#### Analysis of Running/Rolling/Moving Windows

{TTR}

Analysis of Running/Rolling/Moving Windows

Description

Various functions to analyze data over a moving window of periods.

Usage

runSum(x, n = 10, cumulative = FALSE)

runMin(x, n = 10, cumulative = FALSE)

runMax(x, n = 10, cumulative = FALSE)

runMean(x, n = 10, cumulative = FALSE)

runMedian(x, n = 10, non.unique = "mean", cumulative = FALSE)

runCov(x, y, n = 10, use = "all.obs", sample = TRUE, cumulative = FALSE)

runCor(x, y, n = 10, use = "all.obs", sample = TRUE, cumulative = FALSE)

runVar(x, y = NULL, n = 10, sample = TRUE, cumulative = FALSE)

runSD(x, n = 10, sample = TRUE, cumulative = FALSE)

runMAD(x, n = 10, center = NULL, stat = "median", constant = 1.4826,

non.unique = "mean", cumulative = FALSE)

wilderSum(x, n = 10)

#### running function

1. cumulative sum-cumsum

Cumulative Sum or Rolling Sum or Running Total

group\_by(assignee.login, created\_at) %>%

summarize(counts = n()) %>%

mutate(running\_total = cumsum(counts))

1. Percent (%) of Total

mutate(total\_amounts = Adjusted \* Volume, date\_by\_month = round\_date(date,unit="month")) %>%

group\_by(date\_by\_month, symbol) %>%

summarize(total\_amounts = sum(total\_amounts)) %>%

mutate(percent\_total = total\_amounts / sum(total\_amounts) \* 100)

1. Difference from Beginning

group\_by(symbol, date\_by\_month) %>%

summarize(adjusted\_average = mean(Adjusted)) %>%

mutate(difference\_from\_first = adjusted\_average - **first**(adjusted\_average))

**Percent (%) Difference from Beginning or Previous:**

group\_by(symbol, date\_by\_month) %>%

summarize(adjusted\_average = mean(Adjusted)) %>%

mutate(

difference\_from\_first = adjusted\_average - first(adjusted\_average),

percent\_diff\_from\_first = difference\_from\_first /first(adjusted\_average)\*100)

**we can always change the beginning of the period by using ‘filter’ command like below:**

group\_by(symbol, date\_by\_month) %>%

summarize(adjusted\_average = mean(Adjusted)) %>%

filter(date\_by\_month >= "2013-01-01") %>%

mutate(

difference\_from\_first = adjusted\_average - first(adjusted\_average),

percent\_diff\_from\_first = difference\_from\_first /first(adjusted\_average)\*100)

**if we want to filter only for the last 2 years:**

group\_by(symbol, date\_by\_month) %>%

summarize(adjusted\_average = mean(Adjusted)) %>%

filter(date\_by\_month >= today() - years(2)) %>%

mutate(

difference\_from\_first = adjusted\_average - first(adjusted\_average),

percent\_diff\_from\_first = difference\_from\_first /first(adjusted\_average)\*100)

1. Difference from Previous

group\_by(symbol, date\_by\_month) %>%

summarize(adjusted\_average = mean(Adjusted)) %>%

mutate(diff = Adjusted - **lag**(Adjusted), percent\_diff = diff / lag(Adjusted) \* 100)

#### create multiple lagged version of a time series

> x <- (1:20)^2

> embed (x, 3)

[,1] [,2] [,3]

[1,] 9 4 1

[2,] 16 9 4

[3,] 25 16 9

[4,] 36 25 16

[5,] 49 36 25

[6,] 64 49 36

[7,] 81 64 49

[8,] 100 81 64

[9,] 121 100 81

[10,] 144 121 100

[11,] 169 144 121

[12,] 196 169 144

[13,] 225 196 169

[14,] 256 225 196

[15,] 289 256 225

[16,] 324 289 256

[17,] 361 324 289

[18,] 400 361 324

#### moving-average-over-a-categorical-variable

df <- structure(list(Group = c("exceeded", "exceeded", "exceeded",

"exceeded", "exceeded", "exceeded", "exceeded", "exceeded", "exceeded",

"exceeded", "exceeded", "exceeded", "othergroup", "othergroup",

"othergroup", "othergroup", "othergroup", "othergroup", "othergroup",

"othergroup", "othergroup", "othergroup", "othergroup", "othergroup"

), Days = c(0L, 1L, 2L, 3L, 4L, 5L, 6L, 7L, 8L, 9L, 10L, 11L,

0L, 1L, 2L, 3L, 4L, 5L, 6L, 7L, 8L, 9L, 10L, 11L), DV = c(2859L,

2948L, 4412L, 5074L, 5098L, 5147L, 4459L, 4730L, 4643L, 4698L,

4818L, 4521L, 2859L, 2948L, 4412L, 5074L, 5098L, 5147L, 4459L,

4730L, 4643L, 4698L, 4818L, 4521L), X5DayMA = c(NA, NA, NA, NA,

4078L, 4536L, 4838L, 4902L, 4815L, 4735L, 4670L, 4682L, NA, NA,

NA, NA, 4078L, 4536L, 4838L, 4902L, 4815L, 4735L, 4670L, 4682L

)), .Names = c("Group", "Days", "DV", "X5DayMA"), class = "data.frame", row.names = c(NA,

-24L))

library(dplyr)

library(zoo)

df %>% group\_by(Group) %>% mutate('5daymean' = rollmean(DV, 5, align="right", na.pad=TRUE ))

OR the super fast data.table

library(data.table)

dft <- data.table(df)

dft[ , `:=` ('5daymean' = rollmean(DV, 5, align="right", na.pad=TRUE )) , by=Group ]

<http://stackoverflow.com/questions/23619855/constructing-moving-average-over-a-categorical-variable-in-r>

more:

<http://stackoverflow.com/questions/26198551/rolling-mean-moving-average-by-group-id-with-dplyr>

#### moving average for the previous days include or exclude the current day

library(dplyr)

library(zoo)

library(DataCombine)

df = expand.grid(site = factor(seq(10)),

year = 2000:2004,

day = 1:50)

# use Poisson to make math easy to check moving means of temperature

df$temp = rpois(dim(df)[1], 5)

# Assume rains 33% of the days and averages 5 mm each time but highly variable

df$precip = rbinom(dim(df)[1], 1, 1/3) \* rlnorm(dim(df)[1], log(5), 1)

head(df, 75)

# moving mean for that day and previous days (e.g. 5 represents the mean of that day and the for previous days)

df2 = df %>%

group\_by(site, year) %>%

arrange(site, year, day) %>%

mutate(temp.5 = rollmean(x = temp, 5, align = "right", fill = NA))

head(df2, 75)

# moving mean for the previous days not including the current day (e.g. 5 represents the mean of the 5 previous days)

df2 %>%

mutate(temp.lag1 = **lag**(temp, n = 1)) %>%

mutate(temp.5.previous = rollapply(data = temp.lag1,

width = 5,

FUN = mean,

align = "right",

fill = NA,

na.rm = T))

**等价于：**

library(RcppRoll)

df2 %>%

mutate(temp.lag1 = lag(temp, n = 1)) %>%

mutate(temp.5.previous2 = roll\_meanr(temp.lag1,5,na.rm=TRUE))

**注意：**

**zoo package**

**The default method of rollmean does not handle inputs that contain NAs. In such cases, use rollapply instead.**

**The default method ofrollmedian is an interface to**[**runmed**](http://127.0.0.1:47661/help/library/zoo/help/runmed)**.**

#### moving median for the previous days include the current day

> library(RcppRoll)

> library(seismicRoll)

> library(RollingWindow)

> library(zoo)

> set.seed(123)

> input <- sample(c(1:100, rep(NA, 10)),30,replace=TRUE)

> input

[1] 32 87 45 98 NA 6 59 99 61 51 NA 50 75 63 12 99 28 5 37 NA 98 77 71 NA 73

[26] 78 60 66 32 17

> RcppRoll:::roll\_median(input, n = 5, fill = NA, align = 'right', na.rm = TRUE)

[1] NA NA NA NA 87 87 NA 59 59 59 61 61 NA 51 NA 63 63 28 28 37 37 NA 71 77 77

[26] NA 71 NA 66 60

> seismicRoll:::roll\_median(input,n=5,increment = 1)

[1] NA NA 87 87 98 NA 59 59 61 61 61 NA 50 63 63 28 28 37 37 NA 71 77 98 NA NA

[26] 66 66 60 NA NA

> rollapply(input, width=5, median, fill = NA, align="right",na.rm=TRUE)

[1] NA NA NA NA 66.0 66.0 52.0 78.5 60.0 59.0 60.0 56.0 56.0 57.0 56.5

[16] 63.0 63.0 28.0 28.0 32.5 32.5 57.0 74.0 77.0 75.0 75.0 72.0 69.5 66.0 60.0

> RollingMedian(x=input,window = 5, na\_method = "ignore")

# 去除na值后，加入非na值并凑齐5个值，再进行计算

> RollingMedian(x=input,window = 5, na\_method = "window")

# 只对连续5个无na值的window进行计算

> median(c(32, 87, 45, 98, NA),na.rm=TRUE)

[1] 66

> median(c(NA, 6, 59, 99, 61),na.rm=TRUE)

[1] 60

**注意：**

**RcppRoll package**

**慎用roll\_median!!!结果与预期不一致！**

ref:

<http://stackoverflow.com/questions/35086576/na-rm-behavior-of-roll-median>

<https://github.com/cran/seismicRoll/blob/master/R/roll_median.R>

#### moving count

(!x) \* unlist(lapply(rle(x)$lengths, seq\_len))

or

f7 <- function(x){ tmp<-cumsum(x);tmp-cummax((!x)\*tmp)}

or

cumul\_zeros <- function(x) {

x <- !x

rl <- rle(x)

len <- rl$lengths

v <- rl$values

cumLen <- cumsum(len)

z <- x

# replace the 0 at the end of each zero-block in z by the

# negative of the length of the preceding 1-block....

iDrops <- c(0, diff(v)) < 0

z[ cumLen[ iDrops ] ] <- -len[ c(iDrops[-1],FALSE) ]

# ... to ensure that the cumsum below does the right thing.

# We zap the cumsum with x so only the cumsums for the 1-blocks survive:

x\*cumsum(z)

}

> x <- sample(0:1, 1000000, TRUE)

> system.time(res7 <- f7(!x))

user system elapsed

0.076 0.000 0.077

> system.time(res0 <- cumul\_zeros(x))

user system elapsed

0.345 0.003 0.349

> identical(res7, res0)

[1] TRUE

> x <- sample(0:1, 1000000,T)

> system.time( z <- cumul\_zeros(x))

user system elapsed

0.15 0.00 0.14

> system.time( z <- (!x) \* unlist( lapply( rle(x)$lengths, seq\_len)))

user system elapsed

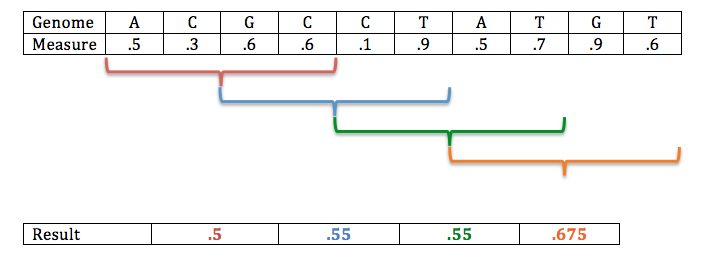
0.75 0.00 0.75

<http://stackoverflow.com/questions/5012516/count-how-many-consecutive-values-are-true>

<http://tolstoy.newcastle.edu.au/R/e4/devel/08/04/1206.html>

#### moving average for different step size

Using a window size of 4 and a step size of 2:

[](http://1.bp.blogspot.com/-eSp1w_oJsBc/U0NmHgWv76I/AAAAAAAAAzU/XyFsegtRKrU/s1600/Screen+Shot+2014-04-07+at+9.59.13+PM.png)

> x

[1] 0.5 0.3 0.6 0.6 0.1 0.9 0.5 0.7 0.9 0.6

> roll\_mean(x,n=4,by=1,align = "right")

[1] 0.500 0.400 0.550 0.525 0.550 0.750 0.675

> roll\_mean(x,n=4,by=2,align = "right")

[1] 5.000000e-01 2.514837e-315 5.500000e-01 2.496528e-315

[5] 5.500000e-01 2.496528e-315 6.750000e-01

> rollmean(x,k=4,align="right")

[1] 0.500 0.400 0.550 0.525 0.550 0.750 0.675

> rollmean(x,k=4,by=2,align="right")

[1] 0.500 0.400 0.550 0.525 0.550 0.750 0.675

#### adaptive-rolling-window-function-top-performance-in-r

<http://stackoverflow.com/questions/21368245/adaptive-rolling-window-function-top-performance-in-r>