Package 'RandomWalker'

May 7, 2026

Title Generate Random Walks Compatible with the 'tidyverse'

Version 1.0.0

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

Language en

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 2025-08-18 23:50:57 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_beta_walk()`, `random_binomial_walk()`, `random_cauchy_walk()`, `random_chisquared_walk()`, `random_displacement_walk()`, `random_exponential_walk()`, `random_f_walk()`, `random_gamma_walk()`, `random_geometric_walk()`, `random_hypergeometric_walk()`, `random_logistic_walk()`, `random_lognormal_walk()`, `random_multinomial_walk()`, `random_negbinomial_walk()`, `random_normal_drift_walk()`, `random_normal_walk()`, `random_poisson_walk()`, `random_smirnov_walk()`, `random_t_walk()`, `random_uniform_walk()`, `random_weibull_walk()`, `random_wilcox_walk()`, `random_wilcoxon_sr_walk()`

Other Continuous Distribution: `geometric_brownian_motion()`, `random_beta_walk()`, `random_cauchy_walk()`, `random_chisquared_walk()`, `random_exponential_walk()`, `random_f_walk()`, `random_gamma_walk()`, `random_logistic_walk()`, `random_lognormal_walk()`, `random_normal_drift_walk()`, `random_normal_walk()`, `random_t_walk()`, `random_uniform_walk()`, `random_weibull_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()`, `subset_walks()`

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(), subset_walks()

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_beta_walk()`, `random_binomial_walk()`, `random_cauchy_walk()`, `random_chisquared_walk()`, `random_displacement_walk()`, `random_exponential_walk()`, `random_f_walk()`, `random_gamma_walk()`, `random_geometric_walk()`, `random_hypergeometric_walk()`, `random_logistic_walk()`, `random_lognormal_walk()`, `random_multinomial_walk()`, `random_negbinomial_walk()`, `random_normal_drift_walk()`, `random_normal_walk()`, `random_poisson_walk()`, `random_smirnov_walk()`, `random_t_walk()`, `random_uniform_walk()`, `random_weibull_walk()`, `random_wilcox_walk()`, `random_wilcoxon_sr_walk()`

Other Discrete Distribution: `random_binomial_walk()`, `random_displacement_walk()`, `random_geometric_walk()`, `random_hypergeometric_walk()`, `random_multinomial_walk()`, `random_negbinomial_walk()`, `random_poisson_walk()`, `random_smirnov_walk()`, `random_wilcox_walk()`, `random_wilcoxon_sr_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()`, `subset_walks()`

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_beta_walk()`, `random_binomial_walk()`, `random_cauchy_walk()`, `random_chisquared_walk()`, `random_displacement_walk()`, `random_exponential_walk()`, `random_f_walk()`, `random_gamma_walk()`, `random_geometric_walk()`, `random_hypergeometric_walk()`, `random_logistic_walk()`, `random_lognormal_walk()`, `random_multinomial_walk()`, `random_negbinomial_walk()`, `random_normal_drift_walk()`, `random_normal_walk()`, `random_poisson_walk()`, `random_smirnov_walk()`, `random_t_walk()`, `random_uniform_walk()`, `random_weibull_walk()`, `random_wilcox_walk()`, `random_wilcoxon_sr_walk()`

Other Continuous Distribution: `brownian_motion()`, `random_beta_walk()`, `random_cauchy_walk()`, `random_chisquared_walk()`, `random_exponential_walk()`, `random_f_walk()`, `random_gamma_walk()`, `random_logistic_walk()`, `random_lognormal_walk()`, `random_normal_drift_walk()`, `random_normal_walk()`, `random_t_walk()`, `random_uniform_walk()`, `random_weibull_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()`, `subset_walks()`

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_beta_walk` *Generate Multiple Random Beta Walks in Multiple Dimensions*

Description

The `random_beta_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 Beta distribution. The user can specify the number of walks, the number of steps in each walk, and the parameters of the Beta distribution (shape1, shape2, ncp). The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

Usage

```
random_beta_walk(
  .num_walks = 25,
  .n = 100,
  .shape1 = 2,
  .shape2 = 2,
  .ncp = 0,
  .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. Must be >= 1. Default is 100.
`.shape1`	Non-negative parameter of the Beta distribution. Default is 2.
`.shape2`	Non-negative parameter of the Beta distribution. Default is 2.
`.ncp`	Non-centrality parameter (ncp >= 0). Default is 0.
`.initial_value`	A numeric value indicating the initial value of the walks. Default is 0.
`.samp`	A logical value indicating whether to sample the Beta 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.

Details

The shape1, shape2, and ncp parameters can be single values or vectors of length equal to the number of dimensions. If vectors are provided, each dimension uses its corresponding value.

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).

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_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Continuous Distribution: brownian_motion(), geometric_brownian_motion(), random_cauchy_walk(), random_chisquared_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_logistic_walk(), random_lognormal_walk(), random_normal_drift_walk(), random_normal_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk()

Examples

```
set.seed(123)
random_beta_walk()

set.seed(123)
random_beta_walk(.dimensions = 3, .shape1 = c(2, 3, 4), .shape2 = c(2, 3, 4), .ncp = c(0, 1, 2)) |>
  head() |>
  t()
```

`random_binomial_walk` *Generate Multiple Random Binomial Walks in Multiple Dimensions*

Description

The `random_binomial_walk` function generates multiple random walks using the binomial distribution via `rbinom()`. The user can specify the number of walks, the number of steps in each walk, the number of trials, and the probability of success. The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

Usage

```
random_binomial_walk(
  .num_walks = 25,
  .n = 100,
  .size = 10,
  .prob = 0.5,
  .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 observations per walk. Must be greater than 0. Default is 100.
`.size`	An integer specifying the number of trials (zero or more). Default is 10.
`.prob`	A numeric value specifying the probability of success on each trial. Must be 0 <= .prob <= 1. Default is 0.5.
`.initial_value`	A numeric value indicating the initial value of the walks. Default is 0.
`.samp`	A logical value indicating whether to sample the binomial 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.

Details

This function generates random walks where each step is drawn from the binomial distribution using rbinom(). The user can control the number of walks, steps per walk, the number of trials (size), and the probability of success (prob). The function supports 1, 2, or 3 dimensions, and augments the output with cumulative statistics for each walk. Sampling can be performed with or without replacement, and a proportion of steps can be sampled if desired.

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).

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_beta_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(),

random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Discrete Distribution: discrete_walk(), random_displacement_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_poisson_walk(), random_smirnov_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Examples

```
set.seed(123)
random_binomial_walk()

set.seed(123)
random_binomial_walk(.dimensions = 2) |>
  head() |>
  t()
```

---

random_cauchy_walk *Generate Multiple Random Cauchy Walks in Multiple Dimensions*

Description

The random_cauchy_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 Cauchy distribution. The user can specify the number of walks, the number of steps in each walk, and the parameters of the Cauchy distribution (location and scale). The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

Usage

```
random_cauchy_walk(
  .num_walks = 25,
  .n = 100,
  .location = 0,
  .scale = 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.

.location	A numeric value indicating the location parameter of the Cauchy distribution. Default is 0.
.scale	A numeric value indicating the scale parameter of the Cauchy distribution. Default is 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 Cauchy 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.

Details

The location and scale parameters can be single values or vectors of length equal to the number of dimensions. If vectors are provided, each dimension uses its corresponding value.

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).

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_beta_walk(), random_binomial_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Continuous Distribution: brownian_motion(), geometric_brownian_motion(), random_beta_walk(), random_chisquared_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_logistic_walk(), random_lognormal_walk(), random_normal_drift_walk(), random_normal_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk()

Examples

```
set.seed(123)
random_cauchy_walk()

set.seed(123)
random_cauchy_walk(.dimensions = 3, .location = c(0, 1, 2), .scale = c(1, 2, 3)) |>
  head() |>
  t()
```

---

random_chisquared_walk

Generate Multiple Random Chi-Squared Walks in Multiple Dimensions

---

Description

The random_chisquared_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 chi-squared distribution. The user can specify the number of walks, the number of steps in each walk, and the parameters of the chi-squared distribution (df and ncp). The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

Usage

```
random_chisquared_walk(
  .num_walks = 25,
  .n = 100,
  .df = 5,
  .ncp = 0,
  .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.
.df	Degrees of freedom for the chi-squared distribution. Default is 5.
.ncp	Non-centrality parameter (non-negative). Default is 0.
.initial_value	A numeric value indicating the initial value of the walks. Default is 0.
.samp	A logical value indicating whether to sample the chi-squared 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.

Details

This function is a flexible generator for random walks where each step is drawn from a chi-squared distribution. The user can control the number of walks, steps per walk, degrees of freedom (df), and the non-centrality parameter (ncp). The function supports 1, 2, or 3 dimensions, and augments the output with cumulative statistics for each walk. Sampling can be performed with or without replacement, and a proportion of steps can be sampled if desired.

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).

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Continuous Distribution: brownian_motion(), geometric_brownian_motion(), random_beta_walk(), random_cauchy_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_logistic_walk(), random_lognormal_walk(), random_normal_drift_walk(), random_normal_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk()

Examples

```
set.seed(123)
random_chisquared_walk()

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

---

random_displacement_walk

Generate a Random Displacement Walk in 2D

Description

The random_displacement_walk function generates a single random walk in 2 dimensions (x, y), where each step is a random displacement in both x and y directions, sampled from the provided displacement and distance spaces. The walk disregards steps where both x and y displacements are zero.

Usage

```
random_displacement_walk(
  .num_walks = 25,
  .seed = NULL,
  .n = 100,
  .distance_space = c(0, 1, 2, 3, 4),
  .displacement = c(-1, 1),
  .dimensions = 1
)
```

Arguments

.num_walks	An integer specifying the number of random walks to generate. Default is 25.
.seed	An optional value to set the random seed. If NULL, no seed is set. Default is NULL.
.n	The number of steps in the walk. Must be >= 0. Default is 100.
.distance_space	A numeric vector of possible step distances. Default is c(0, 1, 2, 3, 4).
.displacement	A numeric vector of possible step directions. Default is c(-1, 1).
.dimensions	An integer specifying the number of dimensions (1, 2, or 3). Default is 1.

Value

A tibble with columns depending on the number of dimensions:

- 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 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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Discrete Distribution: discrete_walk(), random_binomial_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_poisson_walk(), random_smirnov_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Examples

```
random_displacement_walk(.n = 10, .seed = Sys.Date())
```

`random_exponential_walk` *Generate Multiple Random Exponential Walks in Multiple Dimensions*

Description

The `random_exponential_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 an exponential distribution.

Usage

```
random_exponential_walk(
  .num_walks = 25,
  .n = 100,
  .rate = 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. Must be >= 1. Default is 100.
`.rate`	A numeric value or vector indicating the rate parameter(s) of the exponential distribution. Default is 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 exponential 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.

Details

The `rate` parameter can be a single value or a vector of length equal to the number of dimensions. If a vector is provided, each dimension uses its corresponding rate.

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)`.

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_beta_walk()`, `random_binomial_walk()`, `random_cauchy_walk()`, `random_chisquared_walk()`, `random_displacement_walk()`, `random_f_walk()`, `random_gamma_walk()`, `random_geometric_walk()`, `random_hypergeometric_walk()`, `random_logistic_walk()`, `random_lognormal_walk()`, `random_multinomial_walk()`, `random_negbinomial_walk()`, `random_normal_drift_walk()`, `random_normal_walk()`, `random_poisson_walk()`, `random_smirnov_walk()`, `random_t_walk()`, `random_uniform_walk()`, `random_weibull_walk()`, `random_wilcox_walk()`, `random_wilcoxon_sr_walk()`

Other Continuous Distribution: `brownian_motion()`, `geometric_brownian_motion()`, `random_beta_walk()`, `random_cauchy_walk()`, `random_chisquared_walk()`, `random_f_walk()`, `random_gamma_walk()`, `random_logistic_walk()`, `random_lognormal_walk()`, `random_normal_drift_walk()`, `random_normal_walk()`, `random_t_walk()`, `random_uniform_walk()`, `random_weibull_walk()`

Examples

```
set.seed(123)
random_exponential_walk()

set.seed(123)
random_exponential_walk(.dimensions = 3, .rate = c(0.1, 0.1, 0.2)) |>
  head() |>
  t()
```

`random_f_walk` *Generate Multiple Random F Walks in Multiple Dimensions*

Description

The `random_f_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 an F distribution. The user can specify the number of walks, the number of steps in each walk, and the parameters of the F distribution (df1, df2, ncp). The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

Usage

```
random_f_walk(
  .num_walks = 25,
  .n = 100,
  .df1 = 5,
  .df2 = 5,
  .ncp = NULL,
  .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.
`.df1`	Degrees of freedom 1 for the F distribution. Default is 5.
`.df2`	Degrees of freedom 2 for the F distribution. Default is 5.
`.ncp`	Non-centrality parameter. Default is NULL (central F).
`.initial_value`	A numeric value indicating the initial value of the walks. Default is 0.
`.samp`	A logical value indicating whether to sample the F 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.

Details

This function is a flexible generator for random walks where each step is drawn from an F distribution. The user can control the number of walks, steps per walk, degrees of freedom (df1, df2), and optionally the non-centrality parameter (ncp). If .ncp is left as NULL, the function generates F values using the ratio of chi-squared distributions as described in base R documentation. The function supports 1, 2, or 3 dimensions, and augments the output with cumulative statistics for each walk. Sampling can be performed with or without replacement, and a proportion of steps can be sampled if desired.

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).

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Continuous Distribution: brownian_motion(), geometric_brownian_motion(), random_beta_walk(), random_cauchy_walk(), random_chisquared_walk(), random_exponential_walk(), random_gamma_walk(), random_logistic_walk(), random_lognormal_walk(), random_normal_drift_walk(), random_normal_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk()

Examples

```
set.seed(123)
random_f_walk()

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

`random_gamma_walk` *Generate Multiple Random Gamma Walks in Multiple Dimensions*

Description

The `random_gamma_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 gamma distribution. The user can specify the number of walks, the number of steps in each walk, and the parameters of the gamma distribution (shape, scale, rate). The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

Usage

```
random_gamma_walk(
  .num_walks = 25,
  .n = 100,
  .shape = 1,
  .scale = 1,
  .rate = NULL,
  .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. Must be greater than 0. Default is 100.
`.shape`	A positive numeric value for the shape parameter. Default is 1.
`.scale`	A positive numeric value for the scale parameter. Default is 1.
`.rate`	A positive numeric value for the rate parameter. Default is NULL (ignored if scale is provided).
`.initial_value`	A numeric value indicating the initial value of the walks. Default is 0.

`.samp`	A logical value indicating whether to sample the gamma 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.

Details

This function is a flexible generator for random walks where each step is drawn from a gamma distribution. The user can control the number of walks, steps per walk, and the shape, scale, and rate parameters for the distribution. The function supports 1, 2, or 3 dimensions, and augments the output with cumulative statistics for each walk. Sampling can be performed with or without replacement, and a proportion of steps can be sampled if desired.

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)`.

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Continuous Distribution: brownian_motion(), geometric_brownian_motion(), random_beta_walk(), random_cauchy_walk(), random_chisquared_walk(), random_exponential_walk(), random_f_walk(), random_logistic_walk(), random_lognormal_walk(), random_normal_drift_walk(), random_normal_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk()

Examples

```
set.seed(123)
random_gamma_walk()

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

---

random_geometric_walk *Generate Multiple Random Geometric Walks in Multiple Dimensions*

Description

The random_geometric_walk function generates multiple random walks using the geometric distribution via rgeom(). The user can specify the number of walks, the number of steps in each walk, and the probability of success in each trial. The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

Usage

```
random_geometric_walk(
  .num_walks = 25,
  .n = 100,
  .prob = 0.5,
  .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 observations per walk. Default is 100.
.prob	A numeric value specifying the probability of success in each trial. Must be 0 < .prob <= 1. Default is 0.5.
.initial_value	A numeric value indicating the initial value of the walks. Default is 0.
.samp	A logical value indicating whether to sample the geometric 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.

Details

This function generates random walks where each step is drawn from the geometric distribution using `rgeom()`. The user can control the number of walks, steps per walk, and the probability of success (`prob`). The function supports 1, 2, or 3 dimensions, and augments the output with cumulative statistics for each walk. Sampling can be performed with or without replacement, and a proportion of steps can be sampled if desired.

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)`.

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Discrete Distribution: discrete_walk(), random_binomial_walk(), random_displacement_walk(), random_hypergeometric_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_poisson_walk(), random_smirnov_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Examples

```
set.seed(123)
random_geometric_walk()

set.seed(123)
random_geometric_walk(.dimensions = 2) |>
  head() |>
  t()
```

---

random_hypergeometric_walk

Generate Multiple Random Hypergeometric Walks in Multiple Dimensions

---

Description

The random_hypergeometric_walk function generates multiple random walks using the hypergeometric distribution via rhyper(). The user can specify the number of walks, the number of steps in each walk, and the urn parameters (m, n, k). The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

Usage

```
random_hypergeometric_walk(
  .num_walks = 25,
  .nn = 100,
  .m = 50,
  .n = 50,
  .k = 10,
```

```
  .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.
`.nn`	An integer specifying the number of observations per walk. Default is 100.
`.m`	An integer specifying the number of white balls in the urn. Default is 50.
`.n`	An integer specifying the number of black balls in the urn. Default is 50.
`.k`	An integer specifying the number of balls drawn from the urn. Default is 10.
`.initial_value`	A numeric value indicating the initial value of the walks. Default is 0.
`.samp`	A logical value indicating whether to sample the hypergeometric 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 `.nn` to sample. Default is 0.8.
`.dimensions`	An integer specifying the number of dimensions (1, 2, or 3). Default is 1.

Details

This function generates random walks where each step is drawn from the hypergeometric distribution using `rhyper()`. The user can control the number of walks, steps per walk, and the urn parameters: m (white balls), n (black balls), and k (balls drawn). The function supports 1, 2, or 3 dimensions, and augments the output with cumulative statistics for each walk. Sampling can be performed with or without replacement, and a proportion of steps can be sampled if desired.

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).

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Discrete Distribution: discrete_walk(), random_binomial_walk(), random_displacement_walk(), random_geometric_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_poisson_walk(), random_smirnov_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Examples

```
set.seed(123)
random_hypergeometric_walk()

set.seed(123)
random_hypergeometric_walk(.dimensions = 2) |>
  head() |>
  t()
```

---

random_logistic_walk *Generate Multiple Random Logistic Walks in Multiple Dimensions*

Description

The random_logistic_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 logistic distribution. The user can specify the number of walks, the number of steps in each walk, and the parameters of the logistic distribution (location and scale). The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

Usage

```
random_logistic_walk(
  .num_walks = 25,
  .n = 100,
  .location = 0,
  .scale = 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.
.location	A numeric value indicating the location parameter of the logistic distribution. Default is 0.
.scale	A numeric value indicating the scale parameter of the logistic distribution. Default is 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 logistic 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.

Details

This function is a flexible generator for random walks where each step is drawn from a logistic distribution. The user can control the number of walks, steps per walk, and the location and scale parameters for the distribution. The function supports 1, 2, or 3 dimensions, and augments the output with cumulative statistics for each walk. Sampling can be performed with or without replacement, and a proportion of steps can be sampled if desired.

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).

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_lognormal_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Continuous Distribution: brownian_motion(), geometric_brownian_motion(), random_beta_walk(), random_cauchy_walk(), random_chisquared_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_lognormal_walk(), random_normal_drift_walk(), random_normal_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk()

Examples

```
set.seed(123)
random_logistic_walk()

set.seed(123)
random_logistic_walk(.dimensions = 2) |>
  head() |>
  t()
```

`random_lognormal_walk` *Generate Multiple Random Lognormal Walks in Multiple Dimensions*

Description

The `random_lognormal_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 lognormal distribution. The user can specify the number of walks, the number of steps in each walk, and the parameters of the lognormal distribution (meanlog and sdlog). The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

Usage

```
random_lognormal_walk(
  .num_walks = 25,
  .n = 100,
  .meanlog = 0,
  .sdlog = 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.
`.meanlog`	A numeric value indicating the meanlog parameter of the lognormal distribution. Default is 0.
`.sdlog`	A numeric value indicating the sdlog parameter of the lognormal distribution. Default is 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 lognormal 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.

Details

This function is a flexible generator for random walks where each step is drawn from a lognormal distribution. The user can control the number of walks, steps per walk, and the meanlog and sdlog parameters for the distribution. The function supports 1, 2, or 3 dimensions, and augments the output with cumulative statistics for each walk. Sampling can be performed with or without replacement, and a proportion of steps can be sampled if desired.

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)`.

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_beta_walk()`, `random_binomial_walk()`, `random_cauchy_walk()`, `random_chisquared_walk()`, `random_displacement_walk()`, `random_exponential_walk()`, `random_f_walk()`, `random_gamma_walk()`, `random_geometric_walk()`, `random_hypergeometric_walk()`, `random_logistic_walk()`, `random_multinomial_walk()`, `random_negbinomial_walk()`, `random_normal_drift_walk()`, `random_normal_walk()`, `random_poisson_walk()`, `random_smirnov_walk()`, `random_t_walk()`, `random_uniform_walk()`, `random_weibull_walk()`, `random_wilcox_walk()`, `random_wilcoxon_sr_walk()`

Other Continuous Distribution: `brownian_motion()`, `geometric_brownian_motion()`, `random_beta_walk()`, `random_cauchy_walk()`, `random_chisquared_walk()`, `random_exponential_walk()`, `random_f_walk()`, `random_gamma_walk()`, `random_logistic_walk()`, `random_normal_drift_walk()`, `random_normal_walk()`, `random_t_walk()`, `random_uniform_walk()`, `random_weibull_walk()`

Examples

```
set.seed(123)
random_lognormal_walk()

set.seed(123)
random_lognormal_walk(.dimensions = 2) |>
  head() |>
  t()
```

`random_multinomial_walk` *Generate Multiple Random Multinomial Walks*

Description

A multinomial random walk is a stochastic process in which each step is drawn from the multinomial distribution. This function allows for the simulation of multiple independent random walks in one, two, or three dimensions, with user control over the number of walks, steps, trials, probabilities, and dimensions. Sampling options allow for further customization, including the ability to sample a proportion of steps and to sample with or without replacement. The resulting data frame includes cumulative statistics for each walk.

Usage

```
random_multinomial_walk(
  .num_walks = 25,
  .n = 100,
  .size = 3,
  .prob = rep(1/3, .n),
  .initial_value = 0,
  .samp = TRUE,
  .replace = TRUE,
  .sample_size = 0.8,
  .dimensions = 1
)
```

Arguments

`.num_walks`	Integer. Number of random walks to generate. Default is 25.
`.n`	Integer. Length of each walk (number of steps). Default is 100.
`.size`	Integer. Number of trials for each multinomial draw. Default is 3.
`.prob`	Numeric vector. Probabilities for each outcome. Default is rep(1/3, .n).
`.initial_value`	Numeric. Starting value of the walk. Default is 0.
`.samp`	Logical. Whether to sample the steps. Default is TRUE.

.replace	Logical. Whether sampling is with replacement. Default is TRUE.
.sample_size	Numeric. Proportion of steps to sample (0-1). Default is 0.8.
.dimensions	Integer. Number of dimensions (1, 2, or 3). Default is 1.

Details

The random_multinomial_walk function generates multiple random walks using the multinomial distribution via stats::rmultinom(). Each walk is a sequence of steps where each step is a random draw from the multinomial distribution. The user can specify the number of walks, steps, trials per step, and the probability vector. Sampling options allow for further customization, including the ability to sample a proportion of steps and to sample with or without replacement. The resulting data frame includes cumulative statistics for each walk, making it suitable for simulation studies and visualization.

Value

A tibble containing the generated random walks with columns:

- walk_number: Factor representing the walk number.
- step_number: Step index.
- value: Value of the walk at each step.
- Cumulative statistics: cum_sum, cum_prod, cum_min, cum_max, cum_mean.

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Discrete Distribution: discrete_walk(), random_binomial_walk(), random_displacement_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_negbinomial_walk(), random_poisson_walk(), random_smirnov_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Examples

```
set.seed(123)
random_multinomial_walk()

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

`random_negbinomial_walk` *Generate Multiple Random Negative Binomial Walks*

Description

A Negative Binomial random walk is a stochastic process in which each step is drawn from the Negative Binomial distribution, commonly used for modeling count data with overdispersion. This function allows for the simulation of multiple independent random walks in one, two, or three dimensions, with user control over the number of walks, steps, and the distribution parameters. Sampling options allow for further customization, including the ability to sample a proportion of steps and to sample with or without replacement. The resulting data frame includes cumulative statistics for each walk, making it suitable for simulation studies and visualization.

Usage

```
random_negbinomial_walk(
  .num_walks = 25,
  .n = 100,
  .size = 1,
  .prob = 0.5,
  .mu = NULL,
  .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`	Integer. Number of random variables to return for each walk. Default is 100.
`.size`	Integer. Number of successful trials or dispersion parameter. Default is 1. This must also match the number of dimensions, for example if `.dimensions = 3`, then .size must be a vector of length 3 like `c(1, 2, 3)`.

.prob	Numeric. Probability of success in each trial (0 < prob <= 1). Default is 0.5. This must also match the number of dimensions, for example if .dimensions = 3, then .prob must be a vector of length 3 like c(0.5, 0.7, 0.9).
.mu	Numeric. Alternative parametrization via mean. Default is NULL. This must also match the number of dimensions, for example if .dimensions = 3, then .mu must be a vector of length 3 like c(1, 2, 3).
.initial_value	Numeric. Starting value of the walk. Default is 0.
.samp	Logical. Whether to sample the steps. Default is TRUE.
.replace	Logical. Whether sampling is with replacement. Default is TRUE.
.sample_size	Numeric. Proportion of steps to sample (0-1). Default is 0.8.
.dimensions	Integer. Number of dimensions (1, 2, or 3). Default is 1.

Details

The random_negbinomial_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 the Negative Binomial distribution using stats::rnbinom(). The user can specify the number of samples in each walk (n), the size parameter, the probability of success (prob), and/or the mean (mu), and the number of dimensions. The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Discrete Distribution: discrete_walk(), random_binomial_walk(), random_displacement_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_multinomial_walk(), random_poisson_walk(), random_smirnov_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Examples

```
set.seed(123)
random_negbinomial_walk()

set.seed(123)
random_negbinomial_walk(.dimensions = 3,
  .size = c(1,2,3),
  .prob = c(0.5,0.7,0.9)
  ) |>
  head() |>
  t()
```

---

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Continuous Distribution: brownian_motion(), geometric_brownian_motion(), random_beta_walk(), random_cauchy_walk(), random_chisquared_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_logistic_walk(), random_lognormal_walk(), random_normal_walk(), random_t_walk(), random_uniform_walk(), random_weibull_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:

- 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).

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_normal_drift_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Continuous Distribution: brownian_motion(), geometric_brownian_motion(), random_beta_walk(), random_cauchy_walk(), random_chisquared_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_logistic_walk(), random_lognormal_walk(), random_normal_drift_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk()

Examples

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

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

`random_poisson_walk` *Generate Multiple Random Poisson Walks*

Description

A Poisson random walk is a stochastic process in which each step is drawn from the Poisson distribution, commonly used for modeling count data. This function allows for the simulation of multiple independent random walks in one, two, or three dimensions, with user control over the number of walks, steps, and the lambda parameter for the distribution. Sampling options allow for further customization, including the ability to sample a proportion of steps and to sample with or without replacement. The resulting data frame includes cumulative statistics for each walk, making it suitable for simulation studies and visualization.

Usage

```
random_poisson_walk(
  .num_walks = 25,
  .n = 100,
  .lambda = 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	Integer. Number of random variables to return for each walk. Default is 100.
.lambda	Numeric or vector. Mean(s) for the Poisson distribution. Default is 1.
.initial_value	Numeric. Starting value of the walk. Default is 0.
.samp	Logical. Whether to sample the steps. Default is TRUE.
.replace	Logical. Whether sampling is with replacement. Default is TRUE.
.sample_size	Numeric. Proportion of steps to sample (0-1). Default is 0.8.
.dimensions	Integer. Number of dimensions (1, 2, or 3). Default is 1.

Details

The random_poisson_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 the Poisson distribution using base::rpois(). The user can specify the number of samples in each walk (n), the lambda parameter for the Poisson distribution, and the number of dimensions. The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Discrete Distribution: discrete_walk(), random_binomial_walk(), random_displacement_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_smirnov_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Examples

```
set.seed(123)
random_poisson_walk()

set.seed(123)
random_poisson_walk(.dimensions = 3, .lambda = c(1, 2, 3)) |>
   head() |>
   t()
```

---

random_smirnov_walk *Generate Multiple Random Smirnov Walks in Multiple Dimensions*

Description

The random_smirnov_walk function generates multiple random walks using the Smirnov distribution via rsmirnov(). The user can specify the number of walks, the number of steps in each walk, the sizes parameter, and the alternative hypothesis. The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

Usage

```
random_smirnov_walk(
  .num_walks = 25,
  .n = 100,
  .sizes = c(1, 1),
  .z = NULL,
  .alternative = "two.sided",
  .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.
.sizes	A numeric vector of length 2 specifying the sizes parameter for rsmirnov. Default is c(1, 1).
.z	Optional numeric vector for the z parameter in rsmirnov. Default is NULL.
.alternative	One of "two.sided" (default), "less", or "greater". Indicates the type of test statistic.
.initial_value	A numeric value indicating the initial value of the walks. Default is 0.
.samp	A logical value indicating whether to sample the Smirnov 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.

Details

This function generates random walks where each step is drawn from the Smirnov distribution using `rsmirnov()`. The user can control the number of walks, steps per walk, the `sizes` parameter (default `c(1, 1)`), and the `alternative` hypothesis. The parameter `z` can be provided or left as `NULL` (default). The function supports 1, 2, or 3 dimensions, and augments the output with cumulative statistics for each walk. Sampling can be performed with or without replacement, and a proportion of steps can be sampled if desired.

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)`.

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Discrete Distribution: discrete_walk(), random_binomial_walk(), random_displacement_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_poisson_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Examples

```
set.seed(123)
random_smirnov_walk()

set.seed(123)
random_smirnov_walk(.dimensions = 2) |>
  head() |>
  t()
```

---

random_t_walk *Generate Multiple Random t-Distributed Walks in Multiple Dimensions*

Description

The random_t_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 t-distribution. The user can specify the number of walks, the number of steps in each walk, and the degrees of freedom for the t-distribution. The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

Usage

```
random_t_walk(
  .num_walks = 25,
  .n = 100,
  .df = 5,
  .initial_value = 0,
  .ncp = 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.
.df	Degrees of freedom for the t-distribution. Default is 5.
.initial_value	A numeric value indicating the initial value of the walks. Default is 0.
.ncp	A numeric value for the non-centrality parameter for the t-distribution. Default is 0.
.samp	A logical value indicating whether to sample the t-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.

Details

This function is a flexible generator for random walks where each step is drawn from a t-distribution. The user can control the number of walks, steps per walk, degrees of freedom, and optionally the non-centrality parameter (ncp). If .ncp is left blank, the function uses the default behavior of rt() from base R, which sets ncp = 0. The function supports 1, 2, or 3 dimensions, and augments the output with cumulative statistics for each walk. Sampling can be performed with or without replacement, and a proportion of steps can be sampled if desired.

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).

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Continuous Distribution: brownian_motion(), geometric_brownian_motion(), random_beta_walk(), random_cauchy_walk(), random_chisquared_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_logistic_walk(), random_lognormal_walk(), random_normal_drift_walk(), random_normal_walk(), random_uniform_walk(), random_weibull_walk()

Examples

```
set.seed(123)
random_t_walk()

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

---

random_uniform_walk *Generate Multiple Random Uniform Walks in Multiple Dimensions*

Description

The random_uniform_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 uniform distribution. The user can specify the number of walks, the number of steps in each walk, and the parameters of the uniform distribution (min and max). The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

Usage

```
random_uniform_walk(
  .num_walks = 25,
  .n = 100,
  .min = 0,
  .max = 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.
.min	A numeric value indicating the minimum of the uniform distribution. Default is 0.
.max	A numeric value indicating the maximum of the uniform distribution. Default is 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 uniform 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.

Details

This function is a flexible generator for random walks where each step is drawn from a uniform distribution. The user can control the number of walks, steps per walk, and the minimum and maximum values for the uniform distribution. The function supports 1, 2, or 3 dimensions, and augments the output with cumulative statistics for each walk. Sampling can be performed with or without replacement, and a proportion of steps can be sampled if desired.

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).

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_weibull_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Continuous Distribution: brownian_motion(), geometric_brownian_motion(), random_beta_walk(), random_cauchy_walk(), random_chisquared_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_logistic_walk(), random_lognormal_walk(), random_normal_drift_walk(), random_normal_walk(), random_t_walk(), random_weibull_walk()

Examples

```
set.seed(123)
random_uniform_walk()

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

`random_weibull_walk` *Generate Multiple Random Weibull Walks in Multiple Dimensions*

Description

A Weibull random walk is a stochastic process in which each step is drawn from the Weibull distribution, a flexible distribution commonly used to model lifetimes, reliability, and extreme values. This function allows for the simulation of multiple independent random walks in one, two, or three dimensions, with user control over the number of walks, steps, and the shape and scale parameters of the Weibull distribution. Sampling options allow for further customization, including the ability to sample a proportion of steps and to sample with or without replacement. The resulting data frame includes cumulative statistics for each walk, making it suitable for simulation studies and visualization.

Usage

```
random_weibull_walk(
  .num_walks = 25,
  .n = 100,
  .shape = 1,
  .scale = 1,
  .initial_value = 0,
  .samp = TRUE,
  .replace = TRUE,
  .sample_size = 0.8,
  .dimensions = 1
)
```

Arguments

`.num_walks`	Integer. Number of walks to generate. Default is 25.
`.n`	Integer. Number of steps in each walk. Default is 100.
`.shape`	Numeric. Shape parameter of the Weibull distribution. Default is 1.
`.scale`	Numeric. Scale parameter of the Weibull distribution. Default is 1.
`.initial_value`	Numeric. Starting value of the walk. Default is 0.
`.samp`	Logical. Whether to sample the steps. Default is TRUE.
`.replace`	Logical. Whether sampling is with replacement. Default is TRUE.
`.sample_size`	Numeric. Proportion of steps to sample (0-1). Default is 0.8.
`.dimensions`	Integer. Number of dimensions (1, 2, or 3). Default is 1.

Details

The `random_weibull_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 the Weibull distribution using `stats::rweibull()`. The user can specify the number of walks, the number of steps in each

walk, and the parameters .shape and .scale for the Weibull distribution. The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

Value

A data frame with the random walks and cumulative statistics as columns.

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_wilcox_walk(), random_wilcoxon_sr_walk()

Other Continuous Distribution: brownian_motion(), geometric_brownian_motion(), random_beta_walk(), random_cauchy_walk(), random_chisquared_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_logistic_walk(), random_lognormal_walk(), random_normal_drift_walk(), random_normal_walk(), random_t_walk(), random_uniform_walk()

Examples

```
set.seed(123)
random_weibull_walk()

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

`random_wilcoxon_sr_walk` *Generate Multiple Random Wilcoxon Signed-Rank Walks*

Description

A Wilcoxon signed-rank random walk is a stochastic process in which each step is drawn from the Wilcoxon signed-rank distribution, commonly used in nonparametric statistics. This function allows for the simulation of multiple independent random walks in one, two, or three dimensions, with user control over the number of walks, steps, and the sample size parameter for the distribution. Sampling options allow for further customization, including the ability to sample a proportion of steps and to sample with or without replacement. The resulting data frame includes cumulative statistics for each walk, making it suitable for simulation studies and visualization.

Usage

```
random_wilcoxon_sr_walk(
  .num_walks = 25,
  .nn = 100,
  .n = 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.
`.nn`	An integer specifying the number of steps in each walk. Default is 100.
`.n`	Integer or vector. Number(s) of observations in the sample(s) for rsignrank. Default is 1.
`.initial_value`	Numeric. Starting value of the walk. Default is 0.
`.samp`	Logical. Whether to sample the steps. Default is TRUE.
`.replace`	Logical. Whether sampling is with replacement. Default is TRUE.

.sample_size	Numeric. Proportion of steps to sample (0-1). Default is 0.8.
.dimensions	Integer. Number of dimensions (1, 2, or 3). Default is 1.

Details

The random_wilcoxon_sr_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 the Wilcoxon signed-rank distribution using stats::rsignrank(). The user can specify the number of steps/periods (nn), the number of samples in each walk (n), and the number of dimensions. The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcox_walk()

Other Discrete Distribution: discrete_walk(), random_binomial_walk(), random_displacement_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_poisson_walk(), random_smirnov_walk(), random_wilcox_walk()

Examples

```
set.seed(123)
random_wilcoxon_sr_walk()

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

`random_wilcox_walk` *Generate Multiple Random Wilcoxon Walks in Multiple Dimensions*

Description

The `random_wilcox_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 the Wilcoxon distribution using `stats::rwilcox()`. The user can specify the number of walks, the number of steps in each walk, and the parameters `.m` and `.n` for the Wilcoxon distribution. The function also allows for sampling a proportion of the steps and optionally sampling with replacement.

Usage

```
random_wilcox_walk(
  .num_walks = 25,
  .n = 100,
  .m = 10,
  .k = 10,
  .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. (Maps to `nn` in `rwilcox()`)
`.m`	Number of observations in the first sample for Wilcoxon. Default is 10.
`.k`	Number of observations in the second sample for Wilcoxon. Default is 10.
`.initial_value`	A numeric value indicating the initial value of the walks. Default is 0.
`.samp`	A logical value indicating whether to sample the Wilcoxon 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:

- 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).

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_beta_walk(), random_binomial_walk(), random_cauchy_walk(), random_chisquared_walk(), random_displacement_walk(), random_exponential_walk(), random_f_walk(), random_gamma_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_logistic_walk(), random_lognormal_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_normal_drift_walk(), random_normal_walk(), random_poisson_walk(), random_smirnov_walk(), random_t_walk(), random_uniform_walk(), random_weibull_walk(), random_wilcoxon_sr_walk()

Other Discrete Distribution: discrete_walk(), random_binomial_walk(), random_displacement_walk(), random_geometric_walk(), random_hypergeometric_walk(), random_multinomial_walk(), random_negbinomial_walk(), random_poisson_walk(), random_smirnov_walk(), random_wilcoxon_sr_walk()

Examples

```
set.seed(123)
random_wilcox_walk()

set.seed(123)
random_wilcox_walk(.dimensions = 2) |>
  head() |>
  t()
```

`rand_walk_column_names` *Get Column Names*

Description

This function generates the column names of a rand walk data frame.

Usage

```
rand_walk_column_names(.rand_data, .dim_names, .num_sims, .t)
```

Arguments

`.rand_data`	A data frame from which column names are to be extracted.
`.dim_names`	The dimnames passed from the rand walk function.
`.num_sims`	The number of simulations.
`.t`	Ther periods in the walk

Details

The `rand_walk_column_names` function takes a data frame as input and returns the rand walk data with column names.

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_helper()`, `running_quantile()`, `std_cum_max_augment()`, `std_cum_mean_augment()`, `std_cum_min_augment()`, `std_cum_prod_augment()`, `std_cum_sum_augment()`, `subset_walks()`

`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()`, `subset_walks()`

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()`, `subset_walks()`

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(), subset_walks()

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(), subset_walks()

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(), subset_walks()

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(), subset_walks()

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(), subset_walks()

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"))
```

---

subset_walks *Subset Walks by Extreme Values*

Description

This function subsets random walks to identify the walk with the maximum or minimum value.

Usage

```
subset_walks(.data, .type = "max", .value = "y")
```

Arguments

.data	A data frame containing random walks. It must have columns walk_number and the specified value column.
.type	A character string specifying the type of subset: "max" for maximum value, "min" for minimum value, or "both" for both maximum and minimum values.
.value	A character string specifying the column name to use for finding extreme values. Defaults to "y".

Details

The subset_walks function takes a data frame containing random walks and subsets it to return the walk with the maximum or minimum value based on the specified type. It requires that the input data frame contains columns walk_number and the specified value column.

Value

A data frame containing the subset walk.

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(), std_cum_sum_augment()

Examples

```
set.seed(123)
df <- rw30()
subset_walks(df, .type = "max")
subset_walks(df, .type = "min")
subset_walks(df, .type = "both")

# Example with a specific value column
set.seed(123)
discrete_walk() |>
  subset_walks(.type = "both", .value = "cum_sum_y") |>
  visualize_walks(.pluck = 2)
```

---

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 = c(1, 3))
```

Index