Package ‘RandomWalker’

May 6, 2025

Title Generate Random Walks Compatible with the 'tidyverse'

Version 0.3.0

Description Generates random walks of various types by providing a set of functions that are compatible with the 'tidyverse'. The functions provided in the package make it simple to create random walks with a variety of properties, such as how many simulations to run, how many steps to take, and the distribution of random walk itself.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2.9000

URL https://www.spsanderson.com/RandomWalker/, https://github.com/spsanderson/RandomWalker

BugReports https://github.com/spsanderson/RandomWalker/issues

Depends R (>= 4.1.0)

Imports dplyr, tidyr, purrr, rlang, patchwork, NNS, ggiraph

Suggests knitr, rmarkdown, stats, ggplot2, tidyselect

VignetteBuilder knitr

NeedsCompilation no

Author Steven Sanderson [aut, cre, cph] (ORCID: <https://orcid.org/0009-0006-7661-8247>), Antti Rask [aut, cph]

Maintainer Steven Sanderson <spsanderson@gmail.com>

Repository CRAN

Date/Publication 2025-05-06 13:00:05 UTC

Contents

`brownian_motion` *Brownian Motion*

Description

Create a Brownian Motion Tibble

Usage

```
brownian_motion(
  .num_walks = 25,
  .n = 100,
  .delta_time = 1,
  .initial_value = 0,
  .dimensions = 1
)
```

Arguments

`.num_walks`	Total number of simulations.
`.n`	Total time of the simulation.
`.delta_time`	Time step size.
`.initial_value`	Integer representing the initial value.
`.dimensions`	The default is 1. Allowable values are 1, 2 and 3.

Details

Brownian Motion, also known as the Wiener process, is a continuous-time random process that describes the random movement of particles suspended in a fluid. It is named after the physicist Robert Brown, who first described the phenomenon in 1827.

The equation for Brownian Motion can be represented as:

```
W(t) = W(0) + sqrt(t) * Z
```

Where W(t) is the Brownian motion at time t, W(0) is the initial value of the Brownian motion, sqrt(t) is the square root of time, and Z is a standard normal random variable.

Brownian Motion has numerous applications, including modeling stock prices in financial markets, modeling particle movement in fluids, and modeling random walk processes in general. It is a useful tool in probability theory and statistical analysis.

Value

A tibble containing the generated random walks with columns depending on the number of dimensions:

- `walk_number`: Factor representing the walk number.
- `step_number`: Step index.
- `y`: If `.dimensions` = 1, the value of the walk at each step.
- `x`, `y`: If `.dimensions` = 2, the values of the walk in two dimensions.
- `x`, `y`, `z`: If `.dimensions` = 3, the values of the walk in three dimensions.

The following are also returned based upon how many dimensions there are and could be any of x, y and or z:

- `cum_sum`: Cumulative sum of `dplyr::all_of(.dimensions)`.
- `cum_prod`: Cumulative product of `dplyr::all_of(.dimensions)`.
- `cum_min`: Cumulative minimum of `dplyr::all_of(.dimensions)`.
- `cum_max`: Cumulative maximum of `dplyr::all_of(.dimensions)`.
- `cum_mean`: Cumulative mean of `dplyr::all_of(.dimensions)`.

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Generator Functions: `discrete_walk()`, `geometric_brownian_motion()`, `random_normal_drift_walk()`, `random_normal_walk()`

Examples

```
set.seed(123)
brownian_motion()

set.seed(123)
brownian_motion(.dimensions = 3) |>
  head() |>
  t()
```

---

cgmean *Cumulative Geometric Mean*

Description

A function to return the cumulative geometric mean of a vector.

Usage

```
cgmean(.x)
```

Arguments

`.x`	A numeric vector

Details

A function to return the cumulative geometric mean of a vector. `exp(cummean(log(.x)))`

Value

A numeric vector

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Vector Function: `chmean()`, `ckurtosis()`, `cmean()`, `cmedian()`, `crange()`, `csd()`, `cskewness()`, `cvar()`, `euclidean_distance()`, `kurtosis_vec()`, `rw_range()`, `skewness_vec()`

Examples

```
x <- mtcars$mpg

cgmean(x)
```

chmean — *Cumulative Harmonic Mean*

Description

A function to return the cumulative harmonic mean of a vector.

Usage

```
chmean(.x)
```

Arguments

`.x`	A numeric vector

Details

A function to return the cumulative harmonic mean of a vector. `1 / (cumsum(1 / .x))`

Value

A numeric vector

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Vector Function: `cgmean()`, `ckurtosis()`, `cmean()`, `cmedian()`, `crange()`, `csd()`, `cskewness()`, `cvar()`, `euclidean_distance()`, `kurtosis_vec()`, `rw_range()`, `skewness_vec()`

Examples

```
x <- mtcars$mpg

chmean(x)
```

ckurtosis *Cumulative Kurtosis*

Description

A function to return the cumulative kurtosis of a vector.

Usage

```
ckurtosis(.x)
```

Arguments

`.x`	A numeric vector

Details

A function to return the cumulative kurtosis of a vector.

Value

A numeric vector

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Vector Function: `cgmean()`, `chmean()`, `cmean()`, `cmedian()`, `crange()`, `csd()`, `cskewness()`, `cvar()`, `euclidean_distance()`, `kurtosis_vec()`, `rw_range()`, `skewness_vec()`

Examples

```
x <- mtcars$mpg

ckurtosis(x)
```

`cmean` *Cumulative Mean*

Description

A function to return the cumulative mean of a vector.

Usage

```
cmean(.x)
```

Arguments

`.x`	A numeric vector

Details

A function to return the cumulative mean of a vector. It uses `dplyr::cummean()` as the basis of the function.

Value

A numeric vector

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Vector Function: `cgmean()`, `chmean()`, `ckurtosis()`, `cmedian()`, `crange()`, `csd()`, `cskewness()`, `cvar()`, `euclidean_distance()`, `kurtosis_vec()`, `rw_range()`, `skewness_vec()`

Examples

```
x <- mtcars$mpg

cmean(x)
```

`cmedian` *Cumulative Median*

Description

A function to return the cumulative median of a vector.

Usage

```
cmedian(.x)
```

Arguments

`.x`	A numeric vector

Details

A function to return the cumulative median of a vector.

Value

A numeric vector

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Vector Function: `cgmean()`, `chmean()`, `ckurtosis()`, `cmean()`, `crange()`, `csd()`, `cskewness()`, `cvar()`, `euclidean_distance()`, `kurtosis_vec()`, `rw_range()`, `skewness_vec()`

Examples

```
x <- mtcars$mpg

cmedian(x)
```

`confidence_interval` *Confidence Interval*

Description

Calculate the confidence interval

Usage

```
confidence_interval(.vector, .interval = 0.95)
```

Arguments

`.vector`	A numeric vector of data points
`.interval`	A numeric value representing the confidence level (e.g., 0.95 for 95% confidence interval) The default is 0.95

Details

This function calculates the confidence interval for a given vector and interval.

Value

A named vector with the lower and upper bounds of the confidence interval

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Utility Functions: `convert_snake_to_title_case()`, `generate_caption()`, `get_attributes()`, `rand_walk_column_names()`, `rand_walk_helper()`, `running_quantile()`, `std_cum_max_augment()`, `std_cum_mean_augment()`, `std_cum_min_augment()`, `std_cum_prod_augment()`, `std_cum_sum_augment()`

Examples

```
confidence_interval(rnorm(100), 0.95)
```

convert_snake_to_title_case

Helper function to convert a snake_case string to Title Case

Description

Converts a snake_case string to Title Case.

Usage

```
convert_snake_to_title_case(string)
```

Arguments

`string`	A character string in snake_case format.

Details

This function is useful for formatting strings in a more readable way, especially when dealing with variable names or identifiers that use snake_case. This function takes a snake_case string and converts it to Title Case. It replaces underscores with spaces, capitalizes the first letter of each word, and replaces the substring "cum" with "cumulative" for better readability.

Value

A character string converted to Title Case.

Author(s)

Antti Lennart Rask

See Also

Other Utility Functions: `confidence_interval()`, `generate_caption()`, `get_attributes()`, `rand_walk_column_names()`, `rand_walk_helper()`, `running_quantile()`, `std_cum_max_augment()`, `std_cum_mean_augment()`, `std_cum_min_augment()`, `std_cum_prod_augment()`, `std_cum_sum_augment()`

Examples

```
convert_snake_to_title_case("hello_world") # "Hello World"
convert_snake_to_title_case("this_is_a_test") # "This Is A Test"
convert_snake_to_title_case("cumulative_sum") # "Cumulative Sum"
```

`crange` *Cumulative Range*

Description

A function to return the cumulative range of a vector.

Usage

```
crange(.x)
```

Arguments

`.x`	A numeric vector

Details

A function to return the cumulative range of a vector. It uses `max(.x[1:k]) - min(.x[1:k])` as the basis of the function.

Value

A numeric vector

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Vector Function: `cgmean()`, `chmean()`, `ckurtosis()`, `cmean()`, `cmedian()`, `csd()`, `cskewness()`, `cvar()`, `euclidean_distance()`, `kurtosis_vec()`, `rw_range()`, `skewness_vec()`

Examples

```
x <- mtcars$mpg

crange(x)
```

csd *Cumulative Standard Deviation*

Description

A function to return the cumulative standard deviation of a vector.

Usage

```
csd(.x)
```

Arguments

`.x`	A numeric vector

Details

A function to return the cumulative standard deviation of a vector.

Value

A numeric vector. Note: The first entry will always be NaN.

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Vector Function: `cgmean()`, `chmean()`, `ckurtosis()`, `cmean()`, `cmedian()`, `crange()`, `cskewness()`, `cvar()`, `euclidean_distance()`, `kurtosis_vec()`, `rw_range()`, `skewness_vec()`

Examples

```
x <- mtcars$mpg

csd(x)
```

`cskewness` *Cumulative Skewness*

Description

A function to return the cumulative skewness of a vector.

Usage

```
cskewness(.x)
```

Arguments

`.x`	A numeric vector

Details

A function to return the cumulative skewness of a vector.

Value

A numeric vector

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Vector Function: `cgmean()`, `chmean()`, `ckurtosis()`, `cmean()`, `cmedian()`, `crange()`, `csd()`, `cvar()`, `euclidean_distance()`, `kurtosis_vec()`, `rw_range()`, `skewness_vec()`

Examples

```
x <- mtcars$mpg

cskewness(x)
```

cvar *Cumulative Variance*

Description

A function to return the cumulative variance of a vector.

Usage

```
cvar(.x)
```

Arguments

`.x`	A numeric vector

Details

A function to return the cumulative variance of a vector. `exp(cummean(log(.x)))`

Value

A numeric vector. Note: The first entry will always be NaN.

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Vector Function: `cgmean()`, `chmean()`, `ckurtosis()`, `cmean()`, `cmedian()`, `crange()`, `csd()`, `cskewness()`, `euclidean_distance()`, `kurtosis_vec()`, `rw_range()`, `skewness_vec()`

Examples

```
x <- mtcars$mpg

cvar(x)
```

`discrete_walk` *Discrete Sampled Walk*

Description

The `discrete_walk` function generates multiple random walks over discrete time periods. Each step in the walk is determined by a probabilistic sample from specified upper and lower bounds. This function is useful for simulating stochastic processes, such as stock price movements or other scenarios where outcomes are determined by a random process.

Usage

```
discrete_walk(
  .num_walks = 25,
  .n = 100,
  .upper_bound = 1,
  .lower_bound = -1,
  .upper_probability = 0.5,
  .initial_value = 100,
  .dimensions = 1
)
```

Arguments

`.num_walks`	Total number of simulations.
`.n`	Total time of the simulation.
`.upper_bound`	The upper bound of the random walk.
`.lower_bound`	The lower bound of the random walk.
`.upper_probability`	The probability of the upper bound. Default is 0.5. The lower bound is calculated as 1 - .upper_probability.
`.initial_value`	The initial value of the random walk. Default is 100.
`.dimensions`	The default is 1. Allowable values are 1, 2 and 3.

Details

The function `discrete_walk` simulates random walks for a specified number of simulations (`.num_walks`) over a given total time (`.n`). Each step in the walk is either the upper bound or the lower bound, determined by a probability (`.upper_probability`). The initial value of the walk is set by the user (`.initial_value`), and the cumulative sum, product, minimum, and maximum of the steps are calculated for each walk. The results are returned in a tibble with detailed attributes, including the parameters used for the simulation.

Value

A tibble containing the generated random walks with columns depending on the number of dimensions:

- `walk_number`: Factor representing the walk number.
- `step_number`: Step index.
- `y`: If `.dimensions = 1`, the value of the walk at each step.
- `x`, `y`: If `.dimensions = 2`, the values of the walk in two dimensions.
- `x`, `y`, `z`: If `.dimensions = 3`, the values of the walk in three dimensions.

The following are also returned based upon how many dimensions there are and could be any of x, y and or z:

- `cum_sum`: Cumulative sum of `dplyr::all_of(.dimensions)`.
- `cum_prod`: Cumulative product of `dplyr::all_of(.dimensions)`.
- `cum_min`: Cumulative minimum of `dplyr::all_of(.dimensions)`.
- `cum_max`: Cumulative maximum of `dplyr::all_of(.dimensions)`.
- `cum_mean`: Cumulative mean of `dplyr::all_of(.dimensions)`.

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Generator Functions: `brownian_motion()`, `geometric_brownian_motion()`, `random_normal_drift_walk()`, `random_normal_walk()`

Examples

```
set.seed(123)
discrete_walk()

set.seed(123)
discrete_walk(.dimensions = 3) |>
  head() |>
  t()
```

`euclidean_distance` *Distance Calculations*

Description

A function to calculate the Euclidean distance between two vectors.

Usage

```
euclidean_distance(.data, .x, .y, .pull_vector = FALSE)
```

Arguments

`.data`	A data frame
`.x`	A numeric vector
`.y`	A numeric vector
`.pull_vector`	A boolean of TRUE or FALSE. Default is FALSE which will augment the distance to the data frame. TRUE will return a vector of the distances as the return.

Details

A function to calculate the Euclidean distance between two vectors. It uses the formula `sqrt((x - lag(x))^2 + (y - lag(y))^2)`. The function uses augments the data frame with a new column called `distance`.

Value

A numeric Vector of ditances

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Vector Function: `cgmean()`, `chmean()`, `ckurtosis()`, `cmean()`, `cmedian()`, `crange()`, `csd()`, `cskewness()`, `cvar()`, `kurtosis_vec()`, `rw_range()`, `skewness_vec()`

Examples

```
set.seed(123)
df <- rw30()
euclidean_distance(df, step_number, y)
euclidean_distance(df, step_number, y, TRUE) |> head(10)
```

generate_caption — *Helper function to generate a caption string based on provided attributes*

Description

Generates a caption string based on provided attributes.

Usage

```
generate_caption(attributes)
```

Arguments

`attributes`	A list containing various attributes that may include `dimension`, `num_steps`, `mu`, and `sd`.

Details

This function is useful for creating descriptive captions for plots or outputs based on the attributes provided. It ensures that only non-null attributes are included in the caption. This function constructs a caption string by checking various attributes provided in a list. It formats the caption based on the presence of specific attributes, such as dimensions, number of steps, and statistical parameters like mu and standard deviation (sd).

Value

A character string representing the generated caption. If no attributes are provided, it returns an empty string.

Author(s)

Antti Lennart Rask

See Also

Other Utility Functions: `confidence_interval()`, `convert_snake_to_title_case()`, `get_attributes()`, `rand_walk_column_names()`, `rand_walk_helper()`, `running_quantile()`, `std_cum_max_augment()`, `std_cum_mean_augment()`, `std_cum_min_augment()`, `std_cum_prod_augment()`, `std_cum_sum_augment()`

Examples

```
attrs <- list(dimension = 3, num_steps = 100, mu = 0.5, sd = 1.2)
generate_caption(attrs) # "3 dimensions, 100 steps, mu = 0.5, sd = 1.2."

attrs <- list(dimension = NULL, num_steps = 50, mu = NULL, sd = 2.0)
generate_caption(attrs) # "50 steps, sd = 2.0."
```

geometric_brownian_motion *Geometric Brownian Motion*

Description

Create a Geometric Brownian Motion.

Usage

```
geometric_brownian_motion(
  .num_walks = 25,
  .n = 100,
  .mu = 0,
  .sigma = 0.1,
  .initial_value = 100,
  .delta_time = 0.003,
  .dimensions = 1
)
```

Arguments

`.num_walks`	Total number of simulations.
`.n`	Total time of the simulation, how many `n` points in time.
`.mu`	Expected return
`.sigma`	Volatility
`.initial_value`	Integer representing the initial value.
`.delta_time`	Time step size.
`.dimensions`	The default is 1. Allowable values are 1, 2 and 3.

Details

Geometric Brownian Motion (GBM) is a statistical method for modeling the evolution of a given financial asset over time. It is a type of stochastic process, which means that it is a system that undergoes random changes over time.

GBM is widely used in the field of finance to model the behavior of stock prices, foreign exchange rates, and other financial assets. It is based on the assumption that the asset's price follows a random walk, meaning that it is influenced by a number of unpredictable factors such as market trends, news events, and investor sentiment.

The equation for GBM is:

```
dS/S = mdt + sdW
```

where S is the price of the asset, t is time, m is the expected return on the asset, s is the volatility of the asset, and dW is a small random change in the asset's price.

GBM can be used to estimate the likelihood of different outcomes for a given asset, and it is often used in conjunction with other statistical methods to make more accurate predictions about the future performance of an asset.

This function provides the ability of simulating and estimating the parameters of a GBM process. It can be used to analyze the behavior of financial assets and to make informed investment decisions.

Value

A tibble containing the generated random walks with columns depending on the number of dimensions:

- `walk_number`: Factor representing the walk number.
- `step_number`: Step index.
- `y`: If `.dimensions = 1`, the value of the walk at each step.
- `x`, `y`: If `.dimensions = 2`, the values of the walk in two dimensions.
- `x`, `y`, `z`: If `.dimensions = 3`, the values of the walk in three dimensions.

The following are also returned based upon how many dimensions there are and could be any of x, y and or z:

- `cum_sum`: Cumulative sum of `dplyr::all_of(.dimensions)`.
- `cum_prod`: Cumulative product of `dplyr::all_of(.dimensions)`.
- `cum_min`: Cumulative minimum of `dplyr::all_of(.dimensions)`.
- `cum_max`: Cumulative maximum of `dplyr::all_of(.dimensions)`.
- `cum_mean`: Cumulative mean of `dplyr::all_of(.dimensions)`.

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Generator Functions: `brownian_motion()`, `discrete_walk()`, `random_normal_drift_walk()`, `random_normal_walk()`

Examples

```
set.seed(123)
geometric_brownian_motion()

set.seed(123)
geometric_brownian_motion(.dimensions = 3) |>
  head() |>
  t()
```

`get_attributes` *Get Attributes*

Description

The `get_attributes` function takes an R object as input and returns its attributes, omitting the row.names attribute.

Usage

```
get_attributes(.data)
```

Arguments

`.data`	An R object from which attributes are to be extracted.

Details

This function retrieves the attributes of a given R object, excluding the row.names attribute.

Value

A list of attributes of the input R object, excluding row.names.

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Utility Functions: `confidence_interval()`, `convert_snake_to_title_case()`, `generate_caption()`, `rand_walk_column_names()`, `rand_walk_helper()`, `running_quantile()`, `std_cum_max_augment()`, `std_cum_mean_augment()`, `std_cum_min_augment()`, `std_cum_prod_augment()`, `std_cum_sum_augment()`

Examples

```
get_attributes(rw30())
get_attributes(iris)
get_attributes(mtcars)
```

kurtosis_vec *Compute Kurtosis of a Vector*

Description

This function takes in a vector as it's input and will return the kurtosis of that vector. The length of this vector must be at least four numbers. The kurtosis explains the sharpness of the peak of a distribution of data.
`((1/n) * sum(x - mu})^4) / ((()1/n) * sum(x - mu)^2)^2`

Usage

```
kurtosis_vec(.x)
```

Arguments

`.x`	A numeric vector of length four or more.

Details

A function to return the kurtosis of a vector.

Value

The kurtosis of a vector

Author(s)

Steven P. Sanderson II, MPH

See Also

https://en.wikipedia.org/wiki/Kurtosis

Other Vector Function: `cgmean()`, `chmean()`, `ckurtosis()`, `cmean()`, `cmedian()`, `crange()`, `csd()`, `cskewness()`, `cvar()`, `euclidean_distance()`, `rw_range()`, `skewness_vec()`

Examples

```
set.seed(123)
kurtosis_vec(rnorm(100, 3, 2))
```

random_normal_drift_walk

Generate Multiple Random Walks with Drift

Description

This function generates a specified number of random walks, each consisting of a specified number of steps. The steps are generated from a normal distribution with a given mean and standard deviation. An additional drift term is added to each step to introduce a consistent directional component to the walks.

Usage

```
random_normal_drift_walk(
  .num_walks = 25,
  .n = 100,
  .mu = 0,
  .sd = 1,
  .drift = 0.1,
  .initial_value = 0,
  .dimensions = 1
)
```

Arguments

`.num_walks`	Integer. The number of random walks to generate. Default is 25.
`.n`	Integer. The number of steps in each random walk. Default is 100.
`.mu`	Numeric. The mean of the normal distribution used for generating steps. Default is 0.
`.sd`	Numeric. The standard deviation of the normal distribution used for generating steps. Default is 1.
`.drift`	Numeric. The drift term to be added to each step. Default is 0.1.
`.initial_value`	A numeric value indicating the initial value of the walks. Default is 0.
`.dimensions`	The default is 1. Allowable values are 1, 2 and 3.

Details

This function generates multiple random walks with a specified drift. Each walk is generated using a normal distribution for the steps, with an additional drift term added to each step.

Value

A tibble containing the generated random walks with columns depending on the number of dimensions:

- `walk_number`: Factor representing the walk number.

- `step_number`: Step index.
- `y`: If `.dimensions` = 1, the value of the walk at each step.
- `x`, `y`: If `.dimensions` = 2, the values of the walk in two dimensions.
- `x`, `y`, `z`: If `.dimensions` = 3, the values of the walk in three dimensions.

The following are also returned based upon how many dimensions there are and could be any of x, y and or z:

- `cum_sum`: Cumulative sum of `dplyr::all_of(.dimensions)`.
- `cum_prod`: Cumulative product of `dplyr::all_of(.dimensions)`.
- `cum_min`: Cumulative minimum of `dplyr::all_of(.dimensions)`.
- `cum_max`: Cumulative maximum of `dplyr::all_of(.dimensions)`.
- `cum_mean`: Cumulative mean of `dplyr::all_of(.dimensions)`.

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Generator Functions: `brownian_motion()`, `discrete_walk()`, `geometric_brownian_motion()`, `random_normal_walk()`

Examples

```
set.seed(123)
random_normal_drift_walk()

set.seed(123)
random_normal_drift_walk(.dimensions = 3) |>
  head() |>
  t()
```

`random_normal_walk` *Generate Multiple Random Normal Walks in Multiple Dimensions*

Description

The `random_normal_walk` function generates multiple random walks in 1, 2, or 3 dimensions. Each walk is a sequence of steps where each step is a random draw from a normal distribution. The user can specify the number of walks, the number of steps in each walk, and the parameters of the normal distribution (mean and standard deviation). The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

Usage

```
random_normal_walk(
  .num_walks = 25,
  .n = 100,
  .mu = 0,
  .sd = 0.1,
  .initial_value = 0,
  .samp = TRUE,
  .replace = TRUE,
  .sample_size = 0.8,
  .dimensions = 1
)
```

Arguments

`.num_walks`	An integer specifying the number of random walks to generate. Default is 25.
`.n`	An integer specifying the number of steps in each walk. Default is 100.
`.mu`	A numeric value indicating the mean of the normal distribution. Default is 0.
`.sd`	A numeric value indicating the standard deviation of the normal distribution. Default is 0.1.
`.initial_value`	A numeric value indicating the initial value of the walks. Default is 0.
`.samp`	A logical value indicating whether to sample the normal distribution values. Default is TRUE.
`.replace`	A logical value indicating whether sampling is with replacement. Default is TRUE.
`.sample_size`	A numeric value between 0 and 1 specifying the proportion of `.n` to sample. Default is 0.8.
`.dimensions`	An integer specifying the number of dimensions (1, 2, or 3). Default is 1.

Value

A tibble containing the generated random walks with columns depending on the number of dimensions:

- `walk_number`: Factor representing the walk number.
- `step_number`: Step index.
- `y`: If `.dimensions = 1`, the value of the walk at each step.
- `x`, `y`: If `.dimensions = 2`, the values of the walk in two dimensions.
- `x`, `y`, `z`: If `.dimensions = 3`, the values of the walk in three dimensions.

The following are also returned based upon how many dimensions there are and could be any of x, y and or z:

- `walk_number`: Factor representing the walk number.
- `x`: Step index.

- `y`: Normal distribution values.
- `cum_sum`: Cumulative sum of `y`.
- `cum_prod`: Cumulative product of `y`.
- `cum_min`: Cumulative minimum of `y`.
- `cum_max`: Cumulative maximum of `y`.

The tibble includes attributes for the function parameters.

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Generator Functions: `brownian_motion()`, `discrete_walk()`, `geometric_brownian_motion()`, `random_normal_drift_walk()`

Examples

```
set.seed(123)
random_normal_walk()

set.seed(123)
random_normal_walk(.dimensions = 3) |>
  head() |>
  t()
```

`rand_walk_helper` *Random Walk Helper*

Description

A function to help build random walks by mutating a data frame.

Usage

```
rand_walk_helper(.data, .value)
```

Arguments

`.data`	The data frame to mutate.
`.value`	The .initial_value to use. This is passed from the random walk function being called by the end user.

Details

A function to help build random walks by mutating a data frame. This mutation adds the following columns to the data frame: `cum_sum`, `cum_prod`, `cum_min`, `cum_max`, and `cum_mean`. The function is used internally by certain functions that generate random walks.

Value

A modified data frame/tibble with the following columns added:

- `cum_sum`: Cumulative sum of `y`.
- `cum_prod`: Cumulative product of `y`.
- `cum_min`: Cumulative minimum of `y`.
- `cum_max`: Cumulative maximum of `y`.
- `cum_mean`: Cumulative mean of `y`.

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Utility Functions: `confidence_interval()`, `convert_snake_to_title_case()`, `generate_caption()`, `get_attributes()`, `rand_walk_column_names()`, `running_quantile()`, `std_cum_max_augment()`, `std_cum_mean_augment()`, `std_cum_min_augment()`, `std_cum_prod_augment()`, `std_cum_sum_augment()`

Examples

```
df <- data.frame(
  walk_number = factor(rep(1L:25L, each = 30L)),
  x = rep(1L:30L, 25L),
  y = rnorm(750L, 0L, 1L)
  )

rand_walk_helper(df, 100)
```

`running_quantile` *Running Quantile Calculation*

Description

The `running_quantile` function calculates the quantile of a vector over a sliding window, allowing for various alignment and rule options.

Usage

```
running_quantile(
  .x,
  .window,
  .probs = 0.5,
  .type = 7,
  .rule = "quantile",
  .align = "center"
)
```

Arguments

`.x`	A numeric vector for which the running quantile is to be calculated.
`.window`	An integer specifying the size of the sliding window.
`.probs`	A numeric value between 0 and 1 indicating the desired quantile probability (default is 0.50).
`.type`	An integer from 1 to 9 specifying the quantile algorithm type (default is 7).
`.rule`	A character string indicating the rule to apply at the edges of the window. Possible choices are: • "quantile": Standard quantile calculation. • "trim": Trims the output to remove values outside the window. • "keep": Keeps the original values at the edges of the window. • "constant": Fills the edges with the constant value from the nearest valid quantile. • "NA": Fills the edges with NA values. • "func": Applies a custom function to the values in the window (default is "quantile").
`.align`	A character string specifying the alignment of the window ("center", "left", or "right"; default is "center").

Details

This function computes the running quantile of a numeric vector using a specified window size and probability.

Value

A numeric vector containing the running quantile values.

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Utility Functions: `confidence_interval()`, `convert_snake_to_title_case()`, `generate_caption()`, `get_attributes()`, `rand_walk_column_names()`, `rand_walk_helper()`, `std_cum_max_augment()`, `std_cum_mean_augment()`, `std_cum_min_augment()`, `std_cum_prod_augment()`, `std_cum_sum_augment()`

Examples

```
# Example usage of running_quantile
set.seed(123)
data <- cumsum(rnorm(50))
result <- running_quantile(data, .window = 3, .probs = 0.5)
print(result)

plot(data, type = "l")
lines(result, col = "red")
```

---

rw30 *Generate Random Walks*

Description

Generate Random Walks

Usage

```
rw30()
```

Details

The function generates random walks using the normal distribution with a specified mean (`mu`) and standard deviation (`sd`). Each walk is generated independently and stored in a tibble. The resulting tibble is then pivoted into a long format for easier analysis.

Value

A tibble in long format with columns `walk`, `x`, and `value`, representing the random walks. Additionally, attributes `num_walks`, `num_steps`, `mu`, and `sd` are attached to the tibble.

Author(s)

Steven P. Sanderson II, MPH

This function generates 30 random walks with 100 steps each and pivots the result into a long format tibble.

Examples

```
# Generate random walks and print the result
set.seed(123)
rw30()

set.seed(123)
rw30() |>
 visualize_walks()
```

rw_range *Range*

Description

A function to return the range of a vector.

Usage

```
rw_range(.x)
```

Arguments

`.x`	A numeric vector

Details

A function to return the range of a vector. It uses `max(.x) - min(.x)` as the basis of the function.

Value

A numeric vector

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Vector Function: `cgmean()`, `chmean()`, `ckurtosis()`, `cmean()`, `cmedian()`, `crange()`, `csd()`, `cskewness()`, `cvar()`, `euclidean_distance()`, `kurtosis_vec()`, `skewness_vec()`

Examples

```
x <- mtcars$mpg

rw_range(x)
```

skewness_vec *Compute Skewness of a Vector*

Description

This function takes in a vector as it's input and will return the skewness of that vector. The length of this vector must be at least four numbers. The skewness explains the 'tailedness' of the distribution of data.
`((1/n) * sum(x - mu})^3) / ((()1/n) * sum(x - mu)^2)^(3/2)`

Usage

```
skewness_vec(.x)
```

Arguments

`.x`	A numeric vector of length four or more.

Details

A function to return the skewness of a vector.

Value

The skewness of a vector

Author(s)

Steven P. Sanderson II, MPH

See Also

https://en.wikipedia.org/wiki/Skewness

Other Vector Function: `cgmean()`, `chmean()`, `ckurtosis()`, `cmean()`, `cmedian()`, `crange()`, `csd()`, `cskewness()`, `cvar()`, `euclidean_distance()`, `kurtosis_vec()`, `rw_range()`

Examples

```
set.seed(123)
skewness_vec(rnorm(100, 3, 2))
```

`std_cum_max_augment` *Augment Cumulative Maximum*

Description

This function augments a data frame by adding cumulative maximum columns for specified variables.

Usage

```
std_cum_max_augment(.data, .value, .names = "auto", .initial_value = 0)
```

Arguments

`.data`	A data frame to augment.
`.value`	A column name or names for which to compute the cumulative maximum.
`.names`	Optional. A character vector of names for the new cumulative maximum columns. Defaults to "auto", which generates names based on the original column names.
`.initial_value`	A numeric value to start the cumulative maximum from. Defaults to 0.

Details

The function takes a data frame and a column name (or names) and computes the cumulative maximum for each specified column, starting from an initial value. If the column names are not provided, it will throw an error.

Value

A tibble with the original data and additional columns containing the cumulative maximums.

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Utility Functions: `confidence_interval()`, `convert_snake_to_title_case()`, `generate_caption()`, `get_attributes()`, `rand_walk_column_names()`, `rand_walk_helper()`, `running_quantile()`, `std_cum_mean_augment()`, `std_cum_min_augment()`, `std_cum_prod_augment()`, `std_cum_sum_augment()`

Examples

```
df <- data.frame(x = c(1, 3, 2, 5, 4), y = c(10, 7, 6, 12, 5))
std_cum_max_augment(df, .value = x)
std_cum_max_augment(df, .value = y, .names = c("cummax_y"))
```

`std_cum_mean_augment` *Augment Cumulative Sum*

Description

This function augments a data frame by adding cumulative mean columns for specified variables.

Usage

```
std_cum_mean_augment(.data, .value, .names = "auto", .initial_value = 0)
```

Arguments

`.data`	A data frame to augment.
`.value`	A column name or names for which to compute the cumulative mean.
`.names`	Optional. A character vector of names for the new cumulative mean columns. Defaults to "auto", which generates names based on the original column names.
`.initial_value`	A numeric value to start the cumulative mean from. Defaults to 0.

Details

The function takes a data frame and a column name (or names) and computes the cumulative mean for each specified column, starting from an initial value. If the column names are not provided, it will throw an error.

Value

A tibble with the original data and additional columns containing the cumulative means.

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Utility Functions: `confidence_interval()`, `convert_snake_to_title_case()`, `generate_caption()`, `get_attributes()`, `rand_walk_column_names()`, `rand_walk_helper()`, `running_quantile()`, `std_cum_max_augment()`, `std_cum_min_augment()`, `std_cum_prod_augment()`, `std_cum_sum_augment()`

Examples

```
df <- data.frame(x = c(1, 2, 3, 4, 5), y = c(10, 20, 30, 40, 50))
std_cum_mean_augment(df, .value = x)
std_cum_mean_augment(df, .value = y, .names = c("cummean_y"))
```

std_cum_min_augment *Augment Cumulative Minimum*

Description

This function augments a data frame by adding cumulative minimum columns for specified variables.

Usage

```
std_cum_min_augment(.data, .value, .names = "auto", .initial_value = 0)
```

Arguments

`.data`	A data frame to augment.
`.value`	A column name or names for which to compute the cumulative minimum.
`.names`	Optional. A character vector of names for the new cumulative minimum columns. Defaults to "auto", which generates names based on the original column names.
`.initial_value`	A numeric value to start the cumulative minimum from. Defaults to 0.

Details

The function takes a data frame and a column name (or names) and computes the cumulative minimum for each specified column, starting from an initial value. If the column names are not provided, it will throw an error.

Value

A tibble with the original data and additional columns containing the cumulative minimums.

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Utility Functions: `confidence_interval()`, `convert_snake_to_title_case()`, `generate_caption()`, `get_attributes()`, `rand_walk_column_names()`, `rand_walk_helper()`, `running_quantile()`, `std_cum_max_augment()`, `std_cum_mean_augment()`, `std_cum_prod_augment()`, `std_cum_sum_augment()`

Examples

```
df <- data.frame(x = c(5, 3, 8, 1, 4), y = c(10, 7, 6, 12, 5))
std_cum_min_augment(df, .value = x)
std_cum_min_augment(df, .value = y, .names = c("cummin_y"))
```

`std_cum_prod_augment` *Augment Cumulative Product*

Description

This function augments a data frame by adding cumulative product columns for specified variables.

Usage

```
std_cum_prod_augment(.data, .value, .names = "auto", .initial_value = 1)
```

Arguments

`.data`	A data frame to augment.
`.value`	A column name or names for which to compute the cumulative product.
`.names`	Optional. A character vector of names for the new cumulative product columns. Defaults to "auto", which generates names based on the original column names.
`.initial_value`	A numeric value to start the cumulative product from. Defaults to 1.

Details

The function takes a data frame and a column name (or names) and computes the cumulative product for each specified column, starting from an initial value. If the column names are not provided, it will throw an error.

Value

A tibble with the original data and additional columns containing the cumulative products.

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Utility Functions: `confidence_interval()`, `convert_snake_to_title_case()`, `generate_caption()`, `get_attributes()`, `rand_walk_column_names()`, `rand_walk_helper()`, `running_quantile()`, `std_cum_max_augment()`, `std_cum_mean_augment()`, `std_cum_min_augment()`, `std_cum_sum_augment()`

Examples

```
df <- data.frame(x = 1:5, y = 6:10)
std_cum_prod_augment(df, .value = x)
std_cum_prod_augment(df, .value = y, .names = c("cumprod_y"))
```

`std_cum_sum_augment` *Augment Cumulative Sum*

Description

This function augments a data frame by adding cumulative sum columns for specified variables.

Usage

```
std_cum_sum_augment(.data, .value, .names = "auto", .initial_value = 0)
```

Arguments

`.data`	A data frame to augment.
`.value`	A column name or names for which to compute the cumulative sum.
`.names`	Optional. A character vector of names for the new cumulative sum columns. Defaults to "auto", which generates names based on the original column names.
`.initial_value`	A numeric value to start the cumulative sum from. Defaults to 0.

Details

The function takes a data frame and a column name (or names) and computes the cumulative sum for each specified column, starting from an initial value. If the column names are not provided, it will throw an error.

Value

A tibble with the original data and additional columns containing the cumulative sums.

Author(s)

Steven P. Sanderson II, MPH

See Also

Other Utility Functions: `confidence_interval()`, `convert_snake_to_title_case()`, `generate_caption()`, `get_attributes()`, `rand_walk_column_names()`, `rand_walk_helper()`, `running_quantile()`, `std_cum_max_augment()`, `std_cum_mean_augment()`, `std_cum_min_augment()`, `std_cum_prod_augment()`

Examples

```
df <- data.frame(x = 1:5, y = 6:10)
std_cum_sum_augment(df, .value = x)
std_cum_sum_augment(df, .value = y, .names = c("cumsum_y"))
```

summarize_walks *Summarize Walks Data*

Description

Summarizes random walk data by computing statistical measures.

Usage

```
summarize_walks(.data, .value, .group_var)

summarise_walks(.data, .value, .group_var)
```

Arguments

`.data`	A data frame or tibble containing random walk data.
`.value`	A column name (unquoted) representing the value to summarize.
`.group_var`	A column name (unquoted) representing the grouping variable.

Details

This function requires that the input data frame contains a column named 'walk_number' and that the value to summarize is provided. It computes statistics such as mean, median, variance, and quantiles for the specified value variable. #' This function summarizes a data frame containing random walk data by computing various statistical measures for a specified value variable, grouped by a specified grouping variable. It checks for necessary attributes and ensures that the data frame is structured correctly.

Value

A tibble containing the summarized statistics for each group, including mean, median, range, quantiles, variance, standard deviation, and more.

Author(s)

Steven P. Sanderson II, MPH

Examples

```
library(dplyr)

# Example data frame
walk_data <- random_normal_walk(.initial_value = 100)

# Summarize the walks
summarize_walks(walk_data, cum_sum_y, walk_number) |>
 glimpse()
summarize_walks(walk_data, y) |>
```

```
  glimpse()

# Example with missing value variable
# summarize_walks(walk_data, NULL, group) # This will trigger an error.
```

visualize_walks *Visualize Walks*

Description

visualize_walks() visualizes the output of the random walk functions in the RandomWalker package, resulting in one or more ggplot2 plots put together in a patchwork composed of 1 or more patches.

Usage

```
visualize_walks(.data, .alpha = 0.7, .interactive = FALSE, .pluck = FALSE)
```

Arguments

`.data`	The input data. Assumed to be created by one of the random walk functions in the RandomWalker package, but can be any data frame or tibble that contains columns `walk_number`, `x`, and one or more numeric columns like `y`, `cum_sum`, `cum_prod`, `cum_min`, `cum_max` and `cum_mean`, for instance.
`.alpha`	The alpha value for all the line charts in the visualization. Values range from 0 to 1. Default is 0.7.
`.interactive`	A boolean value. TRUE if you want the patches to be interactive. FALSE if you don't. Default is FALSE.
`.pluck`	If you want to visualize only one of the You can choose one of the values (`y`, `cum_sum`, `cum_prod`, `cum_min`, `cum_max`, `cum_mean`). Default is FALSE.

Details

visualize_walks() generates visualizations of the random walks generated by the random walk functions in the RandomWalker package. These are the functions at the moment of writing:

- brownian_motion()
- discrete_walk()
- geometric_brownian_motion()
- random_normal_drift_walk()
- random_normal_walk()
- rw30()

It is possible there are more when you read this, but you can check the rest of the documentation for the current situation.

The visualization function is meant to be easy to use. No parameters needed, but you can set `.alpha` if the default value of 0.7 isn't to your liking.

You can also choose whether you want the visualization to be interactive or not by setting `.interactive` to TRUE. The function uses the `ggiraph` package for making the patches interactive.

If you want to visualize only one of the attributes, you can choose use one of these values (`y`, `cum_sum`, `cum_prod`, `cum_min`, `cum_max`, `cum_mean`) for the `.pluck` parameter.

Value

A patchwork composed of 1 or more patches

Author(s)

Antti Lennart Rask

Examples

```
# Generate random walks and visualize the result
set.seed(123)
rw30() |>
 visualize_walks()

# Set the alpha value to be other than the default 0.7
set.seed(123)
rw30() |>
 visualize_walks(.alpha = 0.5)

# Use the function with an input that has alternatives for y
set.seed(123)
random_normal_walk(.num_walks = 5, .initial_value = 100) |>
 visualize_walks()

# Use the function to create interactive visualizations
set.seed(123)
random_normal_walk(.num_walks = 5, .initial_value = 100) |>
 visualize_walks(.interactive = TRUE)

# Use .pluck to pick just one visualization
set.seed(123)
random_normal_walk(.num_walks = 5, .initial_value = 100) |>
 visualize_walks(.pluck = "cum_sum")
```

Index