Package 'MazamaRollUtils'

May 7, 2026

Type Package

Title Efficient Rolling Functions

Version 1.0.0

Description Fast rolling-window functions for numeric vectors.
Designed for efficient processing of environmental time-series data.

License GPL-3

URL https://github.com/MazamaScience/MazamaRollUtils,
https://mazamascience.github.io/MazamaRollUtils/

BugReports https://github.com/MazamaScience/MazamaRollUtils/issues

Depends R (>= 4.0.0)

Imports Rcpp (>= 1.0.10)

Suggests knitr, markdown, rmarkdown, roxygen2, testthat (>= 3.2.0),
zoo

LinkingTo Rcpp

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation yes

Author Jonathan Callahan [aut, cre],
Hans Martin [aut]

Maintainer Jonathan Callahan <jonathan.s.callahan@gmail.com>

Repository CRAN

Date/Publication 2026-03-17 17:00:11 UTC

Contents

`MazamaRollUtils-package` *Mazama Science Rolling Utilities*

Description

A suite of utility functions for calculating rolling mins, means, maxes and other functions written with an efficient Rcpp/C++ backend.

Author(s)

Jonathan Callahan, Hans Martin

`example_pm25` *Example timeseries dataset*

Description

The `example_pm25` dataset provides example timeseries data for practicing and code examples. This dataset represents hourly air quality measurements.

The dataset was generated on 2021-09-22 by running:

```
library(AirSensor)

example_pm25 <- example_sensor$data
names(example_pm25) <- c("datetime", "pm25")

save(example_pm25, file = "data/example_pm25.rda")
```

Usage

```
example_pm25
```

Format

A data frame with 2 columns:

datetime POSIXct timestamp of the observation

pm25 PM2.5 concentration (µg/m³)

`findOutliers` *Outlier detection with a rolling Hampel filter*

Description

A wrapper around `roll_hampel()` that identifies outliers using either a fixed threshold or a threshold derived from the input data.

Usage

```
findOutliers(
  x,
  width = 25,
  thresholdMin = 7,
  selectivity = NA,
  fixedThreshold = TRUE
)
```

Arguments

`x`	Numeric vector.
`width`	Integer width of the rolling window.
`thresholdMin`	Numeric threshold for outlier detection
`selectivity`	Value between 0 and 1 used in determining outliers, or `NA` if `fixedThreshold=TRUE`.
`fixedThreshold`	Logical specifying whether outlier detection uses `selectivity` (see Details).

Details

The `thresholdMin` level is similar to a sigma value for normally distributed data. Hampel filter values above 6 indicate a data value that is extremely unlikely to be part of a normal distribution (~ 1/500 million) and therefore very likely to be an outlier. By choosing a relatively large value for `thresholdMin` we make it less likely that we will generate false positives. False positives can include high frequency environmental noise.

With the default setting of `fixedThreshold = TRUE` any value above the threshold is considered an outlier and the `selectivity` is ignored.

The `selectivity` is a value between 0 and 1 and is used to generate an appropriate threshold for outlier detection based on the Hampel filter values computed from the incoming data. A lower value for `selectivity` will result in more outliers, while a value closer to 1.0 will result in fewer. If `fixedThreshold=TRUE`, `selectivity` may have a value of `NA`.

When the user specifies `fixedThreshold=FALSE`, the `thresholdMin` and `selectivity` parameters work like squelch and volume on a CB radio: `thresholdMin` sets a noise threshold below which you don't want anything returned while `selectivity` adjusts the number of points defined as outliers by setting a new threshold defined by the maximum value of `roll_hampel` multiplied by `selectivity`.

`width`, the window width, is a parameter that is passed to `roll_hampel()`.

Value

A vector of indices associated with outliers in the incoming data `x`.

Note

This function is copied from the **seismicRoll** package.

See Also

`roll_hampel()`

Examples

```
# Noisy sinusoid with outliers
a <- jitter(sin(0.1*seq(1e4)),amount=0.2)
indices <- sample(seq(1e4),20)
a[indices] <- a[indices]*10

# Outlier detection should identify many of these altered indices
sort(indices)
o_indices <- findOutliers(a)
o_indices

plot(a)
points(o_indices, a[o_indices], pch = 16, cex = 0.8, col = 'red')
title("Outlier detection using a Hampel filter")
```

`roll_hampel` *Roll Hampel*

Description

Apply a moving-window Hampel function to a numeric vector.

Usage

```
roll_hampel(
  x,
  width = 1L,
  by = 1L,
  align = c("center", "left", "right"),
  na.rm = FALSE
)
```

Arguments

x	Numeric vector.
width	Integer width of the rolling window.
by	Integer shift by which the window is moved each iteration.
align	Character position of the return value within the window. One of: "left" | "center" | "right".
na.rm	Logical specifying whether NA values should be removed before the calculations within each window.

Details

The Hampel filter is a robust outlier detector using Median Absolute Deviation (MAD).

For every index in the incoming vector x, a value is returned that is the Hampel function of all values in x that fall within a window of width width.

The align parameter determines the alignment of the return value within the window. Thus:

- align = "left" [*------] will cause the returned vector to have width - 1 NA values at the right end.
- align = "center" [---*---] will cause the returned vector to have NA values at either end as needed for centered alignment.
- align = "right" [------*] will cause the returned vector to have width - 1 NA values at the left end.

For large vectors, the by parameter can be used to force the window to jump ahead by indices for the next calculation. Indices that are skipped over will be assigned NA values so that the return vector still has the same length as the incoming vector. This can dramatically speed up calculations for high resolution time series data.

Value

Numeric vector of the same length as x.

Examples

```
x <- c(0, 0, 0, 1, 1, 2, 2, 4, 6, 9, 0, 0, 0)
roll_hampel(x, 3)
```

roll_MAD *Roll MAD*

Description

Apply a moving-window Median Absolute Deviation function to a numeric vector.

Usage

```
roll_MAD(
  x,
  width = 1L,
  by = 1L,
  align = c("center", "left", "right"),
  na.rm = FALSE
)
```

Arguments

x	Numeric vector.
width	Integer width of the rolling window.
by	Integer shift by which the window is moved each iteration.
align	Character position of the return value within the window. One of: "left" | "center" | "right".
na.rm	Logical specifying whether NA values should be removed before the calculations within each window.

Details

For every index in the incoming vector x, a value is returned that is the Median Absolute Deviation (MAD) of all values in x that fall within a window of width width.

The align parameter determines the alignment of the return value within the window. Thus:

- align = "left" [*------] will cause the returned vector to have width - 1 NA values at the right end.
- align = "center" [---*---] will cause the returned vector to have NA values at either end as needed for centered alignment.
- align = "right" [------*] will cause the returned vector to have width - 1 NA values at the left end.

For large vectors, the by parameter can be used to force the window to jump ahead by indices for the next calculation. Indices that are skipped over will be assigned NA values so that the return vector still has the same length as the incoming vector. This can dramatically speed up calculations for high resolution time series data.

Value

Numeric vector of the same length as x.

Examples

```
# Wikipedia example
x <- c(0, 0, 0, 1, 1, 2, 2, 4, 6, 9, 0, 0, 0)
roll_MAD(x, 3)
roll_MAD(x, 5)
roll_MAD(x, 7)
```

---

roll_max *Roll Max*

Description

Apply a moving-window maximum function to a numeric vector.

Usage

```
roll_max(
  x,
  width = 1L,
  by = 1L,
  align = c("center", "left", "right"),
  na.rm = FALSE
)
```

Arguments

x	Numeric vector.
width	Integer width of the rolling window.
by	Integer shift by which the window is moved each iteration.
align	Character position of the return value within the window. One of: "left" | "center" | "right".
na.rm	Logical specifying whether NA values should be removed before the calculations within each window.

Details

For every index in the incoming vector x, a value is returned that is the maximum of all values in x that fall within a window of width width.

The align parameter determines the alignment of the return value within the window. Thus:

- align = "left" [*------] will cause the returned vector to have width - 1 NA values at the right end.

- `align = "center" [---*---]` will cause the returned vector to have `NA` values at either end as needed for centered alignment.
- `align = "right"  [------*]` will cause the returned vector to have width - 1 `NA` values at the left end.

For large vectors, the by parameter can be used to force the window to jump ahead by indices for the next calculation. Indices that are skipped over will be assigned `NA` values so that the return vector still has the same length as the incoming vector. This can dramatically speed up calculations for high resolution time series data.

Value

Numeric vector of the same length as `x`.

Examples

```
# Example air quality time series
t <- example_pm25$datetime
x <- example_pm25$pm25

plot(t, x, pch = 16, cex = 0.5)
lines(t, roll_max(x, width = 12), col = "red")
lines(t, roll_min(x, width = 12), col = "deepskyblue")
title("12-hr Rolling Max and Min")

plot(t, x, pch = 16, cex = 0.5)
points(t, roll_max(x, width = 12, na.rm = TRUE),
       pch = 16, col = "red")
points(t, roll_max(x, width = 12, na.rm = FALSE),
       pch = 16, col = adjustcolor("black", 0.4))
legend("topright", pch = c(1, 16),
       col = c("red", adjustcolor("black", 0.4)),
       legend = c("na.rm = TRUE", "na.rm = FALSE"))
title("12-hr Rolling max with/out na.rm")
```

---

roll_mean *Roll Mean*

Description

Apply a moving-window mean function to a numeric vector.

Usage

```
roll_mean(
  x,
  width = 1L,
  by = 1L,
  align = c("center", "left", "right"),
```

```
  na.rm = FALSE,
  weights = NULL
)
```

Arguments

x	Numeric vector.
width	Integer width of the rolling window.
by	Integer shift by which the window is moved each iteration.
align	Character position of the return value within the window. One of: "left" | "center" | "right".
na.rm	Logical specifying whether NA values should be removed before the calculations within each window.
weights	Numeric vector of length width specifying each window index weight. If NULL, unit weights are used.

Details

For every index in the incoming vector x, a value is returned that is the mean of all values in x that fall within a window of width width.

The align parameter determines the alignment of the return value within the window. Thus:

- align = "left" [*------] will cause the returned vector to have width - 1 NA values at the right end.
- align = "center" [---*---] will cause the returned vector to have NA values at either end as needed for centered alignment.
- align = "right" [------*] will cause the returned vector to have width - 1 NA values at the left end.

For large vectors, the by parameter can be used to force the window to jump ahead by indices for the next calculation. Indices that are skipped over will be assigned NA values so that the return vector still has the same length as the incoming vector. This can dramatically speed up calculations for high resolution time series data.

The roll_mean() function supports an additional weights argument that can be used to calculate a weighted moving average, a convolution of the incoming data with the kernel provided in weights.

Value

Numeric vector of the same length as x.

Examples

```
# Example air quality time series
t <- example_pm25$datetime
x <- example_pm25$pm25

plot(t, x, pch = 16, cex = 0.5)
lines(t, roll_mean(x, width = 3), col = "goldenrod")
```

```
lines(t, roll_mean(x, width = 23), col = "purple")
legend("topright", lty = c(1, 1),
       col = c("goldenrod", "purple"),
       legend = c("3-hr mean", "23-hr mean"))
title("3- and 23-hr Rolling mean")
```

`roll_median` *Roll Median*

Description

Apply a moving-window median function to a numeric vector.

Usage

```
roll_median(
  x,
  width = 1L,
  by = 1L,
  align = c("center", "left", "right"),
  na.rm = FALSE
)
```

Arguments

`x`	Numeric vector.
`width`	Integer width of the rolling window.
`by`	Integer shift by which the window is moved each iteration.
`align`	Character position of the return value within the window. One of: `"left"` | `"center"` | `"right"`.
`na.rm`	Logical specifying whether `NA` values should be removed before the calculations within each window.

Details

For every index in the incoming vector `x`, a value is returned that is the median of all values in `x` that fall within a window of width `width`.

The `align` parameter determines the alignment of the return value within the window. Thus:

- `align = "left"` `[*------]` will cause the returned vector to have width - 1 `NA` values at the right end.
- `align = "center"` `[---*---]` will cause the returned vector to have `NA` values at either end as needed for centered alignment.
- `align = "right"` `[------*]` will cause the returned vector to have width - 1 `NA` values at the left end.

For large vectors, the by parameter can be used to force the window to jump ahead by indices for the next calculation. Indices that are skipped over will be assigned NA values so that the return vector still has the same length as the incoming vector. This can dramatically speed up calculations for high resolution time series data.

Value

Numeric vector of the same length as x.

Examples

```
# Example air quality time series
t <- example_pm25$datetime
x <- example_pm25$pm25

plot(t, x, pch = 16, cex = 0.5)
lines(t, roll_median(x, width = 3), col = "goldenrod")
lines(t, roll_median(x, width = 23), col = "purple")
legend("topright", lty = c(1, 1),
       col = c("goldenrod", "purple"),
       legend = c("3-hr median", "23-hr median"))
title("3- and 23-hr Rolling median")
```

---

roll_min *Roll Min*

Description

Apply a moving-window minimum function to a numeric vector.

Usage

```
roll_min(
  x,
  width = 1L,
  by = 1L,
  align = c("center", "left", "right"),
  na.rm = FALSE
)
```

Arguments

x	Numeric vector.
width	Integer width of the rolling window.
by	Integer shift by which the window is moved each iteration.
align	Character position of the return value within the window. One of: "left" | "center" | "right".
na.rm	Logical specifying whether NA values should be removed before the calculations within each window.

Details

For every index in the incoming vector x, a value is returned that is the minimum of all values in x that fall within a window of width width.

The align parameter determines the alignment of the return value within the window. Thus:

- align = "left" [*------] will cause the returned vector to have width - 1 NA values at the right end.
- align = "center" [---*---] will cause the returned vector to have NA values at either end as needed for centered alignment.
- align = "right" [------*] will cause the returned vector to have width - 1 NA values at the left end.

For large vectors, the by parameter can be used to force the window to jump ahead by indices for the next calculation. Indices that are skipped over will be assigned NA values so that the return vector still has the same length as the incoming vector. This can dramatically speed up calculations for high resolution time series data.

Value

Numeric vector of the same length as x.

Examples

```
# Example air quality time series
t <- example_pm25$datetime
x <- example_pm25$pm25

plot(t, x, pch = 16, cex = 0.5)
lines(t, roll_max(x, width = 12), col = "red")
lines(t, roll_min(x, width = 12), col = "deepskyblue")
title("12-hr Rolling Max and Min")

plot(t, x, pch = 16, cex = 0.5)
points(t, roll_min(x, width = 12, na.rm = TRUE),
       pch = 16, col = "deepskyblue")
points(t, roll_min(x, width = 12, na.rm = FALSE),
       pch = 16, col = adjustcolor("black", 0.4))
legend("topright", pch = c(16, 16),
       col = c("deepskyblue", adjustcolor("black", 0.4)),
       legend = c("na.rm = TRUE", "na.rm = FALSE"))
title("12-hr Rolling min with/out na.rm")
```

---

roll_nowcast *Roll NowCast*

Description

Apply the EPA NowCast algorithm to a numeric vector of hourly particulate matter measurements.

Usage

```
roll_nowcast(x)
```

Arguments

x	Numeric vector of hourly PM measurements.

Details

The EPA NowCast is a weighted average designed to emphasize more recent hourly PM values while still using up to the previous 12 hours of data. The weighting depends on how much concentrations vary within the window: rapidly changing conditions place more weight on the most recent hours, while stable conditions allow older hours to contribute more evenly.

For every index in the incoming vector x, a value is returned that is the NowCast associated with that hour.

This calculation is always right-aligned:

- `[----------*]` where `*` marks the hour receiving the NowCast value.

Early values use all available data back to the beginning of the series, so the first 11 positions may be calculated from fewer than 12 hours.

Missing values are allowed, but at least 2 valid values must be present in the most recent 3 hours or the result for that index will be `NA`.

Returned values are rounded to one decimal place.

Value

Numeric vector of the same length as `x`.

Examples

```
x <- c(10, 12, 11, 13, 15, 18, 20, 25, 30, 28, 26, 24, 22)
roll_nowcast(x)
```

---

roll_prod *Roll Product*

Description

Apply a moving-window product function to a numeric vector.

Usage

```
roll_prod(
  x,
  width = 1L,
  by = 1L,
  align = c("center", "left", "right"),
  na.rm = FALSE
)
```

Arguments

x	Numeric vector.
width	Integer width of the rolling window.
by	Integer shift by which the window is moved each iteration.
align	Character position of the return value within the window. One of: "left" | "center" | "right".
na.rm	Logical specifying whether NA values should be removed before the calculations within each window.

Details

For every index in the incoming vector x, a value is returned that is the product of all values in x that fall within a window of width width.

The align parameter determines the alignment of the return value within the window. Thus:

- align = "left" [*------] will cause the returned vector to have width - 1 NA values at the right end.
- align = "center" [---*---] will cause the returned vector to have NA values at either end as needed for centered alignment.
- align = "right" [------*] will cause the returned vector to have width - 1 NA values at the left end.

For large vectors, the by parameter can be used to force the window to jump ahead by indices for the next calculation. Indices that are skipped over will be assigned NA values so that the return vector still has the same length as the incoming vector. This can dramatically speed up calculations for high resolution time series data.

Value

Numeric vector of the same length as x.

Examples

```
# Example air quality time series
t <- example_pm25$datetime
x <- example_pm25$pm25

x[1:10]
roll_prod(x, width = 5)[1:10]
```

roll_sd *Roll Standard Deviation*

Description

Apply a moving-window standard deviation function to a numeric vector.

Usage

```
roll_sd(x, width = 1L, by = 1L, align = c("center", "left", "right"))
```

Arguments

x	Numeric vector.
width	Integer width of the rolling window.
by	Integer shift by which the window is moved each iteration.
align	Character position of the return value within the window. One of: "left" | "center" | "right".

Details

For every index in the incoming vector x, a value is returned that is the standard deviation of all values in x that fall within a window of width width.

The align parameter determines the alignment of the return value within the window. Thus:

- align = "left" [*------] will cause the returned vector to have width - 1 NA values at the right end.
- align = "center" [---*---] will cause the returned vector to have NA values at either end as needed for centered alignment.
- align = "right" [------*] will cause the returned vector to have width - 1 NA values at the left end.

For large vectors, the by parameter can be used to force the window to jump ahead by indices for the next calculation. Indices that are skipped over will be assigned NA values so that the return vector still has the same length as the incoming vector. This can dramatically speed up calculations for high resolution time series data.

Value

Numeric vector of the same length as x.

Note

A na.rm argument is not provided for roll_sd() because the statistical meaning of standard deviation computed from partially missing windows may be ambiguous.

Examples

```
# Example air quality time series
t <- example_pm25$datetime
x <- example_pm25$pm25

x[1:10]
roll_sd(x, width = 5)[1:10]
```

`roll_sum` *Roll Sum*

Description

Apply a moving-window sum to a numeric vector.

Usage

```
roll_sum(
  x,
  width = 1L,
  by = 1L,
  align = c("center", "left", "right"),
  na.rm = FALSE
)
```

Arguments

`x`	Numeric vector.
`width`	Integer width of the rolling window.
`by`	Integer shift by which the window is moved each iteration.
`align`	Character position of the return value within the window. One of: `"left"` | `"center"` | `"right"`.
`na.rm`	Logical specifying whether `NA` values should be removed before the calculations within each window.

Details

For every index in the incoming vector `x`, a value is returned that is the sum of all values in `x` that fall within a window of width `width`.

The `align` parameter determines the alignment of the return value within the window. Thus:

- `align = "left"` `[*------]` will cause the returned vector to have width - 1 `NA` values at the right end.
- `align = "center"` `[---*---]` will cause the returned vector to have `NA` values at either end as needed for centered alignment.

- `align = "right"` `[------*]` will cause the returned vector to have width - 1 `NA` values at the left end.

For large vectors, the by parameter can be used to force the window to jump ahead by indices for the next calculation. Indices that are skipped over will be assigned `NA` values so that the return vector still has the same length as the incoming vector. This can dramatically speed up calculations for high resolution time series data.

Value

Numeric vector of the same length as `x`.

Examples

```
# Example air quality time series
t <- example_pm25$datetime
x <- example_pm25$pm25

x[1:10]
roll_sum(x, width = 5)[1:10]
```

`roll_var` *Roll Variance*

Description

Apply a moving-window variance function to a numeric vector.

Usage

```
roll_var(x, width = 1L, by = 1L, align = c("center", "left", "right"))
```

Arguments

`x`	Numeric vector.
`width`	Integer width of the rolling window.
`by`	Integer shift by which the window is moved each iteration.
`align`	Character position of the return value within the window. One of: `"left"` | `"center"` | `"right"`.

Details

For every index in the incoming vector `x`, a value is returned that is the variance of all values in `x` that fall within a window of width `width`.

The `align` parameter determines the alignment of the return value within the window. Thus:

- `align = "left"` `[*------]` will cause the returned vector to have width - 1 `NA` values at the right end.

- `align = "center" [---*---]` will cause the returned vector to have `NA` values at either end as needed for centered alignment.
- `align = "right"  [------*]` will cause the returned vector to have width - 1 `NA` values at the left end.

For large vectors, the by parameter can be used to force the window to jump ahead by indices for the next calculation. Indices that are skipped over will be assigned `NA` values so that the return vector still has the same length as the incoming vector. This can dramatically speed up calculations for high resolution time series data.

Value

Numeric vector of the same length as `x`.

Note

A `na.rm` argument is not provided for `roll_var()` because the statistical meaning of variance computed from partially missing windows may be ambiguous.

Examples

```
# Example air quality time series
t <- example_pm25$datetime
x <- example_pm25$pm25

x[1:10]
roll_var(x, width = 5)[1:10]
```

Index