Package: roll (via r-universe)

November 5, 2024

Type Package

Title Rolling and Expanding Statistics

Version 1.1.8

Date YYYY-MM-DD

Author Jason Foster

Maintainer Jason Foster <jason.j.foster@gmail.com>

Description Fast and efficient computation of rolling and expanding statistics for time-series data.

License GPL (>= 2)

URL https://github.com/jasonjfoster/roll

BugReports https://github.com/jasonjfoster/roll/issues

Imports Rcpp, RcppParallel

LinkingTo Rcpp, RcppArmadillo, RcppParallel

SystemRequirements GNU make

Roxygen list(old_usage = TRUE)

RoxygenNote 7.3.2

Encoding UTF-8

Suggests covr, testthat, zoo

Config/pak/sysreqs make

Repository https://jasonjfoster.r-universe.dev

RemoteUrl https://github.com/jasonjfoster/roll

RemoteRef HEAD

RemoteSha 8232f093ddeeee18e24a51a0bc854729013d2ba7

Contents

`roll-package` *Rolling and Expanding Statistics*

Description

Fast and efficient computation of rolling and expanding statistics for time-series data.

Details

`roll` is a package that provides fast and efficient computation of rolling and expanding statistics for time-series data.

The default algorithm in the `roll` package, and suitable for most applications, is an **online algorithm**. Based on the speed requirements and sequential nature of many problems in practice, online algorithms are a natural fit for computing rolling and expanding statistics of time-series data. That is, as observations are added and removed from a window, online algorithms update statistics and discard observations from memory (Welford, 1962; West, 1979); as a result, the amount of time to evaluate each function is significantly faster as the computation is independent of the window. In contrast, an offline algorithm requires all observations in memory to calculate the statistic for each window. Note that online algorithms are prone to loss of precision due to round-off error; hence, users can trade speed for accuracy and select the offline algorithm by setting the `online` argument to `FALSE`. Also, the RcppParallel package is used to parallelize the online algorithms across columns and across windows for the offline algorithms.

As mentioned above, the numerical calculations use the RcppParallel package to parallelize rolling and expanding statistics of time-series data. The RcppParallel package provides a complete toolkit

for creating safe, portable, high-performance parallel algorithms, built on top of the Intel Threading Building Blocks (TBB) and TinyThread libraries. By default, all the available cores on a machine are used for parallel algorithms. If users are either already taking advantage of parallelism or instead want to use a fixed number or proportion of threads, then set the number of threads in the RcppParallel package with the `RcppParallel::setThreadOptions` function.

Author(s)

Jason Foster

References

Welford, B.P. (1962). "Note on a Method for Calculating Corrected Sums of Squares and Products." *Technometrics*, 4(3), 419-420.

West, D.H.D. (1979). "Updating Mean and Variance Estimates: An Improved Method." *Communications of the ACM*, 22(9), 532-535.

`roll_all` *Rolling All*

Description

A function for computing the rolling and expanding all of time-series data.

Usage

```
roll_all(x, width, min_obs = width, complete_obs = FALSE,
  na_restore = FALSE, online = TRUE)
```

Arguments

`x`	logical vector or matrix. Rows are observations and columns are variables.
`width`	integer. Window size.
`min_obs`	integer. Minimum number of observations required to have a value within a window, otherwise result is `NA`.
`complete_obs`	logical. If `TRUE` then rows containing any missing values are removed, if `FALSE` then each value is used.
`na_restore`	logical. Should missing values be restored?
`online`	logical. Process observations using an online algorithm.

Value

An object of the same class and dimension as `x` with the rolling and expanding all.

Examples

```
n <- 15
x <- rnorm(n)

# rolling all with complete windows
roll_all(x < 0, width = 5)

# rolling all with partial windows
roll_all(x < 0, width = 5)

# expanding all with partial windows
roll_all(x < 0, width = n)
```

`roll_any` *Rolling Any*

Description

A function for computing the rolling and expanding any of time-series data.

Usage

```
roll_any(x, width, min_obs = width, complete_obs = FALSE,
  na_restore = FALSE, online = TRUE)
```

Arguments

`x`	logical vector or matrix. Rows are observations and columns are variables.
`width`	integer. Window size.
`min_obs`	integer. Minimum number of observations required to have a value within a window, otherwise result is `NA`.
`complete_obs`	logical. If `TRUE` then rows containing any missing values are removed, if `FALSE` then each value is used.
`na_restore`	logical. Should missing values be restored?
`online`	logical. Process observations using an online algorithm.

Value

An object of the same class and dimension as `x` with the rolling and expanding any.

Examples

```
n <- 15
x <- rnorm(n)

# rolling any with complete windows
roll_any(x < 0, width = 5)

# rolling any with partial windows
roll_any(x < 0, width = 5)

# expanding any with partial windows
roll_any(x < 0, width = n)
```

`roll_cor` *Rolling Correlations*

Description

A function for computing the rolling and expanding correlations of time-series data.

Usage

```
roll_cor(x, y = NULL, width, weights = rep(1, width), center = TRUE,
  scale = TRUE, min_obs = width, complete_obs = TRUE,
  na_restore = FALSE, online = TRUE)
```

Arguments

`x`	vector or matrix. Rows are observations and columns are variables.
`y`	vector or matrix. Rows are observations and columns are variables.
`width`	integer. Window size.
`weights`	vector. Weights for each observation within a window.
`center`	logical. If `TRUE` then the weighted mean of each variable is used, if `FALSE` then zero is used.
`scale`	logical. If `TRUE` then the weighted standard deviation of each variable is used, if `FALSE` then no scaling is done.
`min_obs`	integer. Minimum number of observations required to have a value within a window, otherwise result is `NA`.
`complete_obs`	logical. If `TRUE` then rows containing any missing values are removed, if `FALSE` then pairwise is used.
`na_restore`	logical. Should missing values be restored?
`online`	logical. Process observations using an online algorithm.

Details

The denominator used gives an unbiased estimate of the covariance, so if the weights are the default then the divisor `n - 1` is obtained.

Value

A cube with each slice the rolling and expanding correlations.

Examples

```
n <- 15
x <- rnorm(n)
y <- rnorm(n)
weights <- 0.9 ^ (n:1)

# rolling correlations with complete windows
roll_cor(x, y, width = 5)

# rolling correlations with partial windows
roll_cor(x, y, width = 5, min_obs = 1)

# expanding correlations with partial windows
roll_cor(x, y, width = n, min_obs = 1)

# expanding correlations with partial windows and weights
roll_cor(x, y, width = n, min_obs = 1, weights = weights)
```

---

roll_cov *Rolling Covariances*

Description

A function for computing the rolling and expanding covariances of time-series data.

Usage

```
roll_cov(x, y = NULL, width, weights = rep(1, width), center = TRUE,
  scale = FALSE, min_obs = width, complete_obs = TRUE,
  na_restore = FALSE, online = TRUE)
```

Arguments

`x`	vector or matrix. Rows are observations and columns are variables.
`y`	vector or matrix. Rows are observations and columns are variables.
`width`	integer. Window size.
`weights`	vector. Weights for each observation within a window.

center	logical. If TRUE then the weighted mean of each variable is used, if FALSE then zero is used.
scale	logical. If TRUE then the weighted standard deviation of each variable is used, if FALSE then no scaling is done.
min_obs	integer. Minimum number of observations required to have a value within a window, otherwise result is NA.
complete_obs	logical. If TRUE then rows containing any missing values are removed, if FALSE then pairwise is used.
na_restore	logical. Should missing values be restored?
online	logical. Process observations using an online algorithm.

Details

The denominator used gives an unbiased estimate of the covariance, so if the weights are the default then the divisor n - 1 is obtained.

Value

A cube with each slice the rolling and expanding covariances.

Examples

```
n <- 15
x <- rnorm(n)
y <- rnorm(n)
weights <- 0.9 ^ (n:1)

# rolling covariances with complete windows
roll_cov(x, y, width = 5)

# rolling covariances with partial windows
roll_cov(x, y, width = 5, min_obs = 1)

# expanding covariances with partial windows
roll_cov(x, y, width = n, min_obs = 1)

# expanding covariances with partial windows and weights
roll_cov(x, y, width = n, min_obs = 1, weights = weights)
```

---

roll_crossprod *Rolling Crossproducts*

Description

A function for computing the rolling and expanding crossproducts of time-series data.

Usage

```
roll_crossprod(x, y = NULL, width, weights = rep(1, width),
  center = FALSE, scale = FALSE, min_obs = width, complete_obs = TRUE,
  na_restore = FALSE, online = TRUE)
```

Arguments

x	vector or matrix. Rows are observations and columns are variables.
y	vector or matrix. Rows are observations and columns are variables.
width	integer. Window size.
weights	vector. Weights for each observation within a window.
center	logical. If TRUE then the weighted mean of each variable is used, if FALSE then zero is used.
scale	logical. If TRUE then the weighted standard deviation of each variable is used, if FALSE then no scaling is done.
min_obs	integer. Minimum number of observations required to have a value within a window, otherwise result is NA.
complete_obs	logical. If TRUE then rows containing any missing values are removed, if FALSE then pairwise is used.
na_restore	logical. Should missing values be restored?
online	logical. Process observations using an online algorithm.

Value

A cube with each slice the rolling and expanding crossproducts.

Examples

```
n <- 15
x <- rnorm(n)
y <- rnorm(n)
weights <- 0.9 ^ (n:1)

# rolling crossproducts with complete windows
roll_crossprod(x, y, width = 5)

# rolling crossproducts with partial windows
roll_crossprod(x, y, width = 5, min_obs = 1)

# expanding crossproducts with partial windows
roll_crossprod(x, y, width = n, min_obs = 1)

# expanding crossproducts with partial windows and weights
roll_crossprod(x, y, width = n, min_obs = 1, weights = weights)
```

roll_idxmax *Rolling Index of Maximums*

Description

A function for computing the rolling and expanding index of maximums of time-series data.

Usage

```
roll_idxmax(x, width, weights = rep(1, width), min_obs = width,
  complete_obs = FALSE, na_restore = FALSE, online = TRUE)
```

Arguments

x	vector or matrix. Rows are observations and columns are variables.
width	integer. Window size.
weights	vector. Weights for each observation within a window.
min_obs	integer. Minimum number of observations required to have a value within a window, otherwise result is NA.
complete_obs	logical. If TRUE then rows containing any missing values are removed, if FALSE then each value is used.
na_restore	logical. Should missing values be restored?
online	logical. Process observations using an online algorithm.

Value

An object of the same class and dimension as x with the rolling and expanding index of maximums.

Examples

```
n <- 15
x <- rnorm(n)
weights <- 0.9 ^ (n:1)

# rolling index of maximums with complete windows
roll_idxmax(x, width = 5)

# rolling index of maximums with partial windows
roll_idxmax(x, width = 5, min_obs = 1)

# expanding index of maximums with partial windows
roll_idxmax(x, width = n, min_obs = 1)

# expanding index of maximums with partial windows and weights
roll_idxmax(x, width = n, min_obs = 1, weights = weights)
```

roll_idxmin *Rolling Index of Minimums*

Description

A function for computing the rolling and expanding index of minimums of time-series data.

Usage

```
roll_idxmin(x, width, weights = rep(1, width), min_obs = width,
  complete_obs = FALSE, na_restore = FALSE, online = TRUE)
```

Arguments

x	vector or matrix. Rows are observations and columns are variables.
width	integer. Window size.
weights	vector. Weights for each observation within a window.
min_obs	integer. Minimum number of observations required to have a value within a window, otherwise result is NA.
complete_obs	logical. If TRUE then rows containing any missing values are removed, if FALSE then each value is used.
na_restore	logical. Should missing values be restored?
online	logical. Process observations using an online algorithm.

Value

An object of the same class and dimension as x with the rolling and expanding index of minimums.

Examples

```
n <- 15
x <- rnorm(n)
weights <- 0.9 ^ (n:1)

# rolling index of minimums with complete windows
roll_idxmin(x, width = 5)

# rolling index of minimums with partial windows
roll_idxmin(x, width = 5, min_obs = 1)

# expanding index of minimums with partial windows
roll_idxmin(x, width = n, min_obs = 1)

# expanding index of minimums with partial windows and weights
roll_idxmin(x, width = n, min_obs = 1, weights = weights)
```

roll_lm *Rolling Linear Models*

Description

A function for computing the rolling and expanding linear models of time-series data.

Usage

```
roll_lm(x, y, width, weights = rep(1, width), intercept = TRUE,
  min_obs = width, complete_obs = TRUE, na_restore = FALSE,
  online = TRUE)
```

Arguments

x	vector or matrix. Rows are observations and columns are the independent variables.
y	vector or matrix. Rows are observations and columns are the dependent variables.
width	integer. Window size.
weights	vector. Weights for each observation within a window.
intercept	logical. Either TRUE to include or FALSE to remove the intercept.
min_obs	integer. Minimum number of observations required to have a value within a window, otherwise result is NA.
complete_obs	logical. If TRUE then rows containing any missing values are removed, if FALSE then pairwise is used.
na_restore	logical. Should missing values be restored?
online	logical. Process observations using an online algorithm.

Value

A list containing the following components:

coefficients	A list of objects with the rolling and expanding coefficients for each y. An object is the same class and dimension (with an added column for the intercept) as x.
r.squared	A list of objects with the rolling and expanding r-squareds for each y. An object is the same class as x.
std.error	A list of objects with the rolling and expanding standard errors for each y. An object is the same class and dimension (with an added column for the intercept) as x.

Examples

```
n <- 15
x <- rnorm(n)
y <- rnorm(n)
weights <- 0.9 ^ (n:1)

# rolling regressions with complete windows
roll_lm(x, y, width = 5)

# rolling regressions with partial windows
roll_lm(x, y, width = 5, min_obs = 1)

# expanding regressions with partial windows
roll_lm(x, y, width = n, min_obs = 1)

# expanding regressions with partial windows and weights
roll_lm(x, y, width = n, min_obs = 1, weights = weights)
```

---

roll_max *Rolling Maximums*

Description

A function for computing the rolling and expanding maximums of time-series data.

Usage

```
roll_max(x, width, weights = rep(1, width), min_obs = width,
  complete_obs = FALSE, na_restore = FALSE, online = TRUE)
```

Arguments

x	vector or matrix. Rows are observations and columns are variables.
width	integer. Window size.
weights	vector. Weights for each observation within a window.
min_obs	integer. Minimum number of observations required to have a value within a window, otherwise result is NA.
complete_obs	logical. If TRUE then rows containing any missing values are removed, if FALSE then each value is used.
na_restore	logical. Should missing values be restored?
online	logical. Process observations using an online algorithm.

Value

An object of the same class and dimension as x with the rolling and expanding maximums.

Examples

```
n <- 15
x <- rnorm(n)
weights <- 0.9 ^ (n:1)

# rolling maximums with complete windows
roll_max(x, width = 5)

# rolling maximums with partial windows
roll_max(x, width = 5, min_obs = 1)

# expanding maximums with partial windows
roll_max(x, width = n, min_obs = 1)

# expanding maximums with partial windows and weights
roll_max(x, width = n, min_obs = 1, weights = weights)
```

---

roll_mean *Rolling Means*

Description

A function for computing the rolling and expanding means of time-series data.

Usage

```
roll_mean(x, width, weights = rep(1, width), min_obs = width,
  complete_obs = FALSE, na_restore = FALSE, online = TRUE)
```

Arguments

x	vector or matrix. Rows are observations and columns are variables.
width	integer. Window size.
weights	vector. Weights for each observation within a window.
min_obs	integer. Minimum number of observations required to have a value within a window, otherwise result is NA.
complete_obs	logical. If TRUE then rows containing any missing values are removed, if FALSE then each value is used.
na_restore	logical. Should missing values be restored?
online	logical. Process observations using an online algorithm.

Value

An object of the same class and dimension as x with the rolling and expanding means.

Examples

```
n <- 15
x <- rnorm(n)
weights <- 0.9 ^ (n:1)

# rolling means with complete windows
roll_mean(x, width = 5)

# rolling means with partial windows
roll_mean(x, width = 5, min_obs = 1)

# expanding means with partial windows
roll_mean(x, width = n, min_obs = 1)

# expanding means with partial windows and weights
roll_mean(x, width = n, min_obs = 1, weights = weights)
```

`roll_median` *Rolling Medians*

Description

A function for computing the rolling and expanding medians of time-series data.

Usage

```
roll_median(x, width, weights = rep(1, width), min_obs = width,
  complete_obs = FALSE, na_restore = FALSE, online = TRUE)
```

Arguments

`x`	vector or matrix. Rows are observations and columns are variables.
`width`	integer. Window size.
`weights`	vector. Weights for each observation within a window.
`min_obs`	integer. Minimum number of observations required to have a value within a window, otherwise result is `NA`.
`complete_obs`	logical. If `TRUE` then rows containing any missing values are removed, if `FALSE` then each value is used.
`na_restore`	logical. Should missing values be restored?
`online`	logical. Process observations using an online algorithm.

Value

An object of the same class and dimension as `x` with the rolling and expanding medians.

Examples

```
n <- 15
x <- rnorm(n)
weights <- 0.9 ^ (n:1)

# rolling medians with complete windows
roll_median(x, width = 5)

# rolling medians with partial windows
roll_median(x, width = 5, min_obs = 1)

# expanding medians with partial windows
roll_median(x, width = n, min_obs = 1)

## Not run:
# expanding medians with partial windows and weights
roll_median(x, width = n, min_obs = 1, weights = weights)

## End(Not run)
```

---

roll_min *Rolling Minimums*

Description

A function for computing the rolling and expanding minimums of time-series data.

Usage

```
roll_min(x, width, weights = rep(1, width), min_obs = width,
  complete_obs = FALSE, na_restore = FALSE, online = TRUE)
```

Arguments

x	vector or matrix. Rows are observations and columns are variables.
width	integer. Window size.
weights	vector. Weights for each observation within a window.
min_obs	integer. Minimum number of observations required to have a value within a window, otherwise result is NA.
complete_obs	logical. If TRUE then rows containing any missing values are removed, if FALSE then each value is used.
na_restore	logical. Should missing values be restored?
online	logical. Process observations using an online algorithm.

Value

An object of the same class and dimension as x with the rolling and expanding minimums.

Examples

```
n <- 15
x <- rnorm(n)
weights <- 0.9 ^ (n:1)

# rolling minimums with complete windows
roll_min(x, width = 5)

# rolling minimums with partial windows
roll_min(x, width = 5, min_obs = 1)

# expanding minimums with partial windows
roll_min(x, width = n, min_obs = 1)

# expanding minimums with partial windows and weights
roll_min(x, width = n, min_obs = 1, weights = weights)
```

---

roll_prod *Rolling Products*

Description

A function for computing the rolling and expanding products of time-series data.

Usage

```
roll_prod(x, width, weights = rep(1, width), min_obs = width,
  complete_obs = FALSE, na_restore = FALSE, online = TRUE)
```

Arguments

x	vector or matrix. Rows are observations and columns are variables.
width	integer. Window size.
weights	vector. Weights for each observation within a window.
min_obs	integer. Minimum number of observations required to have a value within a window, otherwise result is NA.
complete_obs	logical. If TRUE then rows containing any missing values are removed, if FALSE then each value is used.
na_restore	logical. Should missing values be restored?
online	logical. Process observations using an online algorithm.

Value

An object of the same class and dimension as x with the rolling and expanding products.

Examples

```
n <- 15
x <- rnorm(n)
weights <- 0.9 ^ (n:1)

# rolling products with complete windows
roll_prod(x, width = 5)

# rolling products with partial windows
roll_prod(x, width = 5, min_obs = 1)

# expanding products with partial windows
roll_prod(x, width = n, min_obs = 1)

# expanding products with partial windows and weights
roll_prod(x, width = n, min_obs = 1, weights = weights)
```

---

roll_quantile *Rolling Quantiles*

Description

A function for computing the rolling and expanding quantiles of time-series data.

Usage

```
roll_quantile(x, width, weights = rep(1, width), p = 0.5,
  min_obs = width, complete_obs = FALSE, na_restore = FALSE,
  online = TRUE)
```

Arguments

x	vector or matrix. Rows are observations and columns are variables.
width	integer. Window size.
weights	vector. Weights for each observation within a window.
p	numeric. Probability between zero and one.
min_obs	integer. Minimum number of observations required to have a value within a window, otherwise result is NA.
complete_obs	logical. If TRUE then rows containing any missing values are removed, if FALSE then each value is used.
na_restore	logical. Should missing values be restored?
online	logical. Process observations using an online algorithm.

Details

The methodology for computing the quantiles is based on the inverse of the empirical distribution function with averaging at discontinuities (see "Definition 2" in Hyndman and Fan, 1996).

Value

An object of the same class and dimension as x with the rolling and expanding quantiles.

References

Hyndman, R.J. and Fan, Y. (1996). "Sample quantiles in statistical packages." *American Statistician*, 50(4), 361-365.

Examples

```
n <- 15
x <- rnorm(n)
weights <- 0.9 ^ (n:1)

# rolling quantiles with complete windows
roll_quantile(x, width = 5)

# rolling quantiles with partial windows
roll_quantile(x, width = 5, min_obs = 1)

# expanding quantiles with partial windows
roll_quantile(x, width = n, min_obs = 1)

## Not run:
# expanding quantiles with partial windows and weights
roll_quantile(x, width = n, min_obs = 1, weights = weights)

## End(Not run)
```

---

roll_scale *Rolling Scaling and Centering*

Description

A function for computing the rolling and expanding scaling and centering of time-series data.

Usage

```
roll_scale(x, width, weights = rep(1, width), center = TRUE,
  scale = TRUE, min_obs = width, complete_obs = FALSE,
  na_restore = FALSE, online = TRUE)
```

Arguments

x	vector or matrix. Rows are observations and columns are variables.
width	integer. Window size.
weights	vector. Weights for each observation within a window.

center	logical. If TRUE then the weighted mean of each variable is used, if FALSE then zero is used.
scale	logical. If TRUE then the weighted standard deviation of each variable is used, if FALSE then no scaling is done.
min_obs	integer. Minimum number of observations required to have a value within a window, otherwise result is NA.
complete_obs	logical. If TRUE then rows containing any missing values are removed, if FALSE then each value is used.
na_restore	logical. Should missing values be restored?
online	logical. Process observations using an online algorithm.

Details

If center is TRUE then centering is done by subtracting the weighted mean from each variable, if FALSE then zero is used. After centering, if scale is TRUE then scaling is done by dividing by the weighted standard deviation for each variable if center is TRUE, and the root mean square otherwise. If scale is FALSE then no scaling is done.

The denominator used gives an unbiased estimate of the standard deviation, so if the weights are the default then the divisor n - 1 is obtained.

Value

An object of the same class and dimension as x with the rolling and expanding scaling and centering.

Examples

```
n <- 15
x <- rnorm(n)
weights <- 0.9 ^ (n:1)

# rolling z-scores with complete windows
roll_scale(x, width = 5)

# rolling z-scores with partial windows
roll_scale(x, width = 5, min_obs = 1)

# expanding z-scores with partial windows
roll_scale(x, width = n, min_obs = 1)

# expanding z-scores with partial windows and weights
roll_scale(x, width = n, min_obs = 1, weights = weights)
```

roll_sd *Rolling Standard Deviations*

Description

A function for computing the rolling and expanding standard deviations of time-series data.

Usage

```
roll_sd(x, width, weights = rep(1, width), center = TRUE,
  min_obs = width, complete_obs = FALSE, na_restore = FALSE,
  online = TRUE)
```

Arguments

x	vector or matrix. Rows are observations and columns are variables.
width	integer. Window size.
weights	vector. Weights for each observation within a window.
center	logical. If TRUE then the weighted mean of each variable is used, if FALSE then zero is used.
min_obs	integer. Minimum number of observations required to have a value within a window, otherwise result is NA.
complete_obs	logical. If TRUE then rows containing any missing values are removed, if FALSE then each value is used.
na_restore	logical. Should missing values be restored?
online	logical. Process observations using an online algorithm.

Details

The denominator used gives an unbiased estimate of the standard deviation, so if the weights are the default then the divisor n - 1 is obtained.

Value

An object of the same class and dimension as x with the rolling and expanding standard deviations.

Examples

```
n <- 15
x <- rnorm(n)
weights <- 0.9 ^ (n:1)

# rolling standard deviations with complete windows
roll_sd(x, width = 5)

# rolling standard deviations with partial windows
```

```
roll_sd(x, width = 5, min_obs = 1)

# expanding standard deviations with partial windows
roll_sd(x, width = n, min_obs = 1)

# expanding standard deviations with partial windows and weights
roll_sd(x, width = n, min_obs = 1, weights = weights)
```

`roll_sum` *Rolling Sums*

Description

A function for computing the rolling and expanding sums of time-series data.

Usage

```
roll_sum(x, width, weights = rep(1, width), min_obs = width,
  complete_obs = FALSE, na_restore = FALSE, online = TRUE)
```

Arguments

`x`	vector or matrix. Rows are observations and columns are variables.
`width`	integer. Window size.
`weights`	vector. Weights for each observation within a window.
`min_obs`	integer. Minimum number of observations required to have a value within a window, otherwise result is `NA`.
`complete_obs`	logical. If `TRUE` then rows containing any missing values are removed, if `FALSE` then each value is used.
`na_restore`	logical. Should missing values be restored?
`online`	logical. Process observations using an online algorithm.

Value

An object of the same class and dimension as `x` with the rolling and expanding sums.

Examples

```
n <- 15
x <- rnorm(n)
weights <- 0.9 ^ (n:1)

# rolling sums with complete windows
roll_sum(x, width = 5)

# rolling sums with partial windows
roll_sum(x, width = 5, min_obs = 1)
```

```
# expanding sums with partial windows
roll_sum(x, width = n, min_obs = 1)

# expanding sums with partial windows and weights
roll_sum(x, width = n, min_obs = 1, weights = weights)
```

`roll_var` *Rolling Variances*

Description

A function for computing the rolling and expanding variances of time-series data.

Usage

```
roll_var(x, width, weights = rep(1, width), center = TRUE,
  min_obs = width, complete_obs = FALSE, na_restore = FALSE,
  online = TRUE)
```

Arguments

`x`	vector or matrix. Rows are observations and columns are variables.
`width`	integer. Window size.
`weights`	vector. Weights for each observation within a window.
`center`	logical. If `TRUE` then the weighted mean of each variable is used, if `FALSE` then zero is used.
`min_obs`	integer. Minimum number of observations required to have a value within a window, otherwise result is `NA`.
`complete_obs`	logical. If `TRUE` then rows containing any missing values are removed, if `FALSE` then each value is used.
`na_restore`	logical. Should missing values be restored?
`online`	logical. Process observations using an online algorithm.

Details

The denominator used gives an unbiased estimate of the variance, so if the weights are the default then the divisor `n - 1` is obtained.

Value

An object of the same class and dimension as `x` with the rolling and expanding variances.

Examples

```
n <- 15
x <- rnorm(n)
weights <- 0.9 ^ (n:1)

# rolling variances with complete windows
roll_var(x, width = 5)

# rolling variances with partial windows
roll_var(x, width = 5, min_obs = 1)

# expanding variances with partial windows
roll_var(x, width = n, min_obs = 1)

# expanding variances with partial windows and weights
roll_var(x, width = n, min_obs = 1, weights = weights)
```

Index