INFO424 Assignment 1 Answers

Shangwen(Chris) Yang

2024-03-01

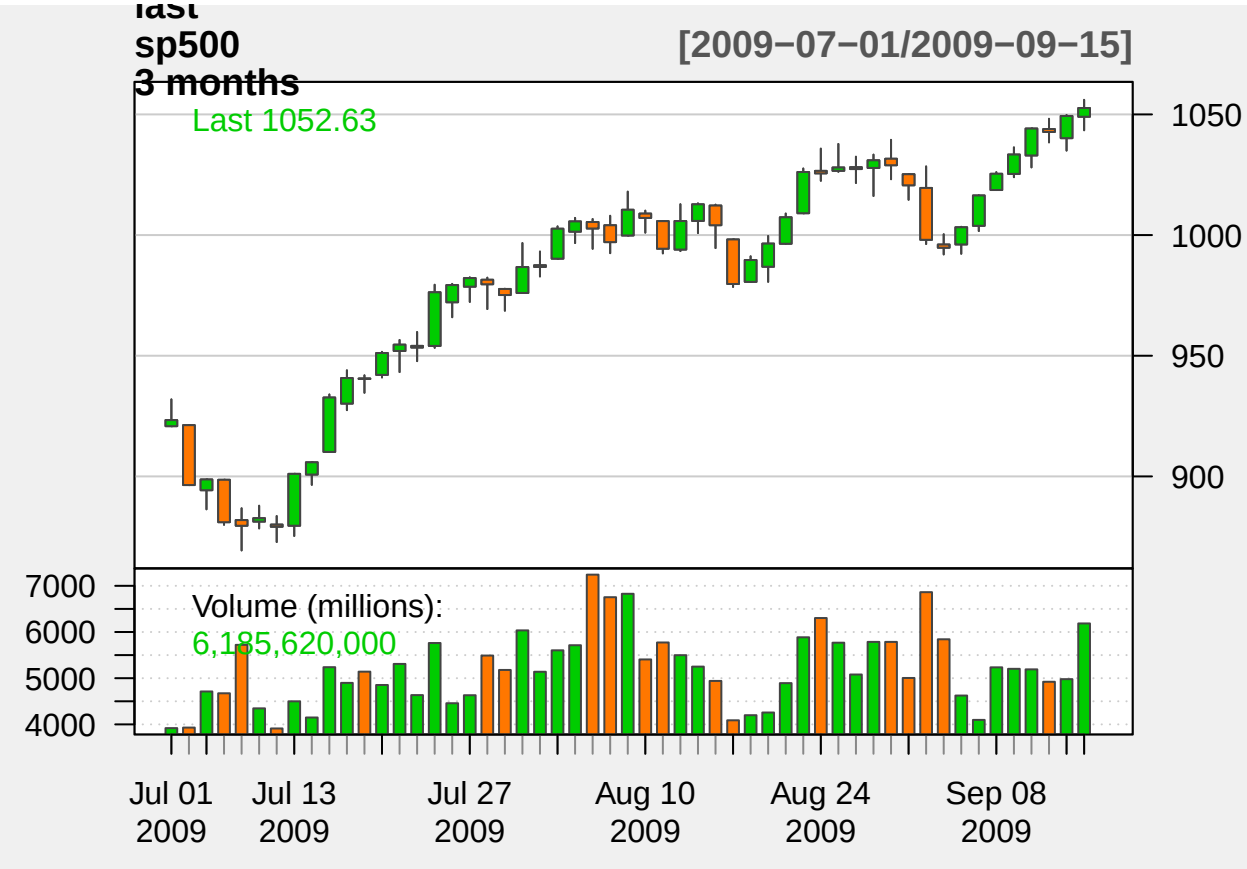
INTRODUCTION

This assignment will examine a range of visualisation and analysis methods for different data types. This will involve writing some R code to explore properties of different data sets and relevant discussions regarding the observed patterns and representation.

PART ONE: TIME SERIES DATA - THE SP500 INDEX (15 MARKS)

Run the script **sp500.R** that has been supplied. This will load some libraries that handle time series data and, in particular, tools for examining stock market data. A plot should be displayed that shows the last 3 months of the SP500 index data (July-Sept, 2009) as a candl plot. Note that the lower part of the plot shows the volume associated with trades.

```
# Run the sp500.R script
source("sp500.r")
```

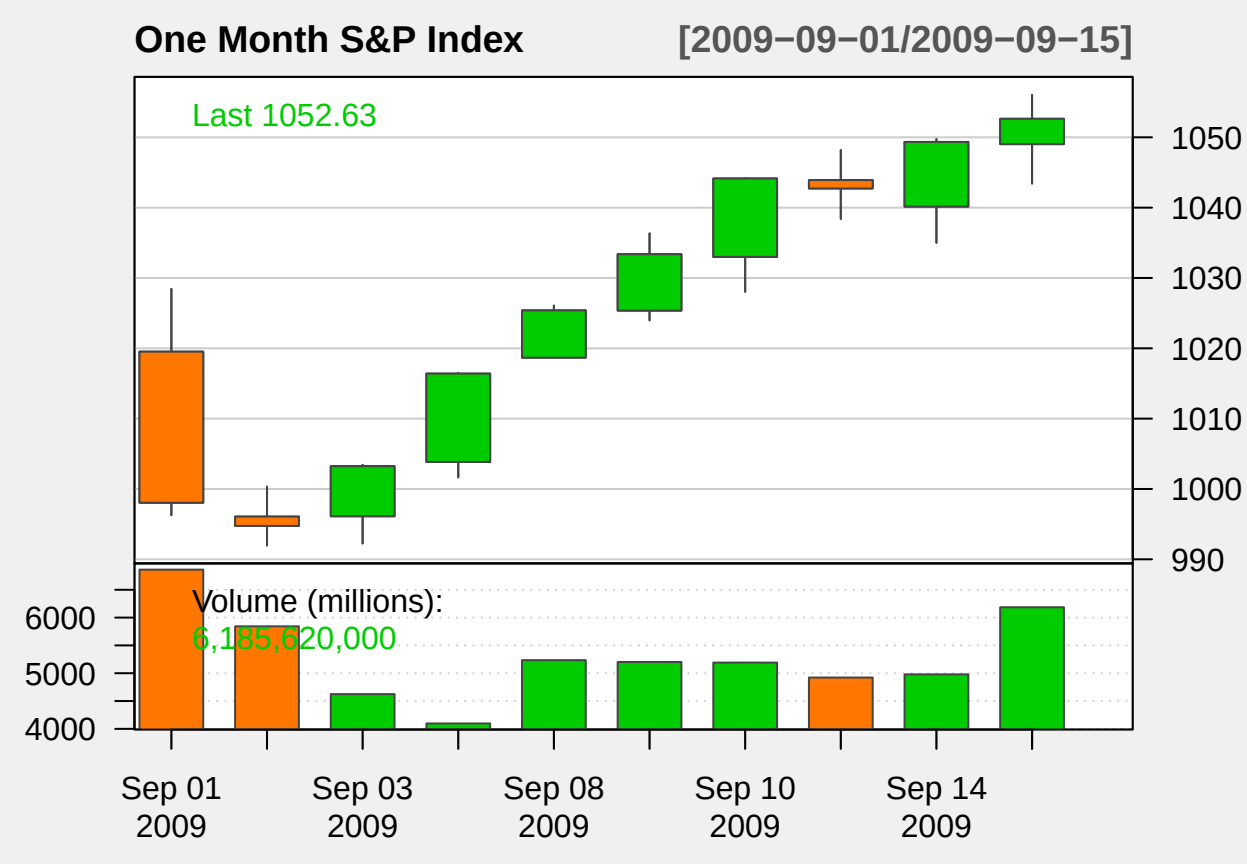


Question One*:* Examine the upper part of the plot and describe what the candles are showing(HINT: change the script to show just the last "1 months" and look at the data for the last "1 months").

Answer:

First, let's subset the last one month of transactions and plot to the candlestick chart

```
# subset the last one month of transactions
last_month <- last(sp500, "1 month")
chartSeries(last_month,
  type = "candlesticks", theme = "white",
  name = paste("One Month S&P Index")
)
```

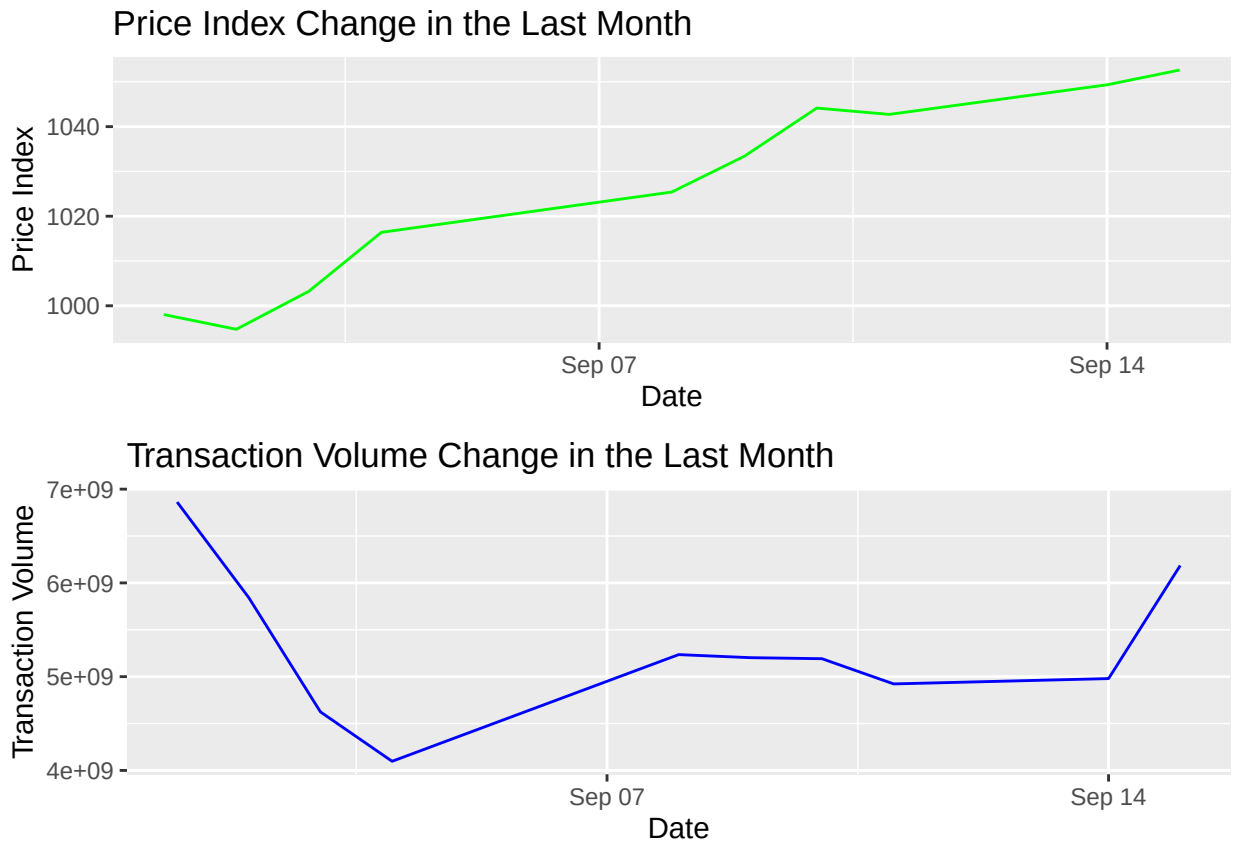


From the candle chart above, we can see that the stop print has a dramatic price drop on the 1st of September 2009, and on the same day it has a very high volume of transactions. This indicates there is a big sell in the market which bring down the stock share prices. However, from the second day onward, the stock prices were getting increased with relatively more stable (but less) volume of transaction. That indicate investors were positive on the market, which pushed the price go up again.

Let's create a line chart to compare the price index and its transaction volume.

```
library(ggplot2)
library("gridExtra")
# generate the price index line chart
plot_index <- ggplot(data = last_month, aes(x = index(last_month), y = AdjClose)) +
  geom_line(color = "green") +
  labs(
    y = "Price Index", x = "Date",
    title = "Price Index Change in the Last Month"
  )
# title("Price Index and Transaction Volume in the Last Month")

# generate the transaction volume line chart
plot_volume <- ggplot(data = last_month, aes(
  x = index(last_month),
  y = Volume
)) +
  geom_line(color = "blue") +
  labs(
    y = "Transaction Volume", x = "Date",
```

```
    title = "Transaction Volume Change in the Last Month"
  )
# title("Price Index and Transaction Volume in the Last Month")

# combine the two line chart
grid.arrange(plot_index, plot_volume, ncol = 1)
```



So we can see from the line chart which is shows the same pattern as the candlestick chart.

Question Two: The daily average price can be approximated by:

$$Av(t) = \frac{C(t) + H(t) + L(t)}{3}$$

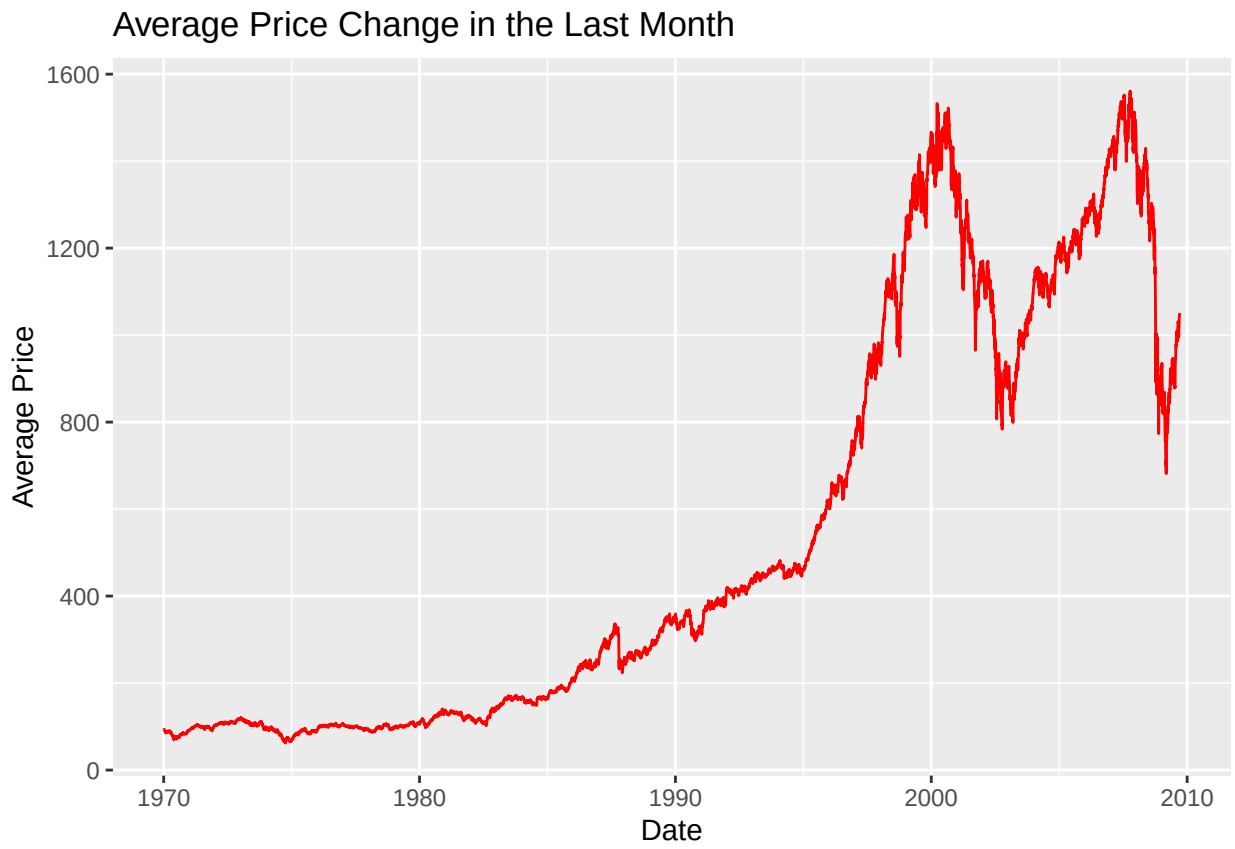
where C(t), H(t) and L(t) are the close, high and low quotes for day t respectively.

Calculate the average price Av(t) for the sp500 data and add this to the sp500 table. **Produce a plot** showing Av(t) and the **R code** used to calculate Av(t). (Hint: This can be done in a single line – there is no need for a for loop)

Answer:

```
# Add Av(t) to the sp500 table
sp500$Av <- (sp500$Close + sp500$High + sp500$Low) / 3
#  Produce a single line plot showing Av(t) from sp500 table.
ggplot(data = sp500, aes(x = index(sp500), y = Av)) +
  geom_line(color = "red") +
```

```
labs(
  y = "Average Price", x = "Date",
  title = "Average Price Change in the Last Month"
)
```



Question Three: Daily returns are commonly used in stock market analysis. The daily return is defined as:

$$Return(t) = \frac{C(t) + C(t-1)}{C(t-1)}$$

Calculate the return for the **last 12 months** of the sp500 data and compare the candleChart for the **last 12 months** and the daily returns. **Discuss the patterns in daily returns in relation to the sp500 price signal.** Include a **figure** showing the **returns plot** and **R cod**e to calculate returns.

Answer:

Based on the formular above, we first calcualte the daily return apply to sp500 dataset and then subset to last 12 months of data as last_year.

```
# add last day closing balance into  sp500 dataset
sp500$lastday_closing <- lag(sp500$AdjClose, 1)
# add daily return column into sp500 dataset
sp500$daily_return <- (sp500$AdjClose - sp500$lastday_closing) / sp500$lastday_closing
last_year <- last(sp500, "12 month")
```

Apart from daily return, I am also interested in understanding how much is the total return for the whole year.

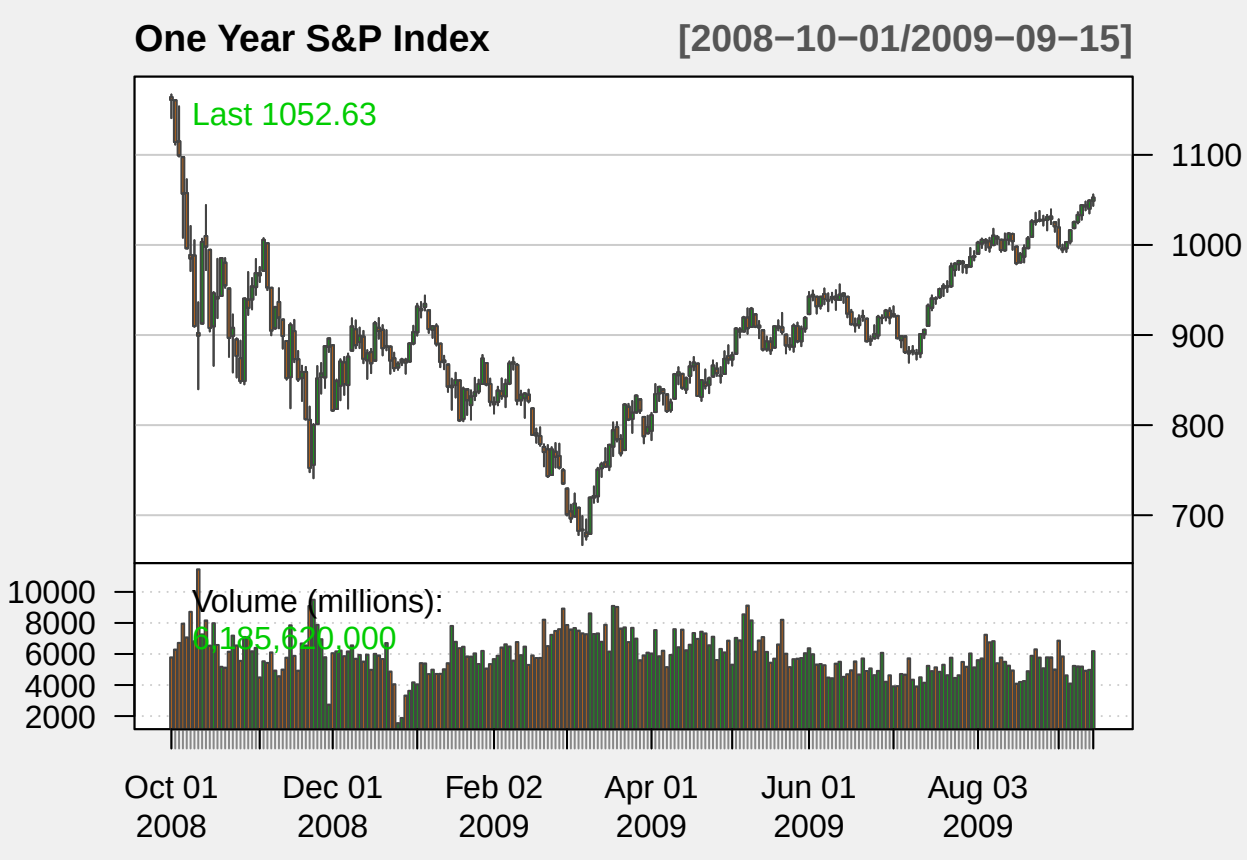
```
# get end of year closing balance
end_of_year_balance <- max(last(last_year$AdjClose))
# get start of year closing balance
start_of_year_balance <- max(first(last_year$AdjClose))
# calculate the total return for the year
total_return <- (end_of_year_balance - start_of_year_balance) / start_of_year_balance
print(paste("The total return since last year is", round(total_return * 100, 2), "%"))
```

```
## [1] "The total return since last year is -9.34 %"
```

Not very good, the S&P Index has decreased for 9.34% since the last year between October 2008 and September 2009.

To check the stock index movement during the last year, let's use last 12 month of index movement data to plot a candleChart.

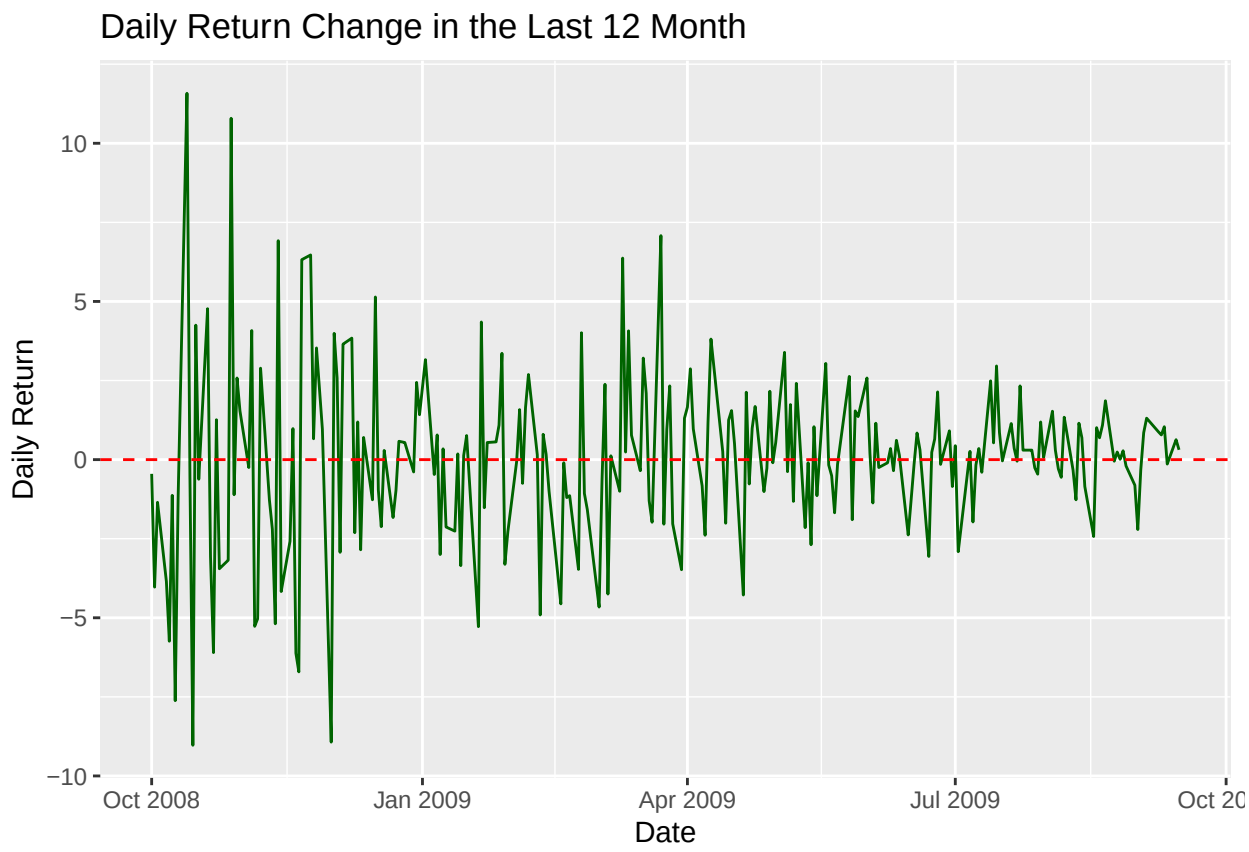
```
chartSeries(last_year,
  type = "candlesticks", theme = "white",
  name = paste("One Year S&P Index")
)
```



As we can see, the index decreased dramatically during the first part of last 12 month period, and then started to claim. However the closing balance did not reach the beginning balance.

To see the daily return movement during the past 12 month, we have build a line plot to demonstrate.

```
# plot the daily return movement during the last 12 month
ggplot(data = last_year, aes(x = index(last_year), y = round(daily_return * 100, 2))) +
  geom_line(color = "darkgreen") +
  labs(
    y = "Daily Return", x = "Date",
    title = "Daily Return Change in the Last 12 Month"
  ) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red")
```



What we can see on the graph above, it demonstrates that the return rate was very fluctuated during the beginning of the last 12 month period, but the return rate was getting stable which indicates the market started to be more stable. However, the return rate was still negative for the whole year.

PART TWO: DEGREE DISTRIBUTION - NETWORK (15 MARKS)

A network (graph) is defined as a set of nodes connected by a set of edges. The degree (k) of a node is defined as the number of edges connected to a node. For example, a social network could be defined by having nodes as people, and the edges link people (nodes) based on whether they are friends. The characteristics of a network are often defined in terms of the distribution of node degree for the entire network. There are many types of networks, but we are interested in are scale-free networks (see https://en.wikipedia.org/wiki/Scale-free_network for additional information), where the probability of a node

having degree k, P(k), is proportional to the degree (k) to some constant power γ. Hence these types of networks are defined by $P(k) \sim k^{-\gamma}$. Complete the following steps:

Question One: Load in the comma separated dataset network.csv into R (HINT: read.csv(...)). This dataset is a network with 1000 nodes. The data shows for each node the degree "k" (number of edges connected to/from the node) for that node.

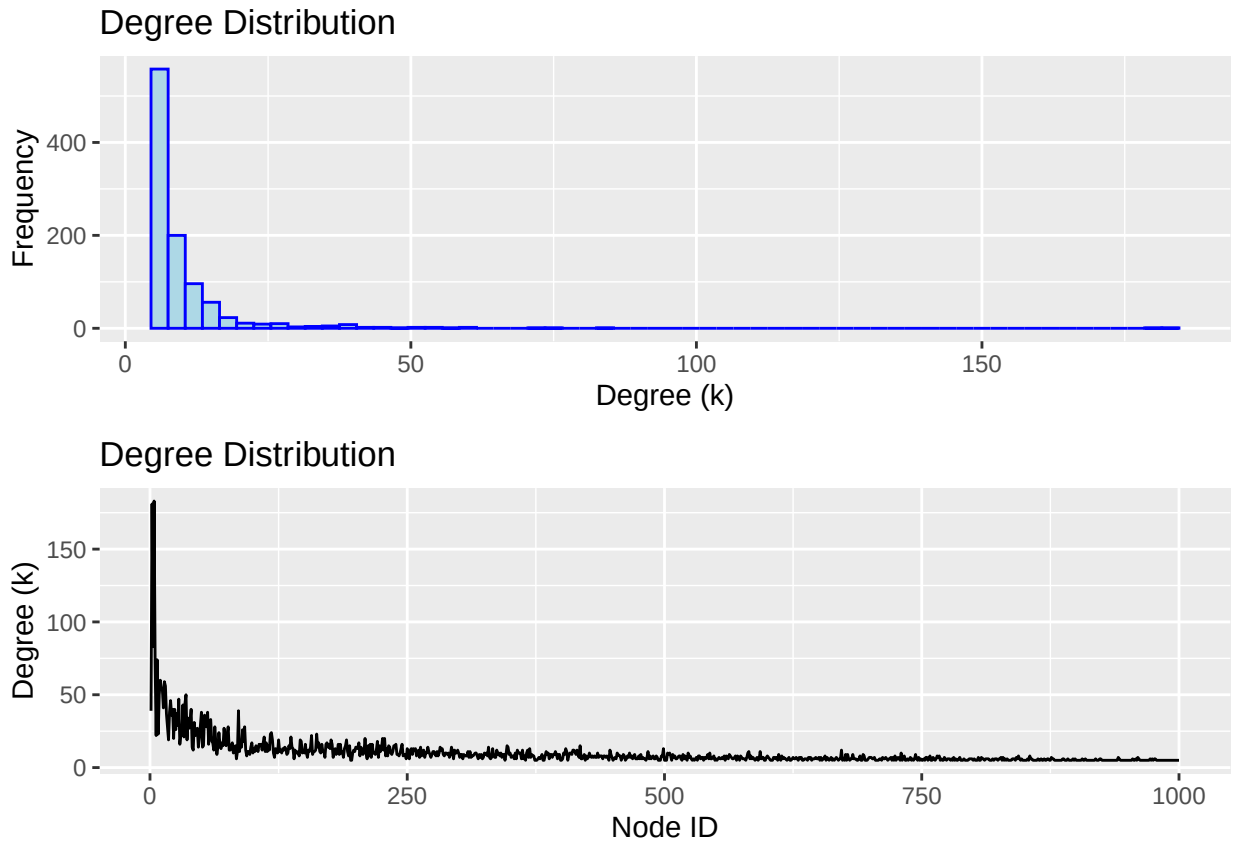
Answer:

```
# Load the network.csv into R
network <- read.csv("network.csv")
```

Question Two: Produce a **single figure showing 2 plots:** the histogram of the degree (k), and a line plot sorted by degree(k). Comment on what this tells you about the network.

Answer:

```
## Warning in geom_histogram(main = "Degree Distribution", binwidth = 3, fill =
## "lightblue", : Ignoring unknown parameters: `main`
```

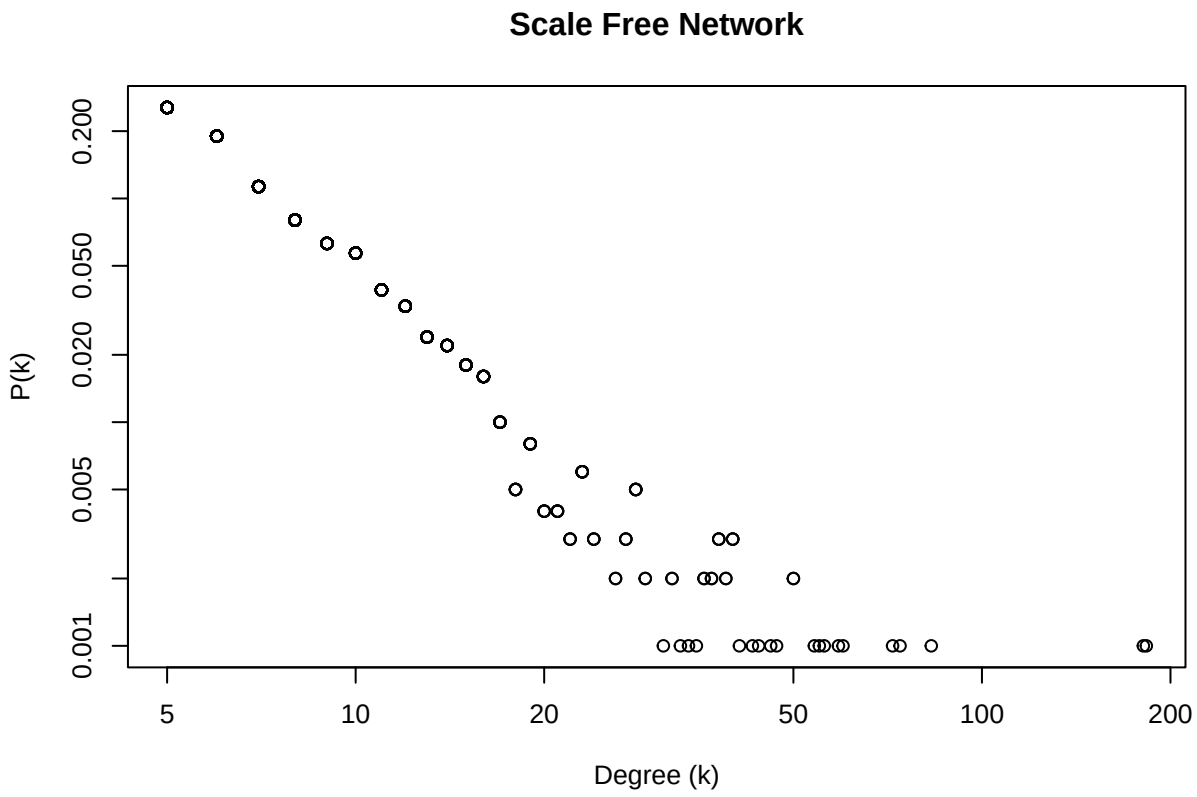


Comments: From the sorted line chart, we can see there is a clear pattern that there is a a small amount of nodes, which have a high degree, and most of the nodes have a low degree. This is consistent with the definition of scale-free network. The histogram shows that most of the nodes have a low degree, and only a few nodes have a high degree. This is consistent with the definition of scale-free network.

Question 3: Calculate P(k), which is the probability of observing a node with degree k, and **produce a plot of P(k) versus k** (as shown below, Figure 1) using a log scale for both the x and y axes. HINT: plot(....,log="xy"). In your assignment include the R code for calculating P(k) and producing the plot.

Answer:

```
# y is a paramter whoese value is typically in the range 2 <y <3. In this example, we set the y = 2.5
y <- 2.5
# calculate P(k)
total_node <- nrow(network)

# calcualte P_k in the network dataset
network$P_k <- sapply(network$k, function(k) {
  sum(network$k == k) / total_node
})

par(cex = 0.8)
plot(network$k, network$P_k, log = "xy", main = "Scale Free Network", xlab = "Degree (k)", ylab = "P(k)
```



Question 4: Explain why the plot from step 4 implies that the network is likely to be scale-free.

Answer:

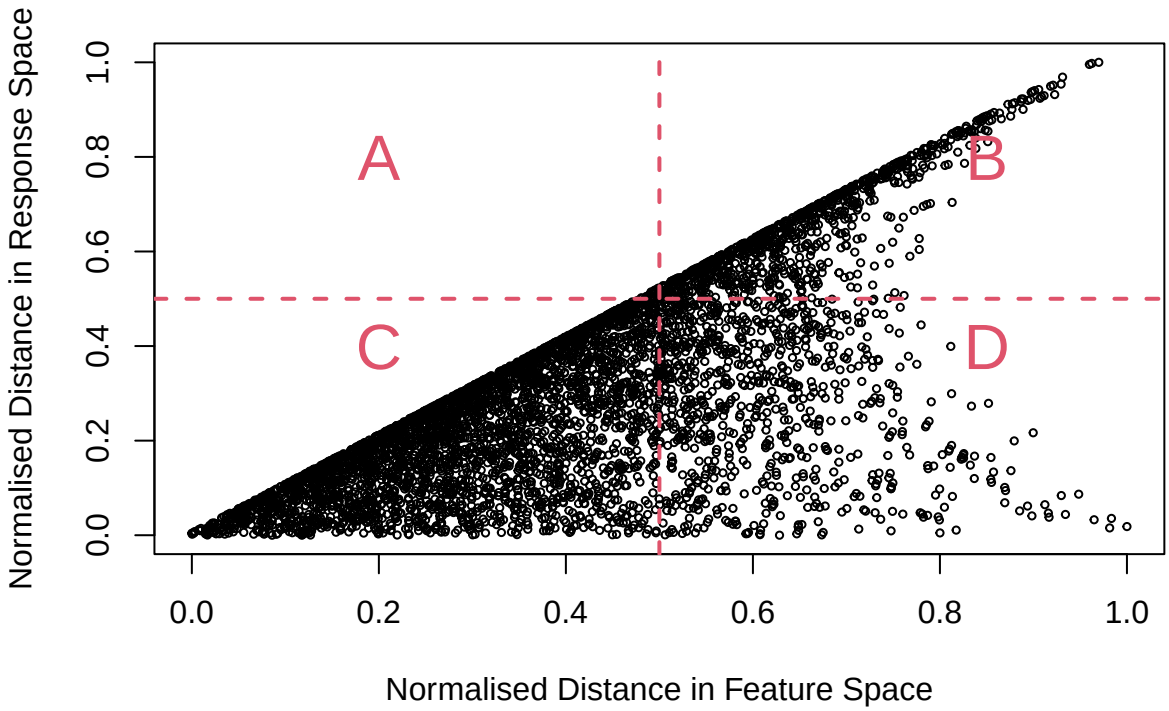
According to the Figure 1. from step 4, we can see that the the network is likely to be scale free because the plot shows a negative replationship between P(k) and k, which is consistent with the definition of scale-free network. The plot shows a stright line in the beginning and then the line starts to spread out when the degree is getting higer.Over all, most of the nodes have a low degree, and only a few nodes have a high degree. This is consistent with the definition of scale-free network.

PART THREE: DATA QUALITY AND STRUCTURE(20 MARKS)

Examine the supplied script **distplots.r**. This contains a definition for a function **dist.table(...)**, that creates a table (data frame) of two columns: the *normalised distance between all pairs of rows in a dataset and the distance between the corresponding response values for each pair*. If you run (source) **distplots.r** it produces a single plot using generated data for a simple linear model of two explanatory variables.

Question One: Explain what the plot represents and why this might be a useful method for examining the quality of a dataset in terms of its potential to be modelled.

```
# Run the distplots.R script
source("distplots.r")
```

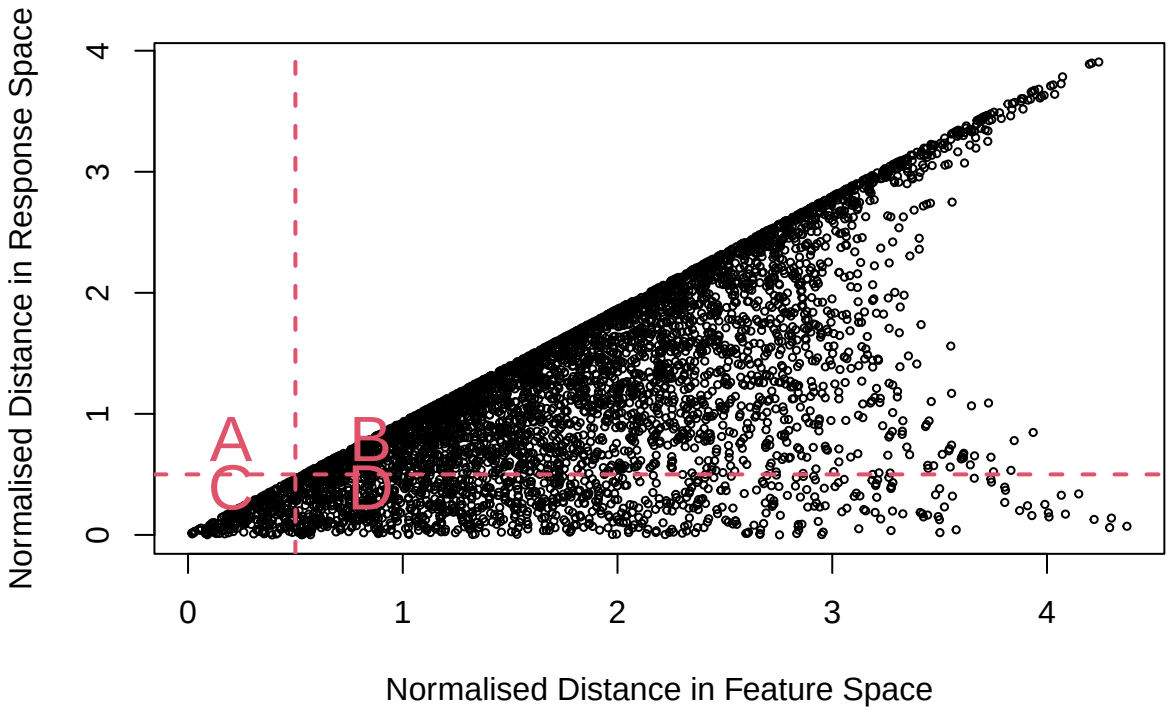


Answer:

The plot represents the relationship between the distance of normalised distance between all pairs of rows in a dataset and the distance between the corresponding response values for each pair. From this plot we can see a clear postive relationship between disctance in feature space and distance in response space. Before ploting, this script file user `dist.table` method to normalised the data, which elimitates the impact of the scale of the data. According to the `distplots.r` script, the response variables is generated from `Y <- (X1 + X2)`, which has different scale compared to the feature variables. To ilustrate the data scale impact of non-normalised data, we can compare the plot of normalised data and non-normalised data.

```
## we modify the original dist.table method to return the non-normalised data
dist.table_v2 <- function(d, response.var = ncol(d)) {
  d <- scale(d) # scale data
  d.dist <- dist(d[, -response.var]) # distance all X values
```

```
  d.resp <- dist(d[, response.var])
  # create a dataframe without normalisation
  data.frame(cbind(d.dist, d.resp))
}
# Now construct the distance table
d1 <- dist.table_v2(ex1, response.var = 3)
# and plot it!
plot.dist.table(d1)
```



From the above graph, Response Spaces in Y axes has more impact to the graph which makes the graph difficult to interpret. However, the normalised data has a clear positive relationship and clear separation around the 0.5 distance in both X axes and Y axes. This is a useful method for examining the quality of a dataset in terms of its potential to be modelled.

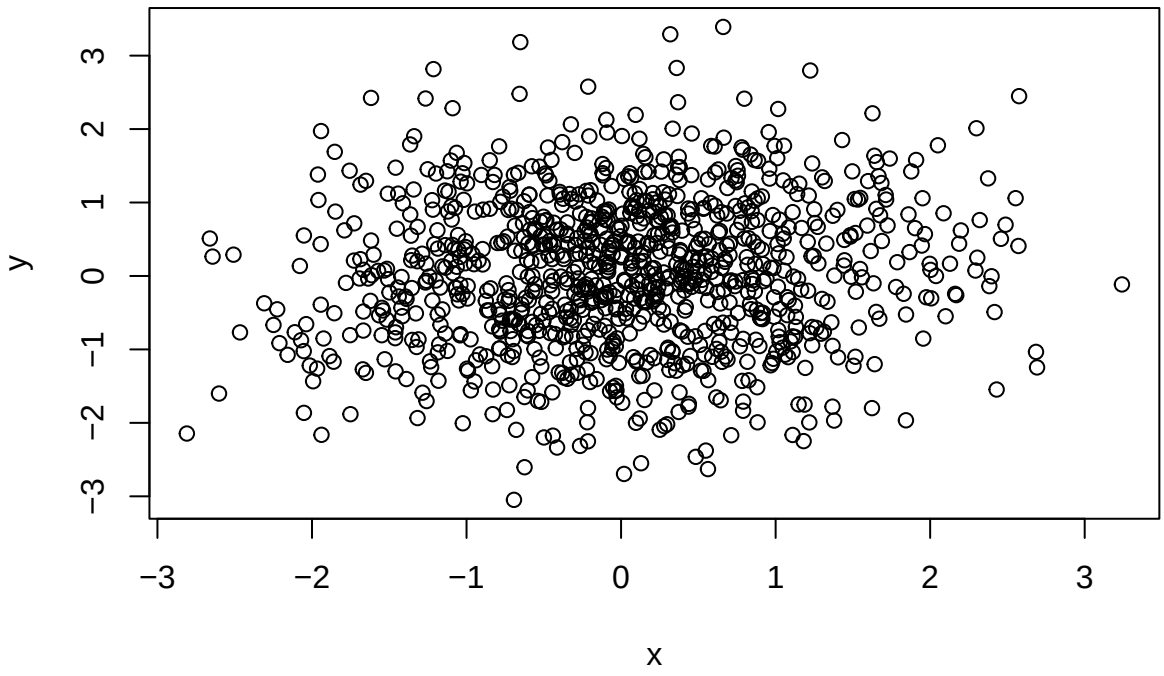
QUESTION TWO: Create a dataset where there is NO relationship between the explanatory and response variables and plot the distance relationship created by dist.table(...). **Explain why you observe this pattern. INCLUDE THE R CODE TO PRODUCE THE DATASET.**

Answer:

We can create a dataset where there is NO relationship between the explanatory and response variables by generating random numbers for the explanatory and response variables. We can use the following R code to produce the dataset.

```
set.seed(123)
# create a dataset contains 1000 records that has no relationship between
# the explanatory and response variables
```

```r
random_x <- rnorm(1000)
random_y <- rnorm(1000)
ex2 <- data.frame(x = random_x, y = random_y)
# plot the datasets
plot(ex2, pch = 1)
```




```r
# Print out the x and y's standard deviation and mean
print(paste("The standard deviation of x is", round(sd(random_x), 2)))
```

```
## [1] "The standard deviation of x is 0.99"
```

```r
print(paste("The standard deviation of y is", round(sd(random_y), 2)))
```

```
## [1] "The standard deviation of y is 1.01"
```

```r
print(paste("The mean of x is", round(mean(random_x), 2)))
```

```
## [1] "The mean of x is 0.02"
```

```r
print(paste("The mean of y is", round(mean(random_y), 2)))
```

```
## [1] "The mean of y is 0.04"
```

```r
# Print out the x and y's correlation
print(paste("The correlation between x and y is", round(cor(random_x, random_y), 2)))
```

```
## [1] "The correlation between x and y is 0.09"
```

```r
print(paste("The correlation between x and y is", round(cor(random_x, random_y), 2)))
```

```
## [1] "The correlation between x and y is 0.09"
```

The above plot show there is no relationship between the explanatory and response variables. This is because both explanatory varibale X and response variable Y are generated from random numbers. The standard deviation of x and y are close to 1 and mean of x and y are close to 0. The correlation between x and y is close to 0. Most importantly, the correlation between x and y is close to 0, which indicates there is no relationship between the explanatory and response variables.

Question Three: Produce a figure with 2 plots using dist.table(...): one using the Boston housing dataset (**library(MASS); data(Boston)**; response variable **medv**) and one using the supplied table bioavailabilility dataset (**bioavailability.txt; response variable last column** (unlabelled)) . **Explain** which dataset you believe would be easier to model and why. Include a discussion on the relationship between the visualisation created by the function dist.table and how it might relate to a **k-nearest neighbour model**.

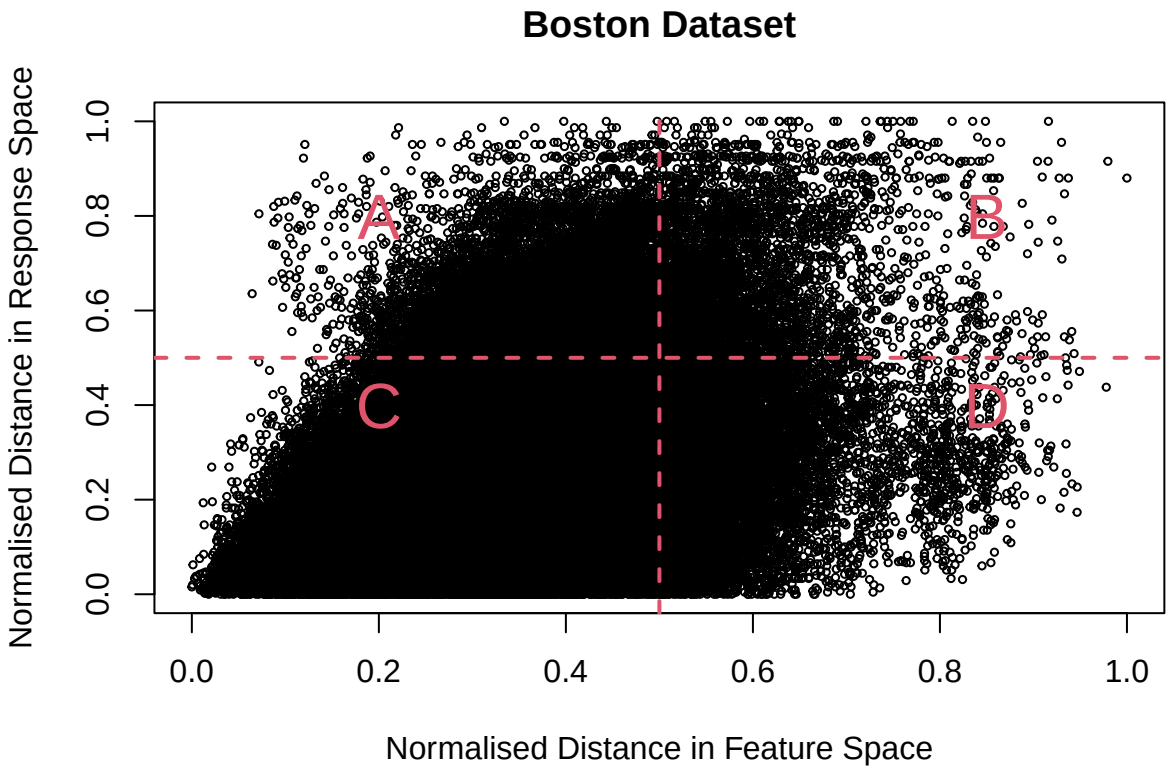
Answer:

We can use the following R code to produce the figure with 2 plots using dist.table(...). The first plot is using the Boston housing dataset and the second plot is using the supplied table bioavailabilility dataset. We can then compare the two plots to see which dataset would be easier to model.

Now, let's use the `dist.table` method to visualise the relationship between the explanatory and response variables for both datasets.

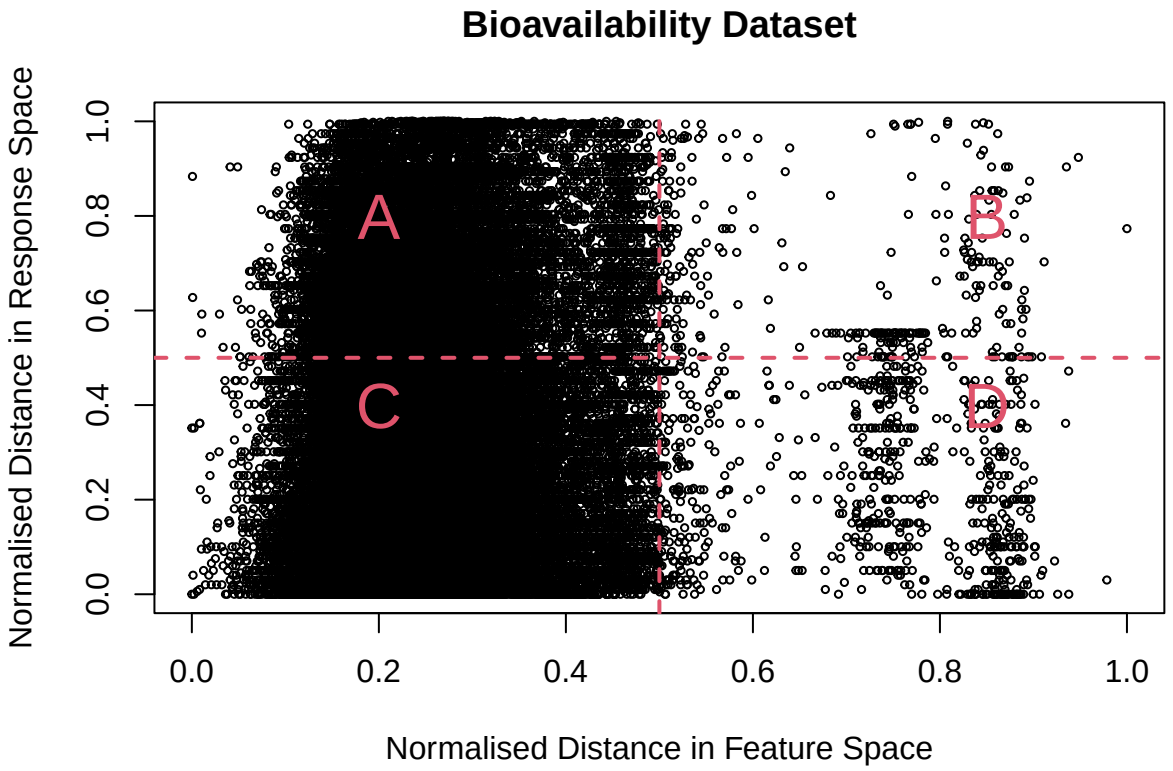
The first plot is for Boston dataset.

```r
# Read both Boston and bioavailability.txt data
library(MASS)
data(Boston)
df_bioavailability <- read.table("bioavailability.txt", sep = "\t", header = FALSE)
# Now construct the distance table
col_num <- dim(Boston)[2]
d <- dist.table(Boston, response.var = col_num)

# and plot it!
plot.dist.table(d)
title("Boston Dataset")
```



Next, let's plot the bioavailability dataset.

```
# Now construct the distance table
col_num <- dim(df_bioavailability)[2]
d <- dist.table(df_bioavailability, response.var = col_num)

# and plot it!
plot.dist.table(d)
title("Bioavailability Dataset")
```



Based on aboved plots, the `dist.table` method create visuals that split distances between response and explanatory variables into four different areas (A, B, C and D). The area C and B are the most important areas for k-nearest neighbour model. The area C is the area where the distance in feature space is **small** and the distance in response space is **small**, which means they are likey to be grouped as one cluster. The area D is the area where the distance in feature space is **large** and the distance in response space is **large**. The area A and D are less important for k-nearest neighbour model. The area A is the area where the distance in feature space is **small** and the distance in response space is **large**. The area D is the area where the distance in feature space is **large** and the distance in response space is **small**.

Base on above analysis, Boston dataset is much easier to model with following reasons:

1. *The Boston data has spread data points in the feature space and response space, which is likely to be modelled by k-nearest neighbour model. The bioavailability dataset has a lot of data points in the area A and D, which is less likely to be modelled by k-nearest neighbour model.*
2. *Boston is the dataset is saved as* `data.frame` *and the dataset has header and each field is labelled with clear meaning. The bioavailability dataset is saved as* `data.frame` *by calling r* `read.table` *method, but the dataset does not have header and the last column is the response variable. The dataset is not labelled and it is difficult to understand the meaning of each field.*
3. *Boston dataset has only 14 columns, which is relatively small and easy to model. The bioavailability dataset has 242 columns, which is relatively large and difficult to model.*

PART FOUR: VISUALISATION AND CLUSTERING (25 MARKS)

This question uses the supplied dataset **"countrystats.csv". This file contains the data for 178 countries, showing their population density (people per km2), income per capita ($), purchasing power parity and change in gross domestic product as a percentage (%change in GDP 2010-2011) for 2011. You will need to read this data into "R", change the row names to be the country names, and delete the country names column prior to working with this data.**

Question One: Briefly state the meaning of each variable for the countrystats data.

Answer:

```
df_countrystats <- read.csv("countrystats.csv")
# change the row name to countrystats column values
rownames(df_countrystats) <- df_countrystats[, 1]
# drop the countrystats column
df_countrystats <- df_countrystats[, -1]
# print each column name and its meaning
print(paste(
  "The country stats data has", dim(df_countrystats)[1],
  "rows and", dim(df_countrystats)[2],
  "columns", "below is each folumn name and its meaning:"
))
```

```
## [1] "The country stats data has 178 rows and 4 columns below is each folumn name and its meaning:"
```

```
print("Column names:")
```

```
## [1] "Column names:"
```

```
print(colnames(df_countrystats))
```

```
## [1] "PopDensity"       "IncomeperCapita"  "PurchasingParity" "ChangeGDP"
```

```
print("The meaning of each variable for the countrystats data:")
```

```
## [1] "The meaning of each variable for the countrystats data:"
```

```
print("Population Density: people per km2")
```

```
## [1] "Population Density: people per km2"
```

```
print("Income per Capita: $")
```

```
## [1] "Income per Capita: $"
```

```
print("Purchasing Power Parity: $")
```

```
## [1] "Purchasing Power Parity: $"
```

```
print("%change in GDP 2010-2011: %")
```

```
## [1] "%change in GDP 2010-2011: %"
```

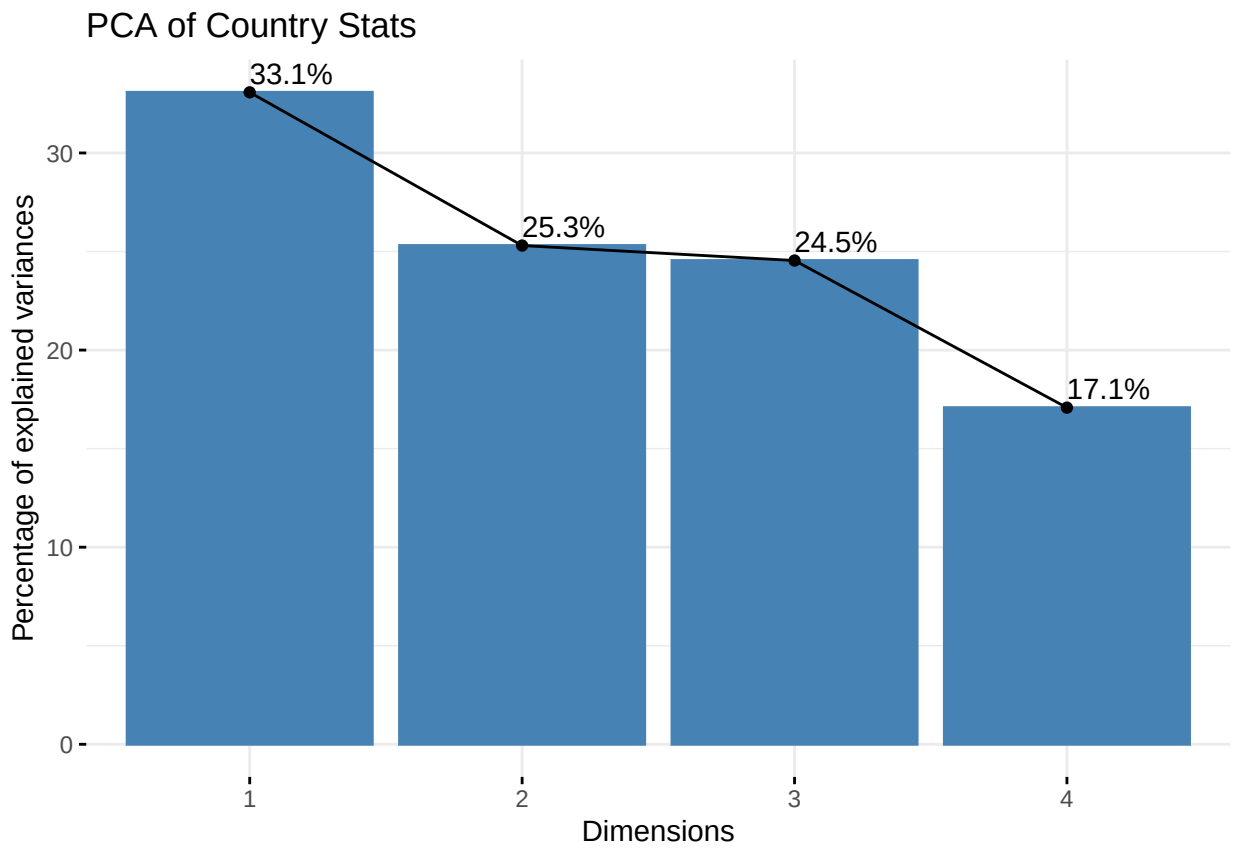
- **PopDensity** column means the population density of each country, which is the number of people per km2.
- **IncomeperCapita** column means the income per capita of each country, which is the average income of each person in the country.
- **PurchasingParity** column means the purchasing power parity of each country, which is the average income of each person in the country.
- **ChangeGDP** column means the change in gross domestic product as a percentage for 2011.

Question Two: Visualise the countrystats data to examine the relationships between (and within) countries. Which countries appear to be the similar to NewZealand? Which countries are the strongest/weakest? You will need to consider which features(explanatory variables) are important when doing this assessment, and explain what you mean by strongest/weakest.

Answer:

To analyse the most important features, we can use the pca method to reduce the dimensionality of the data and then plot the first two principal components to see the relationship between (and within) countries. We can then use the k-means clustering method to cluster the countries into different groups and then examine the relationship between the countries in the same group and the countries in different groups.

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```



By running the code above, we can see that the first two principals only explain 58% of the variance, which is not very good. We will also run Kaiser-Meyer-Olkin (KMO) test to check the sampling adequacy of the data. KMO value close to 1 indicates the data is suitable for factor analysis.

```
library(EFAtools)
kmo <- KMO(df_countrystats)
```

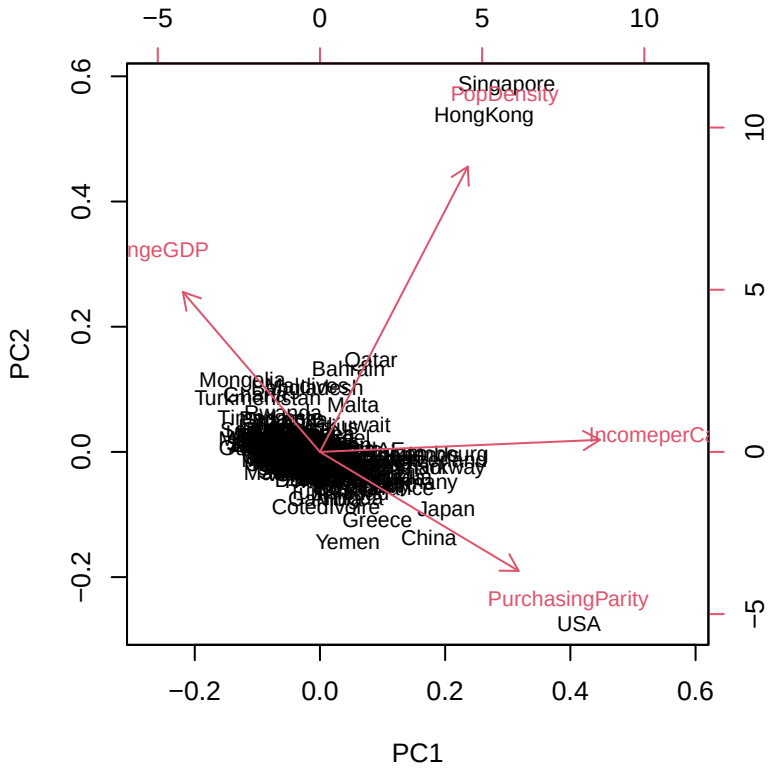
```
## i 'x' was not a correlation matrix. Correlations are found from entered raw data.
```

```
print(paste("The KMO value of the df_countrystats dataset is:", round(kmo$KMO, 4)))
```

```
## [1] "The KMO value of the df_countrystats dataset is: 0.4796"
```

After running the code above, we can get the KMO value for datatset `df_countrystats` is 0.48, which is not very good. This indicates the data is not suitable for factor analysis. However, we can still use the first two principal components to see the relationship between (and within) countries. The code below plots the `biplot` to identify first 2 principal components

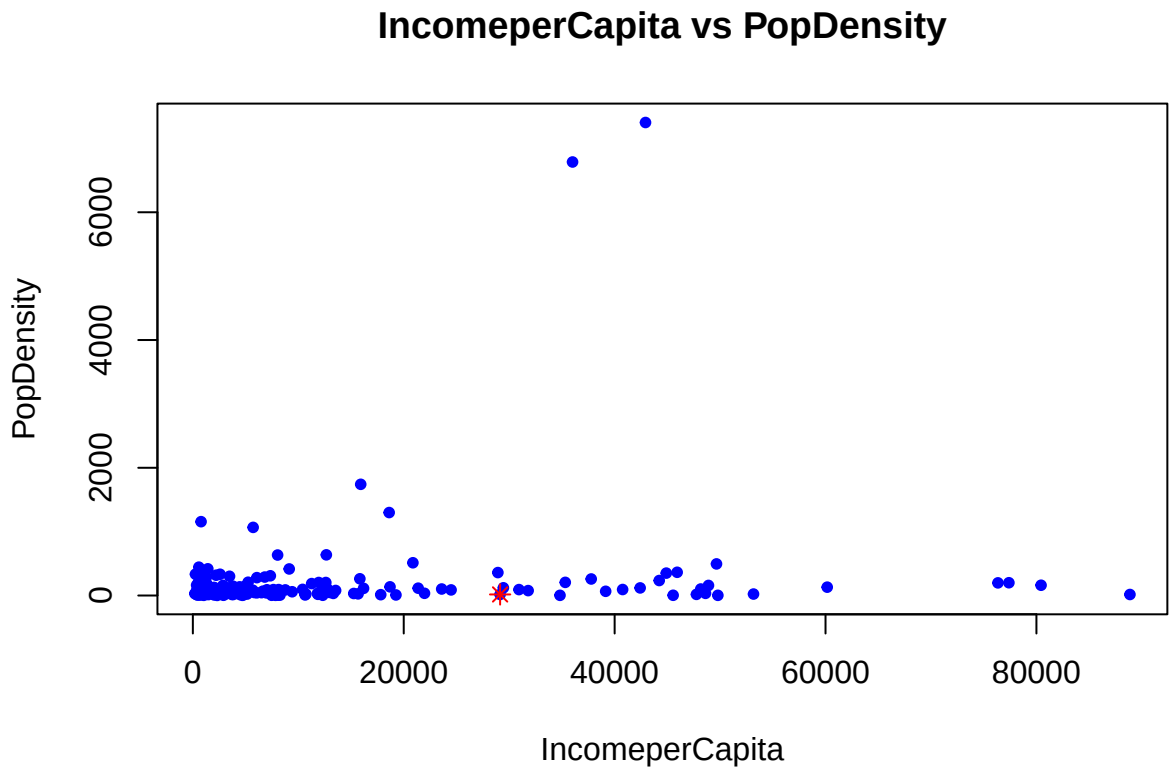
```
par(cex = 0.8)
biplot(pca_countrystats)
```



Base on the plot above, we select longest distance from PCA1 and PCA2 as the top 2 principal components in the dataset. We can see that PopDensity and IncomeperCapita are the most important features in the dataset.

Now plot the data between IncomeperCapita and PopDensity, and print the points that New Zealand belongs to in the red colour. The rest countries are printed in blue dots.

```
# take subset of countrystats_sub by only select IncomeperCapita,PopDensity
df_countrystats_sub <- df_countrystats[, c("IncomeperCapita", "PopDensity")]
# plot the data between IncomeperCapita and PopDensity
plot(df_countrystats_sub$IncomeperCapita, df_countrystats_sub$PopDensity,
  pch = 20, col = "blue", main = "IncomeperCapita vs PopDensity",
  xlab = "IncomeperCapita", ylab = "PopDensity"
)
# Change the plot that New Zealand is in red
points(df_countrystats_sub["NewZealand", ]$IncomeperCapita,
  df_countrystats_sub["NewZealand", ]$PopDensity,
  pch = 8, col = "red"
)
```



The table below demonstrates the top 5 most close to New Zealand based on their Euclidean distance.

```
dist_to_nz <- data.frame(matrix(ncol = 2, nrow = 0))
colnames(dist_to_nz) <- c("country", "distance")
for (country_name in
  rownames(
    df_countrystats_sub[-which(rownames(df_countrystats_sub) == "NewZealand"), ]
  )
) {
  dist_to_nz <- rbind(
```

```
    dist_to_nz,
    data.frame(
      country = country_name,
      distance = dist(rbind(
        df_countrystats_sub["NewZealand", ],
        df_countrystats_sub[country_name, ]
      ))[1]
    )
  )
}
rownames(dist_to_nz)= dist_to_nz$country
dist_to_nz <- dist_to_nz[order(dist_to_nz$distance, decreasing = FALSE), ]
dist_to_nz <- subset(dist_to_nz, select = -country)
head(dist_to_nz, 5)
```

```
##         distance
## Cyprus  326.9801
## Israel  401.3278
## Spain  1791.6127
## Brunei 2660.6766
## Greece 4650.5420
```

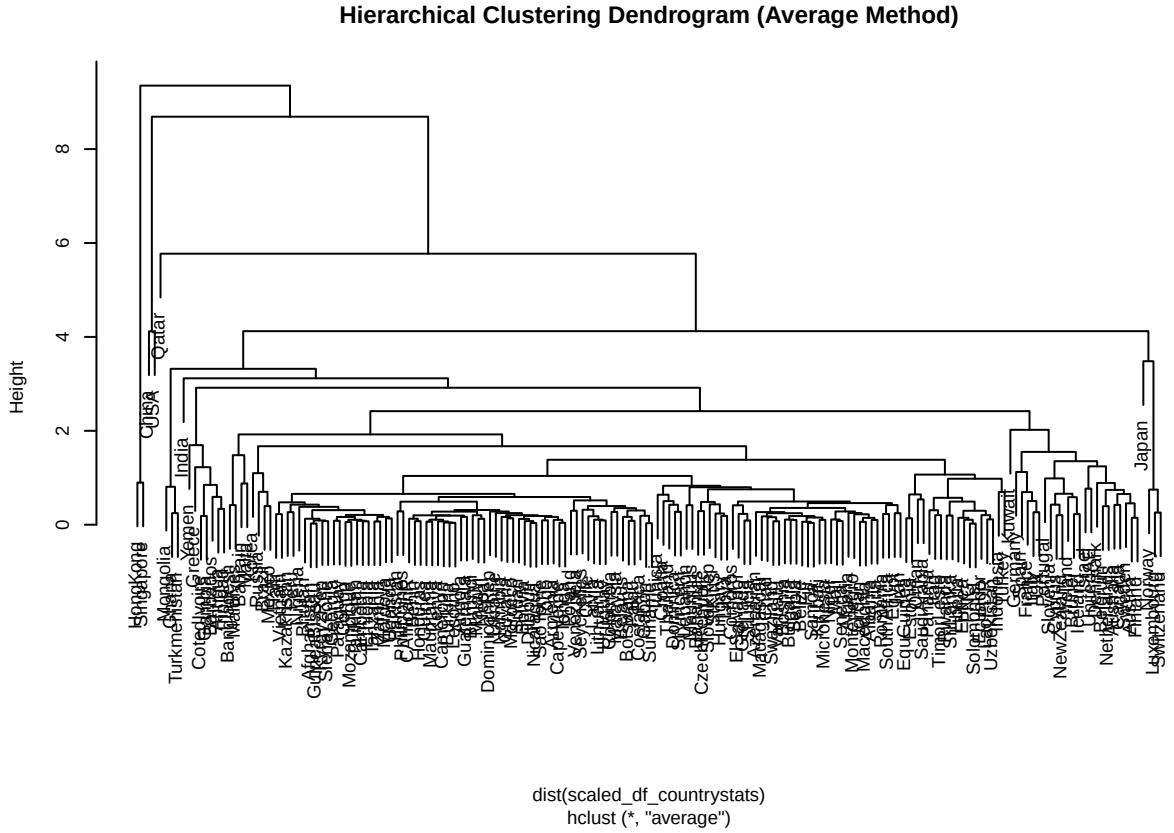
Question Three: Create two hierarchically clustered dendrograms for the country data **after scaling the data** to have a mean of zero and standard deviation of one (for each variable). Use the agglomeration methods "average" and "complete". **Produce figures** showing each of these dendrograms, and **explain why they are different**. Finally, using the "complete" linkage dendrogram, cut the dendrogram into 50 clusters (using cutree), and state the countries that are in the same cluster as New Zealand. **Provide the R code to do this dendrogram/cutree and analysis to get the country names.**

Answer:

We can use the following R code to create two hierarchically clustered dendrograms for the country data after scaling the data to have a mean of zero and standard deviation of one (for each variable). We can then use the agglomeration methods "average" and "complete" to create the dendrograms.

First we scale the data by running the scale funtion and plot the hierarchical clustering dendrogram by using the average method.

```
scaled_df_countrystats <- scale(df_countrystats)

par(cex = 0.6)
# plot_1 using average method. Showing the dendrogram by horizontal
plot(hclust(dist(scaled_df_countrystats), method = "average"),
  main = "Hierarchical Clustering Dendrogram (Average Method)"
)
```

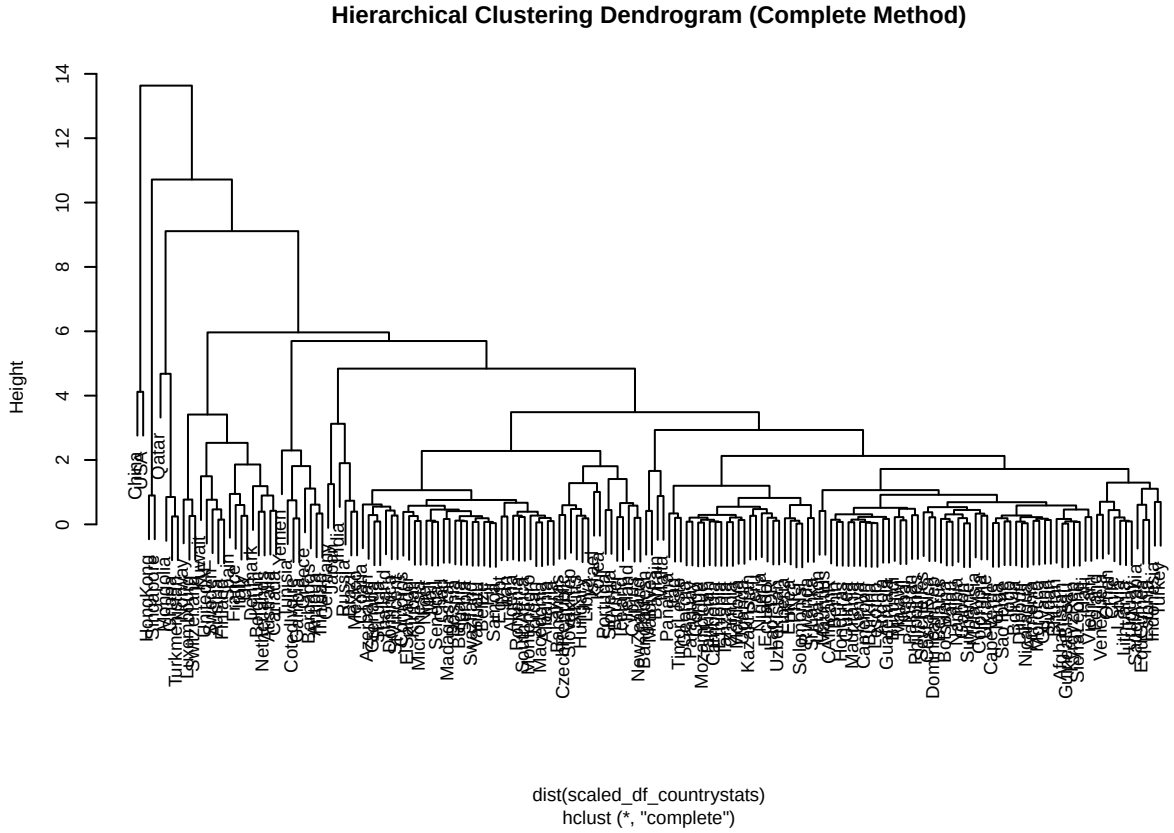


In the second plot below, we are using the "complete" linkage dendrogram, cut the dendrogram into 50 clusters (using cutree), and state the countries that are in the same cluster as New Zealand.

The reason why the two dendrograms are different is because the average method uses the average distance between the two clusters, while the complete method uses the maximum distance between the two clusters. The complete method is more sensitive to outliers than the average method, which is why the two dendrograms are different.

Last by not least, we are cutting the dendrogram into 50 clusters by using the complete method and print the countries that are in the same cluster as New Zealand.

```
# plot_2 using complete method
par(cex = 0.6)
plot(hclust(dist(scaled_df_countrystats), method = "complete"),
  main = "Hierarchical Clustering Dendrogram (Complete Method)"
)
```




```
# cut the dendrogram into 50 clusters
df_tree_50 <- data.frame(cutree(hclust(dist(scaled_df_countrystats),
  method = "complete"
), k = 50))
nz_cluster_num <- df_tree_50[rownames(df_tree_50) == "NewZealand", ]
nz_cluster_country <- rownames(df_tree_50)[df_tree_50[, 1]
== nz_cluster_num & rownames(df_tree_50)
!= "NewZealand"]
df_dengrogram = data.frame(cluster=nz_cluster_num,country=nz_cluster_country)
print(paste("There are",nrow(df_dengrogram), "countries who are in the same cluster. The table below li
```

```
## [1] "There are 4 countries who are in the same cluster. The table below lists all countries."
```

```
df_dengrogram
```

```
##   cluster country
## 1      15  Brunei
## 2      15  Cyprus
## 3      15 Iceland
## 4      15 Ireland
```

Question Four: Use the dimensionality reduction method **t-SNE** to create a 2 dimensional plot of the country data. **Visualise (plot) the result** and compare the nearest 5 countries from New Zealand using the t-SNE mapping to the results in step 3 (HINT: You will need to take the results from t-SNE, build a distance

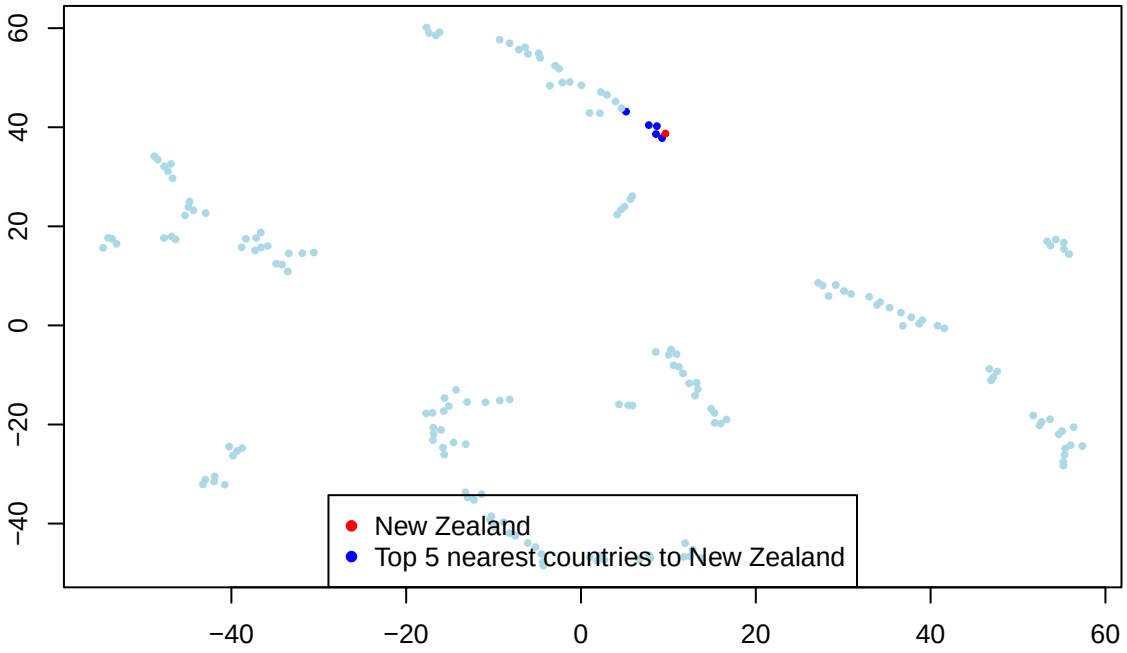
matrix, and find the 5 nearest countries to NZ). Commentonwhytheyarethesame(ordifferent). **Include the plot and R code for t-SNE with your answer.**

Answer:

We can use the following R code to use the dimensionality reduction method t-SNE to create a 2 dimensional plot of the country data. We can then visualise (plot) the result and compare the nearest 5 countries from New Zealand using the t-SNE mapping to the results in step 3.

Once we complete the clustering by using the t-SNE method, we can then plot the result and compare the nearest 5 countries from New Zealand using the t-SNE mapping to the results in step 3. In the plot below, New Zealand is plotted in red, and the top 5 nearest countries to New Zealand are plotted in blue, and the rest of the countries are plotted in light blue. We can see very clear that New Zealand and other 5 nearest countries are in the same cluster, and there are some other countries are in the same cluster as New Zealand but have a relatively larger distance.

```r
# create a datafram to store to store 5 nearnest countries to New Zealand by subsetting df_tsne_countri
dist_to_nz_tsne <- data.frame(matrix(ncol = 2, nrow = 0))
colnames(dist_to_nz_tsne) <- c("country", "distance")

for (country_name in rownames(
  df_tsne_countries[-which(rownames(df_tsne_countries) == "NewZealand"), ]
)) {
  dist_to_nz_tsne <- rbind(
    dist_to_nz_tsne,
    data.frame(
      country = country_name,
      distance = dist(rbind(
        df_tsne_countries["NewZealand", ],
        df_tsne_countries[country_name, ]
      ))[1]
    )
  )
}
# order the dist_to_nz_tsne in acending order and filter the top 5 distance
dist_to_nz_tsne <- dist_to_nz_tsne[order(dist_to_nz_tsne$distance, decreasing = FALSE), ]

par(cex = 0.8)

# Plot tsne_countries but colouring dist_to_nz_tsne countries in different color
plot(df_tsne_countries,
  xlab = "", ylab = "",
  pch = 19, cex = 0.5,
  col = ifelse(rownames(df_tsne_countries) %in% dist_to_nz_tsne$country[1:5],
    "blue", ifelse(rownames(df_tsne_countries) == "NewZealand",
      "red", "lightblue"
    )
  )
)


# Print ledgends on different color plots
legend("bottom",
  legend = c("New Zealand", "Top 5 nearest countries to New Zealand"),
  col = c("red", "blue"), pch = 16
)
```



The following code prints the top 5 nearest countries to New Zealand with their distance to New Zealand.

```
rownames(dist_to_nz_tsne) = dist_to_nz_tsne$country
dist_to_nz_tsne <- subset(dist_to_nz_tsne, select = -country)
head(dist_to_nz_tsne,5)
```

```
##          distance
## Israel  0.9937759
## Cyprus  1.0837711
## Spain   1.7913353
## Brunei  2.5586767
## Iceland 6.3159927
```