scales `value` by dividing by the maximum absolute value. • `"min-max"`: Scales `value` to a 0-1 range by subtracting the minimum and dividing by the range.
`save`	Logical, if TRUE, saves the output as a compressed CSV file. Default is FALSE.

Value

A data frame with the normalized `value` column.

Examples

```
# Create an example data frame for normalization
df <- data.frame(
  value = c(1, 2, 3, 4, 5),
  variable = "var1",
  pseudotime = 1:5
)

# Apply min-max normalization
normalized_df <- normalize_values(df, normalization = "min-max")

# Apply max normalization and save the output as a CSV file
normalized_df <- normalize_values(df, normalization = "max", save = TRUE)
```

---

phase_lines *Generate Phase Lines for Visualization (Binned, Averaged, or All)*

Description

This function generates a data frame for visualizing phase lines by creating either binned, averaged, or all types of phase line summaries, using `x` and `y` variables or their fractional representations. The result can be used to create phase line plots with separate facets for `dx` or `dy`, with flexibility to filter based on a minimum number of observations per bin.

Usage

```
phase_lines(
  df,
  bin_width = 0.25,
  min_bin_n = 1,
  input = "standard",
  output = "all",
  save = FALSE
)
```

Arguments

`df`	A data frame containing columns: `x`, `y`, `dx`, `dy`, `x_variable`, `y_variable`, `x_label`, `y_label`, `dx_label`, `dy_label`.
`bin_width`	Integer specifying the number of bins to discretize `x` and `y`. Default is 4.
`min_bin_n`	Integer specifying the minimum number of observations required in each bin.
`input`	Character string specifying "standard" for direct values or "fraction" for fractional representation.
`output`	Character string specifying output type: "binned", "average", or "all".
`save`	Logical, if TRUE, saves the resulting data frame as a compressed CSV file. Default is FALSE.

Value

A data table with phase line summaries, including a column `output_type` indicating "binned" or "average".

Examples

```
# Example usage
phase_lines(df, bin_width = 5, min_bin_n = 2, input = "standard", output = "all")
```

`phase_portrait` *Generate Phase Portrait Summary (Standard or Fraction-Based)*

Description

This function generates a phase portrait by binning `x` and `y` variables into discrete intervals and calculating the median rate of change (`dx` and `dy`) within each bin. It supports both standard and fraction-based portraits, where `x` values can be shown as fractions of the total (`x + y`), to visualize dynamics across binned states.

Usage

```
phase_portrait(
  df,
  bin_width = 0.25,
  min_bin_n = 1,
  input = "standard",
  save = FALSE
)
```

Arguments

`df`	A data frame containing the following columns: • `condition`: Factor or character column indicating the experimental or observational condition. • `x`: Numeric column representing values for the x-axis variable. • `y`: Numeric column representing values for the y-axis variable. • `dx`: Numeric column representing the rate of change of `x`. • `dy`: Numeric column representing the rate of change of `y`. • `x_variable`, `y_variable`: Character or factor columns identifying the variables.
`bin_width`	Integer specifying the number of bins to discretize `x` and `y` variables. Default is 4.
`min_bin_n`	Integer specifying the minimum number of observations required in each bin for it to be included in the output. Default is 1.
`input`	Character string specifying the input of data, either "standard" for standard binning or "fraction" for fraction-based binning.
`save`	Logical, if TRUE, saves the resulting data frame as a compressed CSV file. Default is FALSE.

Value

A data frame with summarized phase portrait data containing:

- condition, x_variable, y_variable: Grouping columns from the input.
- x, y: Discretized values of x and y or fractional representation of x and total size as y.
- dx, dy: Median values of dx and dy within each bin.
- n: Number of observations in each bin.
- x_label, y_label: Labels for x and y variables, adapted based on the input.
- dx_label, dy_label: Labels for rate of change for x and y variables.
- input: The input of the data ("standard" or "fraction").

Examples

```
# Example usage
phase_portrait(df, bin_width = 5, min_bin_n = 2, input = "standard")
phase_portrait(df, bin_width = 5, min_bin_n = 2, input = "fraction")
```

---

phase_space *Generate Phase Space for Pairwise Variable Analysis with Parallel Processing*

Description

This function calculates the phase space for all pairwise combinations of variables within a data frame. It computes phase portraits by pairing each variable's values and their derivatives, then merges them into a combined phase space data frame. Parallel processing is supported for improved performance.

Usage

```
phase_space(df, n_cores = NULL, save = FALSE)
```

Arguments

df	A data frame containing the following columns: • value: Numeric values representing measurements or observations of interest. • delta: Numeric values representing the derivative (rate of change) of value. • variable: Factor or character column representing the variable or feature name. • condition: Factor or character column identifying different experimental or observational conditions. • sample: Factor or character column identifying different samples. • pseudotime: Numeric values representing temporal or sequential progression.
n_cores	Integer specifying the number of cores to use for parallel processing. If NULL, defaults to the total available cores minus two (if running on Unix). On Windows, n_cores is set to 1.
save	Logical, if TRUE, saves the output as a compressed CSV file. Default is FALSE.

Value

A data frame containing phase space information with columns:

- `x`: Values of the first variable in each pairwise combination.
- `dx`: Derivative of the first variable.
- `x_variable`: Name of the first variable.
- `y`: Values of the second variable in each pairwise combination.
- `dy`: Derivative of the second variable.
- `y_variable`: Name of the second variable.
- Additional columns: `condition`, `sample`, and `pseudotime`.

Examples

```
# Generate an example dataset for phase space analysis
df <- data.frame(
  value = c(runif(50, min = 0, max = 10), runif(50, min = 0, max = 10)),
  delta = c(rnorm(50, mean = 0.5), rnorm(50, mean = -0.5)),
  variable = rep(c("var1", "var2"), each = 50),
  condition = rep("cond1", 100),
  sample = rep(1:10, each = 10),
  pseudotime = rep(seq(1, 10, length.out = 10), 10)
)

# Generate phase space with default parameters
results <- phase_space(df)

# Generate phase space with parallel processing on 4 cores
results <- phase_space(df, n_cores = 4)
```

---

plot_all_trajectories *Plot All Trajectories in Phase Space*

Description

This function creates a phase space plot of all trajectories by visualizing `x` and `y` values across pseudotime. The plot can handle both standard and fraction-based input types, and provides options to customize axis limits, fix aspect ratio, and save the plot to a file.

Arguments

`df`	A data frame containing `x`, `y`, `pseudotime`, `sample`, `condition`, `x_variable`, and `y_variable`.
`input`	A character string specifying the input type: either `"standard"` or `"fraction"`. Default is `"standard"`.
`min_x`	Numeric. Minimum x-axis limit for the plot. Default is NULL.
`max_x`	Numeric. Maximum x-axis limit for the plot. Default is NULL.
`min_y`	Numeric. Minimum y-axis limit for the plot. Default is NULL.
`max_y`	Numeric. Maximum y-axis limit for the plot. Default is NULL.
`save`	Logical, if TRUE, saves the plot as a PDF file. Default is FALSE.
`fix_coord`	Logical, if TRUE enforces a fixed aspect ratio. Default is FALSE.

Value

A ggplot object of the phase space trajectories.

Examples

```
plot_all_trajectories(df, input = "standard", min_x = -10, max_x = 10, min_y = -5, max_y = 5, save = TRUE)
```

`plot_n` *Plot Cell Count (N) in Phase Space*

Description

This function creates a phase space plot displaying the count (`n`) of observations in each bin. The plot can be generated for either standard or fraction-based input types, and provides options to customize axis limits, enforce a fixed aspect ratio, and save the plot to a file.

Arguments

`df`	A data frame containing `x`, `y`, `n`, `condition`, `x_variable`, and `y_variable`.
`bin_width`	Integer specifying the number of bins to use for x and y axes. Default is 4.
`input`	A character string specifying the input type, either "standard" or "fraction". Default is "standard".
`min_x,`	max_x Optional numeric values to set limits for the x-axis.
`min_y,`	max_y Optional numeric values to set limits for the y-axis.
`save`	Logical, if TRUE, saves the plot as a PDF file. Default is FALSE.
`fix_coord`	Logical, if TRUE enforces a fixed aspect ratio. Default is FALSE.

Value

A ggplot object showing the cell count (`n`) in phase space.

Examples

```
plot_n(df, bin_width = 4, input = "standard", min_x = -10, max_x = 10, min_y = -5, max_y = 5, save = TRUE)
```

`plot_phase_line` *Plot Phase Lines in Phase Space*

Description

This function generates a plot of phase lines in phase space, showing changes in `x` and `y` values (change rate) along with the grouping variable. It supports both standard and fraction-based input types and provides options to adjust axis limits, customize the plot's color scale, and save the plot as a PDF.

Usage

```
plot_phase_line(
  df,
  input = "standard",
  min_x = NULL,
  max_x = NULL,
  min_y = NULL,
  max_y = NULL,
  save = FALSE,
  fix_coord = FALSE
)
```

Arguments

df	A data frame containing columns x, y, n, group_var, condition, facet, x_variable, and y_variable.
input	Character string specifying the input type, either "standard" or "fraction". Default is "standard".
min_x,	max_x Optional numeric values to set limits for the x-axis.
min_y,	max_y Optional numeric values to set limits for the y-axis.
save	Logical, if TRUE, saves the plot as a PDF file. Default is FALSE.

Value

A ggplot object representing phase lines in phase space.

Examples

```
plot_phase_line(df, input = "standard", min_x = -10, max_x = 10, min_y = -5, max_y = 5, save = TRUE)
```

---

plot_phase_portrait *Plot Phase Portrait*

Description

This function generates a phase portrait plot, visualizing the dynamics of two variables (x and y) with arrows representing the rate of change (dx and dy). The plot uses color to indicate the speed of change and can optionally enforce axis limits, aspect ratios, and save the plot to a file.

Usage

```
plot_phase_portrait(
  df,
  input = "standard",
  bin_width = 0.25,
  min_x = NULL,
  max_x = NULL,
  min_y = NULL,
  max_y = NULL,
  save = FALSE,
  fix_coord = FALSE
)
```

Arguments

df	A data frame containing x, y, dx, dy, condition, x_variable, and y_variable.
input	A character string indicating the type of input. Defaults to "standard".
bin_width	Numeric. Adjusts the length of arrows. Default is 0.25.
min_x	Numeric. Minimum x-axis limit for the plot. Default is NULL.
max_x	Numeric. Maximum x-axis limit for the plot. Default is NULL.
min_y	Numeric. Minimum y-axis limit for the plot. Default is NULL.
max_y	Numeric. Maximum y-axis limit for the plot. Default is NULL.
save	Logical, if TRUE saves the plot to a PDF file. Default is FALSE.
fix_coord	Logical, if TRUE enforces a fixed aspect ratio. Default is FALSE.

Value

A ggplot object showing the phase portrait.

Examples

```
plot_phase_portrait(df, input = "standard", min_x = -10, max_x = 10, min_y = -5, max_y = 5, save = TRUE)
```

---

plot_stream

Plot Phase Streamline This function creates a phase streamline plot, visualizing the dynamics of two variables (x and y) across conditions. It supports both standard and fraction-based input types, with options for binning, custom axis limits, fixed aspect ratio, and saving the plot.

Description

Plot Phase Streamline This function creates a phase streamline plot, visualizing the dynamics of two variables (x and y) across conditions. It supports both standard and fraction-based input types, with options for binning, custom axis limits, fixed aspect ratio, and saving the plot.

Usage

```
plot_stream(
  df,
  input = "standard",
  bin_width = 0.25,
  min_x = NULL,
  max_x = NULL,
  min_y = NULL,
  max_y = NULL,
  save = FALSE,
  fix_coord = FALSE,
  res = 25
)
```

Arguments

df	A data frame containing columns x, y, dx, dy, condition, x_variable, and y_variable.
input	A character string indicating the type of input, either "standard" or "fraction". Default is "standard".
bin_width	Integer specifying the number of bins for discretizing x and y. Default is 0.25.
min_x	Numeric. Minimum x-axis limit for the plot. Default is NULL.
max_x	Numeric. Maximum x-axis limit for the plot. Default is NULL.
min_y	Numeric. Minimum y-axis limit for the plot. Default is NULL.
max_y	Numeric. Maximum y-axis limit for the plot. Default is NULL.
save	Logical, if TRUE, saves the plot as a PDF file. Default is FALSE.

Value

A ggplot object showing the phase streamline.

Examples

```
plot_stream(df, input = "standard", bin_width = 4, min_x = 0, max_x = 1, min_y = 0, max_y = 1, save = TRUE)
```

---

plot_trajectory *Plot Trajectory in Phase Space*

Description

This function creates a phase space trajectory plot, visualizing x and y values along pseudotime. It connects x1 to x and x to x3 as well as y1 to y and y to y3, representing directional flows with arrows. This enables a clear depiction of how each point in phase space progresses over time.

Usage

```
plot_trajectory(
  df,
  input = "standard",
  min_x = NULL,
  max_x = NULL,
  min_y = NULL,
  max_y = NULL,
  save = FALSE,
  fix_coord = FALSE
)
```

Arguments

df	A dataframe containing columns for coordinates (x, y), pseudotime, and directional endpoints (x1, x3, y1, y3). Optionally includes input (plot label), condition (facet label), x_variable, and y_variable.
input	Character. Input type for the plot label. Default is "standard". If df$input exists, it will override this parameter.

min_x	Numeric. Minimum x-axis limit for the plot. Default is NULL.
max_x	Numeric. Maximum x-axis limit for the plot. Default is NULL.
min_y	Numeric. Minimum y-axis limit for the plot. Default is NULL.
max_y	Numeric. Maximum y-axis limit for the plot. Default is NULL.
save	Logical. If TRUE, saves the plot as a PDF file. Default is FALSE.

Details

This function creates a phase space trajectory plot where points and paths are visualized with arrows to indicate directionality. It uses pseudotime as a color gradient to show progression through phase space. Faceting by the `condition` column allows for separate plots by specified condition.

Value

A ggplot object of the trajectory plot, with customizable color and axis limits.

Examples

```
# Example usage:
df <- data.frame(
  x = c(1, 2, 3), y = c(3, 2, 1),
  x1 = c(1.1, 2.1, 3.1), x3 = c(0.9, 1.9, 2.9),
  y1 = c(3.1, 2.1, 1.1), y3 = c(2.9, 1.9, 0.9),
  pseudotime = c(0.1, 0.5, 0.9),
  condition = c("A", "B", "A"),
  x_variable = "X Axis", y_variable = "Y Axis",
  input = "custom_label"
)
plot_trajectory(df, min_x = 0, max_x = 5, min_y = 0, max_y = 5)
```

---

pseudotime_trajectory *Calculate Pseudotime Trajectory Summary Statistics*

Description

This function calculates summary statistics (median and quantiles) for `x` and `y` variables along a pseudotime trajectory within each condition, computing the 25th and 75th percentiles, as well as the median for each group, to analyze trends in `x` and `y` over pseudotime.

Usage

```
pseudotime_trajectory(df, input = "standard", save = FALSE)
```

Arguments

df	A data frame containing the following columns: • `condition`: Factor or character column indicating the experimental or observational condition. • `pseudotime`: Numeric column indicating pseudotime or a sequential progression. • `x`: Numeric column representing values for the x-axis variable.

- y: Numeric column representing values for the y-axis variable.
- x_variable: Character or factor column identifying the name of the x variable.
- y_variable: Character or factor column identifying the name of the y variable.

input	Either "standard" or "fraction" to specify the type of analysis.
save	Logical, if TRUE saves the resulting data frame as a CSV file. Default is FALSE.

Value

A data frame containing the calculated summary statistics for each combination of condition and pseudotime. Columns in the returned data frame include:

- x1: 25th percentile of x values.
- y1: 25th percentile of y values.
- x3: 75th percentile of x values.
- y3: 75th percentile of y values.
- x: Median of x values.
- y: Median of y values.
- n: Number of observations in each group.

Examples

```
# Example usage:
pseudotime_trajectory(df)
```

---

run_analysis *Run Analysis and Generate Phase Portrait Plots This function performs an in-depth analysis on a dataset to produce various visualizations, including phase portraits, trajectory plots, and other dynamic plots in phase space. Users can specify different analysis options (standard or fraction-based), save individual or combined plot outputs, and control the granularity of binning and parallel processing.*

Description

Run Analysis and Generate Phase Portrait Plots This function performs an in-depth analysis on a dataset to produce various visualizations, including phase portraits, trajectory plots, and other dynamic plots in phase space. Users can specify different analysis options (standard or fraction-based), save individual or combined plot outputs, and control the granularity of binning and parallel processing.

Usage

```
run_analysis(
  df,
  bin_width = 0.25,
  min_bin_n = 1,
  option = "all",
```

```
  phase_line_tables = "all",
  save_tables = TRUE,
  plot_individual = TRUE,
  plot_combined = TRUE,
  n_cores = NULL,
  stream_resolution = 25
)
```

Arguments

df	A data frame containing the dataset with columns x, y, dx, dy, x_variable, y_variable.
bin_width	Integer specifying the number of bins to discretize x and y in phase line calculations. Default is 4.
min_bin_n	Integer specifying the minimum number of observations required in each bin. Default is 1.
option	A character string specifying the analysis type. Options include "all" (default), "standard", or "fraction".
phase_line_tables	Character string specifying phase line table output format: "binned", "average", or "all". Default is "all".
save_tables	Logical. If TRUE, saves intermediary data tables generated during analysis. Default is FALSE.
plot_individual	Logical. If TRUE, saves each individual plot generated during the analysis. Default is FALSE.
plot_combined	Logical. If TRUE, saves combined plots into one file for easy comparison. Default is TRUE.
n_cores	Integer specifying the number of cores to use for parallel processing. Default is NULL (auto-detect).

Details

The run_analysis function processes and visualizes phase space data comprehensively, offering flexibility for standard and fraction-based representations. Customizable options include binning, minimum observations per bin, and parallel processing. Users can save intermediary data tables, individual plots, or combined plots as needed. The main components of this function include:

- **Phase Space Calculation**: Calls phase_space() to compute the core phase space dataset, using parallel processing if specified.
- **Data Preparation for Plotting**: Computes additional summary tables, including pseudotime_trajectory(), phase_portrait(), and phase_lines() based on the selected option. Standard and fraction-based versions are available, providing an in-depth view of phase space dynamics.
- **Plot Generation**: Generates various types of plots to visualize dynamics, including:
  - trajectory_plots: Shows pseudotime trajectories in phase space.
  - all_trajectory_plots: Displays all sample trajectories over pseudotime.
  - stream_plots: Shows dynamic phase portrait streams, illustrating variable changes over time.
  - phase_portrait_plots: Presents a static view of the phase portrait, summarizing variable interactions.

  - phase_line_plots: Visualizes phase lines across binned data, highlighting directional trends.
  - n_plots: Displays observation counts within each phase space bin.
- **Fraction-Based Plots**: Mirrors the standard plots but uses a fraction-based data representation, with plot types prefixed by fract_.
- **Plot Merging**: If plot_combined is TRUE, combines selected plots into a single layout for easier comparison, saved as a single PDF file.
- **Error Handling**: Each data processing and plotting step includes error handling, ensuring that issues are reported without stopping the entire workflow.
- **Parallel Processing**: Supports parallel processing with n_cores to optimize performance for large datasets.

This function is ideal for exploratory data analysis and visual interpretation of phase space dynamics, particularly for examining complex systems over time. Enabling save_tables or plot_individual can produce multiple file outputs for detailed analysis, while combined plots are saved as a single PDF for convenience.

Value

This function saves the generated plots and does not return a value.

Examples

```
# Example usage:
df <- data.frame(x = rnorm(100), y = rnorm(100), dx = rnorm(100), dy = rnorm(100),
                 x_variable = "X Axis", y_variable = "Y Axis")
run_analysis(df, option = "all", bin_width = 4, min_bin_n = 1, save_tables = TRUE, plot_individual = TRUE)
```

---

scale_and_diff *Normalize Values and Calculate Derivatives*

Description

This function normalizes the value column within each group in a data frame and calculates the derivative (difference) between consecutive values. It first shifts values within each group to compute differences, then performs min-max normalization for consistency across groups. Optionally, results can be saved as a compressed CSV file.

Usage

```
scale_and_diff(df, save = FALSE)
```

Arguments

df	A data frame with a value column to be normalized and a variable column for grouping.
save	Logical, if TRUE, saves the output as a compressed CSV file named "delta_lead.csv.gz". Default is FALSE.

Value

A data frame with normalized value and calculated delta columns.

Examples

```
# Create an example data frame for normalization and differentiation
df <- data.frame(
  value = c(1, 2, 3, 4, 5),
  variable = "var1",
  pseudotime = 1:5
)

# Apply normalization and differentiation
normalized_df <- scale_and_diff(df)

# Apply and save the output
normalized_df <- scale_and_diff(df, save = TRUE)
```

---

self_test *Run Self-Test for Phase Portrait Analysis Package*

Description

This function performs a self-test of the phase portrait analysis workflow. It simulates example data, processes it, and runs a comprehensive analysis with specified parameters, saving the resulting plots and data summaries.

Usage

```
self_test()
```

Details

The function uses the Lotka-Volterra model to generate an example dataset, normalizes and computes derivatives on the data, and then performs a complete analysis pipeline including phase portrait and trajectory plotting. The outputs are saved automatically as files for verification.

Value

No return value. Saves analysis results and plots to the working directory.

Examples

```
# Run the self-test to verify package functionality:
self_test()
```