Computer Trading, Homework 1, Team 3

Kevin, Mukil, Shubham, Yifei

Jan 30, 2025

# Question 1

Identify the top 10 stocks and bottom 10 stocks in terms of total annual return (open-to-close) for the entire year. Report each set of stocks separately, from highest to lowest (in terms of absolute value) return along with the value of the return itself, as a percentage to 1 decimal. Ignore all stocks in which the corresponding data begins after the first trading day of the year or ends prior to the last trading day of the year

First, load the “OLHC.rdata” dataset, which contains trading data for S&P 500 stocks over the entirety of 2019. Let’s also define a constant TRADING\_DAYS to indicate the number of trading days in 2019.

**load**("OHLC.rdata")  
  
TRADING\_DAYS <- 250

Next, let’s get a list of all the unique symbols contained in our dataset. We can use the head() function to look at the first few symbols.

symbols <- **levels**(stock**$**symbol)  
**head**(symbols, n = 5)

## [1] "A" "AAL" "AAP" "AAPL" "ABBV"

The next step is a little more complicated; let’s break it down.

1. Filter out symbols that do not have at least 250 days of trading data available for 2019.
2. Calculate the annual return for each remaining symbol, as defined by:
3. Store our results in a data frame called returns

returns <- **data.frame**(  
 symbol = **character**(),  
 annual\_return = **double**()  
)  
**for** (sym **in** symbols) {  
 data <- **subset**(stock, stock**$**symbol **==** sym)  
  
 *# Filter out stocks that don't have enough data points*  
 **if** (**nrow**(data) **<** TRADING\_DAYS) **next**  
  
 first\_day <- **head**(data, n = 1)  
 last\_day <- **tail**(data, n = 1)  
  
 *# Calculate each stock's annual return*  
 ret <- (last\_day**$**close **-** first\_day**$**open) **/** first\_day**$**open **\*** 100  
 returns <- **rbind**(returns, **data.frame**(symbol = sym, annual\_return = ret))  
}

Sort returns by the value of annual\_return in decreasing order (largest returns first).

returns <- returns[**order**(returns**$**annual\_return, decreasing = TRUE), ]

Filter out the 10 stocks with the highest returns and lowest returns, respectively. Sort the lowest\_returns dataframe to display the largest absolute values in decreasing order.

highest\_returns <- **head**(returns, n = 10)  
lowest\_returns <- **tail**(returns, n = 10)  
lowest\_returns <- lowest\_returns[**order**(lowest\_returns**$**annual\_return), ]

Round the resulting dataframes to 1 decimal place.

highest\_returns**$**annual\_return <- **round**(  
 highest\_returns**$**annual\_return,  
 digits = 1  
)  
highest\_returns

## symbol annual\_return  
## 275 LRCX 121.0  
## 254 KLAC 106.7  
## 4 AAPL 105.5  
## 409 TGT 101.1  
## 367 QRVO 100.5  
## 108 CPRT 92.5  
## 467 XRX 90.9  
## 400 SWKS 90.6  
## 31 AMAT 89.6  
## 95 CMG 89.2

lowest\_returns**$**annual\_return <- **round**(  
 lowest\_returns**$**annual\_return,  
 digits = 1  
)  
lowest\_returns

## symbol annual\_return  
## 7 ABMD -45.6  
## 279 M -40.3  
## 140 DXC -31.1  
## 311 MYL -30.2  
## 336 OXY -30.1  
## 196 GPS -27.1  
## 299 MOS -26.9  
## 263 LB -26.8  
## 14 ADS -26.2  
## 101 COG -26.1

Finally, clear out unused temporary variables.

vars <- **ls**()  
**rm**(list = vars[vars **!=** "highest\_returns" **&** vars **!=** "lowest\_returns"])  
**rm**(vars)

# Question 2

Report the average annual return, across stocks for each financial sector. Use the same reporting style as per question 1.

First, let’s filter out stocks whose trading data either doesn’t begin on the first day available or doesn’t end on the last day available.

**load**("OHLC.rdata")  
  
first\_day <- **aggregate**(date **~** symbol, data = stock, min)  
first\_day <- **subset**(first\_day, date **==** **min**(stock**$**date))  
last\_day <- **aggregate**(date **~** symbol, data = stock, max)  
last\_day <- **subset**(last\_day, date **==** **max**(stock**$**date))

After that, let’s calculate the annual returns for each filtered stock and round our results to 1 decimal place.

open\_prices <- **merge**(stock[, **c**("symbol", "date", "open")], first\_day)  
close\_prices <- **merge**(stock[, **c**("symbol", "date", "close")], last\_day)  
filtered <- **merge**(open\_prices, close\_prices, by = "symbol")  
filtered**$**return <- **round**(  
 100 **\*** (filtered**$**close **-** filtered**$**open) **/** filtered**$**open, digits = 1  
)

Next, load in our sector data, which associates every stock in our dataset with a sector. We then merge it with our filtered stock data frame.

sectors <- **read.csv**("sectors.csv")  
filtered <- **merge**(filtered, sectors)

Finally, let’s aggregate the average returns for each sector and sort the resulting data frame by returns in descending order.

returns\_by\_sector <- **aggregate**(return **~** sector, data = filtered, mean)  
returns\_by\_sector**$**return <- **round**(returns\_by\_sector**$**return, digits = 1)  
returns\_by\_sector <- returns\_by\_sector[  
 **order**(returns\_by\_sector**$**return, decreasing = TRUE),  
]  
returns\_by\_sector

## sector return

## 8 Information Technology 43.0

## 7 Industrials 34.5

## 5 Financials 32.5

## 10 Real Estate 30.4

## 3 Consumer Staples 28.3

## 9 Materials 28.1

## 11 Utilities 27.8

## 2 Consumer Discretionary 26.6

## 1 Communication Services 26.4

## 6 Health Care 25.9

## 4 Energy 7.7

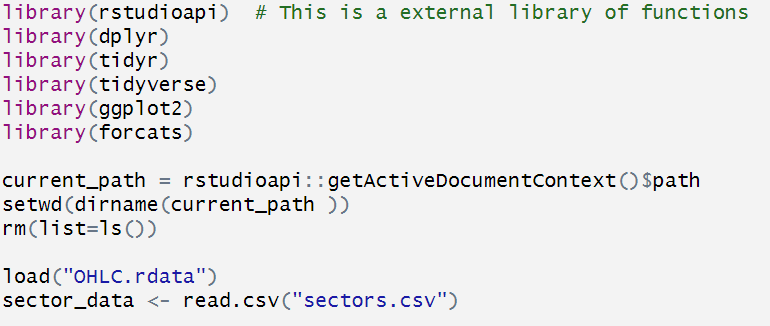
# 

# Question 3

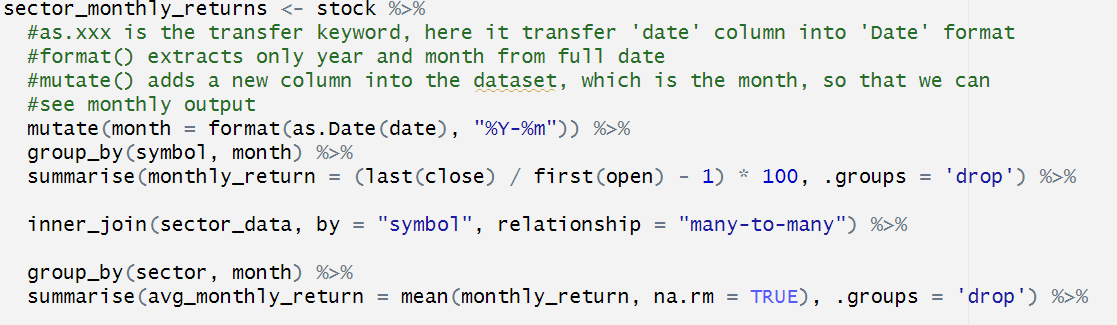
Create a matrix or average returns by sector (rows) and month (columns). Use open-

to-close for the monthly returns of each stock. Sort rows alphabetically by sector.

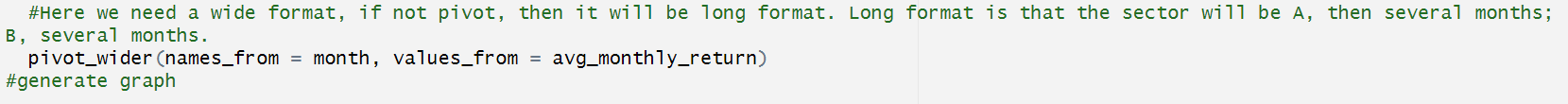
First load the library and dataset

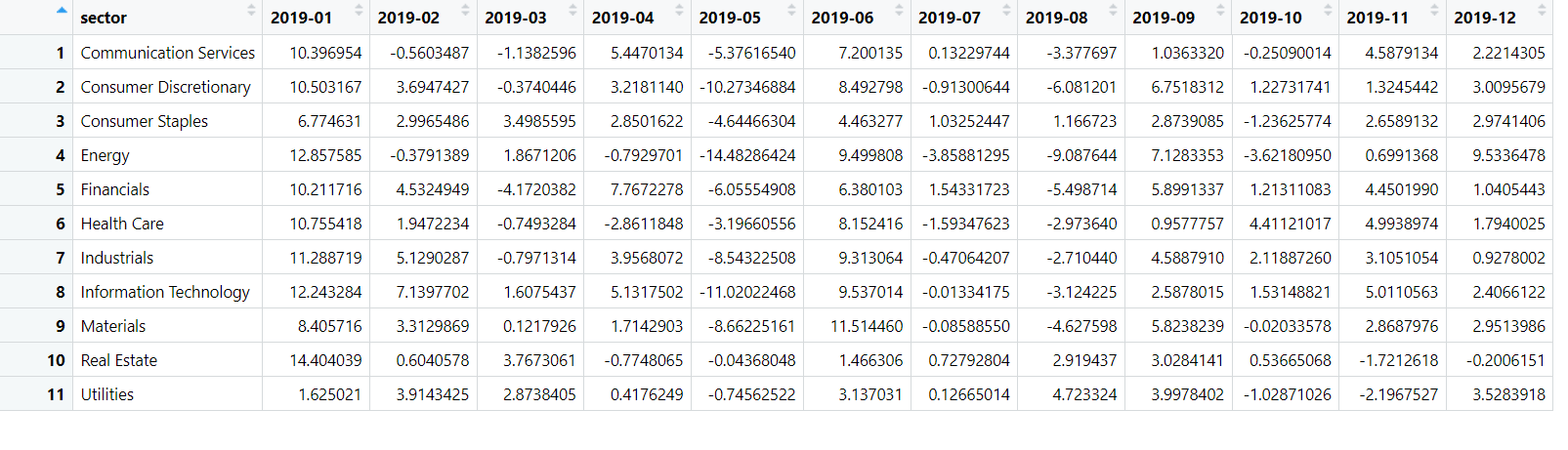


Then, using GroupBY and Inner Join method to create the metrix that has sector, sector month returns and month.

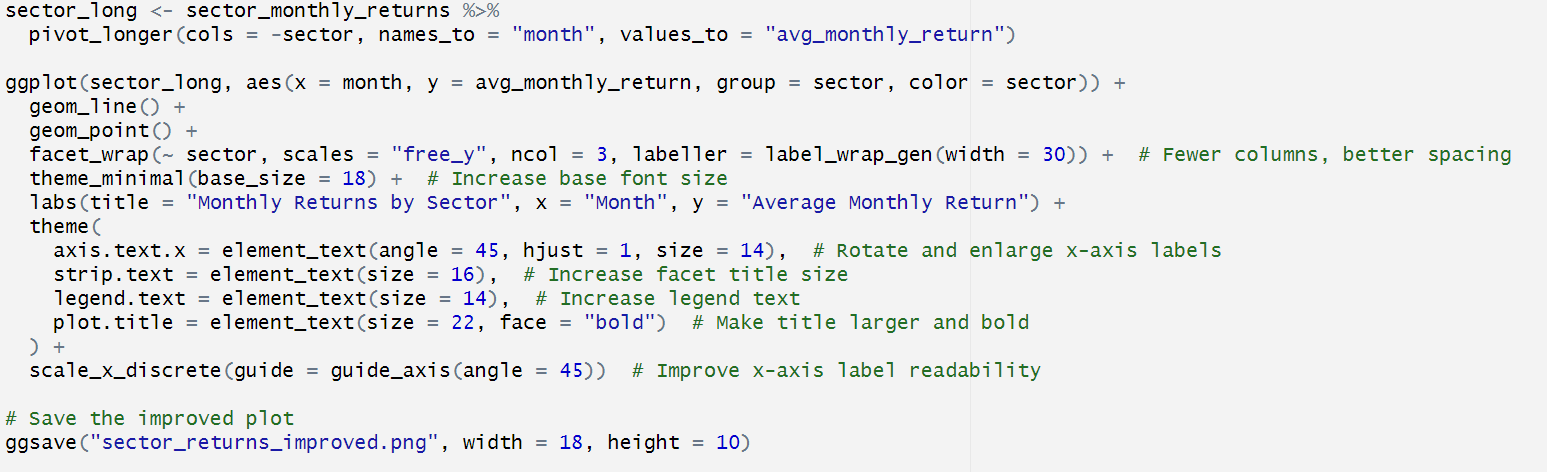


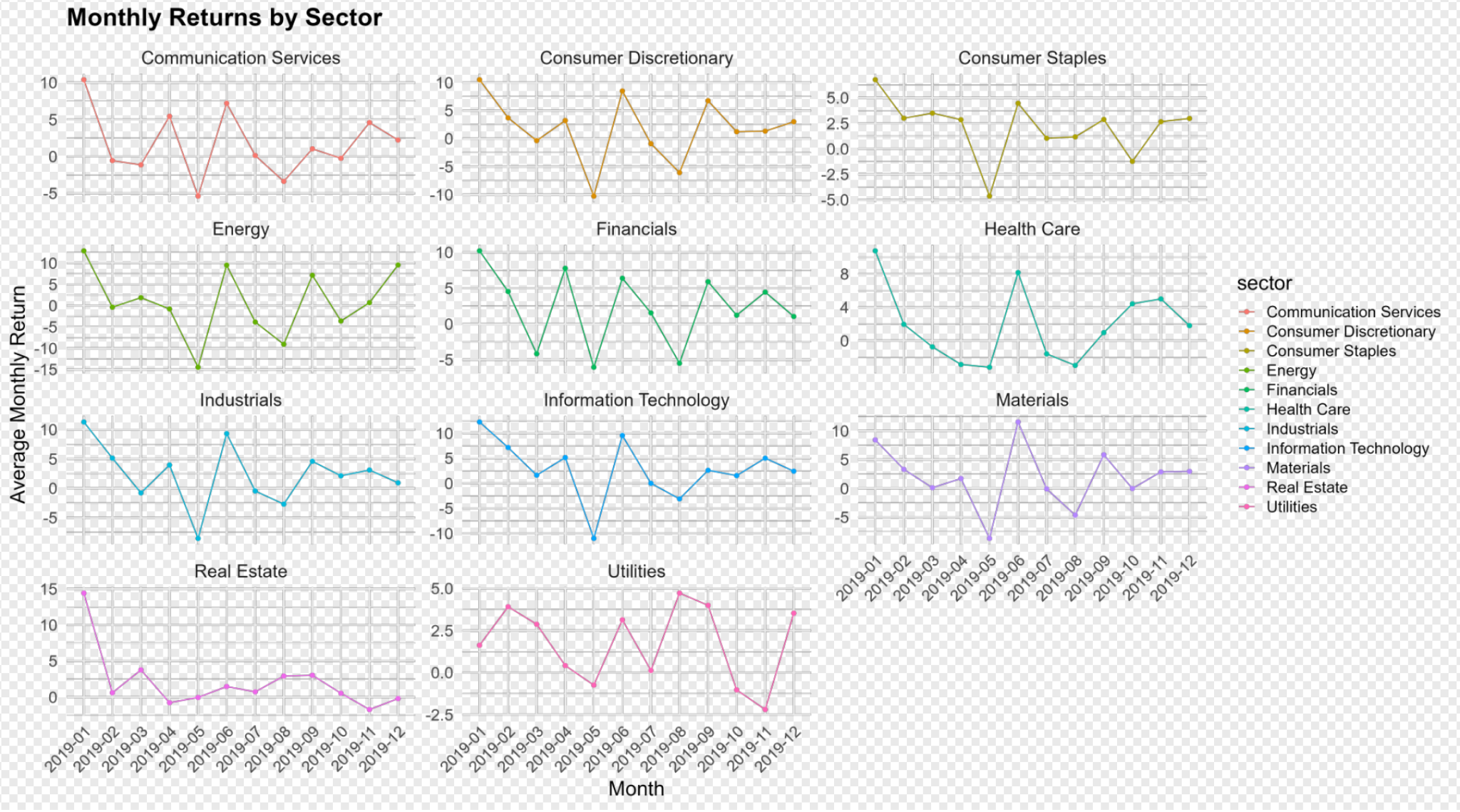
Next, create the matrix. This pivot wider method is to generate nice metrix graph in R. So that we can see different month's return in different sector more easily





Here are the codes to generate graph in R markdown. I used ggplot to adjust the visual output of the graph. This will save the output as a png file in the current path.





# Question 4

Now, let’s assume you would like to know the daily and cumulative returns that would arise if you had day traded Apple stock. Assume you purchase one share every day at the open and sell that same share at the close. Plot the daily return and cumulative return for the entire year. In addition, keep track of and plot the maximum cumulative return achieved through each trading day. So let say the daily return on the first day is 0.9875 (relative to 1.00 so this is a loser day), then the cumulative return is also 0.9875 and so too is the maximum return. Now on the second day, let’s say the daily return is 1.0258. Then the cumulative return is 0.9875\*1.0258 = 1.013 (with a little rounding). Now the maximum cumulative return is also 1.013 since it is greater than 0.9875. Note that if the daily return in a trading day is less than 1, then the cumulative maximum return will be the same as the prior trading day. In addition to the plot, be sure to provide an explanation of what information this cumulative maximum return provides and how it might be useful in evaluating a trading strategy.

Introduction

In this tutorial, we will analyze Apple stock data to compute and plot the daily return, cumulative return, and the maximum cumulative return achieved over each trading day.

1. Clearing the Environment and Loading Required Libraries

Let’s begin by clearing our environment to remove any existing objects. We then load the rstudioapi package, which is useful for interacting with the RStudio IDE. This helps ensure that we start with a clean slate.

**rm**(list=**ls**())  
**library**(rstudioapi)

2. Loading Data and Viewing Available Objects

Next, we load the necessary data files. The load("OHLC.rdata") command loads our market data, and we also read in sector data from a CSV file. After loading, we print the list of objects available in the environment to verify that the data has been successfully loaded.

**load**("OHLC.rdata")  
stockdata <- **read.csv**("sectors.csv")  
**print**(**ls**())

## [1] "stock" "stockdata"

3. Subsetting Data for Apple (AAPL) and Ordering by Date

After verifying that we have both objects from the loaded data, we move onto extracting the data specific to Apple by using the subset function. The given data is already ordered by date, but we order it again to ensure there are no entries diverting from the ordering scheme.

apple\_data <-**subset**(stock, symbol**==**"AAPL")  
apple\_data <-apple\_data[**order**(apple\_data**$**date),]

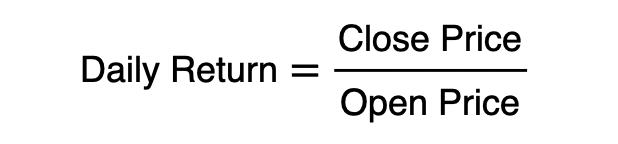
4. Attaching the Apple Data

The attach() function makes the variables in apple\_data available by name, so you can refer to columns directly (e.g., close and open). Although using attach() is not always recommended due to potential naming conflicts, it is used here for simplicity in this tutorial.

**attach**(apple\_data)

5. Calculating the Daily Return

The daily return is generally calculated using the formula:



The question requires us to start by calculating the daily return for the stock. So, using the formula explained above, we write an equation to compute it. This gives us a sense of how much the stock has gained or lost relative to its opening price.

daily\_return<-close**/**open

6. Calculating the Cumulative Return

The cumulative return is calculated by taking the cumulative product of the daily returns. This shows the compounded return if you had held the position continuously throughout the period.

cumulative\_return<-**cumprod**(daily\_return)

7. Tracking the Maximum Cumulative Return

The maximum cumulative return up to each trading day is determined using the cummax() function. This function keeps track of the highest cumulative return achieved so far. This metric is useful because it provides a benchmark of the best performance reached up to any given day, which helps in assessing drawdowns and overall strategy performance.

max\_cumulative\_return<-**cummax**(cumulative\_return)

8. Combining the New Metrics with the Original Data

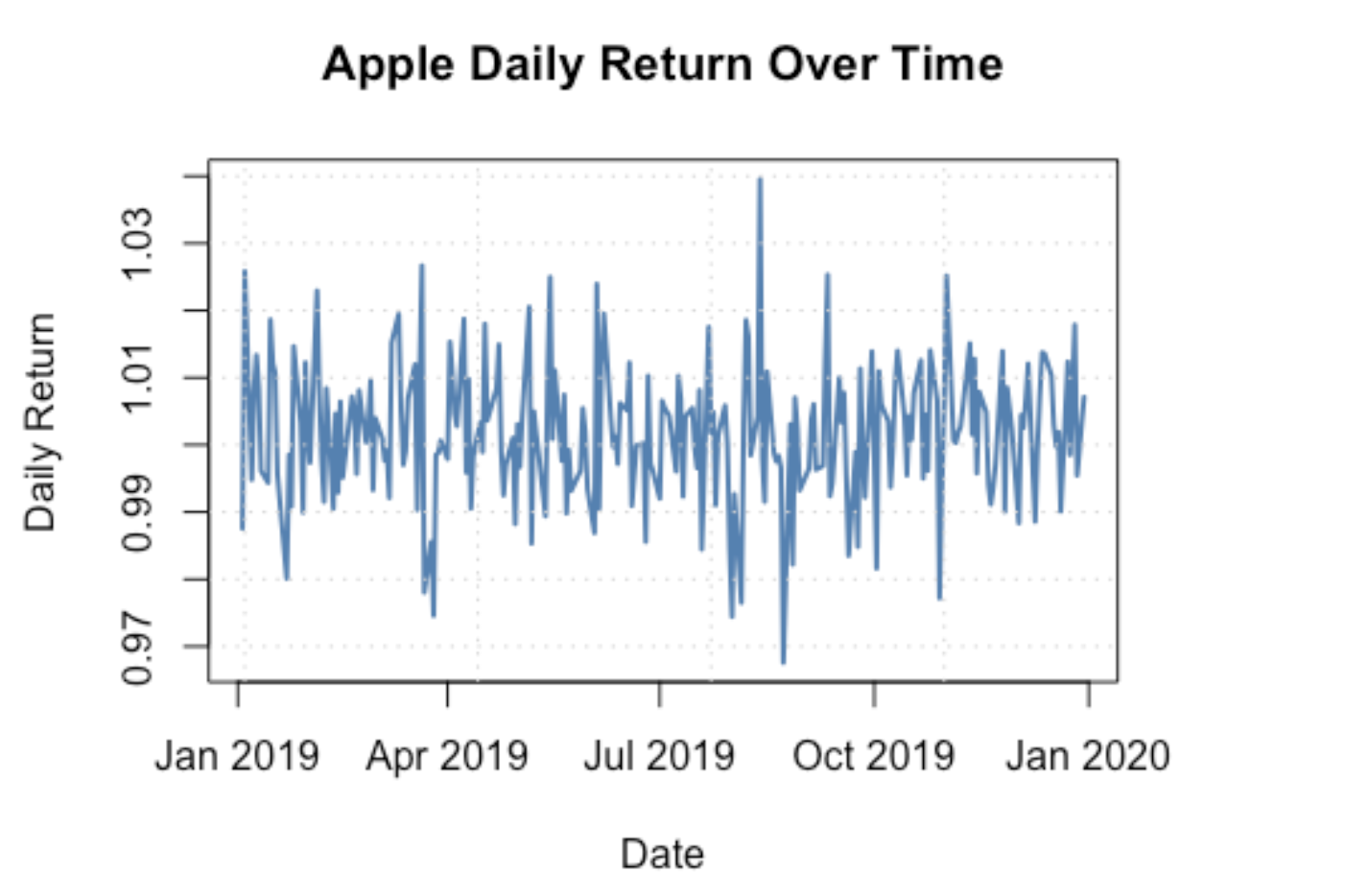
We combine our calculated metrics (daily return, cumulative return, and maximum cumulative return) with the original apple\_data data frame. This allows us to have all our variables together for further analysis and plotting.

apple\_data<-**data.frame**(apple\_data,daily\_return,cumulative\_return,max\_cumulative\_return)

9. Plotting the Daily Return

We create a plot of the daily returns over time. This plot is formatted with a steel blue line, increased line width, grid lines for better readability, and proper axis labels and a title.

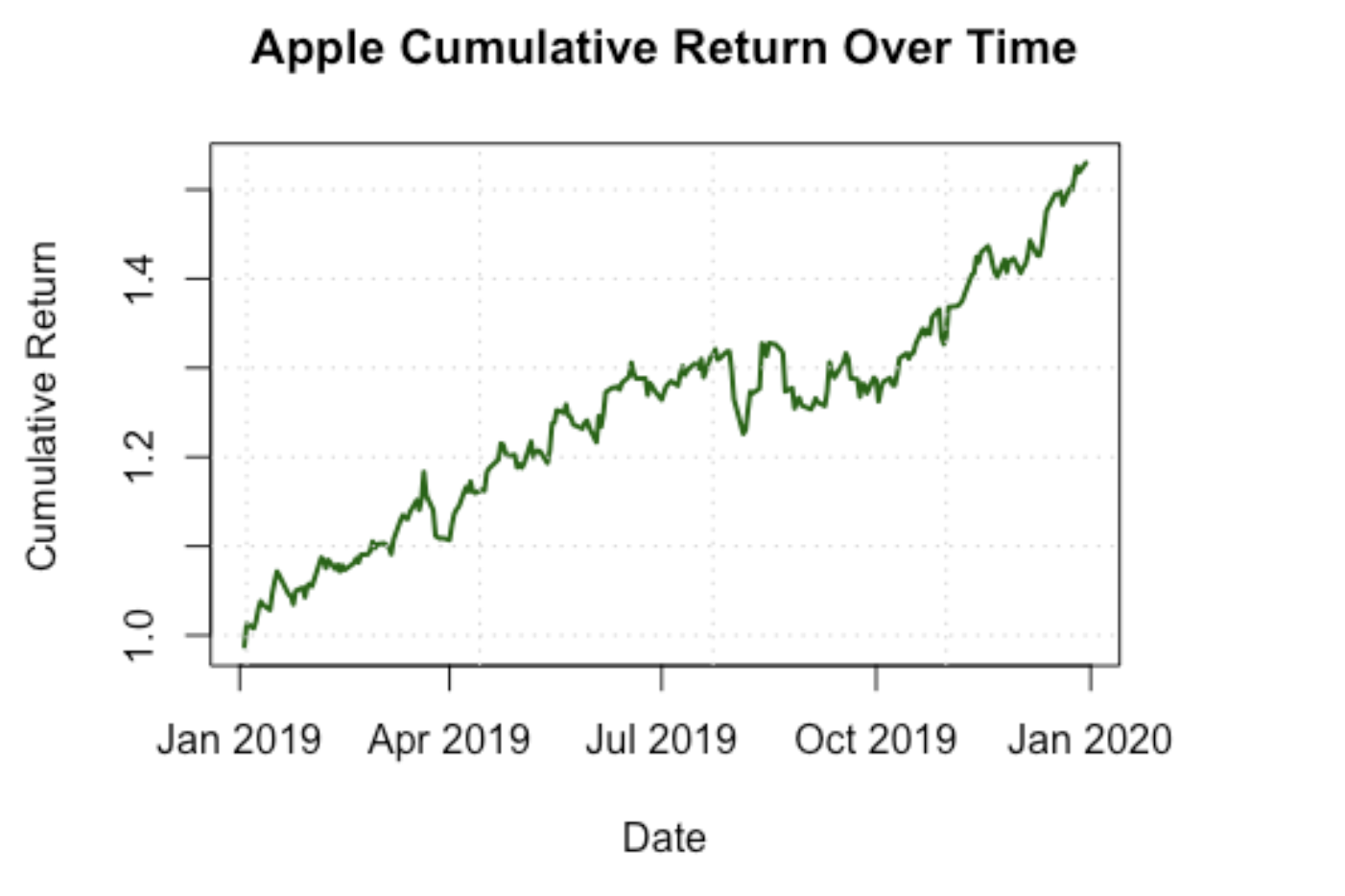
**plot**(apple\_data**$**date, apple\_data**$**daily\_return,  
 type = "l", *# Line plot* col = "steelblue", *# Line color* lwd = 2, *# Line width* xlab = "Date",  
 ylab = "Daily Return",  
 main = "Apple Daily Return Over Time")  
**grid**()



10. Plotting the Cumulative Return

Next, we plot the cumulative return over time using a dark green line. This plot shows the compounded effect of the daily returns throughout the period.

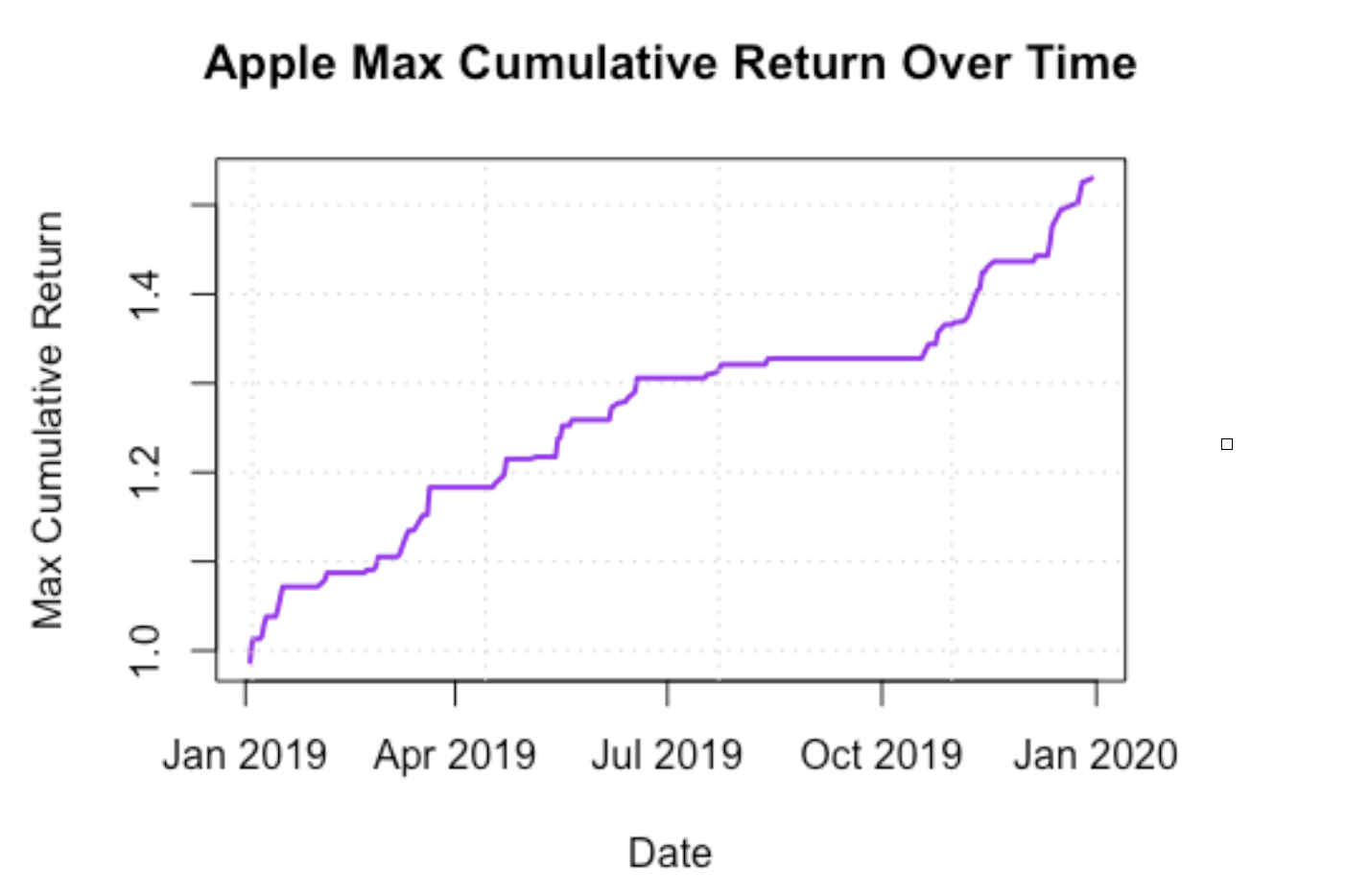
**plot**(apple\_data**$**date, apple\_data**$**cumulative\_return,  
 type = "l", *# Line plot* col = "darkgreen", *# Line color (choose any color you like)* lwd = 2, *# Line width* xlab = "Date",  
 ylab = "Cumulative Return",  
 main = "Apple Cumulative Return Over Time")  
**grid**()



11. Plotting the Maximum Cumulative Return

Finally, we plot the maximum cumulative return. This plot, shown in purple, displays the highest cumulative return achieved up to each trading day. This metric is particularly useful for evaluating the peak performance of the trading strategy and understanding drawdowns.

**plot**(apple\_data**$**date, apple\_data**$**max\_cumulative\_return,  
 type = "l", *# Line plot* col = "purple", *# Line color (you can choose any color)* lwd = 2, *# Line width* xlab = "Date",  
 ylab = "Max Cumulative Return",  
 main = "Apple Max Cumulative Return Over Time")  
**grid**()



12. Information from the Max Cumulative Return Plot

The plot gives us an idea of the highest return that can be achieved at any given point during trading, making it easier to gauge the performance of our trading strategy.

If we see long stretches without any rise in the plot, we can deduce that the strategy is not very efficient over time.

We can also use this graph to compare how the strategy works for other stocks or how the strategy has improved or declined compared to previous plots or past analyses.