# Package 'MazamaRollUtils'

May 9, 2025

**Type** Package

**Title** Efficient Rolling Functions

**Version** 0.1.4

**Date** 2025-05-07

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

**Description** A suite of compiled functions calculating rolling mins, means, maxes and other statistics. This package is designed to meet the needs of data processing systems for environmental time series.

**License** GPL-3

**URL** https://github.com/MazamaScience/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

**Encoding** UTF-8

**VignetteBuilder** knitr

**LazyData** true

**RoxygenNote** 7.3.1

**NeedsCompilation** yes

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

**Repository** CRAN

**Date/Publication** 2025-05-09 14:00:05 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_data` dataset provides example timeseries data for practicing and code examples. This data represents hourly air quality measurements.

This dataset was 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 dataframe with columns "datetime" and "pm25".

---

`findOutliers` *Outlier Detection with a Rolling Hampel Filter*

---

## Description

A wrapper for the `roll_hampel()` function that counts outliers using either a user specified threshold value or a threshold value based on the statistics of the incoming 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 [0-1] 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 statistics of 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 to use when sliding the window to the next location |
| 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 funcion 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 = -1 [*------] will cause the returned vector to have width-1 NA values at the right end.
- align = 0 [---*---] will cause the returned vector to have width/2 NA values at either end.
- align = 1 [------*] will cause the returned vector to have width-1 NA values at the left end.

For large vectors, theby 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

```
library(MazamaRollUtils)

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 to use when sliding the window to the next location |
| 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 = -1 [*------] will cause the returned vector to have width-1 NA values at the right end.
- align = 0 [---*---] will cause the returned vector to have width/2 NA values at either end.
- align = 1 [------*] will cause the returned vector to have width-1 NA values at the left end.

For large vectors, theby 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

```
library(MazamaRollUtils)

# 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 to use when sliding the window to the next location |
| `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 = -1` `[*------]` will cause the returned vector to have width-1 `NA` values at the right end.
- `align = 0` `[---*---]` will cause the returned vector to have width/2 `NA` values at either end.

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

For large vectors, theby 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

```
library(MazamaRollUtils)

# 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 size 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 = -1 [*------] will cause the returned vector to have width-1 NA values at the right end.
- align = 0 [---*---] will cause the returned vector to have width/2 NA values at either end.
- align = 1 [------*] will cause the returned vector to have width-1 NA values at the left end.

For large vectors, theby 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* (weighting function) provided in weights.

## Value

Numeric vector of the same length as x.

## Examples

```
library(MazamaRollUtils)

# 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", "12-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 = -1` `[*------]` will cause the returned vector to have width-1 `NA` values at the right end.
- `align = 0` `[---*---]` will cause the returned vector to have width/2 `NA` values at either end.
- `align = 1` `[------*]` 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

```
library(MazamaRollUtils)

# 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", "12-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 = -1 [*------] will cause the returned vector to have width-1 NA values at the right end.
- align = 0 [---*---] will cause the returned vector to have width/2 NA values at either end.
- align = 1 [------*] will cause the returned vector to have width-1 NA values at the left end.

For large vectors, theby 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

```
library(MazamaRollUtils)

# 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_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 = -1 [*------] will cause the returned vector to have width-1 NA values at the right end.
- align = 0 [---*---] will cause the returned vector to have width/2 NA values at either end.
- align = 1 [------*] will cause the returned vector to have width-1 NA values at the left end.

For large vectors, theby 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

```
library(MazamaRollUtils)

# 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 = -1 [*------] will cause the returned vector to have width-1 NA values at the right end.
- align = 0 [---*---] will cause the returned vector to have width/2 NA values at either end.
- align = 1 [------*] will cause the returned vector to have width-1 NA values at the left end.

For large vectors, theby 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

No na.rm argument is provided as interpretation of the results is not at all clear.

## Examples

```
library(MazamaRollUtils)

# 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` = `-1` `[*------]` will cause the returned vector to have width-1 `NA` values at the right end.
- `align` = `0` `[---*---]` will cause the returned vector to have width/2 `NA` values at either end.

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

For large vectors, theby 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

```
library(MazamaRollUtils)

# 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 = -1` `[*------]` will cause the returned vector to have width-1 `NA` values at the right end.
- `align = 0` `[---*---]` will cause the returned vector to have width/2 `NA` values at either end.
- `align = 1` `[------*]` 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

No `na.rm` argument is provided as interpretation of the results is not at all clear.

## Examples

```
library(MazamaRollUtils)

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

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

# Index