data607assignment3B

AUTHOR
xiaofei

Approach

This assignment requires time series data. I'm planning to choose a stock price dataset. , I'll load data into R, check data type of each columns. Library tidyquant provide stock price, I choose stock Apple(AAPL) for this assignment to analyze price in year 2023. To use window functions calculate moving average, I will use cumean ( ) to calcualte YTD average and lag ( ) to look back 6 days window calculate 6 days moving average. In the end I'll show result through plots.

Data Exploration

From a glance, I have 250 rows of data which looks about right amount of total trading days in year 2023. There are 8 columns in different data type such as char, date and number. Lowest price occurred toward beginning of the year on date 1/3, and the highest price on 12/29. Overall this looks like a quality data set to work with.

```r
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```r
library(tidyquant)
```

```
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo

── Attaching core tidyquant packages ─────────────────────── tidyquant 1.0.11 ──
✔ PerformanceAnalytics 2.0.8      ✔ TTR                  0.24.4
✔ quantmod             0.4.28     ✔ xts                  0.14.1
── Conflicts ────────────────────────────────────────── tidyquant_conflicts() ──
✖ zoo::as.Date()                 masks base::as.Date()
✖ zoo::as.Date.numeric()         masks base::as.Date.numeric()
✖ dplyr::filter()                masks stats::filter()
✖ xts::first()                     masks dplyr::first()
✖ dplyr::lag()                     masks stats::lag()
✖ xts::last()                      masks dplyr::last()
✖ PerformanceAnalytics::legend() masks graphics::legend()
✖ quantmod::summary()              masks base::summary()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become
errors
```

```r
library(lubridate)
```

```

Attaching package: 'lubridate'

The following objects are masked from 'package:base':

    date, intersect, setdiff, union
```

```r
library(ggplot2)

stock_data <- tq_get("AAPL",
                     from = "2023-01-01",
                     to = "2023-12-31",
                     get = "stock.prices")

#check data
str(stock_data)
```

```
tibble [250 × 8] (S3: tbl_df/tbl/data.frame)
 $ symbol  : chr [1:250] "AAPL" "AAPL" "AAPL" "AAPL" ...
 $ date    : Date[1:250], format: "2023-01-03" "2023-01-04" ...
 $ open    : num [1:250] 130 127 127 126 130 ...
 $ high    : num [1:250] 131 129 128 130 133 ...
 $ low     : num [1:250] 124 125 125 125 130 ...
 $ close   : num [1:250] 125 126 125 130 130 ...
 $ volume  : num [1:250] 1.12e+08 8.91e+07 8.10e+07 8.78e+07 7.08e+07 ...
 $ adjusted: num [1:250] 123 124 123 128 128 ...
```

```r
head(stock_data)
```

```
# A tibble: 6 × 8
  symbol date        open  high   low close    volume adjusted
  <chr>  <date>     <dbl> <dbl> <dbl> <dbl>     <dbl>    <dbl>
1 AAPL   2023-01-03  130.  131.  124.  125. 112117500     123.
2 AAPL   2023-01-04  127.  129.  125.  126.  89113600     124.
3 AAPL   2023-01-05  127.  128.  125.  125.  80962700     123.
4 AAPL   2023-01-06  126.  130.  125.  130.  87754700     128.
5 AAPL   2023-01-09  130.  133.  130.  130.  70790800     128.
6 AAPL   2023-01-10  130.  131.  128.  131.  63896200     129.
```

```r
stock_data_clean <- stock_data |>
  select(date, close) |>  #use closeing price
  arrange(date) |>       #make sure date in order
  mutate(
    year = year(date),
    month = month(date),
    day = day(date)
  )
```

Window Functions

Use window function, in this case cumean ( ), to calculate YTD average. First need make sure data are sorted by date , then we will calculates a value for each row based on a window of previous rows. The window expands as date moving forward.

Similarly, we will use lab ( ) function to look back 6 days window. Each row's value depends on previous rows and windows slides forward as we move through the sorted dates. I notice there are some missing values for this one, due to there aren't enough days to calculate 6 days moving average at the beginning of the year.

```r
library(dplyr)
library(lubridate)

stock_analysis <- stock_data_clean |>
  # ensure ordering date
  arrange(date) |>

  group_by(year) |>
  mutate(
    ytd_avg = cummean(close),
    day_of_year = row_number(),
    ytd_min = cummin(close),
    ytd_max = cummax(close),

    # 6-day
    ma_6 = (close +
            lag(close, 1) +
            lag(close, 2) +
            lag(close, 3) +
            lag(close, 4) +
            lag(close, 5)) / 6
  )|>
  ungroup() |>

  mutate(
    overall_avg = mean(close, na.rm = TRUE)
  )
```

```r
# Check the results
stock_analysis |>
  select(date, year, close, ytd_avg, ma_6) |>
  head(15)
```

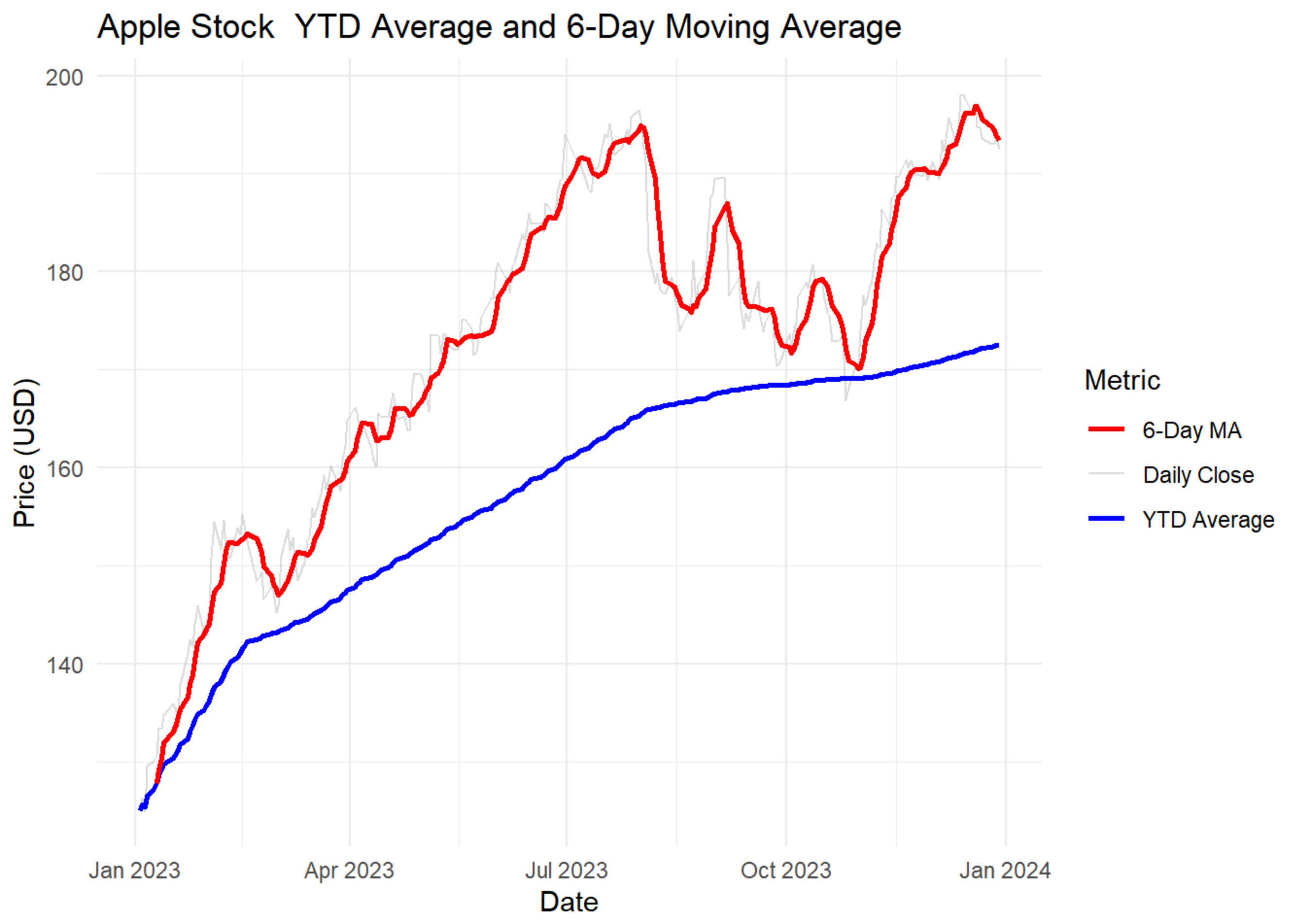
```
# A tibble: 15 × 5
   date        year close ytd_avg  ma_6
   <date>     <dbl> <dbl>   <dbl> <dbl>
 1 2023-01-03  2023  125.    125.   NA 
 2 2023-01-04  2023  126.    126.   NA 
 3 2023-01-05  2023  125.    125.   NA 
 4 2023-01-06  2023  130.    127.   NA 
 5 2023-01-09  2023  130.    127.   NA 
 6 2023-01-10  2023  131.    128.  128.
 7 2023-01-11  2023  133.    129.  129.
 8 2023-01-12  2023  133.    129.  130.
 9 2023-01-13  2023  135.    130.  132.
10 2023-01-17  2023  136.    130.  133.
11 2023-01-18  2023  135.    131.  134.
12 2023-01-19  2023  135.    131.  135.
13 2023-01-20  2023  138.    132.  135.
14 2023-01-23  2023  141.    132.  137.
15 2023-01-24  2023  143.    133.  138.
```

To show result in visualization :

```r
# plot for both averages
ggplot(stock_analysis, aes(x = date)) +
  geom_line(aes(y = close, color = "Daily Close"), alpha = 0.5) +
  geom_line(aes(y = ytd_avg, color = "YTD Average"), size = 1) +
  geom_line(aes(y = ma_6, color = "6-Day MA"), size = 1) +
  labs(title = "Apple Stock  YTD Average and 6-Day Moving Average",
       x = "Date",
       y = "Price (USD)",
       color = "Metric") +
  theme_minimal() +
  scale_color_manual(values = c("Daily Close" = "gray",
                                "YTD Average" = "blue",
                                "6-Day MA" = "red"))
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
ℹ Please use `linewidth` instead.

Warning: Removed 5 rows containing missing values or values outside the scale range
(`geom_line()`).
```



Conclusion

The goal of the project is do window function on a time series data. I choose stock price because it’s readyly available, well recorded with price and date. cumean ( ) and lag ( ) are both use of window function to calculate closing price of Apple stock in the chosen year - 2023. The plots shows the average price are trending up from Jan to Dec of 2023.

One small but crucial details is to make sure sort the data by date at the begin since the moving window will just expand by rows and won’t be checking on the dates during the process.