

*Texas A&M University*

**FINC 689 : Systematic Trading Strategies**

**Homework 1 Tutorial Submission (Spring 2023)**

Instructor

1. Dr. Michael Ketzenberg

Team 4

1. Sankalp Chapalgaonkar (233005696)
2. Kushagra Jain (532008695)
3. Kailesh Hariharan (933003377)

***Introduction***

This report serves as a tutorial to help the reader understand the solutions to problem statements for Homework 1. The four questions addressed in this tutorial focus on performing different types of data manipulation using R programming and attempt to derive insights out of financial data.

R programming is an open-source software environment widely used for statistical analysis which, as it turns out, is perfect for the kind of work that will be performed in this course. Data manipulation and transformation in conjunction with preparing visualizations is an important part of data analysis which is a key outcome of this tutorial.

The dataset being used in this tutorial is the Open-High-Low-Close stock dataset for S&P 500 during the year 2019. The S&P 500 is a stock index which tracks the stock performance of 500 large companies listed in the United States stock market. As we progress through building solutions for the four questions, we will explore different aspects of the dataset and as a result understand how returns are calculated and reported over time.

***Learning Objectives***

Before we dive into the problems, we are going to address we want to outline the major learnings you can expect to be gaining at the end of this Tutorial. Here is the list of Learning Objectives for this tutorial:

* At the end of this tutorial, you will have an understanding of how we can compute our trading returns.
* At the end of this tutorial, you will understand the different timeframes for which we can calculate trading returns.
* At the end of this tutorial, you will learn how to use R programming language to program your trading strategy.
* At the end of this tutorial, you will learn about different R functions that can help perform fast computations and filters on your data.
* At the end of this tutorial, you will learn how to plot our results using R.
* At the end of this tutorial, you will learn how to gain insights from the plots of the different kinds of returns you calculate and use that insight to determine the correctness of your trading strategy.
* At the end of this tutorial, you will learn how to read, clean and manipulate trading datasets.

#### ***Problem 1***

In first problem we are supposed to find the annual returns for the stocks given in the **OHLC.rdata** file which contains the information about the name of the stock, open, high, low and close for a day and the date for which this data is presented.

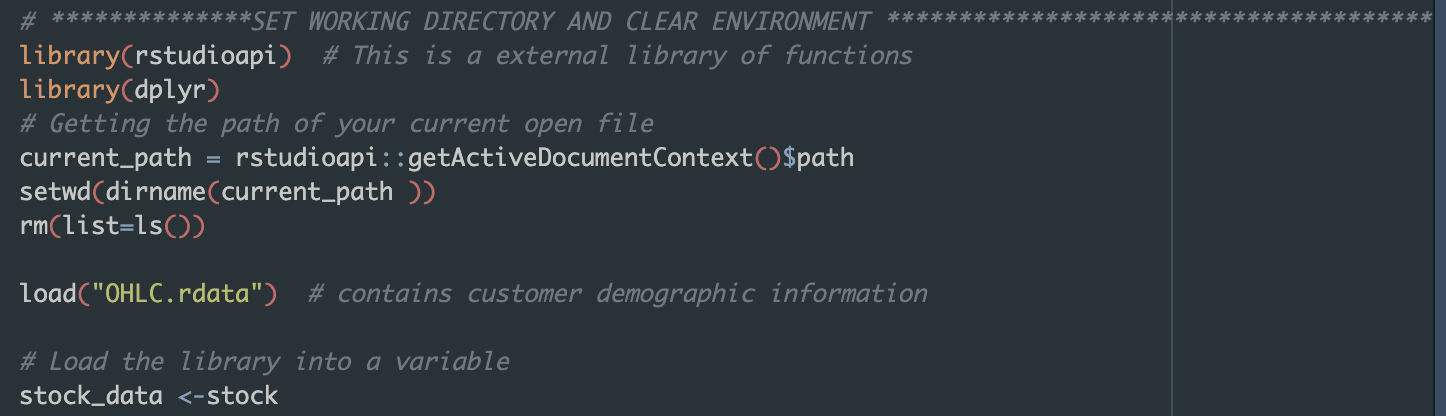
*Steps and Implementation*

Here are the steps we followed to tackle this problem:

1. Load the various libraries needed for the processing and Load the stocks data from **OHLC.rdata** File

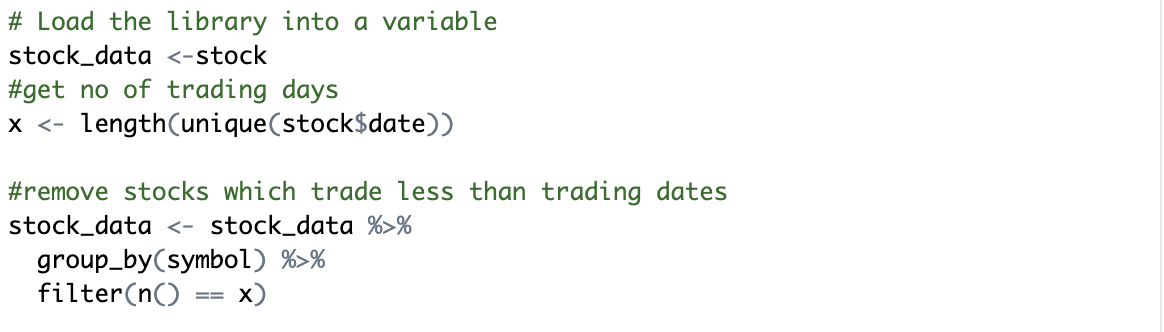
We are using the following two libraries

* **rstudioapi** – This library is an R package that provides an API (Application Programming Interface) to interact with the RStudio IDE (Integrated Development Environment). The rstudioapi library can be used to access various components of the RStudio environment, such as the source editor, the console, and the environment, and to interact with these components programmatically. We are using this package to get the activeDirectoryPath and setting it as the working directory for the current RStudio environment.
* **dplyr** – This package in the R programming language provides a set of functions for data manipulation and cleaning. It can be used to clean, filter, and aggregate data, making it easier to perform data analysis. We used group\_by function to aggregate the records on stock level and filter function to filter the records on date column.



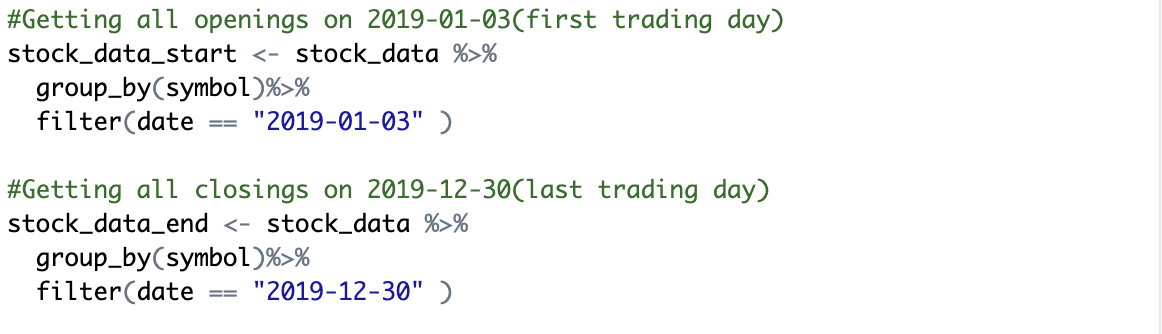
1. Reading the data from the file and group them on stock level

Here the stock data is stored into stock\_data variable and we calculate the unique number of dates by using the unique function on the date column. Then we group the stocks on stock level and filter the stocks which are traded on each trading day of the year(2019).



1. Fetch the opening on the first and last trading date

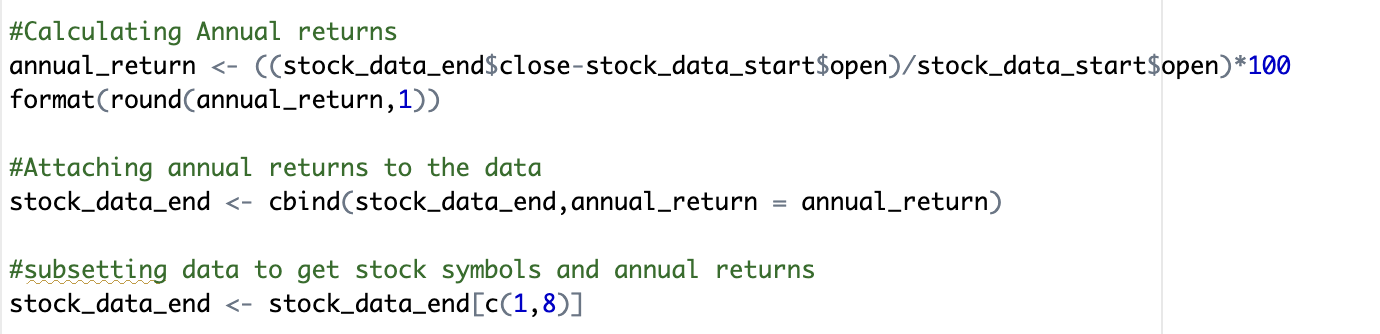
Here the first trading date is January 3, 2019 and the last trading date is December 30, 2019 according to the data file. We filter the data using the filter function applied on the date column in the data and the filtered results are stored in the stock\_data\_start and stock\_data\_end variable.



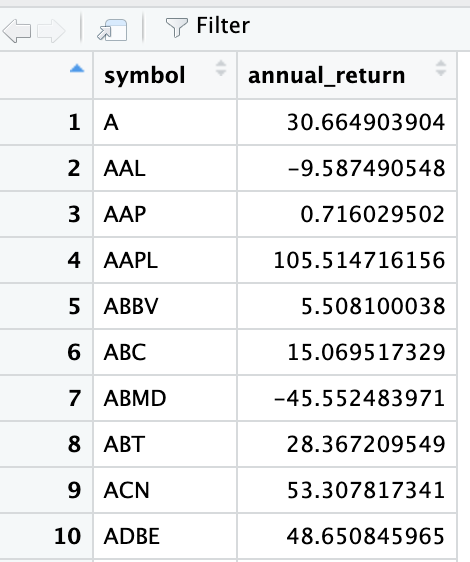
1. Calculate the total annual return for each stock by subtracting the opening price of the first trading day of the year from the closing price of the last trading day of the year and dividing the result by the opening price of the first trading day of the year

Here annual returns are calculated from the following formula

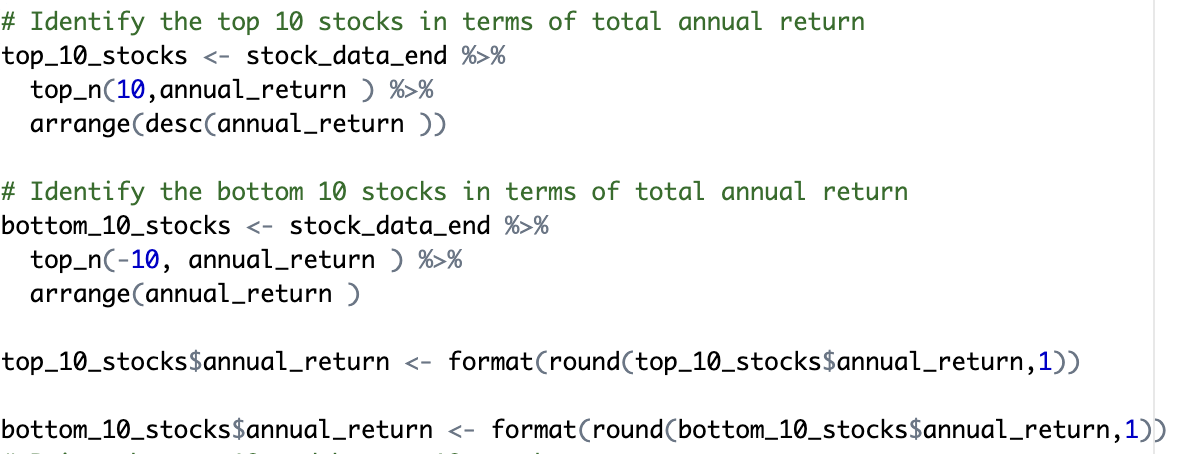
We format the calculated annual returns to one decimal place and append it to the stock\_data\_end using **cbind** function and keep only the stock name and the value of annual returns as the only two columns.



The final stock\_data\_end looks like the following:

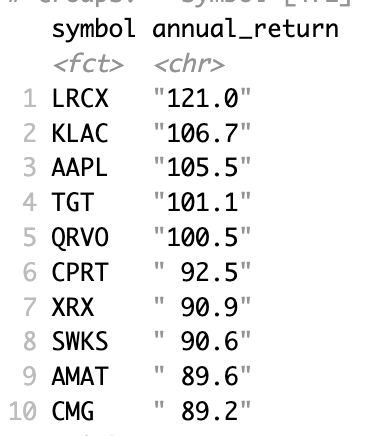


1. Rank the stocks based on their total annual return in terms of absolute value, from highest to lowest and print the top 10 stocks with the highest total annual return and the bottom 10 stocks with the lowest total annual return. Annual Returns are expressed as a percentage to 1 decimal place.

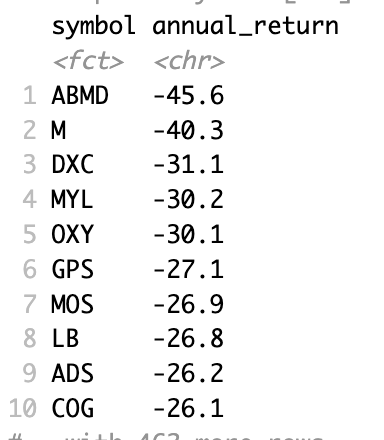


Here we use the **top\_n** function to sort the data in descending order based on the annual return values for getting the top 10 stocks and in ascending order(descending by absolute value) for getting the bottom 10 stocks. Once we have top and bottom 10 stocks based on the annual return values, we use the **print** function to output the results as expected. Results are as follows

**Top 10 Stocks**



**Bottom 10 Stocks**



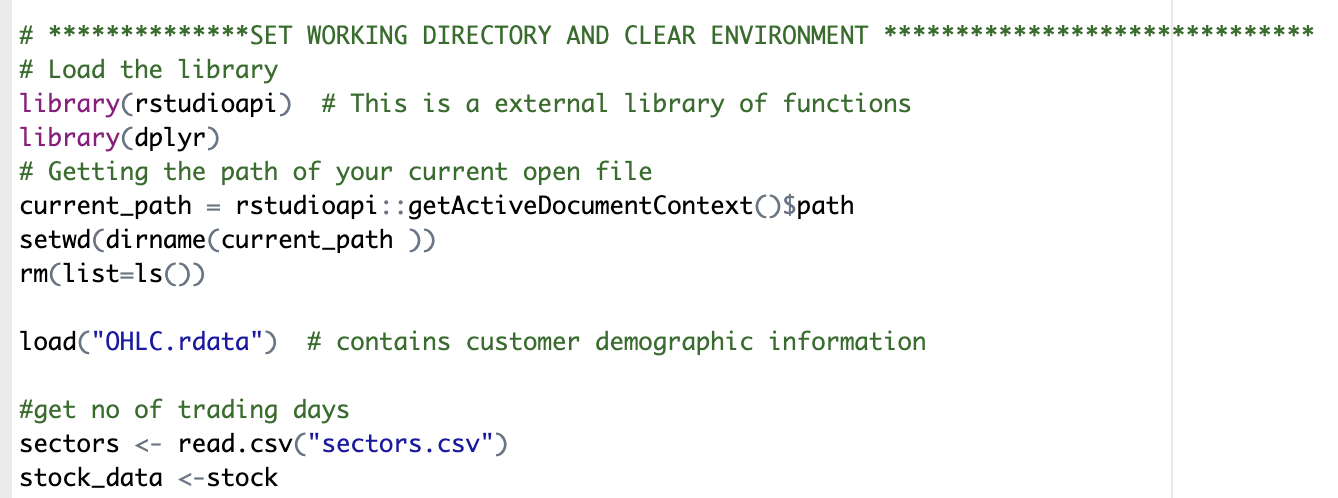
#### ***Problem 2***

Here similar to problem 1 we are required to calculate the annual returns for each sectors. The sectors data (stock-sector mappings) is given in the separate csv file named **sectors.csv**

*Steps and Implementation*

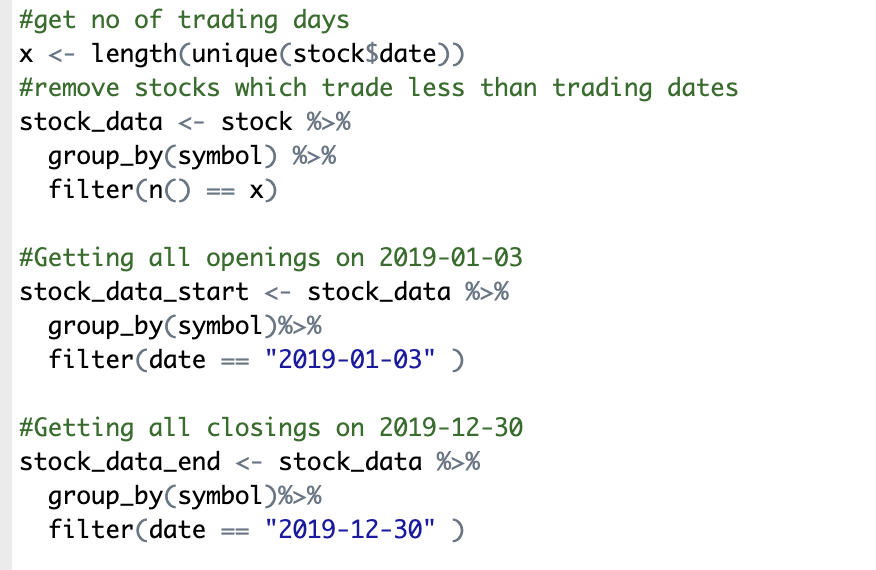
1. Importing the required R packages, setting the working environment details and loading the data files

Same as previous problem, we are using **rstudioapi** and **dplyr** packages. We load the **OHLC.rdata** and **sectors.csv** to read the stocks and sectors data respectively.



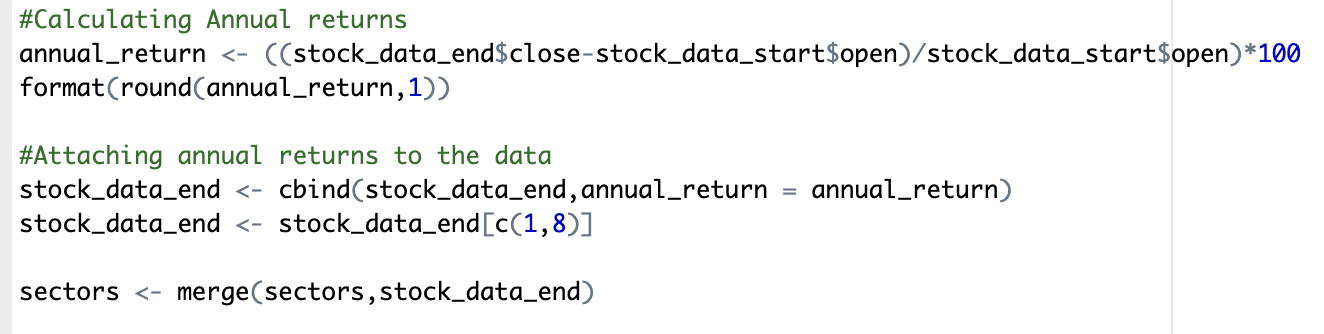
1. Group the stocks data on stock level and fetch the data for first and last trading day

We calculate the number of days in the rdata file and filter the stocks which have the corresponding data for each day. We fetch the open and close data for each stock on the first (January 3, 2019) and last (December 30, 2019) trading day respectively.



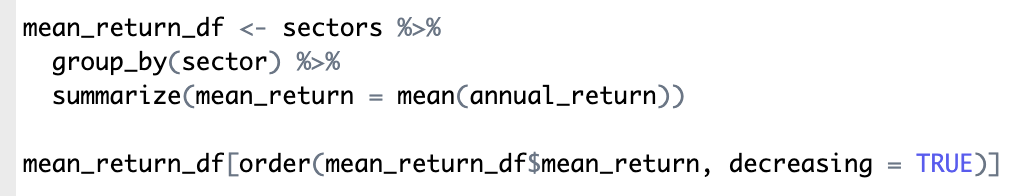
1. Calculate the annual returns using the formula mentioned in problem one

Here we first calculate the annual returns for each stock and then attach this data to the **stock\_data\_end** variable using **cbind** function. After this we merge this with the sectors data so that we have the annual returns for each stock and corresponding sector in the sectors dataset.

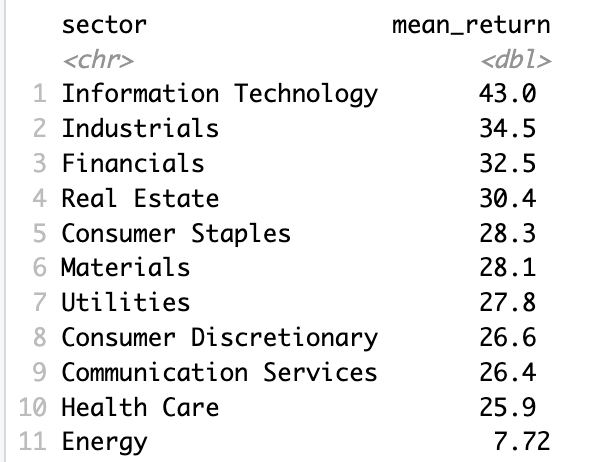


1. Calculate the mean annual returns and print the results

Here we use group\_by function to group the sectors annual returns on sector level and print the result in the descending order.



Here are the average annual returns across stocks for each financial sector in descending order



#### ***Problem 3***

The third problem we have is to calculate the monthly returns for each sector and then we try to present it in form of a matrix with row names as sectors and column names as months sorted alphabetically by row names. Before we dive into the implementation details and steps, let's define what monthly returns are. So, a monthly return is defined by the following formula

We basically try to access how much our stock has gained in a month by considering that we sell at the closing of the last trading day of the month and assuming that we bought at the opening of the first trading day of the month. Let me also point out that we do this for sectors, not individual stocks, so we need to calculate the monthly return for each stock and then combine the results sector-wise.

*Approach*

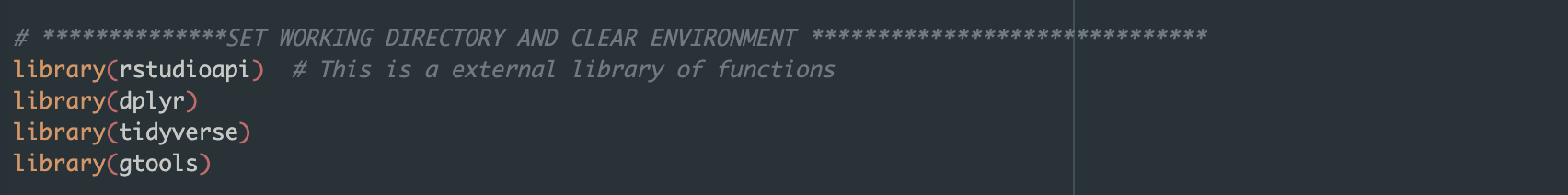
To calculate the monthly returns sector wise we first calculate the monthly returns for each stock by grouping the daily return records by month and then we group the stocks by sectors and take mean of all the monthly returns for all the stocks in that sector.

*Steps and Implementation*

1. Importing the required libraries

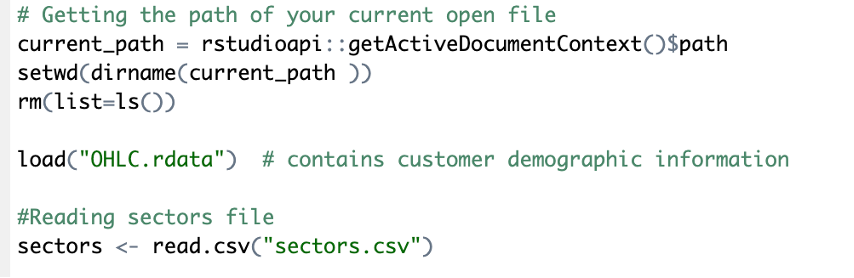
Here we import 2 additional libraries namely tidyverse and gtools.

* **tidyverse** – will help us identify the opening and closing dates of the stocks in a month
* **gtools** – will help us sort the matrix rows alphabetically



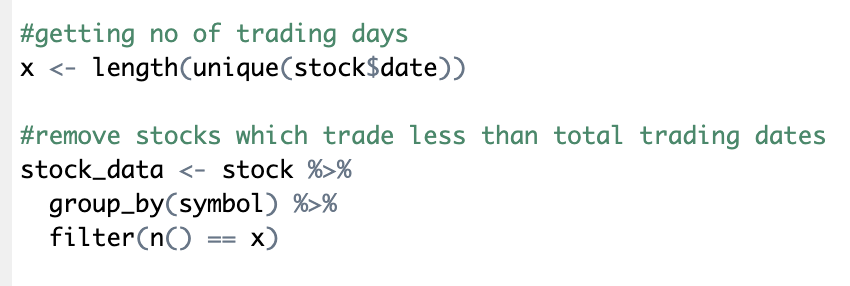
1. Set the Working directory

As done in the above problems we set the working directory and load the data into the working environment



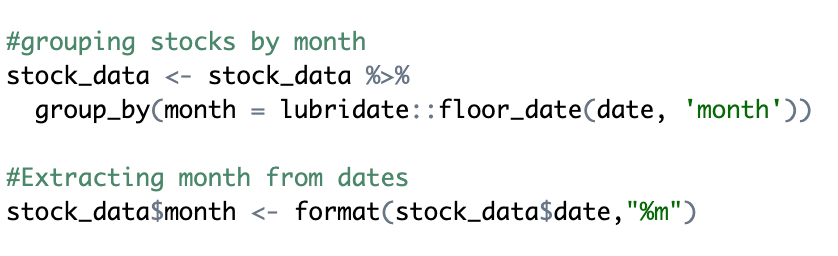
1. Clean the data

For the purpose of our tutorial and to maintain consistency in data we consider only those stocks that are traded on each trading day of the year. The logic remains the same as in previous problems.

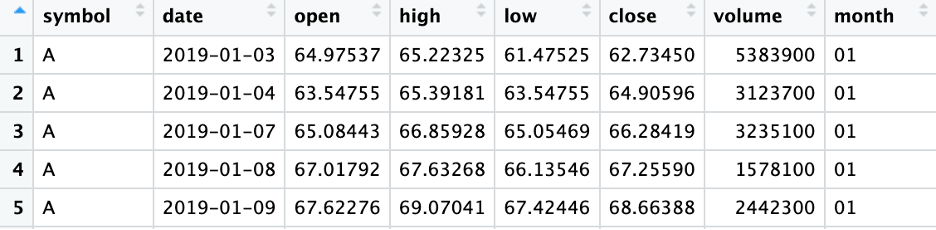


1. Group stock data by month

We group all the stocks by month to make aggregation easier

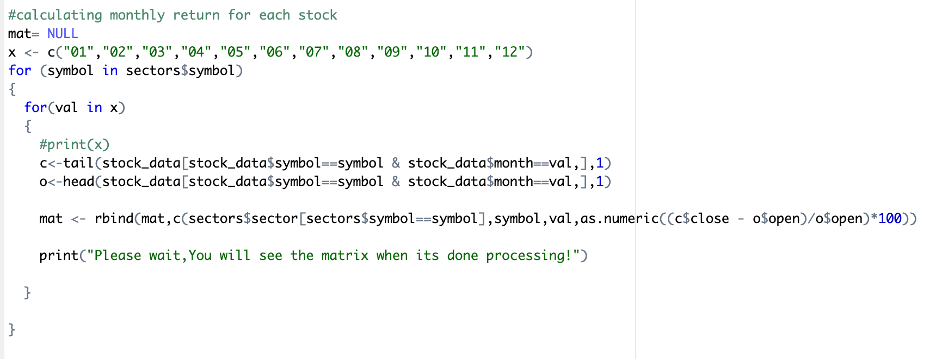


**lubridate::floor\_date** helps us get the first date of the month where that particular date lies; we then extract the month from the modified date. For example if date ==”2019-01-15” then floor date makes it “2019-01-01” and then we extract “01” for Januaray from it. So at the end of this statement, we have the data grouped by month for each record.

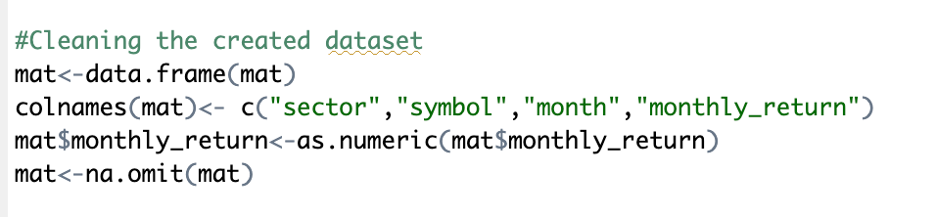


1. We now calculate the monthly return for each stock

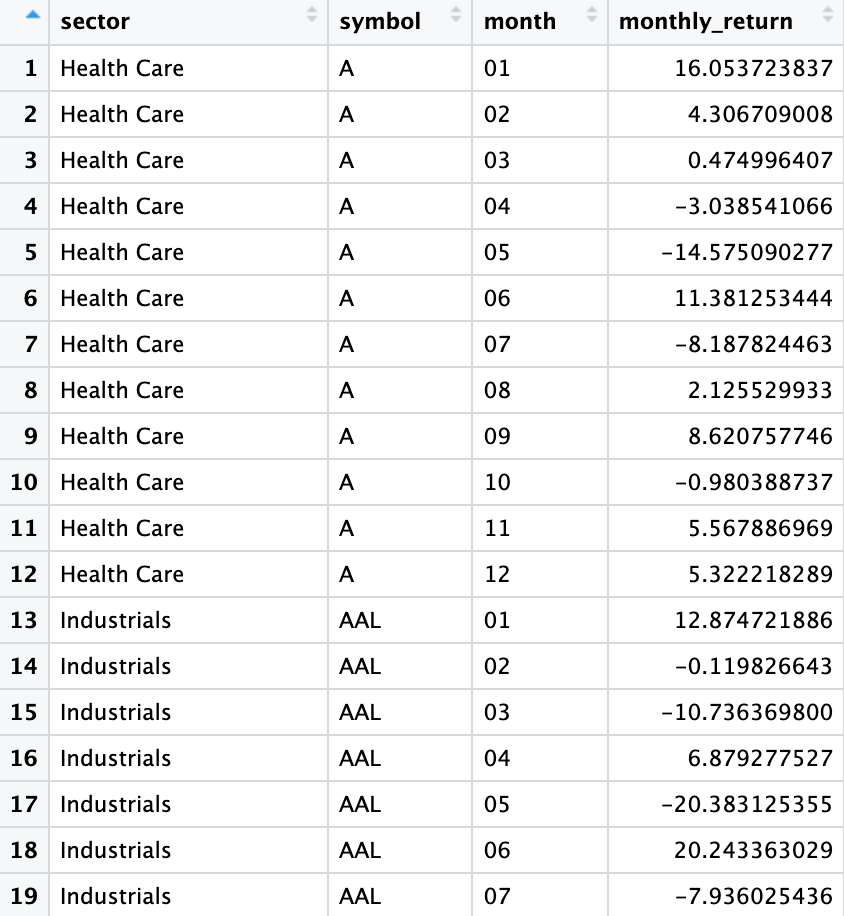
Here we loop through each stock and for each month we calculate the monthly return, storing the value all along.



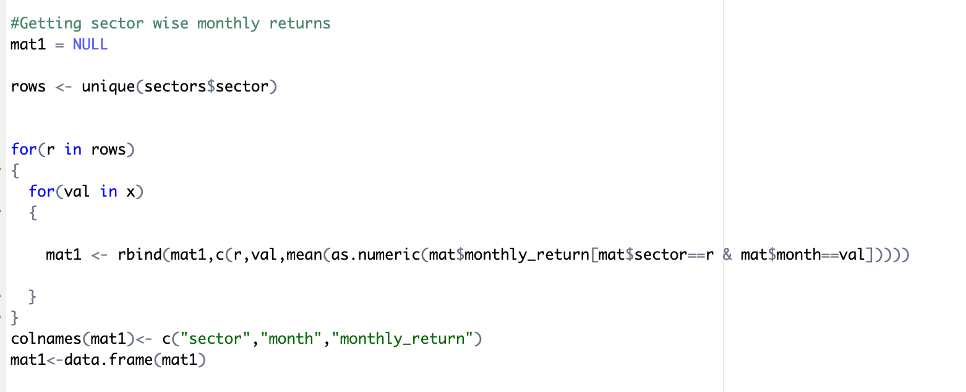
We then clean our newly created dataset by naming columns and omitting N/A values



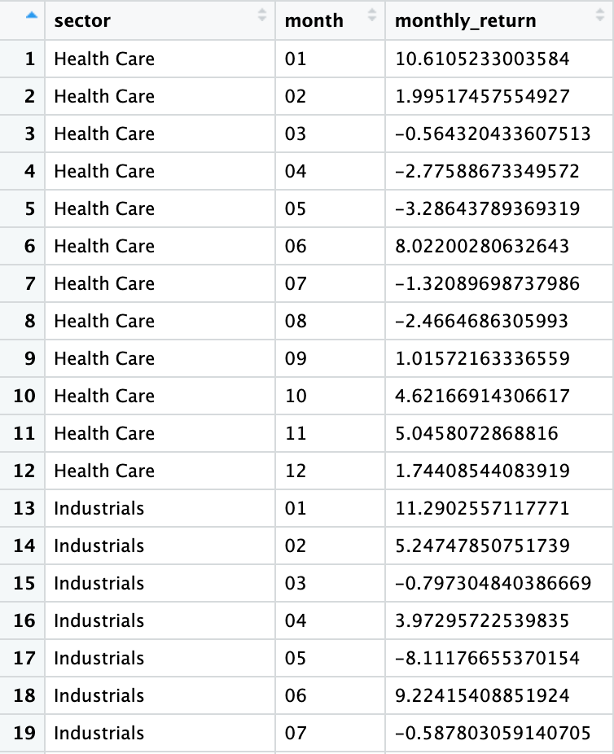
Here the dataset looks like following image:



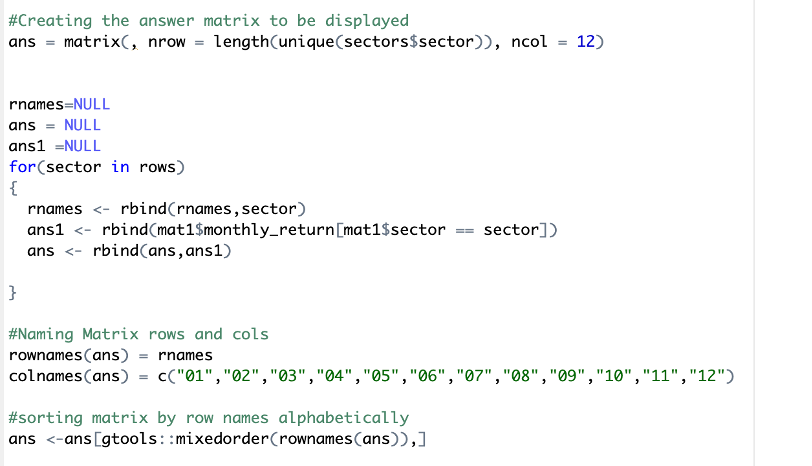
1. Calculate the mean return for each sector for each month



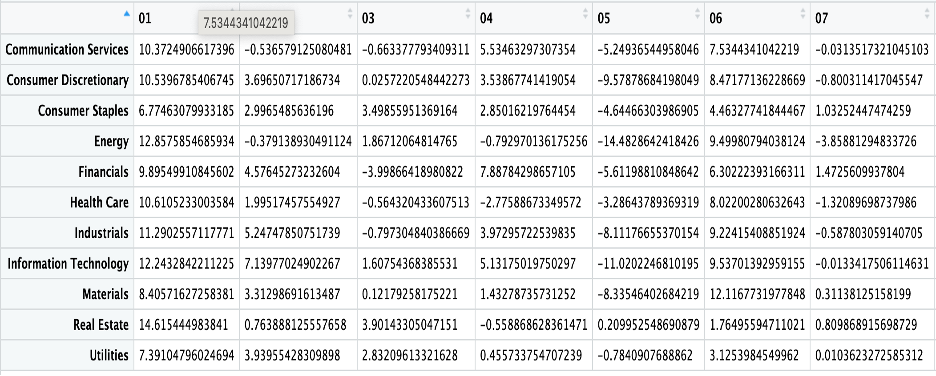
Here we are calculating the mean return for each sector for every month by filtering the monthly return by sector and month value.



1. Format the data to create the matrix( as asked)



Here we form a matrix with sectors as row names and months as col names and then we sort the row names to get the order alphabetically.

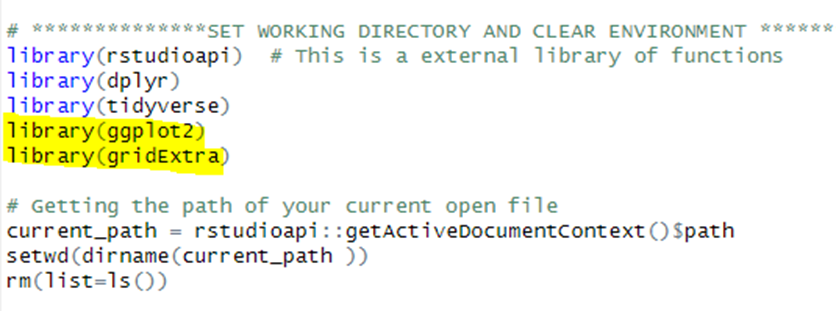


#### ***Problem 4***

For the 4th question, we are required to evaluate daily returns for Apple (Symbol: AAPL) stock for the year 2019. The daily returns are then plotted on a graph along with the cumulative return for the stock. Further, a new quantity – Maximum cumulative return is computed and plotted in the same graph to show what the maximum cumulative return for Apple is over different times in the year.

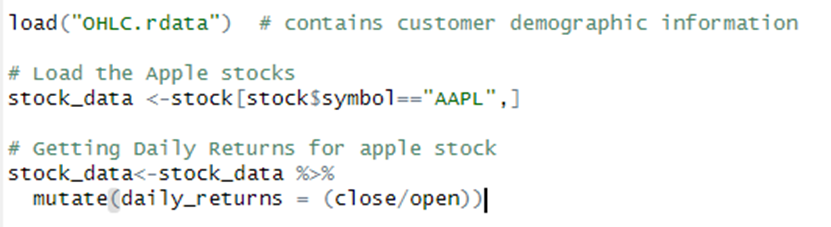
In order to plot this information in a graph, it is required to format the data to simplify the job of plotting it on a graph. The very first step is to load all the required libraries, set the working directory, and clear the working environment, similar to the previous solutions.

Here, we install and load two new libraries namely – ggplot2 and gridextra. These libraries assist us with plotting returns over the full year.

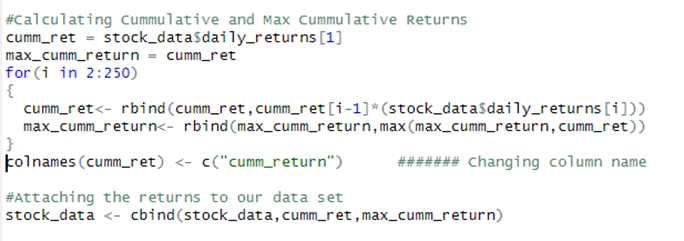


We then move forward to load OHLC data and pick only the rows pertaining to Apple by filtering on symbol = AAPL as shown below. We then calculate daily returns using the formula –

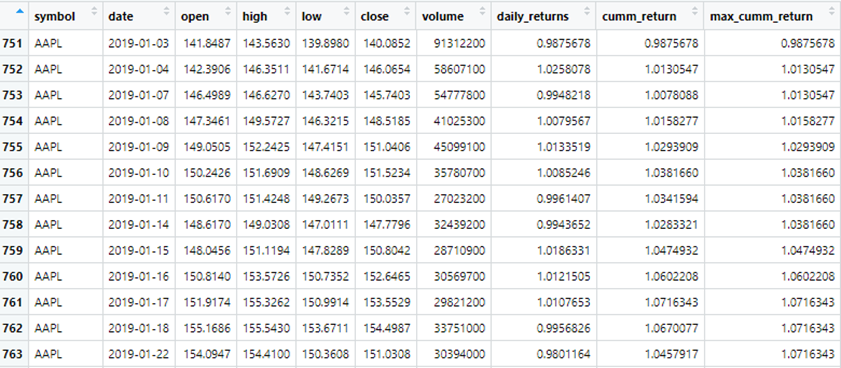
Daily return = (closing price of the day - opening price of the day)/opening price of the day



In order to calculate the cumulative return and maximum cumulative return, we make use of a for loop to make use of daily returns over the 250 trading days and determine both values. For cumulative return, we multiply the return of the current day by the cumulative return of the previous day. Maximum cumulative return is determined by assigning the maximum value of cumulative return at a given point and is only updated if the new cumulative return exceeds the maximum cumulative return from the previous day. The logic to obtain these values is shown below along with the respective values for the first 10 days.

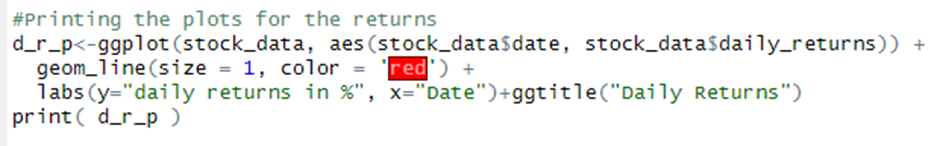


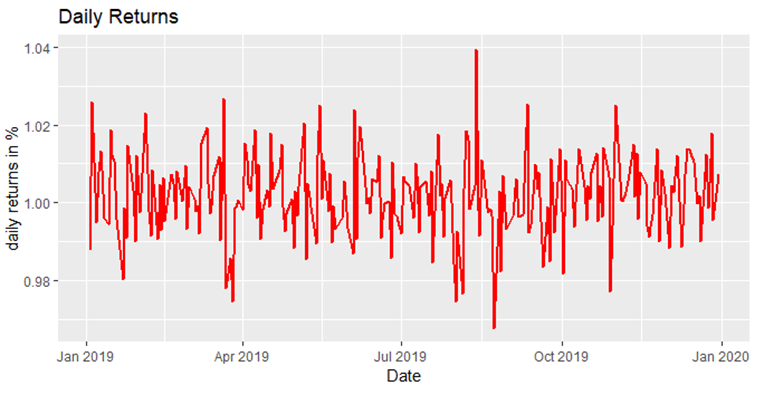
Cumulative and maximum cumulative return values for the first 10 days –



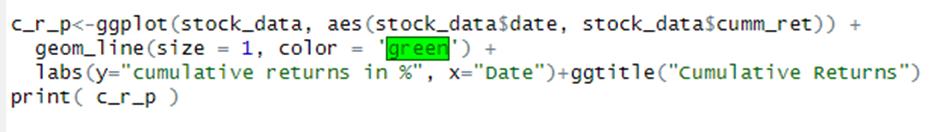
The next step is to move forward with plotting these values on different graphs and then combining all 3 plots into one. In order to plot the values, we use the ggplot() function where we specify the x-axis and y-axis values. In addition to this, we also specify the colour of the graph, the type of plot, the title, and the labels for each axis. The same is done for all 3 graphs before integrating the plots into one graph. The code and plot for each value – daily return, cumulative return, and maximum cumulative return are given below one after the other.

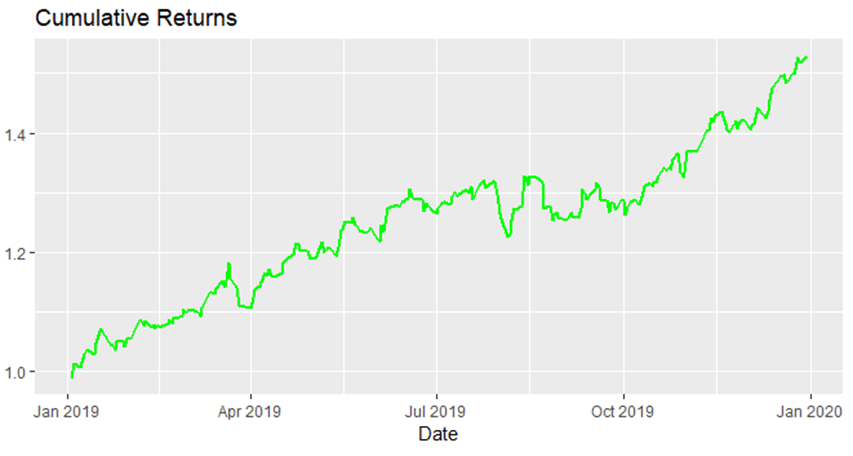
Daily Return -



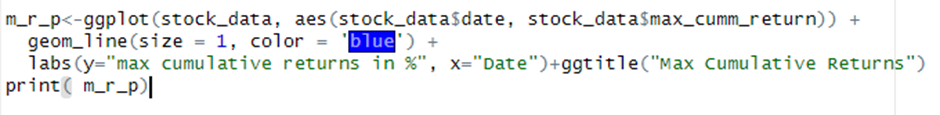


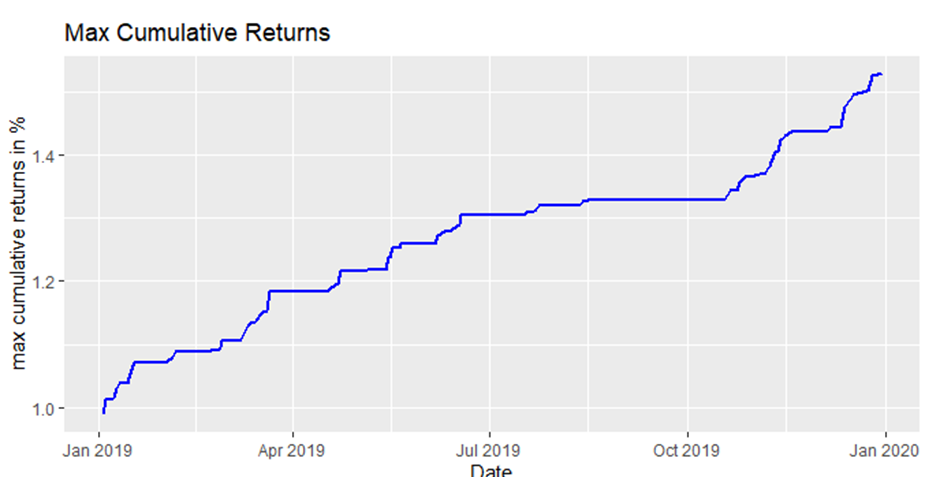
Cumulative Return -





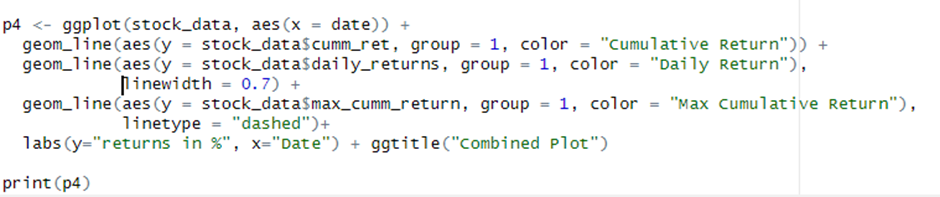
Maximum Cumulative Return –

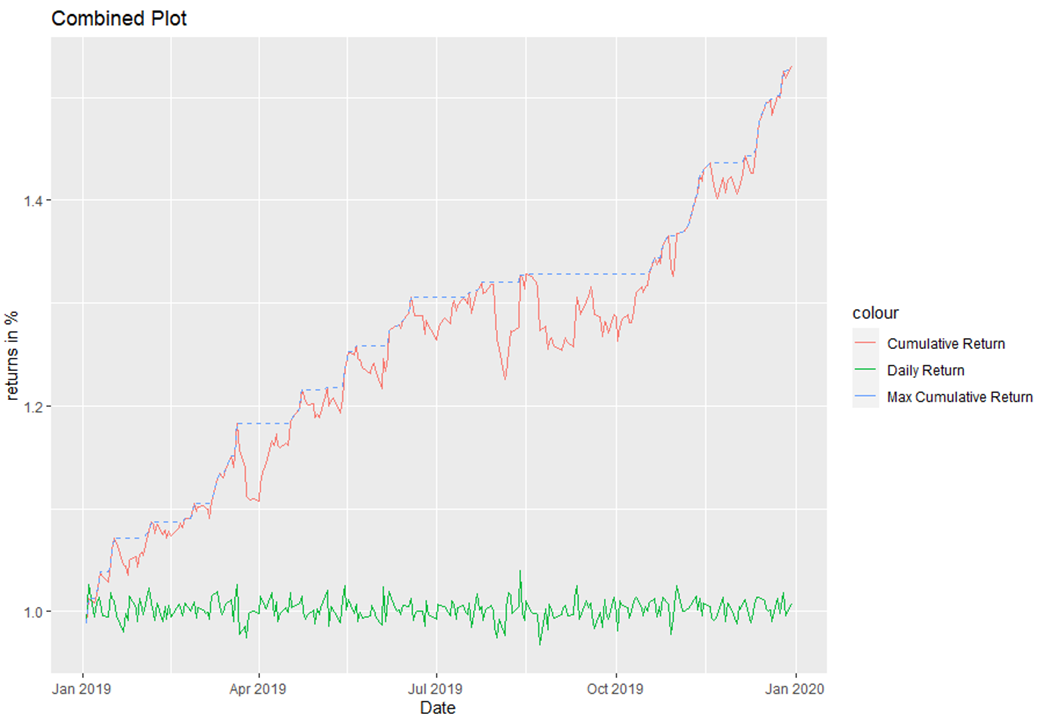




Combined plot –

As can be seen below, we use different line types and line widths along with colors in order to differentiate between each plot.





**INSIGHT FROM THE PLOT:**

Cumulative maximum return is a metric that measures the largest overall gain that an investment portfolio has achieved over time. It is calculated by comparing the current value of the portfolio to its maximum value at any point in the past.

This information is useful in evaluating a trading strategy because it gives insight into the risk-reward profile of the strategy. A high cumulative maximum return indicates that the strategy has generated large returns in the past, which is generally a positive sign. However, if the cumulative maximum return is achieved through high-risk investments, it could also mean that the strategy is subject to large losses if the market turns.

By analyzing cumulative maximum return, investors can determine if a trading strategy has a track record of consistent gains or if its returns are volatile. They can also assess if the strategy's returns are sustainable in the long term and if it is suitable for their investment goals.