

# Package ‘roll’

January 17, 2020

**Type** Package

**Title** Rolling Statistics

**Version** 1.1.4

**Date** 2020-01-16

**Author** Jason Foster

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

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

**License** GPL (>= 2)

**URL** <https://github.com/jjf234/roll>

**BugReports** <https://github.com/jjf234/roll/issues>

**Imports** Rcpp, RcppParallel

**LinkingTo** Rcpp, RcppArmadillo, RcppParallel

**SystemRequirements** GNU make, C++11

**RoxygenNote** 7.0.2

**Encoding** UTF-8

**Suggests** covr, testthat, zoo

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2020-01-17 09:40:08 UTC

## R topics documented:

roll-package . . . . .	2
roll_all . . . . .	3
roll_any . . . . .	4
roll_cor . . . . .	5
roll_cov . . . . .	6
roll_idxmax . . . . .	7
roll_idxmin . . . . .	8

roll_lm . . . . .	9
roll_max . . . . .	11
roll_mean . . . . .	12
roll_median . . . . .	13
roll_min . . . . .	14
roll_prod . . . . .	15
roll_scale . . . . .	16
roll_sd . . . . .	17
roll_sum . . . . .	18
roll_var . . . . .	19
<b>Index</b>	<b>21</b>

---

 roll-package

*Rolling Statistics*


---

## Description

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

## Details

Based on the speed requirements and sequential nature of many problems in practice, **online algorithms** are a natural fit for computing rolling statistics of time-series data. That is, as observations are added and removed from a rolling window, online algorithms update statistics and discard observations from memory. The default algorithm in the `roll` package, and suitable for most applications, is an online algorithm; however, in some cases it is impossible to recover the information needed to update each statistic. Specifically, if the `weights` vector is an arbitrarily changing sequence then an offline algorithm is used instead to calculate the rolling statistic. In the former case, the algorithm is parallelized across columns via `RcppParallel` and across windows in the latter case. 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`.

As mentioned above, the numerical calculations use `RcppParallel` to parallelize rolling statistics of time-series data. `RcppParallel` 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 `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	<i>Rolling All</i>
----------	--------------------

---

## Description

A function for computing the rolling 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 all.

## Examples

```
n_vars <- 3  
n_obs <- 15  
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)  
  
# rolling all  
result <- roll_all(x < 0, 5)
```

---

`roll_any`*Rolling Any*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

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

**Examples**

```
n_vars <- 3  
n_obs <- 15  
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)  
  
# rolling any  
result <- roll_any(x < 0, 5)
```

**Description**

A function for computing the rolling correlation matrices 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 correlation matrix.

**Examples**

```
n_vars <- 3
n_obs <- 15
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)

# rolling correlation matrices
result <- roll_cor(x, width = 5)

# rolling correlation matrices with exponential decay
weights <- 0.9 ^ (5:1)
result <- roll_cor(x, width = 5, weights = weights)
```

---

roll\_cov

*Rolling Covariance Matrices*

---

**Description**

A function for computing the rolling covariance matrices 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 covariance matrix.

### Examples

```
n_vars <- 3
n_obs <- 15
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)

# rolling covariance matrices
result <- roll_cov(x, width = 5)

# rolling covariance matrices with exponential decay
weights <- 0.9 ^ (5:1)
result <- roll_cov(x, width = 5, weights = weights)
```

---

roll_idxmax	<i>Rolling Index of Maximums</i>
-------------	----------------------------------

---

### Description

A function for computing the rolling 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 index of maximums.

**Examples**

```
n_vars <- 3
n_obs <- 15
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)

# rolling index of maximums
result <- roll_idxmax(x, 5)

# rolling index of maximums with exponential decay
weights <- 0.9 ^ (5:1)
result <- roll_idxmax(x, 5, weights)
```

---

roll_idxmin	<i>Rolling Index of Minimums</i>
-------------	----------------------------------

---

**Description**

A function for computing the rolling 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 index of minimums.

**Examples**

```
n_vars <- 3
n_obs <- 15
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)

# rolling index of minimums
result <- roll_idxmin(x, 5)

# rolling index of minimums with exponential decay
weights <- 0.9 ^ (5:1)
result <- roll_idxmin(x, 5, weights)
```

---

roll\_lm

*Rolling Linear Models*


---

**Description**

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

**Examples**

```
n_vars <- 3
n_obs <- 15
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)
y <- matrix(rnorm(n_obs), nrow = n_obs, ncol = 1)

# rolling regressions
result <- roll_lm(x, y, 5)

# rolling regressions with exponential decay
weights <- 0.9 ^ (5:1)
result <- roll_lm(x, y, 5, weights)
```

---

roll_max	<i>Rolling Maximums</i>
----------	-------------------------

---

**Description**

A function for computing the rolling 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 maximums.

**Examples**

```
n_vars <- 3  
n_obs <- 15  
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)  
  
# rolling maximums  
result <- roll_max(x, 5)  
  
# rolling maximums with exponential decay  
weights <- 0.9 ^ (5:1)  
result <- roll_max(x, 5, weights)
```

---

`roll_mean`*Rolling Means*

---

**Description**

A function for computing the rolling 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**

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

**Value**

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

**Examples**

```
n_vars <- 3  
n_obs <- 15  
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)  
  
# rolling means  
result <- roll_mean(x, 5)  
  
# rolling means with exponential decay  
weights <- 0.9 ^ (5:1)  
result <- roll_mean(x, 5, weights)
```

---

roll_median	<i>Rolling Medians</i>
-------------	------------------------

---

**Description**

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

**Usage**

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

**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 medians.

**Examples**

```
n_vars <- 3  
n_obs <- 15  
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)  
  
# rolling medians  
result <- roll_median(x, 5)  
  
# rolling medians with exponential decay  
weights <- 0.9 ^ (5:1)  
result <- roll_median(x, 5, weights)
```

---

roll_min	<i>Rolling Minimums</i>
----------	-------------------------

---

**Description**

A function for computing the rolling 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 minimums.

**Examples**

```
n_vars <- 3  
n_obs <- 15  
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)  
  
# rolling minimums  
result <- roll_min(x, 5)  
  
# rolling minimums with exponential decay  
weights <- 0.9 ^ (5:1)  
result <- roll_min(x, 5, weights)
```

---

roll_prod	<i>Rolling Products</i>
-----------	-------------------------

---

**Description**

A function for computing the rolling 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 products.

**Examples**

```
n_vars <- 3  
n_obs <- 15  
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)  
  
# rolling products  
result <- roll_prod(x, 5)  
  
# rolling products with exponential decay  
weights <- 0.9 ^ (5:1)  
result <- roll_prod(x, 5, weights)
```

---

roll_scale	<i>Rolling Scaling and Centering</i>
------------	--------------------------------------

---

**Description**

A function for computing the rolling 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**

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

**Examples**

```
n_vars <- 3
n_obs <- 15
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)

# rolling z-scores
result <- roll_scale(x, 5)

# rolling z-scores with exponential decay
weights <- 0.9 ^ (5:1)
result <- roll_scale(x, 5, weights)
```

---

roll_sd	<i>Rolling Standard Deviations</i>
---------	------------------------------------

---

**Description**

A function for computing the rolling 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**

<code>x</code>	vector or matrix. Rows are observations and columns are variables.
<code>width</code>	integer. Window size.
<code>weights</code>	vector. Weights for each observation within a window.
<code>center</code>	logical. If TRUE then the weighted mean of each variable is used, if FALSE then zero is used.
<code>min_obs</code>	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 standard deviations.

### Examples

```
n_vars <- 3
n_obs <- 15
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)

# rolling standard deviations
result <- roll_sd(x, 5)

# rolling standard deviations with exponential decay
weights <- 0.9 ^ (5:1)
result <- roll_sd(x, 5, weights)
```

---

roll\_sum

*Rolling Sums*

---

### Description

A function for computing the rolling 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 sums.

**Examples**

```
n_vars <- 3
n_obs <- 15
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)

# rolling sums
result <- roll_sum(x, 5)

# rolling sums with exponential decay
weights <- 0.9 ^ (5:1)
result <- roll_sum(x, 5, weights)
```

---

roll\_var

*Rolling Variances*


---

**Description**

A function for computing the rolling 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 variances.

**Examples**

```
n_vars <- 3
n_obs <- 15
x <- matrix(rnorm(n_obs * n_vars), nrow = n_obs, ncol = n_vars)

# rolling variances
result <- roll_var(x, 5)

# rolling variances with exponential decay
weights <- 0.9 ^ (5:1)
result <- roll_var(x, 5, weights)
```

# Index

roll (roll-package), 2  
roll-package, 2  
roll\_all, 3  
roll\_any, 4  
roll\_cor, 5  
roll\_cov, 6  
roll\_idxmax, 7  
roll\_idxmin, 8  
roll\_lm, 9  
roll\_max, 11  
roll\_mean, 12  
roll\_median, 13  
roll\_min, 14  
roll\_prod, 15  
roll\_scale, 16  
roll\_sd, 17  
roll\_sum, 18  
roll\_var, 19  
  
setThreadOptions, 2