Package: RcppRoll (via r-universe)

October 7, 2026

Type Package

Title Efficient Rolling / Windowed Operations

Version 0.4.0.9000

Description Provides fast and efficient routines for common rolling / windowed operations. Routines for the efficient computation of windowed mean, median, sum, product, minimum, maximum, standard deviation and variance are provided.

License GPL (>= 2)

URL https://kevinushey.github.io/RcppRoll/, https://github.com/kevinushey/RcppRoll

BugReports https://github.com/kevinushey/RcppRoll/issues

Depends R (>= 2.15.1)

Suggests zoo, testthat

Encoding UTF-8

RoxygenNote 7.3.3

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

Date/Publication 2026-09-07 20:11:08 UTC

RemoteUrl https://github.com/kevinushey/rcpproll

RemoteRef HEAD

RemoteSha 81b85b281844c3f31b617d0f710bd14daa8140cd

Contents

RcppRoll-package 2
RcppRoll-exports 3
roll_threads 9

Index **11**

RcppRoll-package *RcppRoll*

Description

This package implements a number of 'roll'-ing functions for R vectors and matrices.

Details

Currently, the exported functions are:

- roll_max
- roll_mean
- roll_median
- roll_min
- roll_prod
- roll_sd
- roll_sum
- roll_var

Parallelization

When the package is compiled with OpenMP support, the rolling window computations are parallelized across threads. By default, the number of threads is chosen by the OpenMP runtime (typically controlled through the OMP_NUM_THREADS environment variable); it can be set explicitly with options(RcppRoll.threads = <n>), and parallelization can be disabled with options(RcppRoll.threads = 1). Small inputs are always computed serially, and results are identical whatever the number of threads – including on builds without OpenMP support at all.

Use roll_threads() to check how many threads are in use, or whether the installed package has OpenMP support at all. The package also reports this when attached; suppress that message with options(RcppRoll.quiet = TRUE).

Author(s)

Maintainer: Kevin Ushey <kevinushey@gmail.com>

See Also

Useful links:

- https://kevinushey.github.io/RcppRoll/
- https://github.com/kevinushey/RcppRoll
- Report bugs at https://github.com/kevinushey/RcppRoll/issues

RcppRoll-exports *RcppRoll*

Description

Efficient windowed / rolling operations. Each function here applies an operation over a moving window of size `n`, with (customizable) weights specified through `weights`.

Usage

```
roll_mean(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = numeric(0),
  partial = FALSE,
  align = c("center", "left", "right"),
  normalize = TRUE,
  na.rm = FALSE
)

roll_meanr(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = NA,
  partial = FALSE,
  align = "right",
  normalize = TRUE,
  na.rm = FALSE
)

roll_meanl(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = NA,
  partial = FALSE,
  align = "left",
  normalize = TRUE,
  na.rm = FALSE
)

roll_median(
```

```
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = numeric(0),
  partial = FALSE,
  align = c("center", "left", "right"),
  normalize = TRUE,
  na.rm = FALSE
)

roll_medianr(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = NA,
  partial = FALSE,
  align = "right",
  normalize = TRUE,
  na.rm = FALSE
)

roll_medianl(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = NA,
  partial = FALSE,
  align = "left",
  normalize = TRUE,
  na.rm = FALSE
)

roll_min(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = numeric(0),
  partial = FALSE,
  align = c("center", "left", "right"),
  normalize = TRUE,
  na.rm = FALSE
)

roll_minr(
```

```
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = NA,
  partial = FALSE,
  align = "right",
  normalize = TRUE,
  na.rm = FALSE
)

roll_minl(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = NA,
  partial = FALSE,
  align = "left",
  normalize = TRUE,
  na.rm = FALSE
)

roll_max(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = numeric(0),
  partial = FALSE,
  align = c("center", "left", "right"),
  normalize = TRUE,
  na.rm = FALSE
)

roll_maxr(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = NA,
  partial = FALSE,
  align = "right",
  normalize = TRUE,
  na.rm = FALSE
)

roll_maxl(
```

```
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = NA,
  partial = FALSE,
  align = "left",
  normalize = TRUE,
  na.rm = FALSE
)

roll_prod(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = numeric(0),
  partial = FALSE,
  align = c("center", "left", "right"),
  normalize = TRUE,
  na.rm = FALSE
)

roll_prodr(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = NA,
  partial = FALSE,
  align = "right",
  normalize = TRUE,
  na.rm = FALSE
)

roll_prodl(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = NA,
  partial = FALSE,
  align = "left",
  normalize = TRUE,
  na.rm = FALSE
)

roll_sum(
```

```
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = numeric(0),
  partial = FALSE,
  align = c("center", "left", "right"),
  normalize = TRUE,
  na.rm = FALSE
)

roll_sumr(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = NA,
  partial = FALSE,
  align = "right",
  normalize = TRUE,
  na.rm = FALSE
)

roll_suml(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = NA,
  partial = FALSE,
  align = "left",
  normalize = TRUE,
  na.rm = FALSE
)

roll_sd(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = numeric(0),
  partial = FALSE,
  align = c("center", "left", "right"),
  normalize = TRUE,
  na.rm = FALSE
)

roll_sdr(
```

```
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = NA,
  partial = FALSE,
  align = "right",
  normalize = TRUE,
  na.rm = FALSE
)

roll_sdl(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = NA,
  partial = FALSE,
  align = "left",
  normalize = TRUE,
  na.rm = FALSE
)

roll_var(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = numeric(0),
  partial = FALSE,
  align = c("center", "left", "right"),
  normalize = TRUE,
  na.rm = FALSE
)

roll_varr(
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = NA,
  partial = FALSE,
  align = "right",
  normalize = TRUE,
  na.rm = FALSE
)

roll_varl(
```

```
  x,
  n = 1L,
  weights = NULL,
  by = 1L,
  fill = NA,
  partial = FALSE,
  align = "left",
  normalize = TRUE,
  na.rm = FALSE
)
```

Arguments

x	A numeric vector or a numeric matrix.
n	A positive integer scalar giving the window size. Ignored when weights is non-NULL.
weights	A non-empty vector of length n, giving the weights for each element within a window. If NULL, we take unit weights of width n. For an even window with uniform weights, roll_median() returns the lower of the two central values; the unweighted median instead averages those values. Variance and standard deviation require finite, non-negative frequency weights.
by	A positive integer scalar. Calculate at every by-th point rather than every point.
fill	Either an empty vector (no fill), or a vector (recycled to) length 3 giving left, center and right fills.
partial	A single non-missing logical. Compute windows at the edges of x over however many elements are in range, rather than filling them? Cannot be combined with weights, and fill does not apply.
align	Align windows on the "left", "center" or "right".
normalize	A single non-missing logical. Normalize window weights, such that they sum to n? Normalized weights must be finite and have a finite, non-zero sum.
na.rm	A single non-missing logical. Remove missing values?

Details

The functions postfixed with l and r are convenience wrappers that set **l**eft / **r**ight alignment of the windowed operations.

---

roll_threads *Report the Number of Threads Used for Rolling Computations*

Description

Reports how many threads the rolling computations would put to work on a large enough input. The options(RcppRoll.threads) value, when set, must be a positive integer scalar; otherwise the count defers to the OpenMP runtime default (typically controlled through the OMP_NUM_THREADS environment variable). Small inputs are always computed serially, and results do not depend on the number of threads used.

Usage

```
roll_threads()
```

Value

An integer scalar: the maximum number of threads used, or `NA` if the package was compiled without `OpenMP` support – useful for checking what an installation from sources ended up with.

Index

RcppRoll *(RcppRoll-package)*, 2
RcppRoll-exports, 3
RcppRoll-package, 2
roll_max, *2*
roll_max *(RcppRoll-exports)*, 3
roll_maxl *(RcppRoll-exports)*, 3
roll_maxr *(RcppRoll-exports)*, 3
roll_mean, *2*
roll_mean *(RcppRoll-exports)*, 3
roll_meanl *(RcppRoll-exports)*, 3
roll_meanr *(RcppRoll-exports)*, 3
roll_median, *2*
roll_median *(RcppRoll-exports)*, 3
roll_medianl *(RcppRoll-exports)*, 3
roll_medianr *(RcppRoll-exports)*, 3
roll_min, *2*
roll_min *(RcppRoll-exports)*, 3
roll_minl *(RcppRoll-exports)*, 3
roll_minr *(RcppRoll-exports)*, 3
roll_prod, *2*
roll_prod *(RcppRoll-exports)*, 3
roll_prodl *(RcppRoll-exports)*, 3
roll_prodr *(RcppRoll-exports)*, 3
roll_sd, *2*
roll_sd *(RcppRoll-exports)*, 3
roll_sdl *(RcppRoll-exports)*, 3
roll_sdr *(RcppRoll-exports)*, 3
roll_sum, *2*
roll_sum *(RcppRoll-exports)*, 3
roll_suml *(RcppRoll-exports)*, 3
roll_sumr *(RcppRoll-exports)*, 3
roll_threads, *2*, 9
roll_var, *2*
roll_var *(RcppRoll-exports)*, 3
roll_varl *(RcppRoll-exports)*, 3
roll_varr *(RcppRoll-exports)*, 3