

# Package ‘convo’

June 20, 2018

**Type** Package

**Title** Fast Computation of Running Sample Statistics

**Version** 0.1.0

**Description** Provides methods for fast computation of running (aka rolling) sample statistics. These include running sample mean of a single numeric sequence, running sample variance of a single numeric sequence, running sample covariance of two numeric sequences. Implementation of the methods uses convolution via Fast Fourier Transform. The complexity of the convolution can be reduced from  $O(n^2)$  to  $O(n \log n)$  with fast Fourier transform compared to conventional computation.

**License** GPL-3

**Imports** stats

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.0.1

**Suggests** knitr, rmarkdown, testthat, covr

**VignetteBuilder** knitr

**Author** Marta Karas [aut, cre],  
Jacek Urbanek [aut]

**Maintainer** Marta Karas <mkaras2@jhu.edu>

## R topics documented:

DigFilter . . . . .	2
RunningCor . . . . .	2
RunningCov . . . . .	4
RunningL2Norm . . . . .	5
RunningMean . . . . .	6
RunningSd . . . . .	7
RunningVar . . . . .	8
<b>Index</b>	<b>9</b>

---

DigFilter

*Bandpass Digital Filter*


---

### Description

Bandpass digital filter; passes frequencies within a certain range and rejects frequencies outside that range.

### Usage

```
DigFilter(x, fs, LD, LU)
```

### Arguments

x	numeric sequence
fs	sampling frequency of a numeric sequence x
LD	lower bandpass frequency
LU	higher bandpass frequency

### Examples

```
## generate components of a signal x
fs <- 1000
t.seq <- seq(0, 1, length.out = 1000)
x1 <- sin(2 * pi * t.seq * 50)
x2 <- sin(2 * pi * t.seq * 150)
x3 <- sin(2 * pi * t.seq * 250)
x0 <- rnorm(length(t.seq), sd = 0.1)
plot(x0, ylim = range(c(x0, x1, x2, x3)), main = "x components")
lines(x1, col = "blue")
lines(x2, col = "red")
lines(x3, col = "green")
## generate signal x
x <- x1 + x2 + x3 + x0
plot(x, main = "x", type = "l")
## use filter with passband frequencies [100 Hz, 200 Hz]
x.filtered <- DigFilter(x, 1000, 100, 200)
plot(x.filtered, type = "l", main = "x filtered with [100 Hz, 200 Hz] frequency bandpass")
plot(x2, type = "l", main = "compare: x2 component of x")
```

---

RunningCor

*Fast Running Correlation Computation*


---

### Description

Computes running correlation between two sequences in a fixed width window, whose length corresponds to the length of the shorter sequence. Uses convolution via Fast Fourier Transform.

**Usage**

```
RunningCor(x, y, circular = FALSE)
```

**Arguments**

<code>x</code>	numeric sequence
<code>y</code>	numeric sequence, of equal or shorter length than <code>x</code> sequence
<code>circular</code>	logical; whether running correlation is computed assuming circular nature of <code>x</code> sequence (see Details)

**Details**

Computes running correlation between two sequences in a fixed width window. The length of a window is equal to the shorter of the two sequences (`y`), and window "runs" over the length of longer sequence (`x`).

Parameter `circular` determines whether `x` sequence is assumed to have a circular nature. Assume  $l_x$  is the length of sequence `x`,  $l_y$  is the length of shorter sequence `y`.

If `circular` equals `TRUE` then

- output sequence length equals  $l_x$ ,
- first element of the output sequence corresponds to sample correlation between `x[1:l_y]` and `y`,
- last element of the output sequence corresponds to sample correlation between `c(x[l_x], x[1:(l_y - 1)])` and `y`.

If `circular` equals `FALSE` then

- output sequence length equals  $l_x - l_y + 1$ ,
- first element of the output sequence corresponds to sample correlation between `x[1:l_y]` and `y`,
- last element of the output sequence corresponds to sample correlation between `x[(l_x - l_y + 1):l_x]`.

**Value**

numeric sequence

**Examples**

```
x <- sin(seq(0, 1, length.out = 1000) * 2 * pi * 6)
y <- x[1:100]
out1 <- RunningCor(x, y, circular = TRUE)
out2 <- RunningCor(x, y, circular = FALSE)
plot(out1, type = "l"); points(out2, col = "red")
```

RunningCov

*Fast Running Covariance Computation***Description**

Computes running covariance between two sequences in a fixed width window, whose length corresponds to the length of the shorter sequence. Uses convolution via Fast Fourier Transform.

**Usage**

```
RunningCov(x, y, circular = FALSE)
```

**Arguments**

x	numeric sequence
y	numeric sequence, of equal or shorter length than x sequence
circular	logical; whether running variance is computed assuming circular nature of x sequence (see Details)

**Details**

Computes running covariance between two sequences in a fixed width window. The length of a window is equal to the shorter of the two sequences (y), and window "runs" over the length of longer sequence (x).

Parameter circular determines whether x sequence is assumed to have a circular nature. Assume  $l_x$  is the length of sequence x,  $l_y$  is the length of shorter sequence y.

If circular equals TRUE then

- output sequence length equals  $l_x$ ,
- first element of the output sequence corresponds to sample covariance between  $x[1:l_y]$  and y,
- last element of the output sequence corresponds to sample covariance between  $c(x[l_x], x[1:(l_y - 1)])$  and y.

If circular equals FALSE then

- output sequence length equals  $l_x - l_y + 1$ ,
- first element of the output sequence corresponds to sample covariance between  $x[1:l_y]$  and y,
- last element of the output sequence corresponds to sample covariance between  $x[(l_x - l_y + 1):l_x]$ .

**Value**

numeric sequence

**Examples**

```
x <- sin(seq(0, 1, length.out = 1000) * 2 * pi * 6)
y <- x[1:100]
out1 <- RunningCov(x, y, circular = TRUE)
out2 <- RunningCov(x, y, circular = FALSE)
plot(out1, type = "l"); points(out2, col = "red")
```

RunningL2Norm

*Fast Running L2 Norm Computation***Description**

Computes running L2 norm between two sequences in a fixed width window, whose length corresponds to the length of the shorter sequence. Uses convolution via Fast Fourier Transform.

**Usage**

```
RunningL2Norm(x, y, circular = FALSE)
```

**Arguments**

x	numeric sequence
y	numeric sequence, of equal or shorter length than x sequence
circular	logical; whether running L2 norm is computed assuming circular nature of x sequence (see Details)

**Details**

Computes running L2 norm between two sequences in a fixed width window. The length of a window is equal to the shorter of the two sequences (y), and window "runs" over the length of longer sequence (x).

Parameter circular determines whether x sequence is assumed to have a circular nature. Assume  $l_x$  is the length of sequence x,  $l_y$  is the length of shorter sequence y.

If circular equals TRUE then

- output sequence length equals  $l_x$ ,
- first element of the output sequence corresponds to sample L2 norm between  $x[1:l_y]$  and y,
- last element of the output sequence corresponds to sample L2 norm between  $c(x[l_x], x[1:(l_y - 1)])$  and y.

If circular equals FALSE then

- output sequence length equals  $l_x - l_y + 1$ ,
- first element of the output sequence corresponds to sample L2 norm between  $x[1:l_y]$  and y,
- last element of the output sequence corresponds to sample L2 norm between  $x[(l_x - l_y + 1):l_x]$ .

**Value**

numeric sequence

**Examples**

```
x <- sin(seq(0, 1, length.out = 1000) * 2 * pi * 6)
y1 <- x[1:100] + rnorm(100)
y2 <- rnorm(100)
out1 <- RunningL2Norm(x, y1)
out2 <- RunningL2Norm(x, y2)
plot(out1, type = "l"); points(out2, col = "blue")
```

---

RunningMean	<i>Fast Running Mean Computation</i>
-------------	--------------------------------------

---

**Description**

Computes running sample mean of a sequence in a fixed width window. Uses convolution via Fast Fourier Transform.

**Usage**

```
RunningMean(x, W, circular = FALSE)
```

**Arguments**

x	numeric sequence
W	numeric; width of x sequence window
circular	logical; whether running sample mean is computed assuming circular nature of x sequence (see Details)

**Details**

Parameter circular determines whether x sequence is assumed to have a circular nature. Assume  $l_x$  is the length of sequence x, W is a fixed length of x sequence window.

If circular equals TRUE then

- output sequence length equals  $l_x$ ,
- first element of the output sequence corresponds to sample mean of  $x[1:W]$ ,
- last element of the output sequence corresponds to sample mean of  $c(x[l_x], x[1:(W - 1)])$ .

If circular equals FALSE then

- output sequence length equals  $l_x - W + 1$ ,
- first element of the output sequence corresponds to sample mean of  $x[1:W]$ ,
- last element of the output sequence corresponds to sample mean of  $x[(l_x - W + 1):l_x]$ .

**Value**

numeric sequence

**Examples**

```
x <- rnorm(1000)
RunningMean(x, 100)
length(RunningMean(x, 100, circular = FALSE))
length(RunningMean(x, 100, circular = TRUE))
```

RunningSd

*Fast Running Standard Deviation Computation***Description**

Computes running sample standard deviation of a sequence in a fixed width window. Uses convolution via Fast Fourier Transform.

**Usage**

```
RunningSd(x, W, circular = FALSE)
```

**Arguments**

x	numeric sequence
W	numeric; width of x sequence window
circular	logical; whether running sample standard deviation is computed assuming circular nature of x sequence (see Details)

**Details**

Parameter circular determines whether x sequence is assumed to have a circular nature. Assume  $l_x$  is the length of sequence x, W is a fixed length of x sequence window.

If circular equals TRUE then

- output sequence length equals  $l_x$ ,
- first element of the output sequence corresponds to sample standard deviation of  $x[1:W]$ ,
- last element of the output sequence corresponds to sample standard deviation of  $c(x[1_x], x[1:(W - 1)])$ .

If circular equals FALSE then

- output sequence length equals  $l_x - W + 1$ ,
- first element of the output sequence corresponds to sample standard deviation of  $x[1:W]$ ,
- last element of the output sequence corresponds to sample standard deviation of  $x[(1_x - W + 1):1_x]$ .

**Value**

numeric sequence

**Examples**

```
x <- rnorm(1000)
RunningSd(x, 100)
length(RunningSd(x, 100, circular = FALSE))
length(RunningSd(x, 100, circular = TRUE))
```

RunningVar

*Fast Running Variance Computation***Description**

Computes running sample variance of a sequence in a fixed width window. Uses convolution via Fast Fourier Transform.

**Usage**

```
RunningVar(x, W, circular = FALSE)
```

**Arguments**

x	numeric sequence
W	numeric; width of x sequence window
circular	logical; whether running sample variance is computed assuming circular nature of x sequence (see Details)

**Details**

Parameter circular determines whether x sequence is assumed to have a circular nature. Assume  $l_x$  is the length of sequence x, W is a fixed length of x sequence window.

If circular equals TRUE then

- output sequence length equals  $l_x$ ,
- first element of the output sequence corresponds to sample variance of  $x[1:W]$ ,
- last element of the output sequence corresponds to sample variance of  $c(x[1_x], x[1:(W - 1)])$ .

If circular equals FALSE then

- output sequence length equals  $l_x - W + 1$ ,
- first element of the output sequence corresponds to sample variance of  $x[1:W]$ ,
- last element of the output sequence corresponds to sample variance of  $x[(1_x - W + 1):1_x]$ .

**Value**

numeric sequence

**Examples**

```
x <- rnorm(1000)
RunningVar(x, 100)
length(RunningVar(x, 100, circular = FALSE))
length(RunningVar(x, 100, circular = TRUE))
```



# Index

DigFilter, [2](#)

RunningCor, [2](#)

RunningCov, [4](#)

RunningL2Norm, [5](#)

RunningMean, [6](#)

RunningSd, [7](#)

RunningVar, [8](#)