4. Basic Visualisation and Exploratory Data Analysis

Ali Ataullah and Hang Le

01/04/2020

Contents

1 Introduction

After preparing data for analysis, the first step is usually an "Exploratory Data Analysis" (EDA). This is the stage where the analyst summarises and/or visualises the data in order to assess the integrity of data (e.g. detect outliers or errors) and/or to generate interesting research questions. For details, see (Chihara and Hesterberg, 2018, chapter 2) and (Ruppert and Matteson, 2015, chapter 4). There are several excellent `R` packages for data visualisation. We will work with the `ggplot2` package, which is a part of the `tidyverse`.

2 Intended Learning Outcomes

By the end of this session, students should be able to

1. plot graphs of functions,

2. produce descriptive statistics (e.g. mean and standard deviation),and

3. produce and label basic plots (e.g. scatterplots and histograms).

3 Template of a ggplot Code

Wickham and Grolemund (2016) summarise that all ggplot2 graphics are generated the following coding template:

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(
     mapping = aes(<MAPPINGS>),
     stat = <STAT>,
     position = <POSITION>
  ) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION>
```

The above is available here. The examples and exercises covered in this session will clarify the above template. We start by loading the tidyverse package.

```r
# Load the package `tidyverse` that includes ggplot2
library(tidyverse)
```

4 Plotting Functions

4.1 Net Returns vs Log Returns

In financial analysis, we regularly work with return on assets. Suppose the price of an asset at time $t-1$ was P_{t-1} and the price today at time t is $P-t$. The **net return** on this asset for the given time interval is

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} = \frac{P_t}{P_{t-1}} - 1 \tag{1}$$

The **log return** is

$$log(1 + R_t) = log(\frac{P_t}{P_{t-1}}) \tag{2}$$

We will now show that when net return R_t is close to 0, then

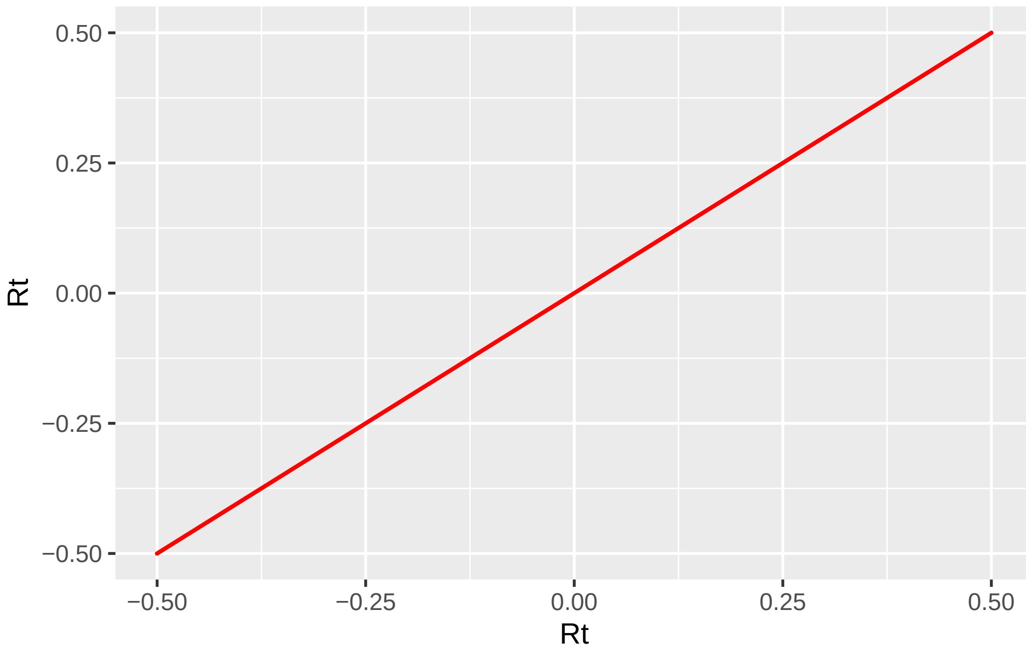
$$log(1 + R_t) \approx R_t \tag{3}$$

The first step in creating a `ggplot2` plot is to have a dataframe. Let us create a dataframe in order to check whether $log(1 + R_t) \approx R_t$ when R_t is close to 0. We use the `tibble()` function to create a dataframe. It is sufficient for us to remember that `tibble` is a special kind of dataframe. We will use the `seq()` function to create a sequence of real numbers from -0.5 to 0.5.

```
# Create a column Rt u, which is net return from -0.5 to 0.5.

# We have 500 observations of net returns

df1 <- tibble(Rt = seq(from = -0.5, to = 0.5, length.out = 500))

# Use mutate() to create logRt = log(1+Rt)

df1 <- df1 %>%
  mutate(logRt  = log(1+Rt))
```

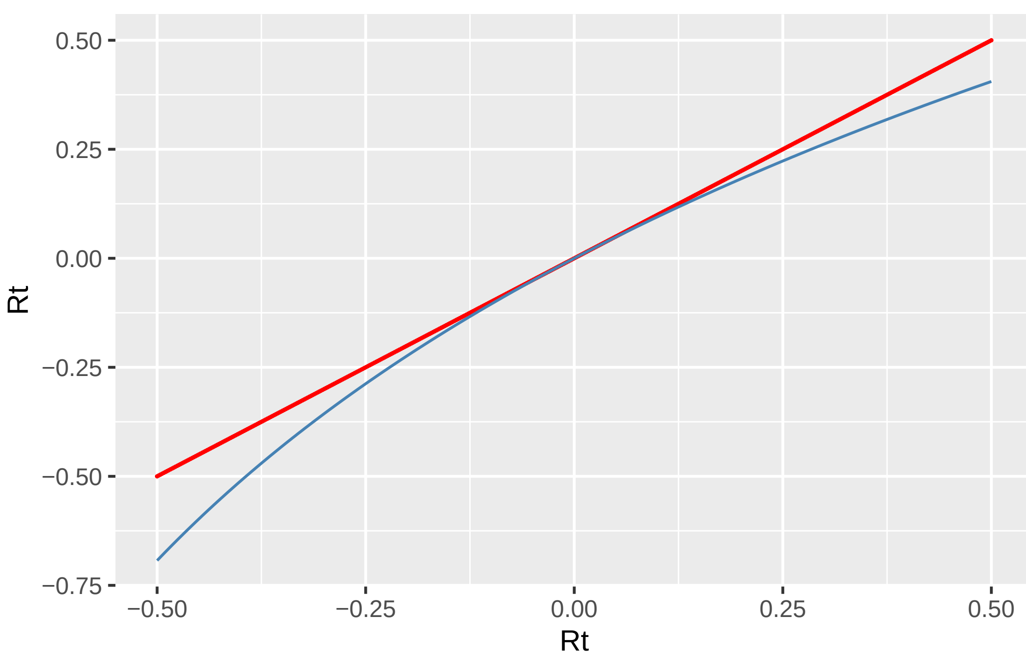
We now use `ggplot()` function to plot the

```
fig1 <- ggplot(data = df1) +
  geom_point(aes(x = Rt,y = Rt), size = 0.1, color = "red")

fig1
```



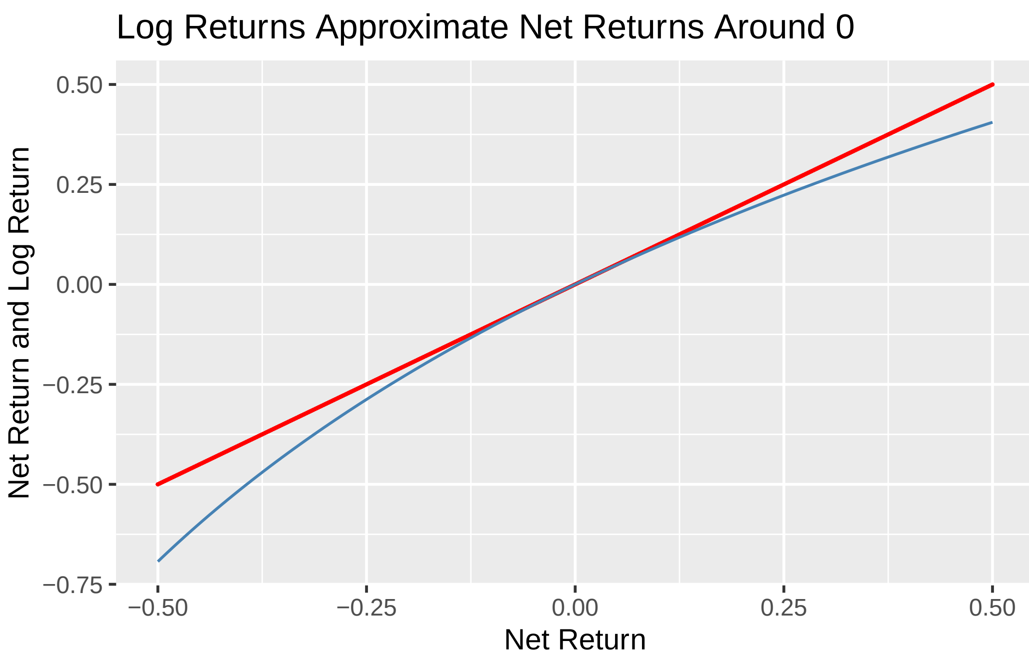
We now **add** another layer $log(1 + R_t)$ to the object `fig1`.

```
fig1 <- fig1 +
  geom_line(aes(x = Rt,y = logRt), size = 0.5, color = "steelblue")



fig1
```



The above plot is OK but the label for y-axis is not correct because we show both R_t and $log(1 + R_t)$. So, we add another layer of labels to the above plot as follows. We also add a title.

```
fig1 <- fig1 +
  labs(title = "Log Returns Approximate Net Returns Around 0",
       y = "Net Return and Log Return",
       x = "Net Return")



fig1
```



4.2 Exercise

Financial theory builds models of investors' demand for risky assets. An important element of the financial theory is the way it captures investors' attitude towards risk. Many models in finance begin with the assumption that investors are **risk averse**. To understand the basic idea, consider the following game. Suppose you are offered a lottery that pays \$100 with probability 0.5 and 0 with probability 0.5. On average, the lottery pays \$50. Will you pay \$50 to play this lottery?

It is very likely that you will pay less than \$50 to play this lottery. How much less depends on your degree of risk aversion. Investors with high risk aversion would pay less (high risk premium) than investors with low risk aversion (low risk premium). In finance , we assume that investors' risk attitude can be captured in their **utility function**. One class of utility function represents **Constant Relative Risk Aversion (CRRA)**. Investors with CRRA utility function invest a fixed percentage of their wealth in risky assets.

Let W denote the wealth of an investor. A widely used CRRA utility function in finance is

$$U(W) = \frac{W^{1-\lambda}}{1-\lambda} \quad , \lambda > 0 \ and \ \lambda \neq 1 \tag{4}$$

For this exercise, suppose there are two investors: Jack and Jill. For Jack, $\lambda_{jack} = 0.5$. For Jill, $\lambda_{jill} = 0.7$. We seek to plot utility of the two investors for wealth $W \in [0, 20]$. Both investors have the CRRA utility function shown in Equation (4). Draw a plot that shows utility functions for Jack and Jill. Comment on the difference between Jack's and Jill's utility function.

```
# Your code - Carefully look at the code below

# You will need to remove # and modify "???"

# Hints below

# Create a tibble

# df2 <- tibble(W = seq(???, ???, length.out = 500))

# Create new columns, for Jack's and Jill's utility function


# df2 <- df2 %>%
#     mutate(Jack = W^(1 - ???)/(1 - ???),
#     Jill = W^(1 - ???)/(1 - ???))

# Create ggplot object

# ggplot(df2) +
```

```r
#   geom_point(aes(x = W, y = ???), color = "red", size = 0.1) +
#   geom_point(aes(x = W, y = ???), color = "steelblue", size = 0.1)
```

We now clear our `R` environment.

```r
rm(list = ls())
```

5 Descriptive Statistics

This section uses a dataset containing hypothetical financial data for 100 firms. Data contains 4 variables: Sales,CAPX (Capital Expenditures), DebtEquity (Debt to Equity Ratio), and Industry that the firm belongs to. We start by loading the data. Sales and CAPX are in millions of Pounds.

```r
financialData <- read_csv("financialData.csv")
```

```
## Parsed with column specification:
## cols(
##   Sales = col_double(),
##   CAPX = col_double(),
##   DebtEquity = col_double(),
##   Industry = col_character()
## )
```

5.1 Centre, Spread and Shape of Data

Two important measures of the centre of the distribution of a variable are *mean* and *median*. In `R`, mean is obtained using `mean()` and median is computed using `median()`. The spread of the distribution of a variable is usually summarised with the *standard deviation*, which is the average squared deviation from the mean. In `R`, standard deviation is computed with `sd()` function. The shape of the distribution of a variable is usually summarised by *skewness* and *kurtosis*. Skewness determines whether the distribution is symmetrical or not, while kurtosis captures whether the distribution has heavy tails. We create a new data frame that contains summary statistics for the `Sales` variable. We use the `moments` package for skewness and kurtosis.

```r
library(moments)

df_summary <- financialData %>%
  summarise(mean_Sales = mean(Sales),
            med_Sales = median(Sales),
            sd_Sales = sd(Sales),
            skew_Sales = skewness(Sales),
            kurt_Sales = kurtosis(Sales))
```

5.2 Exercise

Follow the above code and create a data frame that contains summary statistics for Debt to Equity ratio in `financialData`.

```r
# Your code
```

5.3 Summarising Multiple Columns

It is not ideal to summarise variables one by one. There are several variations of the `summarise()` function. For example, we can use the `summarise_at()` to summarise several variables as follows. We will apply three functions `mean()`, `median()` and `sd()` to three variables below.

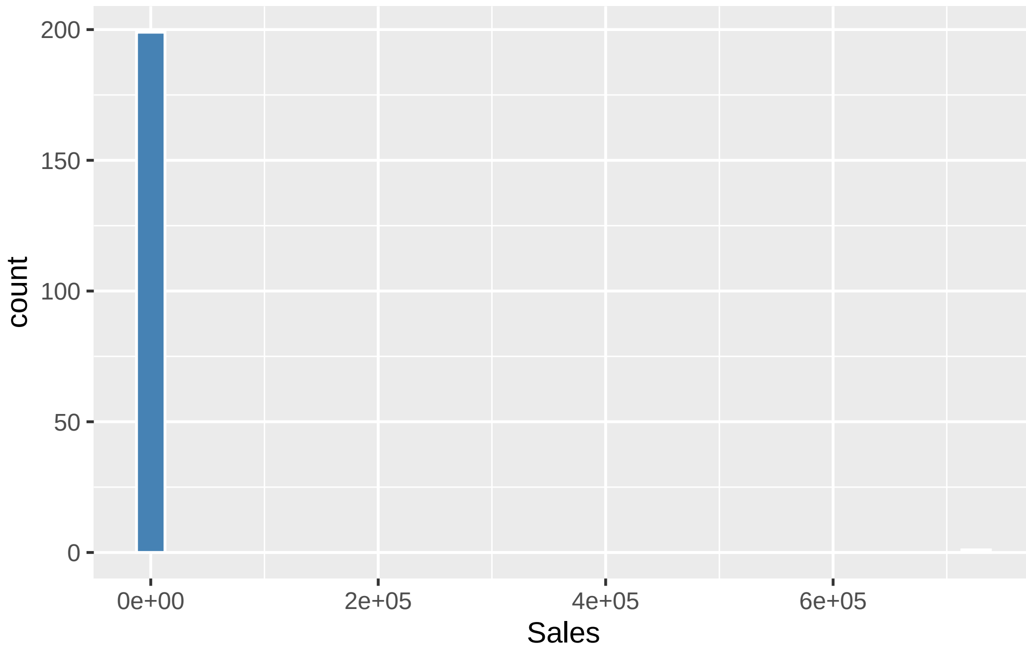
```r
df_summary <- financialData %>%
  summarise_at(c("Sales", "DebtEquity", "CAPX"),
               list(mean, median,sd), na.rm = TRUE)
```

6 Basic Plots

6.1 Histogram

Histogram is a widely used device to visualise the variations in observed values of a variable. To draw a histogram, we divide the x-axis into equally spaced intervals (called bins) and then show the frequency of observations falling in each bin as bars. We plot the histogram for `Sales` below and leave histogram of pother variables for practice.

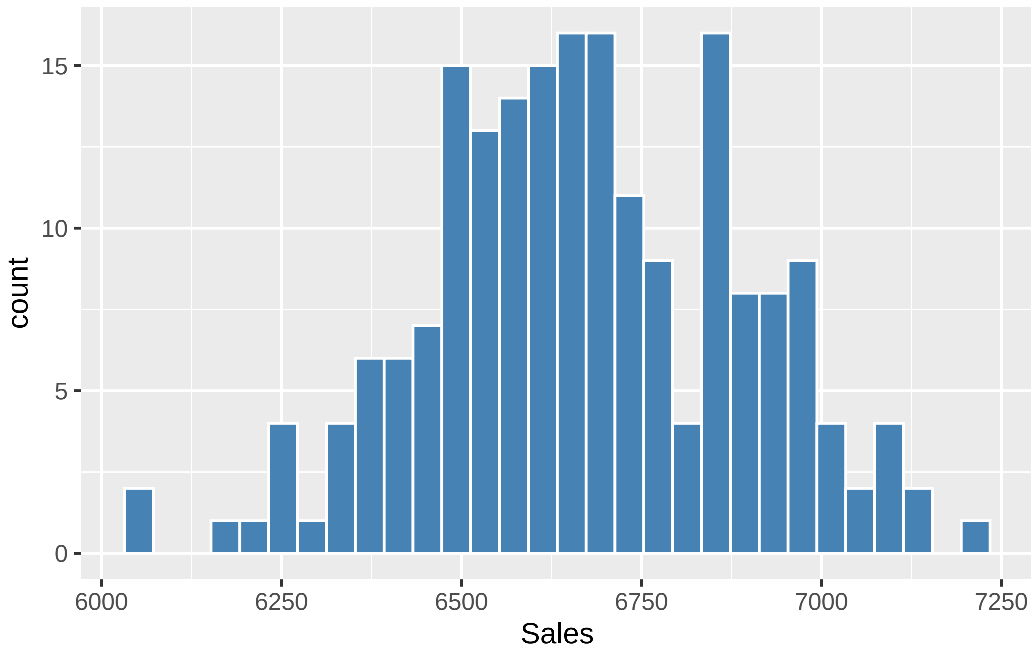
```
ggplot(financialData) +
  geom_histogram(aes(Sales), bins = 30, color = "white", fill = "steelblue")
```



The above histogram suggest that there seems to be a problem with our data. If you go back to your summary statistics, you will notice a big difference between the mean and the median values of Sales. This is because there is one extremely large value of Sales (over 42,000). This seems to be an **outlier** (i.e. an exceptionally large value compare to other observations). We can use the `filter()` function to remove this from our data and then plot the histogram again.

```
# Keep observations where Sales < 1000 million.
financialData <- financialData %>%
  filter(Sales <= 9000)
```

```
ggplot(financialData) +
  geom_histogram(aes(Sales), bins = 30, color = "white", fill = "steelblue")
```



6.2 Exercise

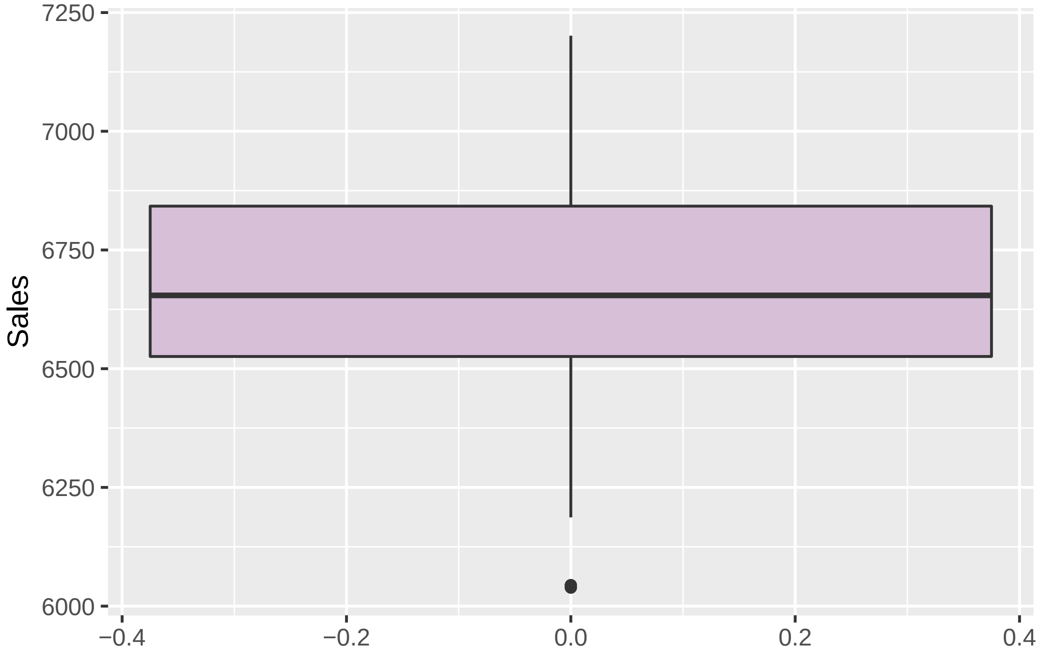
Plot histograms `CAPX` and `DebtEquity` in `financialData`.

```
# Your code
```

6.3 Boxplot

Boxplot is a very useful tool to visualise the distribution and summary statistics of a variable. A boxplot shows the median, the first quartile, the third quartile and 1.5 times the interquartile range (IQR). The plot also shows possible outliers. Note that quartiles divide the data into four parts. Around 25% of the data is below the first quartile, around 50% of the data is below the second quartile (which is also the median), and around 75% of the data is below the third quartile. IQR is the difference between the third and the first quartile. A boxplot for `Sales` is given below.

```
box_sales <- ggplot(financialData) +
  geom_boxplot(aes(Sales), fill = "thistle") +
  coord_flip()
box_sales
```



6.4 Exercise

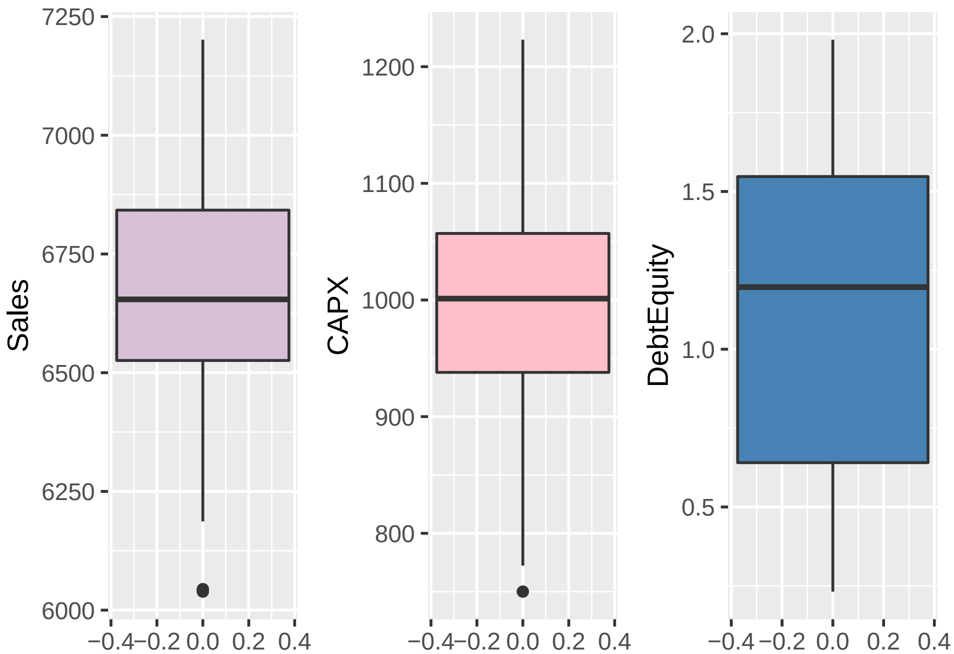
Add boxplots for `CAPX` (call it `box_capx`) and `DebtEquity`(call it `box_de`).

```
# Your code
```

We can put the three boxplots together using the `gridExtra()` package (you will need to install and load this).

```
library(gridExtra)
```

```
grid.arrange(box_sales, box_capx, box_de, nrow = 1)
```



We now clear our `R` environment.

```
rm(list = ls())
```

7 Plots of Stock Prices

7.1 The Evolution of Asset Prices

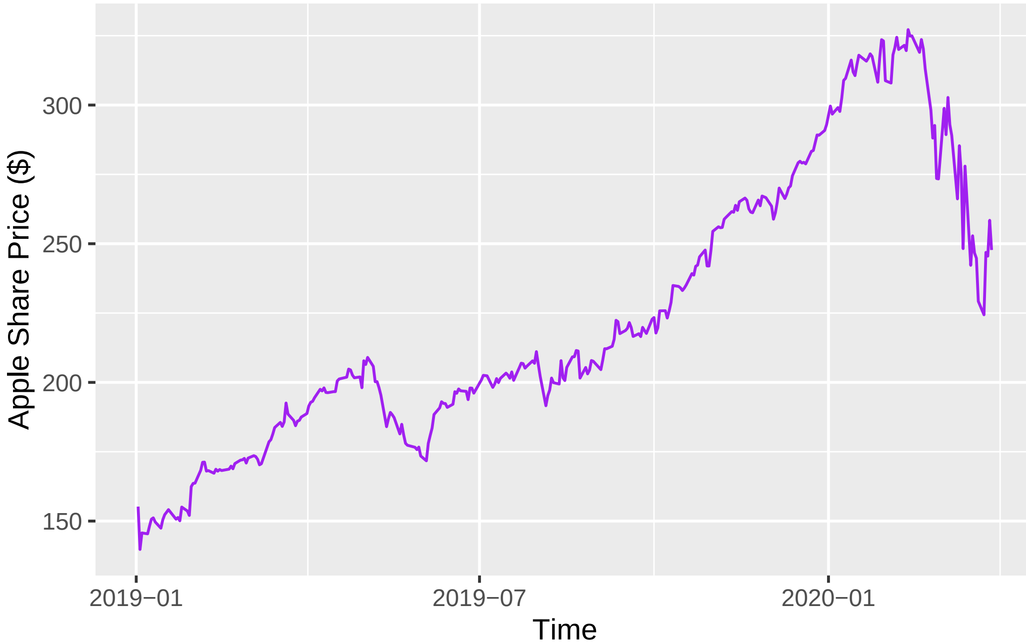
This section will use a data set that contains stock prices of four companies: Apple (AAPL), Amazon (AMZN), Netflix (NFLX) and Microsoft (MSFT). The data contains daily closing prices from 2000-12-01 to 2020-03-27.

```
df_stocks <- read_csv("assets.csv")
```

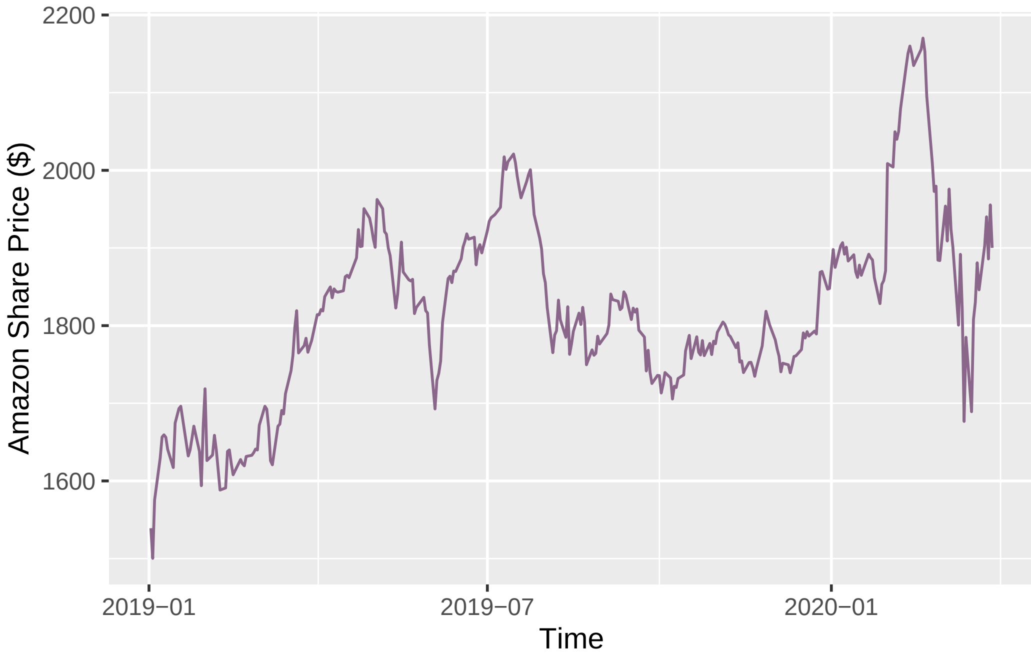
Let us plot the four prices for the period 2019-01-01 to 2020-03-27 to see how stock prices evolved during the period. We create plots for AAPL and AMZN. Plots for NFLX and MSFT are left for you to practice.

```
# Note how we use the filter() to obtain the relevant time period.
fig_APPL <- ggplot( data = filter(df_stocks, date >= "2019-01-01")) +
  geom_line(aes(x = date, y = AAPL), color = "purple") +
  labs(x = "Time",
       y = "Apple Share Price ($)")
```

```r
fig_APPL
```




```r
fig_AMZN <- ggplot( data = filter(df_stocks, date >= "2019-01-01")) +
  geom_line(aes(x = date, y = AMZN), color = "plum4") +
  labs(x = "Time",
       y = "Amazon Share Price ($)")

fig_AMZN
```



7.2 Exercise

Use `df_stocks` to plot the share prices of Netflix (call it `fig_NFLX`) and Microsoft (call it `fig_MSFT`) for the period 2019-01-01 to 2020-03-27.

```
# Your code
# There are many colours avaiable.
# You may choose from the following colours:
# "thistle4", bisque4", "red", "green", "lightsalmon", "steelblue".
```

Now, we use the `grid.arrange()` to put the four figures together as follows.

```
grid.arrange(fig_AMZN,fig_APPL,fig_MSFT,fig_NFLX, nrow = 2)
```

8 Plots of Returns

8.1 Data in Long and Wide Format

For this section, we create a new dataframe that contains returns for the four stocks. Note that we have four stocks. We could compute return for each assets by applying `mutate()` to each price column. This is fine for

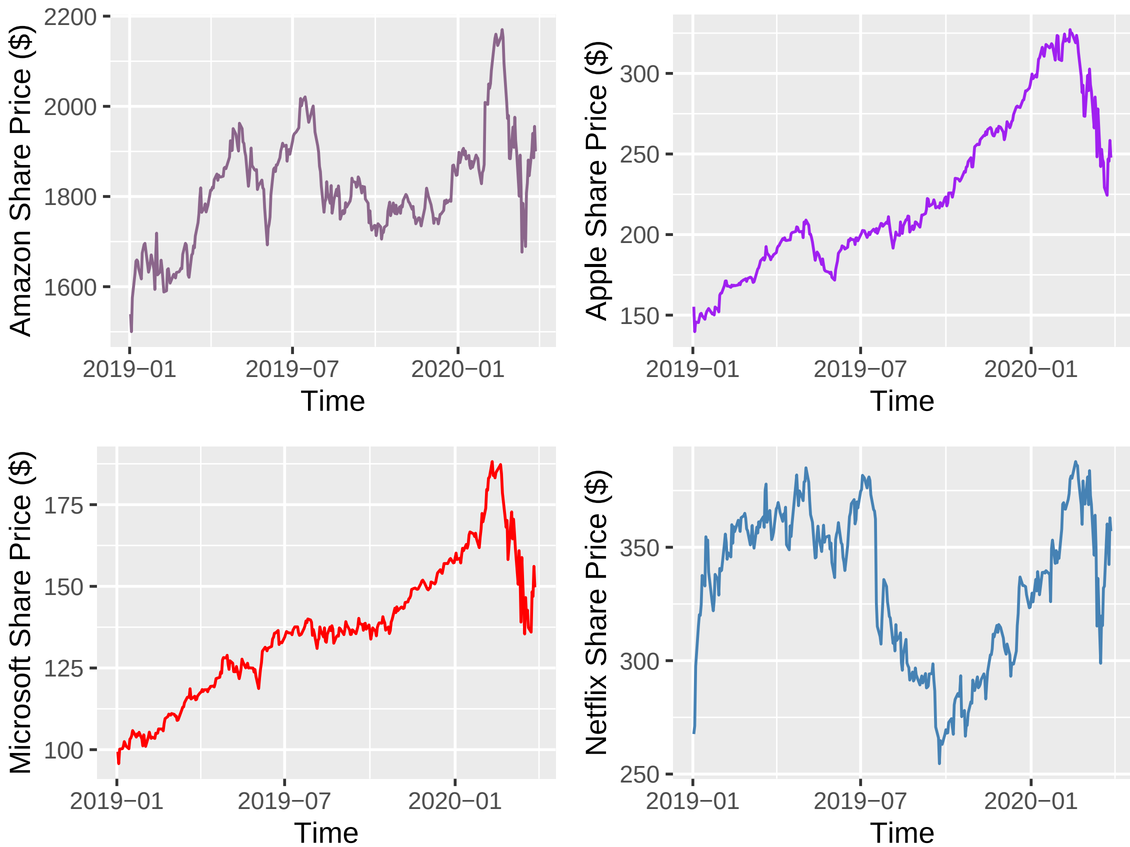


Figure 1: Prices of Four Stocks

four assets. But if we have many assets, then such repetitive computation would be very time consuming. We highly recommend learning the `purrr` package to perform repetitive tasks.

With experience, you would be able to write your own functions and use tools that will enable you to perform repetitive task efficiently. For our session, we use the `gather()` function to facilitate the computation of log returns for four assets. The `gather()` function transform our data from `wide` to `long` format. We then use `group_by()` and `mutate()` to compute returns. We explain our procedure with a simple data containing prices of two assets. You will practice with `df_stocks` in the exercise.

8.1.1 Step 1: Create a Tibble

```
simple_df <- tibble(date = c("2020-01-01", "2020-01-02", "2020-01-03", "2020-01-04"),
                    Asset1 = c(3,4,3,5),
                    Asset2 = c(7,5,6,7))

simple_df
```

date	Asset1	Asset2
2020-01-01	3	7
2020-01-02	4	5
2020-01-03	3	6
2020-01-04	5	7

8.1.2 Step 2: Change to Long Format

```
simple_df <- simple_df %>%
  gather(Stocks, Price, -date)

simple_df
```

date	Stocks	Price
2020-01-01	Asset1	3
2020-01-02	Asset1	4
2020-01-03	Asset1	3
2020-01-04	Asset1	5
2020-01-01	Asset2	7
2020-01-02	Asset2	5
2020-01-03	Asset2	6
2020-01-04	Asset2	7

8.1.3 Step 3: Compute Return for Each Asset

```r
simple_df <- simple_df %>%
  group_by(Stocks) %>%
  mutate(Return = log(Price) - log(lag(Price)))
```

8.1.4 Step 4: Remove `NA`s

```r
simple_df <- simple_df %>%
  drop_na()
```

8.2 Exercise

Use the steps outlined above to compute returns for the four stocks in `df_stocks`.

```r
# Your code
# Hint: Incomplete code provided below.
# Replace ??? with appropriate code



# df_returns <- ??? %>%
#  gather(???, ???, -date) %>%
```

```
#   group_by(Stocks) %>%
#   mutate(Return = log(Price)-log(lag(Price))) %>%
#   select(date, ???, ???) %>%
#   drop_na()
```

9 Plot Returns Using Facets

Instead of creating a separate plot for each stock, we can use the `facet_wrap()` function in `ggplot` to draw separate plot for each of the four assets. We will use data in `df_returns`.

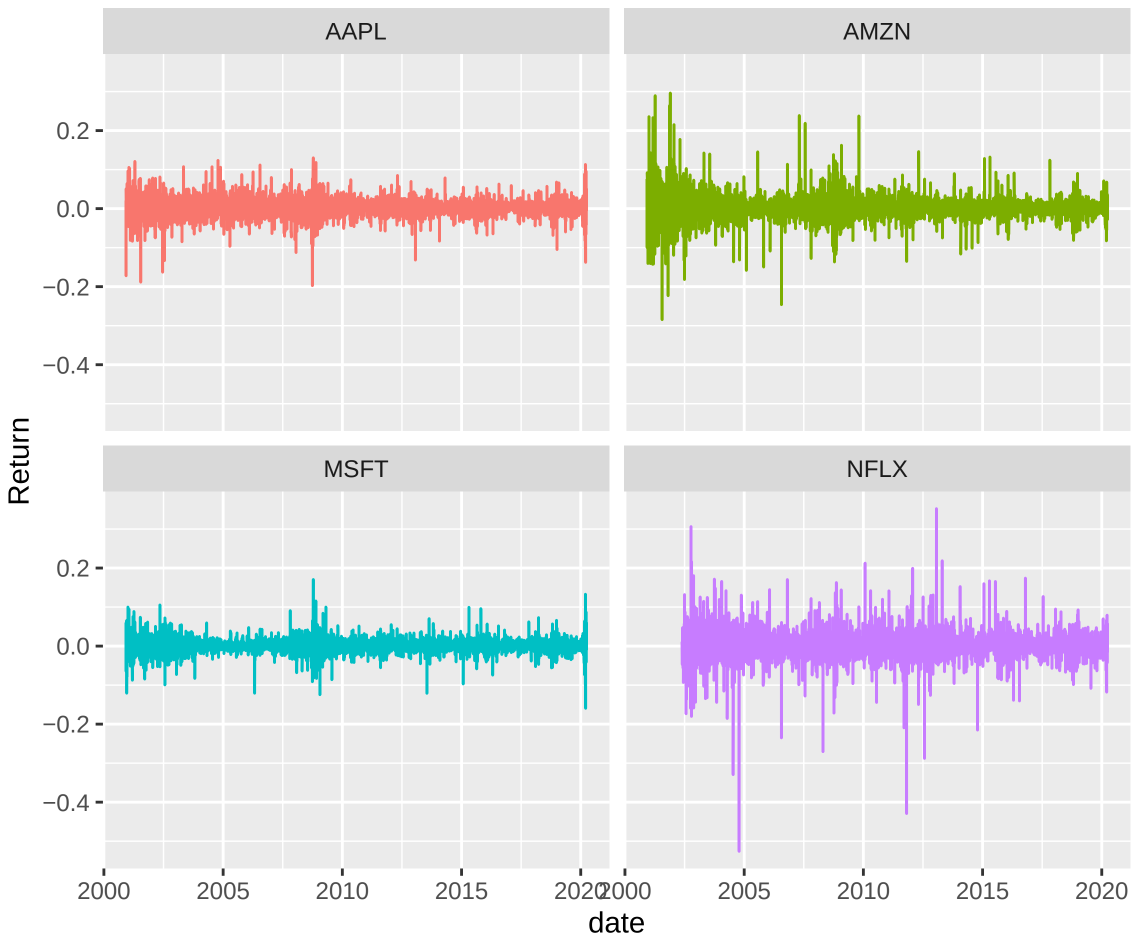


Figure 2: Returns for Four Assets

10 Density Plot of Returns

Earlier we used histograms to visualise the distribution of financial data. A better way to visualise distribution is to create a density plot. We do this for returns of four stocks below.

```
ggplot(df_returns) +
  geom_density(aes(x = Return,
                   group = Stocks,
                   color = Stocks) ) +
  facet_wrap(~Stocks, ncol = 2) +
  theme(legend.position = "none")
```

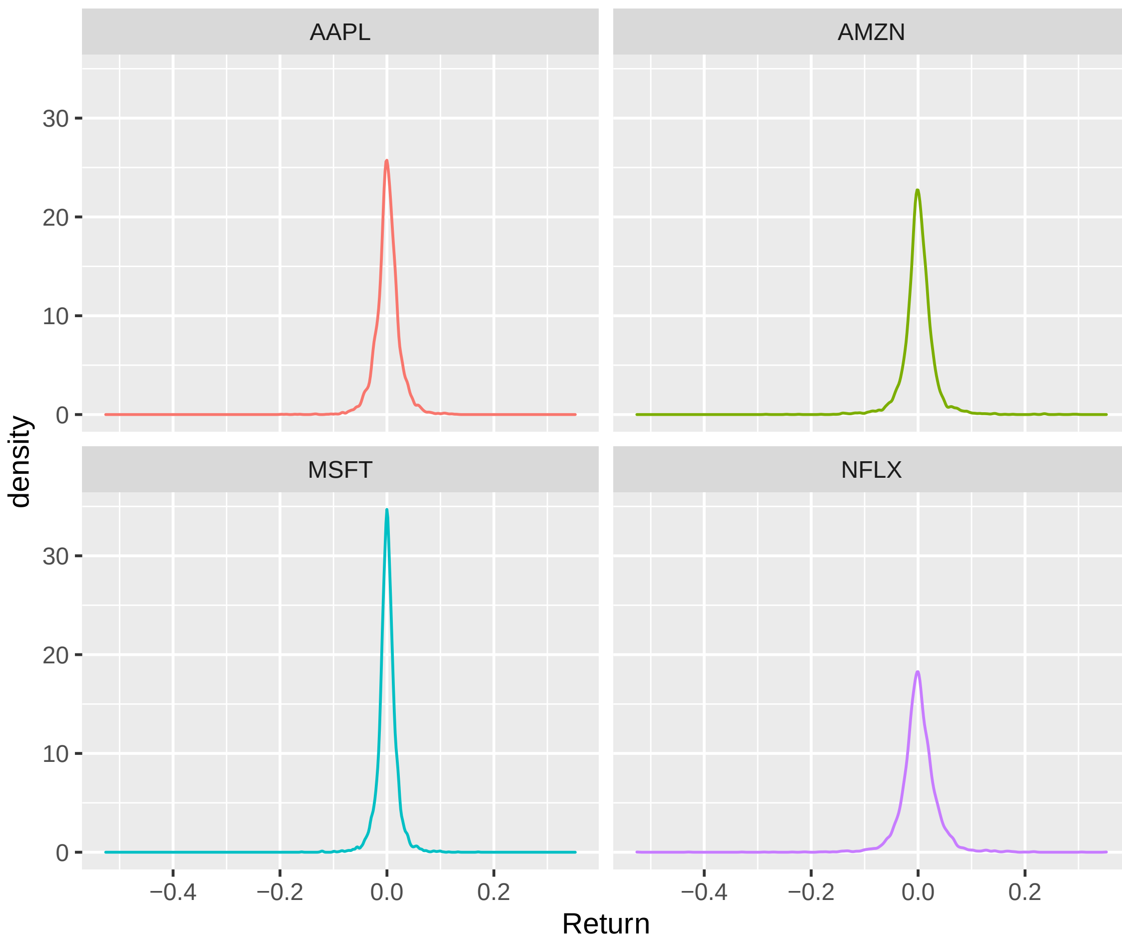


Figure 3: Density Plot for Returns

11 Scatterplot

11.1 Possible Relationship Between Two Variables

A **scatterplot** is a useful visualisation tool to explore possible relationship between two variables. We draw a scatterplot for returns for Apple and Amazon. You will draw scatterplot for Microsoft and Netflix.

However, we first need to change our data from `long` to `wide` format. This is done below using the `spread()` function.

```
df_returns <- df_returns %>%
  spread(Stocks, Return) %>%
  drop_na()
```

```
ggplot(df_returns) +
  geom_point(aes(x = AAPL, y = AMZN),
             color = "steelblue", alpha = 0.4) +
  labs(title = "Scatterplot for Apple & Amazon",
       x = "Return on Apple Stock",
       y = "Return on Amazon Stock")
```

11.2 Exercise

Use `df_stock` to draw a scatterplot for Netflix and Microsoft.

```
# Your code
```

12 Next Steps

This session provided a brief overview of some of the visualisation tools available in the `ggplot2` package. The plot considered in this sessions can be used for preliminary exploration of data before conducting financial analysis. The next session provides an overview of essential concepts from probability and mathematical statistics. These are frequently used not only in finance but in many other disciplines.

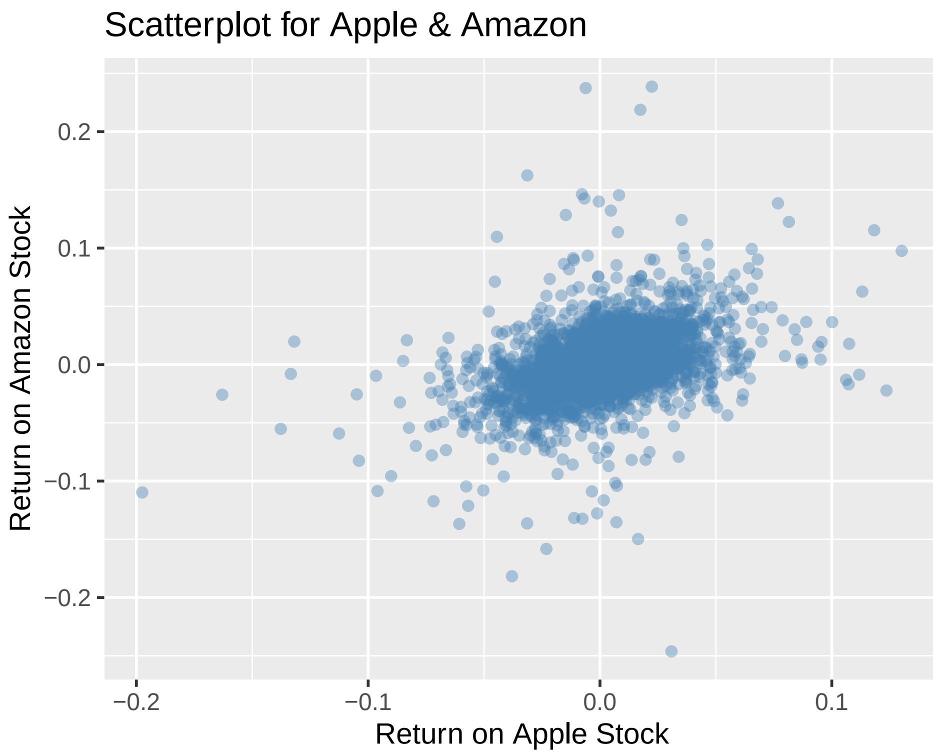


Figure 4: Scatterplot for Apple and Amazon

References

Chihara, L. and Hesterberg, T. (2018). *Mathematical Statistics with Resampling and R*. Wiley Online Library.

Ruppert, D. and Matteson, D. S. (2015). Statistics and data analysis for financial engineering with r examples.

Wickham, H. and Grolemund, G. (2016). *R for Data Dcience: Import, Tidy, Transform, Visualize, and Model Data*. O'Reilly Media, Inc.