Package: RandomWalker (via r-universe)

September 16, 2024

Title Generate Random Walks Compatible with the 'tidyverse'

Version 0.1.0

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

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2.9000

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

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

Depends R (>= 4.1.0)

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

Suggests knitr, rmarkdown, stats, ggplot2

VignetteBuilder knitr

NeedsCompilation no

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

Maintainer Steven Sanderson <spsanderson@gmail.com>

Repository CRAN

Date/Publication 2024-09-15 20:30:24 UTC

Contents

brownian_motion . . . 2
cgmean . . . 4
chmean . . . 5
ckurtosis . . . 6
cmean . . . 7
cmedian . . . 8
convert_snake_to_title_case . . . 9
crange . . . 10
csd . . . 11
cskewness . . . 12
cvar . . . 13
discrete_walk . . . 14
euclidean_distance . . . 15
generate_caption . . . 16
geometric_brownian_motion . . . 17
kurtosis_vec . . . 19
random_normal_drift_walk . . . 20
random_normal_walk . . . 21
rand_walk_helper . . . 23
rw30 . . . 24
rw_range . . . 25
skewness_vec . . . 26
summarize_walks . . . 27
visualize_walks . . . 28

Index **30**

`brownian_motion` *Brownian Motion*

Description

Create a Brownian Motion Tibble

Usage

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

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.
.return_tibble	The default is TRUE. If set to FALSE then an object of class matrix will be returned.

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/matrix

Author(s)

Steven P. Sanderson II, MPH

See Also

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

Examples

```
library(ggplot2)

set.seed(123)
brownian_motion()

set.seed(123)
brownian_motion() |>
  ggplot(aes(x = x, y = y, group = walk_number, color = walk_number)) +
  geom_line() +
  labs(title = "Brownian Motion", x = "Time", y = "Value") +
```

```
  theme_minimal() +
  theme(legend.position = "none")
```

---

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)
```

`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: `generate_caption()`, `rand_walk_helper()`

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
)
```

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.

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 simulated walks, with columns for the walk number, time period, and various cumulative metrics (sum, product, min, max).

Author(s)

Steven P. Sanderson II, MPH

See Also

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

Examples

```
library(ggplot2)

set.seed(123)
discrete_walk()

set.seed(123)
discrete_walk(.num_walks = 30, .n = 250, .upper_probability = 0.55) |>
ggplot(aes(x = x, y = cum_sum)) +
 geom_line(aes(group = walk_number), alpha = .618, color = "steelblue") +
 theme_minimal() +
 theme(legend.position = "none") +
 geom_smooth(method = "lm", se = FALSE)
```

`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, x, y)
euclidean_distance(df, x, 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: `convert_snake_to_title_case()`, `rand_walk_helper()`

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,
  .return_tibble = TRUE
)
```

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.
`.return_tibble`	The default is TRUE. If set to FALSE then an object of class matrix will be returned.

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/matrix

Author(s)

Steven P. Sanderson II, MPH

See Also

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

Examples

```
library(ggplot2)

set.seed(123)
geometric_brownian_motion()

set.seed(123)
geometric_brownian_motion() |>
  ggplot(aes(x = x, y = y, group = walk_number, color = walk_number)) +
  geom_line() +
  labs(title = "Geometric Brownian Motion", x = "Time", y = "Value") +
  theme_minimal() +
  theme(legend.position = "none")
```

kurtosis_vec *Compute Kurtosis of a Vector*

Description

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

Usage

```
kurtosis_vec(.x)
```

Arguments

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

Details

A function to return the kurtosis of a vector.

Value

The kurtosis of a vector

Author(s)

Steven P. Sanderson II, MPH

See Also

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

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

Examples

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

`random_normal_drift_walk` *Generate Multiple Random Walks with Drift*

Description

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

Usage

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

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.

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 in long format with columns `walk_number`, `x` (step index), and `y` (walk value). The tibble has attributes for the number of walks, number of steps, mean, standard deviation, and drift.

Author(s)

Steven P. Sanderson II, MPH

See Also

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

Examples

```
library(ggplot2)

set.seed(123)
walks <- random_normal_drift_walk(.num_walks = 10, .n = 50, .mu = 0, .sd = 1.2,
                                  .drift = 0.05)
ggplot(walks, aes(x = x, y = y, group = walk_number, color = walk_number)) +
  geom_line() +
  labs(title = "Random Walks with Drift", x = "Time", y = "Value") +
  theme_minimal() +
  theme(legend.position = "none")
```

---

random_normal_walk *Generate Multiple Random Normal Walks*

Description

The random_normal_walk function is useful for simulating random processes and can be applied in various fields such as finance, physics, and biology to model different stochastic behaviors.

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
)
```

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.

Details

This function generates multiple random walks, which are sequences 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.

The output tibble includes several computed columns for each walk, such as the cumulative sum, product, minimum, and maximum of the steps.

Value

A tibble containing the generated random walks with the following columns:

- walk_number: Factor representing the walk number.
- x: Step index.
- y: Normal distribution values.
- cum_sum: Cumulative sum of y.
- cum_prod: Cumulative product of y.
- cum_min: Cumulative minimum of y.
- cum_max: Cumulative maximum of y.

The tibble includes attributes for the function parameters.

Author(s)

Steven P. Sanderson II, MPH

See Also

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

Examples

```
library(ggplot2)

# Generate 10 random walks with 50 steps each
set.seed(123)
random_normal_walk(.num_walks = 10, .n = 50)
```

```
# Generate random walks with different mean and standard deviation
set.seed(123)
random_normal_walk(.num_walks = 10, .n = 50, .samp = FALSE)

set.seed(123)
random_normal_walk(.num_walks = 2, .n = 100) |>
  ggplot(aes(x = x, y = y, group = walk_number, color = walk_number)) +
  geom_line() +
  labs(title = "Random Normal Walk", x = "Time", y = "Value") +
  theme_minimal() +
  theme(legend.position = "none")
```

`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: `convert_snake_to_title_case()`, `generate_caption()`

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)
```

`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

```
library(ggplot2)

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

set.seed(123)
rw30() |>
 ggplot(aes(x = x, y = y, color = walk_number, group = walk_number)) +
 geom_line() +
 theme_minimal() +
 theme(legend.position = "none") +
 labs(
   title = "30 Random Walks",
   x = "Step",
   y = "Value",
   color = "Walk Number"
 )
```

`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))
```

`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, 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)
```

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 x, 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.

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 the `.alpha` if the default value of 0.7 isn't to your liking.

You can combine this function with many tidyverse functions (either before or after). There's one example below.

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() |>
 visualize_walks()

# Use the pluck function from purrr to pick just one visualization
set.seed(123)
random_normal_walk() |>
 visualize_walks() |>
 purrr::pluck(2)
```

Index

∗ **Auto Random Walk**
  rw30, 24
∗ **Generator Functions**
  brownian_motion, 2
  discrete_walk, 14
  geometric_brownian_motion, 17
  random_normal_drift_walk, 20
  random_normal_walk, 21
∗ **Statistic Functions**
  summarize_walks, 27
∗ **Utility Functions**
  convert_snake_to_title_case, 9
  generate_caption, 16
  rand_walk_helper, 23
∗ **Vector Function**
  cgmean, 4
  chmean, 5
  ckurtosis, 6
  cmean, 7
  cmedian, 8
  crange, 10
  csd, 11
  cskewness, 12
  cvar, 13
  euclidean_distance, 15
  kurtosis_vec, 19
  rw_range, 25
  skewness_vec, 26
∗ **Visualization Functions**
  visualize_walks, 28

brownian_motion, 2, *15*, *18*, *21, 22*

cgmean, 4, *5–8*, *10–13*, *16*, *19*, *26*
chmean, *4*, 5, *6–8*, *10–13*, *16*, *19*, *26*
ckurtosis, *4, 5*, 6, *7, 8*, *10–13*, *16*, *19*, *26*
cmean, *4–6*, 7, *8*, *10–13*, *16*, *19*, *26*
cmedian, *4–7*, 8, *10–13*, *16*, *19*, *26*
convert_snake_to_title_case, 9, *17*, *24*
crange, *4–8*, 10, *11–13*, *16*, *19*, *26*
csd, *4–8*, *10*, 11, *12, 13*, *16*, *19*, *26*
cskewness, *4–8*, *10, 11*, 12, *13*, *16*, *19*, *26*
cvar, *4–8*, *10–12*, 13, *16*, *19*, *26*

discrete_walk, *3*, 14, *18*, *21, 22*
dplyr::cummean(), *7*

euclidean_distance, *4–8*, *10–13*, 15, *19*, *26*

generate_caption, *9*, 16, *24*
geometric_brownian_motion, *3*, *15*, 17, *21, 22*

kurtosis_vec, *4–8*, *10–13*, *16*, 19, *26*

rand_walk_helper, *9*, *17*, 23
random_normal_drift_walk, *3*, *15*, *18*, 20, *22*
random_normal_walk, *3*, *15*, *18*, *21*, 21
rw30, 24
rw_range, *4–8*, *10–13*, *16*, *19*, 25, *26*

skewness_vec, *4–8*, *10–13*, *16*, *19*, *26*, 26
summarise_walks *(*summarize_walks*)*, 27
summarize_walks, 27

visualize_walks, 28