Package 'seismicRoll'

December 4, 2018

Type Package

Version 1.1.3

Title Fast Rolling Functions for Seismology using 'Rcpp'

Author Jonathan Callahan [aut],
Rob Casey [aut],
Mary Templeton [aut],
Gillian Sharer [aut, cre]

Maintainer Gillian Sharer <gillian@iris.washington.edu>

Depends R (>= 3.0.0)

Suggests

License GPL (>= 2)

Imports Rcpp (>= 0.12.17)

LinkingTo Rcpp

Description Fast versions of seismic analysis functions that 'roll' over a vector of values. See the 'RcppRoll' package for alternative versions of basic statistical functions such as rolling mean, median, etc.

Collate seismicRoll.R RcppExports.R findOutliers.R roll_hampel.R roll_mean.R roll_median.R roll_sd.R roll_stalta.R roll_range.R

Repository CRAN

Encoding UTF-8

RoxygenNote 6.1.1

NeedsCompilation yes

Date/Publication 2018-12-04 20:00:03 UTC

R topics documented:

`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, n = 41, thresholdMin = 10, selectivity = NA,
  increment = 1, fixedThreshold = TRUE)
```

Arguments

`x`	an R numeric vector
`n`	integer window size
`thresholdMin`	initial value for outlier detection
`selectivity`	value between [0-1] used in determining outliers, or `NA` if `fixedThreshold=TRUE`.
`increment`	integer shift to use when sliding the window to the next location
`fixedThreshold`	logical specifying whether outlier detection uses `selectivity` (see below)

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`.

`n`, the windowSize, is a parameter that is passed to `roll_hampel()`.

The default value of `increment=1` should not be changed. Outliers are defined as individual points that stand apart from their neighbors. Applying the Hampel filter to every other point by using `increment` > 1 will invariably miss some of the outliers.

Value

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

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)
findOutliers(a)
```

`roll_hampel` *Rolling Hampel Filter for Outlier Detection*

Description

Fast, center-aligned hampel filter using C++/Rcpp. The Hampel filter is a robust outlier detector using Median Absolute Deviation (MAD). Additional performance gains can be achieved by skipping `increment` values between calculations.

Usage

```
roll_hampel(x, n, increment = 1)
```

Arguments

`x`	an R numeric vector
`n`	integer window size
`increment`	integer shift to use when sliding the window to the next location

Details

Unlike the version in the **pracma** package, this version does not return the corrected timeseries. Instead, it returns a vector of values that can be tested against different threshold values. Higher values in the return are associated with a higher likelihood that the associated point is an outlier when compared with its neighbors. Outliers can be picked out by comparing the return values against some threshold as seen in the example.

Also *unlike* the **pracma** version, n is interpreted as the full window length and will be increased by one if necessary to have a window representing an odd number of indices.

Value

A vector of values of the same length as x.

Note

A pure R version of the filter is found in the **pracma** package.

See Also

roll_median

Examples

```
a <- sin(0.1*seq(100))
a[20] <- 50

b <- roll_hampel(a,10)
threshold <- 6
which(b > threshold)

## Not run:
  require(microbenchmark)
  require(pracma)

  microbenchmark(hampel(a,10), roll_hampel(a,10), times=10)

  #  Unit: microseconds
  #                 expr      min       lq    median       uq       max neval
  #        hampel(a, 10) 7610.688 7784.374 8037.4035 9453.928 16176.535    10
  #   roll_hampel(a, 10)   36.530   37.443   58.7165   65.418    90.403    10

## End(Not run)
```

roll_mean *Rolling Mean with Alignment*

Description

Fast rolling means with aligment using C++/Rcpp. Additional performance gains can be achieved by skipping `increment` values between calculations.

Usage

```
roll_mean(x, n = 7, increment = 1, align = "center")
```

Arguments

`x`	an R numeric vector
`n`	integer window size
`increment`	integer shift to use when sliding the window to the next location
`align`	window alignment, one of `"left"|"center"|"right"`

Details

The window size `n` is interpreted as the full window length.

Setting `increment` to a value greater than one will result in `NA`s for all skipped-over indices.

The `align` parameter determines the alignment of the current index within the window. Thus:

- `align="left" [*------]` will cause the returned vector to have n-1 `NA` values at the right end.
- `align="center" [---*---]` will cause the returned vector to have (n-1)/2 `NA` values at either end.
- `align="right" [------*]` will cause the returned vector to have n-1 `NA` values at the left end.

Value

A vector of rolling mean values of the same length as `x`.

Note

For `align="center"`, the window size is increased by one if necessary to guarantee an odd window size.

Examples

```
x <- rep(1:5,each=10)
plot(x,cex=0.8,pch=17,main="Test of roll_mean alignment with a 5-point window")
points(roll_mean(x,5,1,'left'),cex=1.5,col='blue',type='b')
points(roll_mean(x,5,1,'center'),cex=1.5,col='forestgreen',type='b')
points(roll_mean(x,5,1,'right'),cex=1.5,col='red',type='b')
legend("topleft",
       legend=c('data','align=left','align=center','align=right'),
       col=c('black','blue','forestgreen','red'),
       pch=c(17,1,1,1))
```

---

roll_median *Rolling Median*

Description

Fast, center-aligned rolling medians using C++/Rcpp. Additional performance gains can be achieved by skipping `increment` values between calculations.

The `roll_median` function can be used to replace outliers detected by the `roll_hampel` function. See example below.

Usage

```
roll_median(x, n = 7, increment = 1)
```

Arguments

`x`	an R numeric vector
`n`	integer window size
`increment`	integer shift to use when sliding the window to the next location

Details

The window size `n` is interpreted as the full window length. Values within `n/2` of the beginning or end of `x` are set to `NA`.

Setting `increment` to a value greater than one will result in `NA`s for all skipped-over indices.

Value

A vector of rolling median values of the same length as `x`.

See Also

`roll_hampel`

Examples

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

# Outlier detection
b <- roll_hampel(a,10)
threshold <- 6
outliers <- which(b > threshold)

# Outlier replacement with median values
a_fixed <- a
a_fixed[outliers] <- roll_median(a,10)[outliers]

# Set up two plots
layout(matrix(seq(2)))
plot(a,type='l', col='gray60', main="Outliers detected")
points(outliers,a[outliers], col='red', lwd=2)

plot(a_fixed,type='l', col='gray60',
     main="Outliers replaced with rolling median")
points(outliers,a_fixed[outliers], col='red', lwd=2)
```

---

roll_range *Rolling Range with Alignment*

Description

Fast rolling range with aligment using C++/Rcpp. Additional performance gains can be achieved by skipping `increment` values between calculations.

Usage

```
roll_range(x, n = 7, increment = 1, align = "center")
```

Arguments

`x`	an R numeric vector
`n`	integer window size
`increment`	integer shift to use when sliding the window to the next location
`align`	window alignment, one of `"left"|"center"|"right"`

Details

The window size `n` is interpreted as the full window length.

Setting `increment` to a value greater than one will result in `NA`s for all skipped-over indices.

The `align` parameter determines the alignment of the current index within the window. Thus:

- `align="left" [*------]` will cause the returned vector to have n-1 `NA` values at the right end.
- `align="center" [---*---]` will cause the returned vector to have (n-1)/2 `NA` values at either end.
- `align="right" [------*]` will cause the returned vector to have n-1 `NA` values at the left end.

Value

A vector of rolling values that difference the maximum and minimum values, of the same length as `x`.

Note

For `align="center"`, the window size is increased by one if necessary to guarantee an odd window size.

`roll_sd` *Rolling Standard Deviation with Alignment*

Description

Fast rolling standard deviations with aligment using C++/Rcpp. Additional performance gains can be achieved by skipping `increment` values between calculations.

Usage

```
roll_sd(x, n = 7, increment = 1, align = "center")
```

Arguments

`x`	an R numeric vector
`n`	integer window size
`increment`	integer shift to use when sliding the window to the next location
`align`	window alignment, one of `"left"|"center"|"right"`

Details

The window size `n` is interpreted as the full window length.

Setting `increment` to a value greater than one will result in `NA`s for all skipped-over indices.

The `align` parameter determines the alignment of the current index within the window. Thus:

- `align="left" [*------]` will cause the returned vector to have n-1 `NA` values at the right end.
- `align="center" [---*---]` will cause the returned vector to have (n-1)/2 `NA` values at either end.
- `align="right" [------*]` will cause the returned vector to have n-1 `NA` values at the left end.

Value

A vector of rolling standard deviation values of the same length as `x`.

Note

For `align="center"`, the window size is increased by one if necessary to guarantee an odd window size.

`roll_stalta` *Rolling STA/LTA*

Description

Fast rolling STA/LTA using C++/Rcpp. Additional performance gains can be achieved by skipping `increment` values between calculations.

The STA/LTA ratio method is used for automatic detection of seismic signal arrival times.

Usage

```
roll_stalta(x, n_sta, n_lta, increment = 1)
```

Arguments

`x`	an R numeric vector
`n_sta`	integer STA window size
`n_lta`	integer LTA window size
`increment`	integer shift to use when sliding the window to the next location

Details

The `roll_stalta` function described here does no preprocessing of the incoming data and merely calculates the ratio of the average value in the STA window to the average value in the LTA window. Windows are aligned so that the index is at the left edge of the STA window and at the right edge of the LTA window.

$$STA(x_i) = \frac{1}{ns} \sum_{j=i}^{i+ns} x_i$$

$$LTA(x_i) = \frac{1}{nl} \sum_{j=i-nl}^{i} x_i$$

$$r_i = \frac{STA_i}{LTA_i}$$

```
[---------- LTA --------*].......
.....................[*- STA --]
```

For proper use of this algorithm seismic data should be preprocessed as in the example below with:

- demean, detrend and taper the raw signal
- square the processed signal to get power

With `increment=1`, this function is equivalent to, eg:

```
sta <- roll_mean(x,3,1,"left")
lta <- roll_mean(x,30,1,"right")
r <- sta/lta
```

For increments greater than one, the rolling means above will not align properly, hence the need for a dedicated `roll_stalta` function.

Values within `n_lta-1` of the beginning or `n_sta-1` of the end of `x` are set to `NA`.

Setting `increment` to a value greater than one will result in `NA`s for all skipped-over indices.

Value

A vector of values of the same length as `x` with each point containing the STA/LTA ratio at that point.

References

First Break Picking

Examples

```
# Contrived example
x <- rep(c(1,5,3,2,1),each=20)
p <- roll_stalta(x,3,6)
plot(x, pch=17, cex=0.8, ylim=c(0,max(x)),
     main="Test of roll_stalta on artificial data")
points(p,cex=1.5,col='red',type='b')
legend('topleft',
       legend=c('data','STA/LTA'),
       pch=c(17,1),
       col=c('black','red'))

# Real example requiring the 'seismic' package
## Not run:
require(seismic)

# Create a new IrisClient
iris <- new("IrisClient")

# Seismic data with a large quake
starttime <- as.POSIXct("2010-02-27 06:30:00", tz="GMT")
endtime <- as.POSIXct("2010-02-27 07:00:00", tz="GMT")
st <- getDataselect(iris,"IU","ANMO","00","BHZ",starttime,endtime)
```

```
tr <- st@traces[[1]]

# Preprocess the data
x <- DDT(tr)@data

# Calculate the first break 'picker'
n_sta <- 3 * tr@stats@sampling_rate
n_lta <- 10 * n_sta
p <- roll_stalta(x^2,n_sta,n_lta)

first_break <- which(p == max(p,na.rm=TRUE))

plot(x,type='l',
     main='Test of STA/LTA first break picker on raw seismic data')
abline(v=first_break,col='red')

## End(Not run)
```

`seismicRoll` *Fast Rolling Statistics for Seismology*

Description

This package implements fast versions of 'roll'-ing functions primarily for use in seismology. It is intended for use with the **seismic** and **seismicMetrics** packages being developed for the IRIS Data Management Center (DMC) (http://www.iris.edu/dms/nodes/dmc/). One advantage of the **seismicRoll** package is that all returned values are of the same dimension as the incoming data with NAs where the rolling function could not be calculated.

Details

Currently exported functions include:

- `findOutliers` – outlier detection wrapper
- `roll_hampel` – outlier detection
- `roll_mean` – rolling mean
- `roll_median` – rolling median (for outlier replacement)
- `roll_sd` – rolling standard deviation
- `roll_stalta` – first break picker
- `roll_range` – rolling difference of max/min values

History

version 1.1.3

- minor changes to how the code compiles
- add roll_range function

version 1.1.2 – bug fix

- `findOutliers()` exits if the `roll_hampel()` return vector consists entirely of `NA` values.

version 1.1.0 – findOutliers update and bug fix

- `findOutliers()` function default argument values changed. Now `thresholdMin=10`, `selectivity=NA`, and `fixedThreshold=TRUE`.
- Bug fix in `roll_hampel()` handling `NA` values.

version 1.0.3 – findOutliers update

- Added `fixedThreshold` argument to `findOutliers()` function.

version 1.0.0 – initial release

Index