Package 'RcppMovStat'

January 15, 2018

Type Package

Title Fast Moing Statitics Caculation via Rcpp

Version 1.0

Date 2018-01-06

Author Pengfei (Leonard) Li

Maintainer Pengfei (Leonard) Li <pli3@tulane.com>

Description This is a package providing several efficient functions to
calculate common moving(or rolling, running) statitics for both EVENLY and UN-
EVENLY SPACED Time
Series: moving average, moving median, and moving maximum(minimum) . Built on C++, these functions would be apparently more efficient than those written in traditional R and also faster than Others written by Rcpp.

Suggests zoo, caTools, Rcpp

License GPL (>= 2)

Imports Rcpp (>= 0.12.14)

LinkingTo Rcpp

RoxygenNote 6.0.1

R topics documented:

RcppMovStat-package *Fast Moing Statitics Caculation via Rcpp*

Description

This is a package providing several efficient functions to calculate common moving(or rolling, running) statitics: moving average, moving median, and moving maximum(minimum) . Built on C++, these functions are apparently more efficient than those written in traditional R.

Details

See the following instructions for further details.

Author(s)

Pengfei (Leonard) Li, pli3@tulane.edu.

Maintainer: Pengfei (Leonard) Li <pli3@tulane.edu>

movCount *Moving Count*

Description

This function returns a moving count of the given vector.

Usage

```
movCount(vec, n = 1L, ss = 1L, na_rm = FALSE, sizeD = FALSE,
  align = "left")

movCountr(vec, n = 1L, ss = 1L, na_rm = FALSE, sizeD = FALSE)
```

Arguments

vec	A numeric vector.
n	An integer: moving window size, with 1 as default
ss	An integer: step size, only calculating at points with an equal distance *ss*. Namely, there are *ss-1* number between each two 'consecutive' points
na_rm	logical. Should missing values (including NaN) be removed?
sizeD	logical. Only applied when *ss* > *1*, it decides whether to get a result of smaller size. If $sizeD = T$, *align* does not affect the output.
align	A string denotes how to align the moving average, three options: "left", "middle", "right"

Details

This function counts the number of non-missing values for each moving window. It is especially designed for *vec* vector with missing values. Otherwise, it will return a trival vector with all elements *n*.

Value

This function returns a vector whose length is the same as that of *vec* or is `ceiling`$((L-n+1)/ss)$, (when $sizeD = T$), where L is the length of vec.

Functions

- `movCountr`: An function equivalent to `movCount(..., align = "right")`

Examples

```
movCount(c(1, 4, 3, NA, 8), 2, na_rm = TRUE)
movCount(c(1, 4, 3, NA, 8), 2, na_rm = TRUE, align = 'right')
movCountr(c(1, 4, 3, NA, 8), 2, na_rm = TRUE)
movCount(c(1, 4, 3, NA, NA), 2, na_rm = TRUE)
```

`movCountUE` *Weighted Simple Moving Count for Unevenly Spaced Time Series*

Description

This function returns A matrix: the first column is the position, the second column the input vectorthe, and third column Moving Count of the given vector. The weight argument is optional.

Usage

```
movCountUE(vec, pos, n = 1L, ss = 1L, na_rm = FALSE, sizeD = FALSE,
  align = "left")

movCountUEr(vec, pos, n = 1L, ss = 1L, na_rm = FALSE, sizeD = FALSE)
```

Arguments

`vec`	A numeric vector.
`pos`	A numeric vector with all integers. Its length must be the SAME as vec. N.B. We use integers to represent the (relative) postions of every point. The first element MUST BE 1, which is design to follow THE 1-INDEXED RULE of R.
`n`	An integer: moving window size, with 1 as default
`ss`	An integer: step size, only calculating at points with an equal distance *ss*. Namely, there are *ss-1* number between each two 'consecutive' points
`na_rm`	logical. Should missing values (including NaN) be removed?
`sizeD`	logical. Only applied when *ss* > *1*, it decides whether to get a result of smaller size. If $sizeD = T$, *align* does not affect the output.
`align`	A string denotes how to align the moving average, three options: "left", "middle", "right"

Details

This function counts the number of non-missing values for each moving window. It is especially designed for *vec* vector with missing values. Otherwise, it will return a trival vector with all elements *n*.
This function is more helpful than `movCount`, as we would have missing values for an Unevenly Spaced Time Series.
For matrix details, please refer to details of `movMeanUE`.

Value

This function returns A MATRIX of size: $L * 3$, where L is the length of vector, or of size: $L1 * 3$, where $L1 = \texttt{ceiling}((nrow - n + 1)/ss)$, (when $sizeD = T$). In the matrix, the first column denotes the position, the second column the original vector, and the third column the moving average.

Functions

- `movCountUEr`: An function equivalent to `movCountUE(..., align = "right")`

Examples

```
movCountUEr(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), 2)
movCountUEr(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2, na_rm = TRUE)
movCountUEr(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2, ss = 3, na_rm = TRUE,
sizeD = TRUE)
movCountUE(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2,
ss = 3, na_rm = TRUE, align = "right")
movCountUE(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2, ss = 3,
na_rm = TRUE, sizeD = TRUE, align = "right")
movCountUEr(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9),  n = 2, ss = 3,
na_rm = TRUE)
movCountUEr(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2, ss = 3, na_rm = TRUE,
sizeD = TRUE)
```

`movEmean` *Basic Exponential Moving Mean*

Description

This function returns a basic exponential moving average (EMA) of the given vector.

Usage

```
movEmean(vec, n = 1L, smFac = NULL)
```

Arguments

`vec`	A numeric vector.
`n`	An integer: moving window size, with 1 as default
`smFac`	A number: smoothing factor, with default $2/(n + 1)$, see **details** below.

Details

This function makes fairly efficient the computaion of EMA, which dubbed as basic exponential smoothing, the same section of https://en.wikipedia.org/wiki/Exponential_smoothing. It provides an access to define *smFac* yourself, i.e the smoothing factor, whose default is $2/(n+1)$.

Value

This function returns a vector whose length is the same as that of *vec*.

Examples

```
movEmean(c(1, 4, 3, 6, 8), 2, smFac = 1/3)
movEmean(c(1, 4, 3, 6, 8), 2)
```

---

movMean *Weighted Simple Moving Mean*

Description

This function returns a simple moving average of the given vector. The weight argument is optional.

Usage

```
movMean(vec, n = 1L, ss = 1L, w = NULL, na_rm = FALSE, sizeD = FALSE,
  align = "left")

movMeanr(vec, n = 1L, ss = 1L, w = NULL, na_rm = FALSE, sizeD = FALSE)
```

Arguments

vec	A numeric vector.
n	An integer: moving window size, with 1 as default
ss	An integer: step size, only calculating at points with an equal distance *ss*. Namely, there are *ss-1* number between each two 'consecutive' points
w	An optional weight vector of length *n*. It will be automatically normalized (sum to 1).
na_rm	logical. Should missing values (including NaN) be removed?
sizeD	logical. Only applied when *ss* > *1*, it decides whether to get a result of smaller size. If $sizeD = T$, *align* does not affect the output.
align	A string denotes how to align the moving average, three options: "left", "middle", "right"

Details

Despite of Efficient computation, usually 5~6 times faster than the moving average function in package roll_mean, it is able to handle potential missing values (*NA* or *NaN*) in the *vec*.
For instace, the output of the second examle is $NA, NA, 2.2000003.7142864.875000$. The last number 5.5 is obtained by using renormalized weight, namely omitting 0.2. The weight applied would be $0.5/(0.5+0.3)$ and $0.3/(0.5+0.3)$. Hence,

$$4.875 = 3 * 0.5/(0.5+0.3) + 8 * 0.3/(0.5+0.3)$$

Value

This function returns a vector whose length is the same as that of *vec* or is `ceiling`$((L-n+1)/ss)$, (when $sizeD = T$), where L is the length of vec.

Functions

- `movMeanr`: An function equivalent to `movMean(..., align = "right")`

Examples

```
movMean(c(1, 4, 3, NA, 8), 3, align = "right", na_rm = TRUE)
movMean(c(1, 4, 3, NA, 8), 3, w = c(0.5, 0.2, 0.3), na_rm = TRUE, align = "right")
movMean(c(1, 4, 3, NA, 8, 4, 5, 9, 6, 0), n = 3, ss = 4, na_rm = TRUE, align = "right")
```

`movMeanUE` *Weighted Simple Moving Mean for Unevenly Spaced Time Series*

Description

This function returns A matrix: the first column is the position, the second column the input vectorthe, and third column simple moving average of the given vector. The weight argument is optional.

Usage

```
movMeanUE(vec, pos, n = 1L, ss = 1L, w = NULL, na_rm = FALSE,
  sizeD = FALSE, align = "left")

movMeanUEr(vec, pos, n = 1L, ss = 1L, w = NULL, na_rm = FALSE,
  sizeD = FALSE)
```

Arguments

`vec`	A numeric vector.
`pos`	A numeric vector with all integers. Its length must be the SAME as vec. N.B. We use integers to represent the (relative) postions of every point. The first element MUST BE 1, which is design to follow THE 1-INDEXED RULE of R.
`n`	An integer: moving window size, with 1 as default
`ss`	An integer: step size, only calculating at points with an equal distance *ss*. Namely, there are *ss-1* number between each two 'consecutive' points
`w`	An optional weight vector of length *n*. It will be automatically normalized (sum to 1).
`na_rm`	logical. Should missing values (including NaN) be removed?
`sizeD`	logical. Only applied when *ss* > *1*, it decides whether to get a result of smaller size. If $sizeD = T$, *align* does not affect the output.
`align`	A string denotes how to align the moving average, three options: "left", "middle", "right"

Details

This function is especially designed for Unevenly Spaced Time Series. It is efficient as it herits the similiar routine of `movMean`.
The result is kind of tricky. To make it clear, it is written to return a MATRIX. For instance, the third column of the output of second example is $NA, 2.5, 4.0, NA, NA, NA, 3.0, 3.0, 8.0$. 2.5 is the average of 1 and 4, and 4.0 the average of 4. The third column of the output of third example is the every third element starting from nth number.
For how weights, w, work, one can refer to `movMean`.

Value

This function returns A MATRIX of size: $L * 3$, where L is the length of vector, or of size: $L1 * 3$, where $L1 =$ `ceiling`$((nrow - n + 1)/ss)$, (when $sizeD = T$). In the matrix, the first column denotes the position, the second column the original vector, and the third column the moving average.

Functions

- `movMeanUEr`: An function equivalent to `movMeanUE(..., align = "right")`

Examples

```
movMeanUEr(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), 2)
movMeanUEr(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2, na_rm = TRUE)
movMeanUEr(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2, ss = 3, na_rm = TRUE,
sizeD = TRUE)
movMeanUE(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), w = c(0, 1),  n = 2,
ss = 3, na_rm = TRUE, align = "right")
movMeanUE(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2, ss = 3,
na_rm = TRUE, sizeD = TRUE, align = "right")
movMeanUEr(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9),  n = 2, ss = 3,
na_rm = TRUE)
movMeanUEr(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2, ss = 3, na_rm = TRUE,
sizeD = TRUE)
```

---

movQt — *Moving Quantile(Moving Median, Moving Minimum, Moving Maximum)*

Description

This function returns a moving quantile of the given vector.

Usage

```
movQt(vec, n = 1L, prob = 0.5, ss = 1L, na_rm = FALSE, sizeD = FALSE,
  align = "left")

movQtr(vec, n = 1L, prob = 0.5, ss = 1L, na_rm = FALSE, sizeD = FALSE)
```

Arguments

vec	A numeric vector.
n	An integer: moving window size, with 1 as default
prob	A number: between *0* and *1*, meaning *prob* quantile
ss	An integer: step size, only calculating at points with an equal distance *ss*. Namely, there are *ss-1* number between each two 'consecutive' points
na_rm	logical. Should missing values (including NaN) be removed?
sizeD	logical. Only applied when *ss* > *1*, it decides whether to get a result of smaller size. If $sizeD = T$, *align* does not affect the output.
align	A string denotes how to align the moving average, three options: "left", "middle", "right"

Details

Despite of Efficient computation, this function can return differents kinds of moving quantile, e.g. moving median($prob = 0.5$), moving minimun($prob = 0$), and moving maximum($prob = 1$). It can handle potential missing values(*NA* or *NaN*) in the *vec*. When we move to one specific fragment, missing values can be removed by setting $na_rm = TRUERUE$. If all values of this fragment is missing, it will return *NA*.
In terms of the quantile algorithm, please consult type *7* in function `quantile`.

Value

This function returns a vector whose length is the same as that of *vec* or is `ceiling`$((L-n+1)/ss)$, (when $sizeD = T$), where L is the length of vec.

Functions

- `movQtr`: An function equivalent to `movQt(..., align = "right")`

Examples

```
movQt(vec = c(1, 4, 3, NA, 8, 4, 5, 9, 6, 0), n = 3, ss = 4, na_rm = TRUE, align = "right")
movQt(vec = c(1, 4, 3, NA, 8, 4, 5, 9, 6, 0), n = 3, na_rm = TRUE, align = "right")
movQt(vec = c(1, 4, 3, NA, NA, NA, 5, 9, 6, 0), n = 3, ss = 4, na_rm = TRUE, align = "middle")
```

---

movQtUE *Moving quantile_UE(Moving Median, Moving Minimum, Moving Maximum) for Unevenly Spaced Time Series*

Description

This function returns A matrix: the first column is the position, the second column the input vectorthe, and third column moving quantile_UE of the given vector.

Usage

```
movQtUE(vec, pos, n = 1L, prob = 0.5, ss = 1L, na_rm = FALSE,
  sizeD = FALSE, align = "left")

movQtUEr(vec, pos, n = 1L, prob = 0.5, ss = 1L, na_rm = FALSE,
  sizeD = FALSE)
```

Arguments

vec	A numeric vector.
pos	A numeric vector with all integers. Its length must be the SAME as vec. N.B. We use integers to represent the (relative) postions of every point. The first element MUST BE 1, which is design to follow THE 1-INDEXED RULE of R.
n	An integer: moving window size, with 1 as default
prob	A number: between *0* and *1*, meaning *prob* quantile_UE
ss	An integer: step size, only calculating at points with an equal distance *ss*. Namely, there are *ss-1* number between each two 'consecutive' points
na_rm	logical. Should missing values (including NaN) be removed?
sizeD	logical. Only applied when *ss* > *1*, it decides whether to get a result of smaller size. If $sizeD = T$, *align* does not affect the output.
align	A string denotes how to align the moving average, three options: "left", "middle", "right"

Details

This function is especially designed for Unevenly Spaced Time Series. It is efficient as it herits the similiar routine of `movQt`.
The result is kind of tricky. To make it clear, it is written to return a MATRIX. For instance, the third column of the output of second example is $2.5, NA, NA, NA, NA, NA, 3.0, NA, NA$. 2.5 is the median of 1 and 4, and 4.0 the average of 4. The third column of the output of third example is the every third element starting from nth number.
For how weights, w, work, one can refer to `movQt`.

Value

This function returns A MATRIX of size: $L * 3$, where L is the length of vector, or of size: $L1 * 3$, where $L1 = \texttt{ceiling}((nrow - n + 1)/ss)$, (when $sizeD = T$). In the matrix, the first column denotes the position, the second column the original vector, and the third column the moving average.

Functions

- `movQtUEr`: An function equivalent to `movQtUE(..., align = "right")`

Examples

```
movQtUE(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2, ss = 3, na_rm = TRUE)
movQtUE(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2, ss = 3, na_rm = TRUE, sizeD = TRUE)
movQtUE(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2, na_rm = TRUE, align = "middle")
movQtUE(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2, ss = 3, na_rm = TRUE, sizeD = TRUE,
 align = "right")
```

movSum *Weighted Simple Moving Sum*

Description

This function returns a simple moving sum of the given vector. The weight argument is optional.

Usage

```
movSum(vec, n = 1L, ss = 1L, w = NULL, na_rm = FALSE, sizeD = FALSE,
  align = "left")

movSumr(vec, n = 1L, ss = 1L, w = NULL, na_rm = FALSE, sizeD = FALSE)
```

Arguments

vec	A numeric vector.
n	An integer: moving window size, with 1 as default
ss	An integer: step size, only calculating at points with an equal distance *ss*. Namely, there are *ss-1* number between each two 'consecutive' points
w	An optional weight vector of length *n*.
na_rm	logical. Should missing values (including NaN) be removed?
sizeD	logical. Only applied when *ss* > *1*, it decides whether to get a result of smaller size. If $sizeD = T$, *align* does not affect the output.
align	A string denotes how to align the moving average, three options: "left", "middle", "right"

Details

This function can obtain the moving sum efficiently. It serves as somehow a generalized version of `movMean`. The difference is that it will not automatically normalized the weights vector, *w* argument. If there is no missing value in *vec*, and *w* is normalized, which means the sum of all elements is *1*, this function will return a moving average.

Value

This function returns a vector whose length is the same as that of *vec* or is `ceiling`$((L-n+1)/ss)$, (when $sizeD = T$), where L is the length of vec.

Functions

- `movSumr`: An function equivalent to `movMean(..., align = "right")`

Examples

```
movSum(c(1, 4, 3, NA, 8), 3, align = "right", na_rm = TRUE)
movSum(c(1, 4, 3, NA, 8), 3, w = c(0.5, 0.2, 0.3), na_rm = TRUE, align = "right")
movSum(c(1, 4, 3, NA, 8, 4, 5, 9, 6, 0), n = 3, ss = 4, na_rm = TRUE, align = "right")
```

movSumUE *Weighted Simple Moving Sum for Unevenly Spaced Time Series*

Description

This function returns A matrix: the first column is the position, the second column the input vectorthe, and third column moving sum of the given vector. The weight argument is optional.

Usage

```
movSumUE(vec, pos, n = 1L, ss = 1L, w = NULL, na_rm = FALSE,
  sizeD = FALSE, align = "left")

movSumUEr(vec, pos, n = 1L, ss = 1L, w = NULL, na_rm = FALSE,
  sizeD = FALSE)
```

Arguments

vec	A numeric vector.
pos	A numeric vector with all integers. Its length must be the SAME as vec. N.B. We use integers to represent the (relative) postions of every point. The first element MUST BE 1, which is design to follow THE 1-INDEXED RULE of R.
n	An integer: moving window size, with 1 as default
ss	An integer: step size, only calculating at points with an equal distance *ss*. Namely, there are *ss-1* number between each two 'consecutive' points
w	An optional weight vector of length *n*. It will be automatically normalized (sum to 1).
na_rm	logical. Should missing values (including NaN) be removed?
sizeD	logical. Only applied when *ss* > *1*, it decides whether to get a result of smaller size. If $sizeD = T$, *align* does not affect the output.
align	A string denotes how to align the moving average, three options: "left", "middle", "right"

Details

This function can obtain the moving sum efficiently. It serves as somehow a generalized version of movMeanUE. The difference is that it will not automatically normalized the weights vector, *w* argument.
If there is no missing value in *vec*, and *w* is normalized, which means the sum of all elements is *1*, this function will return a moving average. For matrix details, please refer to details of movMeanUE.

Value

This function returns A MATRIX of size: $L * 3$, where L is the length of vector, or of size: $L1 * 3$, where $L1 = \texttt{ceiling}((nrow - n + 1)/ss)$, (when $sizeD = T$). In the matrix, the first column denotes the position, the second column the original vector, and the third column the moving average.

Functions

- `movSumUEr`: An function equivalent to `movSumUE(..., align = "right")`

Examples

```
movSumUEr(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), 2)
movSumUEr(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2, na_rm = TRUE)
movSumUEr(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2, ss = 3, na_rm = TRUE,
sizeD = TRUE)
movSumUE(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), w = c(0, 1),  n = 2,
ss = 3, na_rm = TRUE, align = "right")
movSumUE(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2, ss = 3,
na_rm = TRUE, sizeD = TRUE, align = "right")
movSumUEr(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9),  n = 2, ss = 3,
na_rm = TRUE)
movSumUEr(c(1, 4, 3, NA, 8), pos = c(1, 2, 7, 8, 9), n = 2, ss = 3, na_rm = TRUE,
sizeD = TRUE)
```

Index