Coursework (COMP-1832)

Programming Fundamentals for Data Science

Submitted By:
Suresh Regmi (001204526)

Submitted to:
Dr. Jia Wang

December 9, 2022

Contents

List of Figures

1 Python Task 1 Solutions

1.1 Reading the dataset using read_csv

As we are reading a comma-separated values (CSV) source, pandas function read_csv() would be a useful approach in doing so. To perform this, we need to import the pandas library first and can use the head() function to display the first 10 rows. We need to make sure to supply 10 as an input parameter in the head function so that it prints the first 10 rows, otherwise, it will print the first 5 rows by default. Following is the code to implement it and the result we got:

```
# Import required libraries
import pandas as pd

# Read the iris dataset csv file as a dataframe
iris_df=pd.read_csv('https://raw.githubusercontent.com/sureshrgmi/Iris/
main/Iris.csv')

# Printing the first 10 rows of the data structure
iris_df.head(10)
```

	Sepal length	Sepal width	Petal length	Petal width	Species
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa
5	5.4	3.9	1.7	0.4	Setosa
6	4.6	3.4	1.4	0.3	Setosa
7	5.0	3.4	1.5	0.2	Setosa
8	4.4	2.9	1.4	0.2	Setosa
9	4.9	3.1	1.5	0.1	Setosa

Figure 1: First 10 rows of IRIS dataset

1.2 Calculating Statistical information (Central Tendency and Dispersion)

In this task, we are going to calculate the Central Tendency Measures (Mean and Median) and Dispersion Measures (Standard Deviation, Range) for the Petal Width of all three species (Setosa, Versicolor, Virginica).

```
# Import required libraries
import pandas as pd

# Read the iris dataset csv file as a dataframe
iris_df=pd.read_csv('https://raw.githubusercontent.com/sureshrgmi/Iris/
main/Iris.csv')

# Calculating Central Tendency and Dispersion
iris_df.groupby('Species') \
  .apply(lambda x: pd.Series({
      'Mean'         : x['Petal width'].mean(),
      'Median'       : x['Petal width'].median(),
      'Range'        : x['Petal width'].max()-x['Petal width'].min(),
      'Standard Deviation'          : x['Petal width'].std()
})
)
```

Species	Mean	Median	Range	Standard Deviation
Setosa	0.244	0.2	0.5	0.107210
Versicolor	1.326	1.3	0.8	0.197753
Virginica	2.026	2.0	1.1	0.274650

Figure 2: Dispersion and Central Tendency of Petal Width

2 Python Task 2 Solutions

2.1 Best visualisation for a single attribute of single species

The three visualisations we are supposed to choose from are:

- Line Chart: In most cases, the line charts are used to visualize and track the changes in a value over a short or long period of time.
- Bar Chart: Bar charts are also used to track the changes over time or compare the value between two groups.
- Box Plot Chart: Box plot provides high-level information on a numeric data value to show how the values are distributed including symmetry, skewness and variance.

Yet the three visualisations mentioned (Line Chart, Bar Chart and Box Plot) have their own way of representing the data points, I think Box Plot would be the best visualization for a single attribute of single species (Petal width of Setosa in our case).

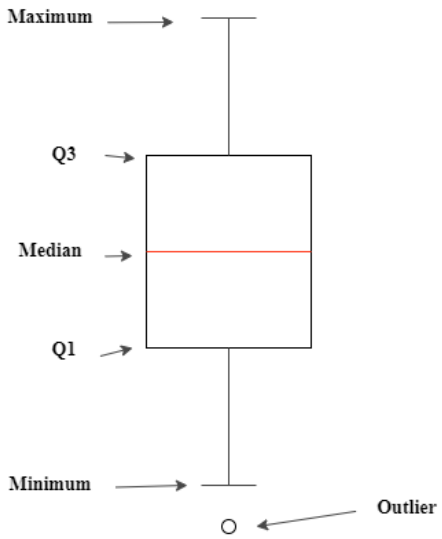

Figure 3: Box Plot example

2.2 Box Plot example

Following is the python code to create a box plot and the resulting visualisation:

```
# Import required libraries
import pandas as pd
import matplotlib.pyplot as plt

# Read the iris dataset csv file as a dataframe
iris_df=pd.read_csv('https://raw.githubusercontent.com/sureshrgmi/Iris/
main/Iris.csv')

# Cerate a new dataframe for Setosa Species
setosa_df=iris_df.where(iris_df['Species']=='Setosa')
```

```
# Visualizing the data (Petal Width of Setosa) using Box Plot
plt.figure(figsize = (10, 7))
plt.title("Petal Width of Setosa")
plt.ylabel("Size in CM")
b_plot = setosa_df.boxplot(column = 'Petal width')
b_plot.plot()
```

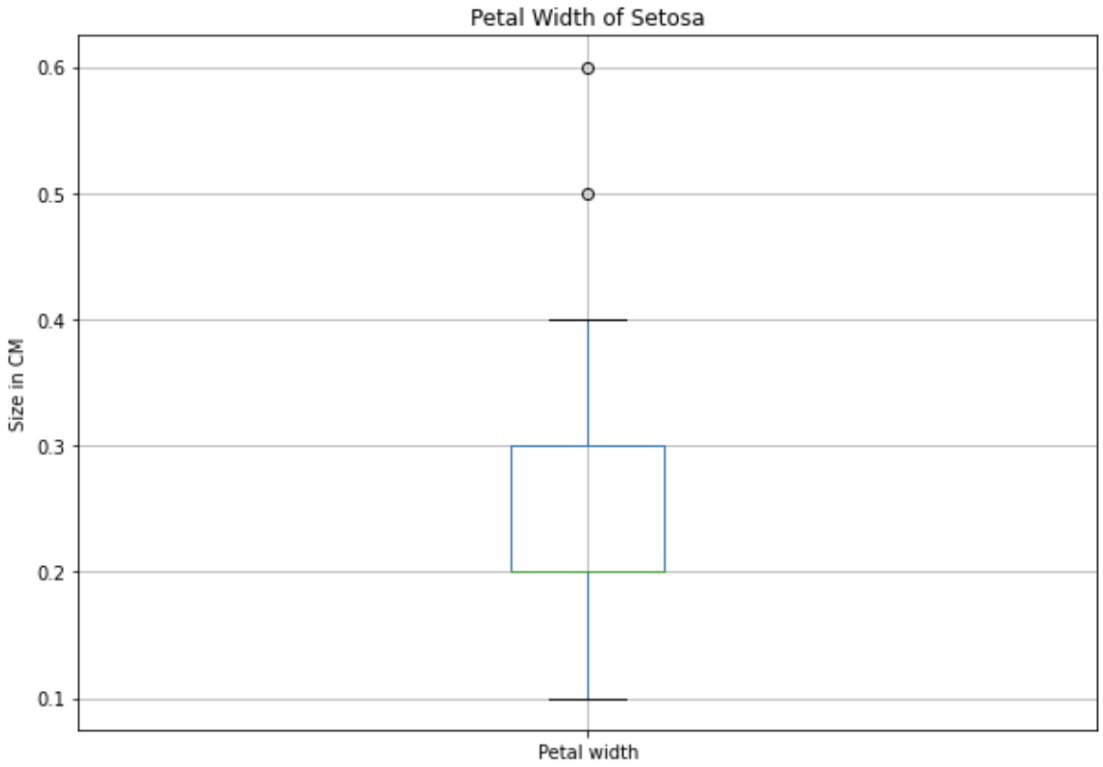

Figure 4: Box Plot for Petal Width of Setosa

Justification

A box plot is a graph that presents the summary of large data using five measures (minimum, maximum, lower quartile, upper quartile and median). It offers a clear summary visually effectively for a large dataset and is easy to understand by looking at it. The box size gives us an understanding of how the data is spread (a larger box size means data is more spread). In addition, it also shows outliers in our data (i.e. data points that occur below or above the minimum and maximum values). In the plot above, we can see that the Petal width for Setosa has 0.1 as the minimum value, 0.4 as the maximum value, 0.2 as both the median and lower quartile, 0.3 as the upper quartile, 0.4 as maximum values and two outliers (0.5 and 0.6).

3 Python Task 3 Solutions

Following is the python code to create parallel coordinate for iris data.

```
# Import required libraries
import pandas as pd
# import numpy as np
import matplotlib.pyplot as plt
# Importing mlines from matplotlib
import matplotlib.lines as mlines

# Read the iris dataset csv file as a dataframe
iris_df=pd.read_csv('https://raw.githubusercontent.com/sureshrgmi/Iris/
main/Iris.csv')

# Setting the figure size
plt.figure(figsize=(20 , 10))
# Specifying columns we want to plot
column_names = ["Sepal length", "Sepal width","Petal length", "Petal
↪  width"]
# Calue range for Y axis
max_value = 8
for i in range(0,4):
    plt.vlines(i, 0, max_value, 'black')
# Setting up plot parameters for each Species
for index, row in iris_df.iterrows():
    if(row['Species']=='Setosa'):
        plt.plot(column_names, row[column_names], color = 'blue',
        ↪  label='Setosa')
    elif (row['Species']=='Versicolor'):
        plt.plot(column_names, row[column_names], color = 'red',
        ↪  label='Versicolor')
    elif (row['Species']=='Virginica'):
        plt.plot(column_names, row[column_names], color = 'green',
        ↪  label='Virgincica')

# Variables and properties for legend
Setosa = mlines.Line2D([], [], color='blue', label='Setosa')
Versicolor = mlines.Line2D([], [], color='red', label='Versicolor')
Virginica = mlines.Line2D([], [], color='green', label='Virginica')
# Plot title
plt.title('Parallel Coordinates Plot for IRIS Data', fontsize=20,
↪  fontweight='bold')
#Plot Legend
plt.legend(loc=9, prop={'size': 15}, frameon=True,shadow=True,
↪  facecolor="white", edgecolor="black",handles=[Setosa, Versicolor,
↪  Virginica])
```

```
# X and Y axis labels
plt.xlabel('Features', fontsize=15)
plt.ylabel('Values', fontsize=15)
# Showing GridLines
plt.grid()


plt.show()
```

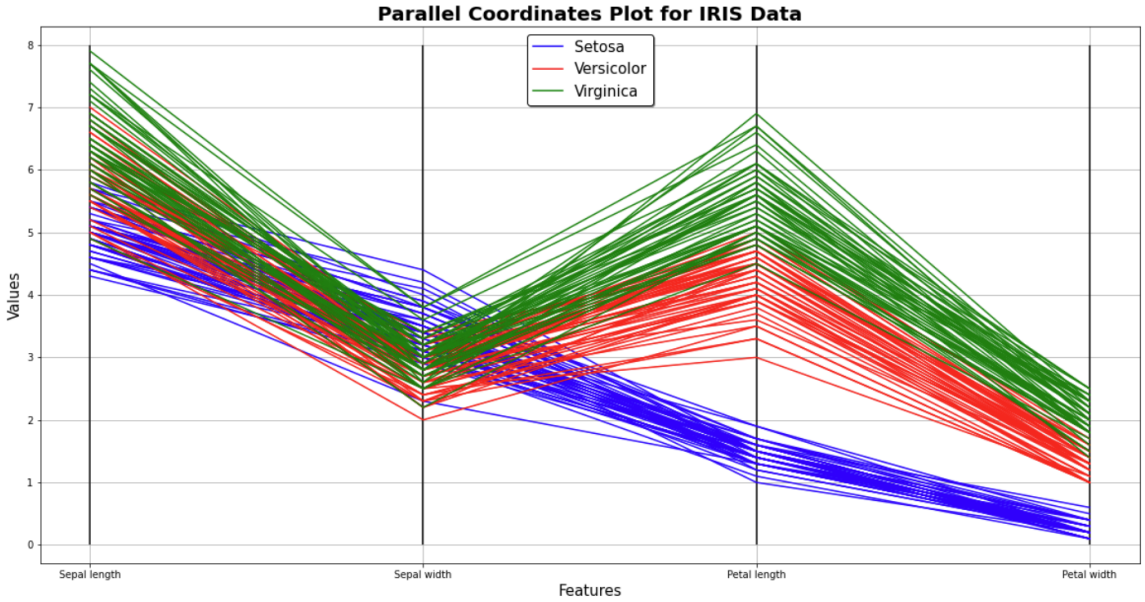

Figure 5: Parallel coordinate diagram for IRIS dataset

Explanation

By observing the parallel coordinates diagram, it is clear that the petal length and petal width are the properties that can be used to identify the species as there is a clear distinction between the values in the diagram with the former one showing the most. Out of these two properties, I would choose **Petal Length** to identify the species as the data is more distributed with a high range. From the diagram, we can say that the petal length for Setosa has lower range values (1 to 2), Versicolor has a middle range value (3 to 4.5) and Virginica has higher range values(4.5 to 7).

4 R Task 1 Solutions

4.1 Gathering columns from the who dataset

```
# Install Packages
install.packages("tidyverse")
```

```
# Import Libraries
library(tidyverse)

# List the columns from the "who" data frame
colnames(who)

# Gathering all columns using pivot_longer
who1 <- pivot_longer(who, 5:60, names_to = "key", values_to="cases",
values_drop_na= T )
```

Before we use any dataset from the tidy package, we need to make sure that the tidyverse package is installed and imported. The first two commands are for installing and importing the package. After that, we listed the columns from the data frame to determine the index of columns new_sp_m014 and newrel_f65 so that we can use them in the pivot longer function. We are storing the column names as keys, and values as cases and are removing values that are not available.

4.2 Making variable names consistent

```
# Creating a new data frame who2 from who1
who2 <- who1

# Making variable names consistent
who2$key <- str_replace(who2$key, "newrel", "new_rel")
```

In this implementation, the data frame who1 is copied into a new data frame who2 and the value from the key column is updated (substituted "newerl" by "new_rel") with the str_replace operation.

4.3 Pipe operation and comments

```
# Running the code
who3 <- who2 %>% separate(key, c("new", "type", "sexage"), sep = "_")

# Commenting the code
# who3 <- who2 %>% separate(key, c("new", "type", "sexage"), sep = "_")
```

Operator %>% also known as a pipe operator is used to pipe together a sequence of operations on a data frame. This operation feeds the result of one operation to another operation below it and makes the code easy to read. Commenting in R can be done by using "#" in front of any code like mentioned above. R does not have a different symbol for multi-line commenting (can be done by using multiple single-line comments).

4.4 Using separate function to split sexage int sex and age

```
# Separating sexage column into sex and age
who4 <- who3 %>% separate(sexage, c("sex", "age"), sep = "(?<=[A-Za-z])
(?=[0-9])")
```

4.5 Printing first 5 and last 5 rows

```
# Print first 5 rows of the data frame who4
head(who4, n=5)

# Print last 5 rows of the data frame who4
tail(who4, n=5)
```

```
> # Print first 5 rows of the data frame who4
> head(who4, n=5)
# A tibble: 5 × 9
  country     iso2  iso3   year new   type  sex   age   cases
  <chr>       <chr> <chr> <int> <chr> <chr> <chr> <chr> <int>
1 Afghanistan AF    AFG    1997 new   sp    m     014       0
2 Afghanistan AF    AFG    1997 new   sp    m     1524     10
3 Afghanistan AF    AFG    1997 new   sp    m     2534      6
4 Afghanistan AF    AFG    1997 new   sp    m     3544      3
5 Afghanistan AF    AFG    1997 new   sp    m     4554      5
> # Print last 5 rows of the data frame who4
> tail(who4, n=5)
# A tibble: 5 × 9
  country  iso2  iso3   year new   type  sex   age   cases
  <chr>    <chr> <chr> <int> <chr> <chr> <chr> <chr> <int>
1 Zimbabwe ZW    ZWE    2013 new   rel   f     2534   4649
2 Zimbabwe ZW    ZWE    2013 new   rel   f     3544   3526
3 Zimbabwe ZW    ZWE    2013 new   rel   f     4554   1453
4 Zimbabwe ZW    ZWE    2013 new   rel   f     5564    811
5 Zimbabwe ZW    ZWE    2013 new   rel   f     65      725
>
```

Figure 6: First and last 5 rows of who dataset

4.6 Exporting who4 as a CSV file

```
# Export who4 as a CSV file to the local directory
write.csv(who4, "path_to_directory/who4.csv", row.names=FALSE)
```

We can use the function write.csv to export the data frame to a CSV file. Here in this code, "path_to_directory" should be replaced with the actual path to the directory where you want to store the file.

5 R Task 2 Solutions

5.1 Calculating mean, median, mode, variance and standard deviation

```
# Install Packages
install.packages("tidyverse")
install.packages("modeest")

# Import Libraries
library(tidyverse)
library(modeest)

# View the Nile dataset
view(Nile)

# Calculating mean, median, mode, variance and standard deviation
# Mean
mean(Nile)
[1] 919.35

# Median
median(Nile)
[1] 893.5

# Mode
mfv(Nile)
[1]  845 1020 1100 1160

# Variance
var(Nile)
[1] 28637.95

# Standard Deviation
sd(Nile)
[1] 169.2275
```

There are separate functions on R to calculate the basic statistics variables. They all come as a part of the **tidyverse** package. However, to calculate the Mode value, we need to install and import another library called **modeest**.

5.2 Computing how "spread out" the data are

```
# Calculate Minimum Value
min(Nile)
```

```
[1] 456

# Calculate Maximum Value
max(Nile)
[1] 1370

# Calculate Range
max(Nile, na.rm=TRUE) - min(Nile, na.rm=TRUE)
[1] 914
```

5.3 Calculating the interquartile range (IQR)

```
# Finding interquartile range (IQR)
IQR(Nile)
[1] 234

# Computing Quantiles
quantile(Nile)
    0%    25%    50%    75%   100%
 456.0  798.5  893.5 1032.5 1370.0
```

The interquartile range (IQR) is the difference between the first and third (lower and upper) quartiles in our dataset, the quantile() function on the other hand divides the data into four parts (quartiles).

5.4 Histogram using built-in R basic functions

```
# Histogram using built-in R functions

hist(Nile,
     main="Histogram showing flow of the river Nile",
     xlab=bquote("Annual flow (10"^"8"*" m"^"3"*")"),
     ylab="Frequency",
     col="blue",
     freq=TRUE
)
```

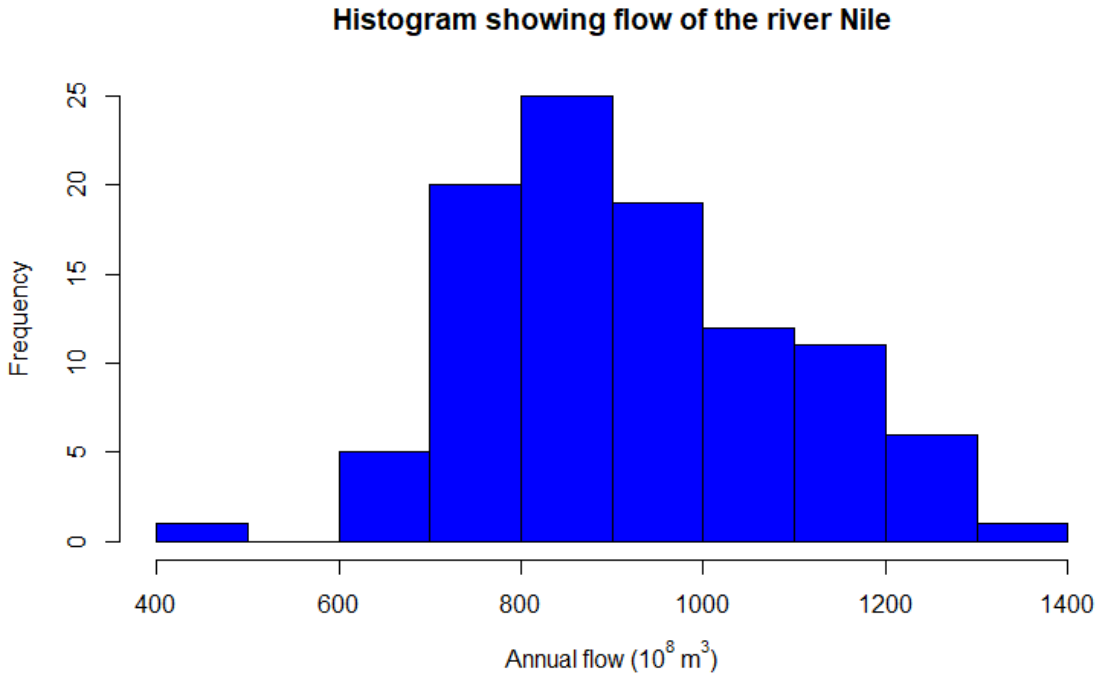

Figure 7: Histogram for Nile Dataset

5.5 Creating quantile-quantile plot

```
# Q-Q Plot using qqnorm and qqline

qqnorm(Nile, pch=1, main = "Flow of Nile River (Q-Q plot)",
       frame=TRUE, col="blue", xlab = "Quantiles",
       ylab = bquote("Annual flow (10"^"8"*" m"^"3"*")"),
       plot.it = TRUE, datax = FALSE)
qqline(Nile, col="red", lwd=2)
```

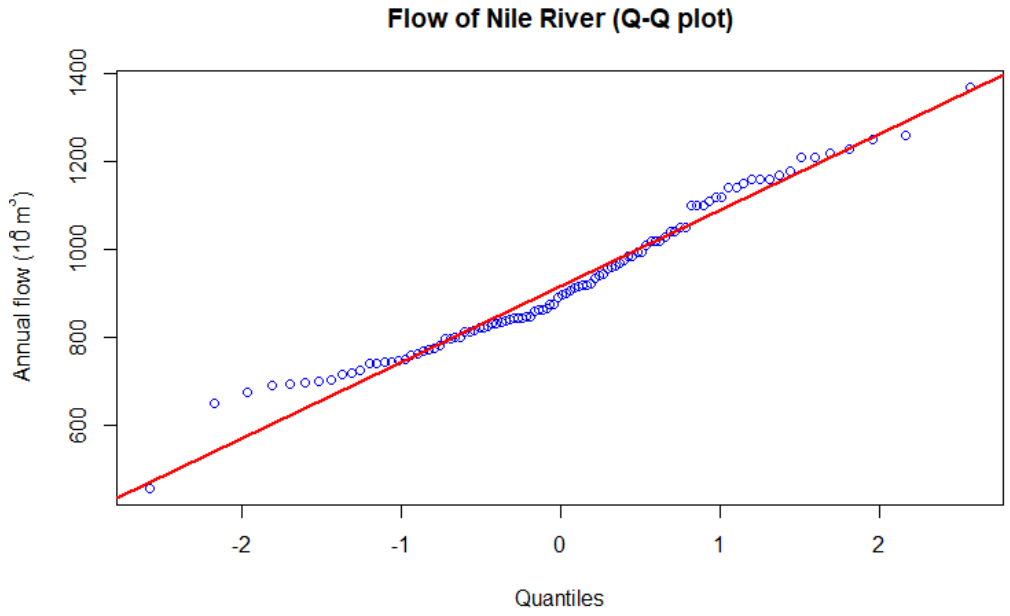

Figure 8: Q-Q Plot

The Q-Q plot (also known as the Quantile-Quantile plot) is the scatter plot between two sets of variables (Theoretical quantiles and Sample Quantiles). The purpose of the line on the Q-Q plot is to provide a reference for two sets of distribution. On our plot, we can see that most of the data points are along the reference and have some minor deviations on the tails. Overall, we can say that our data is normally distributed. From the plot, we could also say that our dataset has low kurtosis with light tails.

5.6 Further exploring the dataset

To further explore the dataset and implement the arguments such as xlab, ylab, main and type, we have created a line plot for the time series Nile dataset. Here, the arguments xlab and ylab are used to set the X and Y axis labels respectively. The main is used to set the title of the plot while the type is used to see what type of plot we want. In our case, we have set the type to b which means the plot should show both points and lines.

```
plot(Nile,
     xlab = "Year",
     ylab = bquote("Annual flow (10"^"8"*" m"^"3"*")"),
     main = "Annual flow of Nile (1871-1970)",
     type = "b")
```

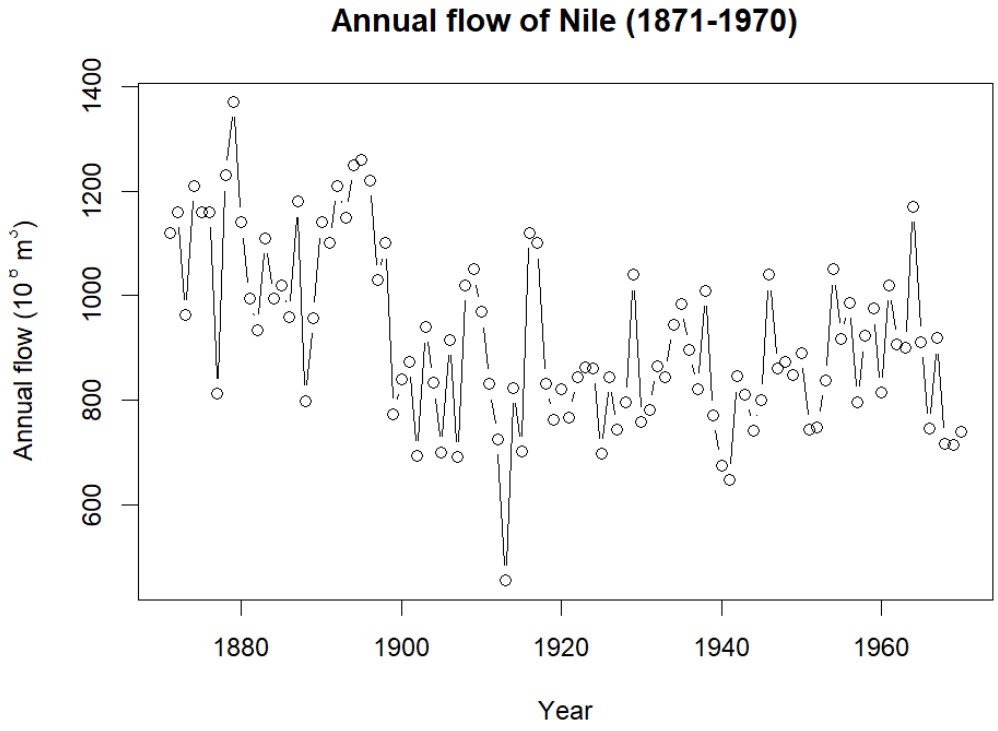

Figure 9: Line plot for time series Nile data

6 R Task 3 Solutions

6.1 Vehicle brand offering best mpg in both city and highway

```
# Import Libraries
library(tidyverse)
library(ggplot2)

# City vs Highway mpg
sp <- ggplot(mpg, aes(x = cty, y = hwy, colour = factor(manufacturer)))+
  geom_point(size=5) + ggtitle("Highway VS City (MPG)") +
  theme(plot.title = element_text(hjust = 0.5))+ xlab("City MPG") +
  ylab("Highway MPG") + scale_color_discrete("Manufacturers")

sp
```

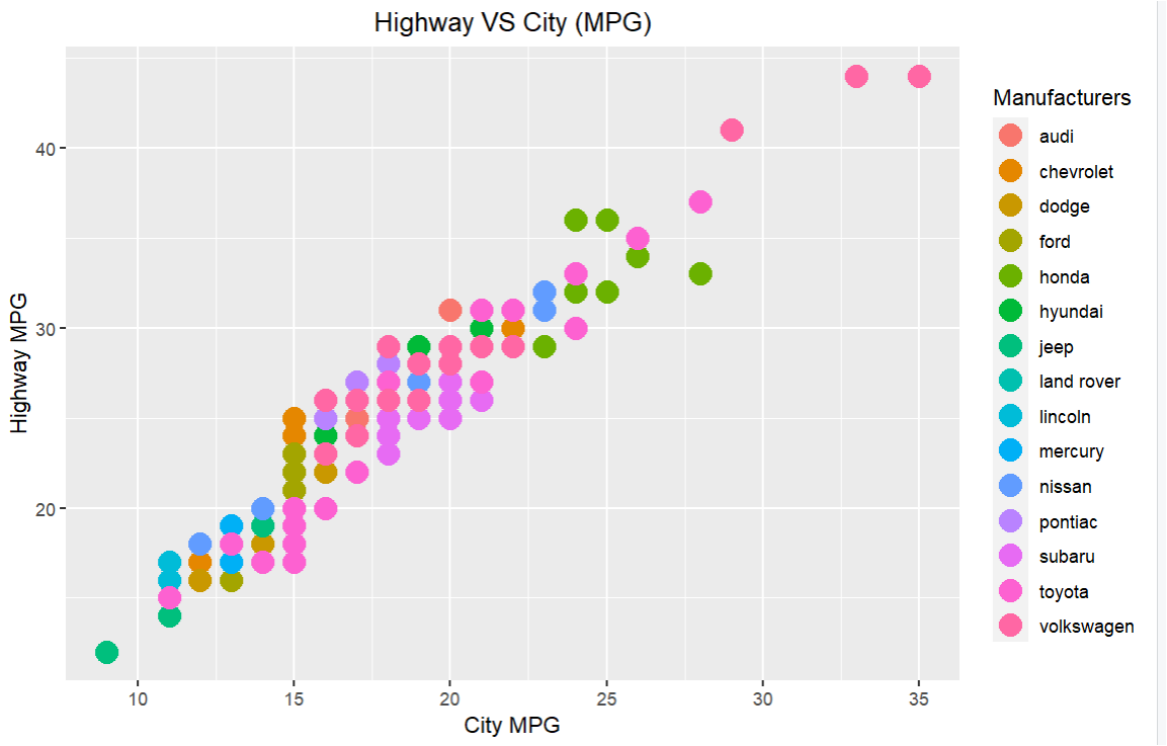

Figure 10: Scatter plot showing City vs Highway MPG

In this plot, we have created a scatter plot of Highway mpg (Hwy) against the City mpg (Cty) from the mpg dataset. We have categorized the points based on the manufacturers and used a different colour for each of them. From the plot, we can see that Volkswagen has the highest values for both city and highway mpg. However, it also has some data points where both the highway and city mpg are not good. Honda, on the other hand, doesn't have the highest mpg values, but, has most of its values in the higher range. Let's explore it further by plotting the average mpg value rather than plotting each point.

```
# Creating a new dataframe with the average value
agg_mpg <- aggregate(cbind(cty,hwy) ~ manufacturer, data = mpg,
FUN = mean, na.rm = TRUE)

# City vs Highway mpg (Average Value)
sp <- ggplot(agg_mpg, aes(x = cty, y = hwy, colour = factor(manufacturer)))+
  geom_point(size=5) + ggtitle("Highway VS City (Average MPG)") +
  theme(plot.title = element_text(hjust = 0.5))+ xlab("City MPG") +
  ylab("Highway MPG") + scale_color_discrete("Manufacturers")

sp
```

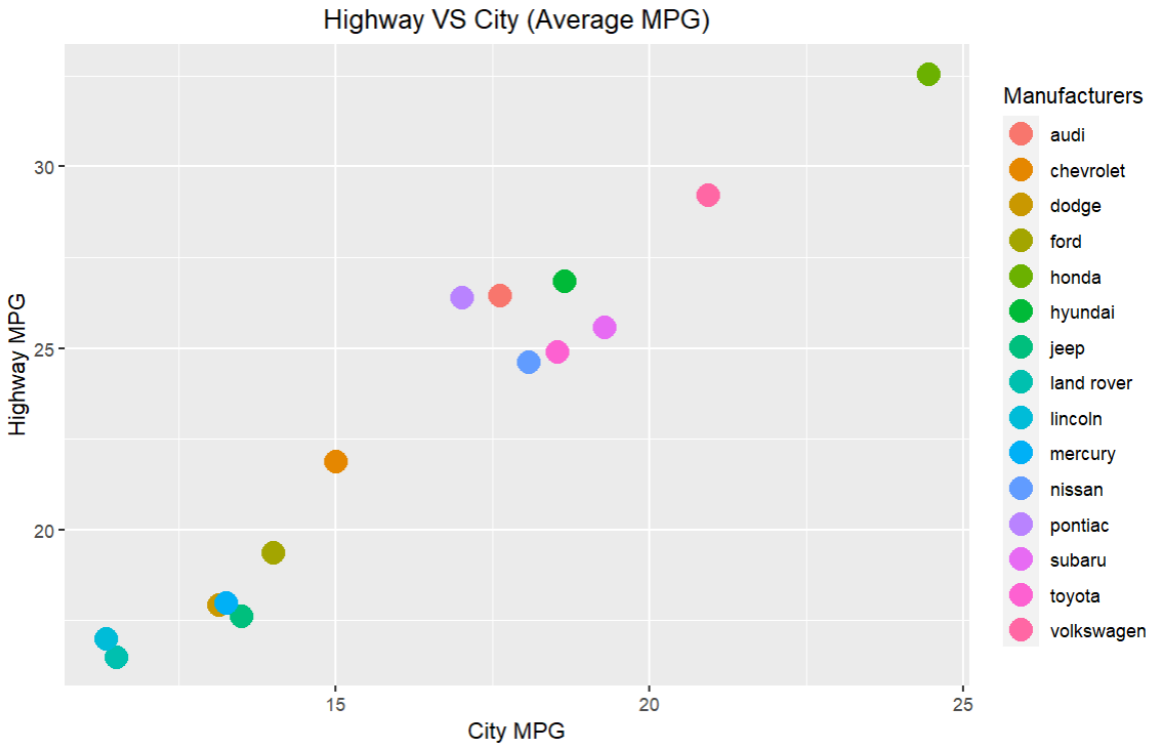

Figure 11: Scatter plot showing City vs Highway MPG (Average)

Here, we can clearly see that Honda provides the best average mpg in both city and highway by a large margin while Volkswagen comes in second. So, by taking average into the consideration, Honda is the best one.

6.2 Type of car with lowest city mpg

```
# Plotting City MPG vs Displ for different classes (Using facet_warp)
p <- ggplot(data = mpg, aes(x = displ, y = cty, color=class,
                            shape=cty<15)) +
  geom_point()+geom_point(color='darkblue')+
  scale_color_discrete("Class") +
  scale_shape_discrete("City MPG < 15")
p + facet_wrap(~class) +
  ggtitle("City MPG Vs Displ for different classes")+
  theme(plot.title = element_text(hjust = 0.5))+
  xlab("Displacement") +
  ylab("City MPG")+geom_smooth(formula = y ~ x)
```

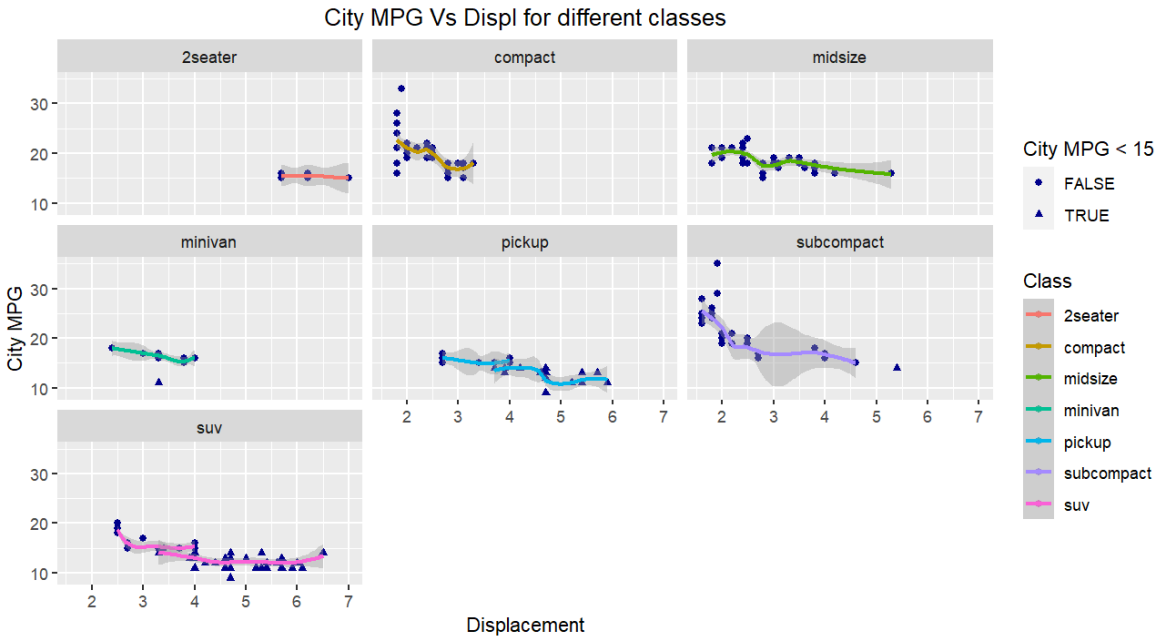

Figure 12: City MPG vs Displacement for different classes

In this case, the plot we have created contains the comparison of City MPG with the displacement value for different classes of vehicles. The points are plotted with different shapes based on the city mpg values which makes it easier to identify the low mileage values. Looking into the plot, we can see that Pickup and SUV are the classes of vehicles with significant no of low mileage records. By further comparing these two classes, we can say that **SUV** has the lowest city mileage values (no. of triangles) even for the higher range of displacement values.

6.3 Type of car with best city and highway mpg

```
# City vs Highway MPF for different Classes
plot <- ggplot(mpg, aes(x= cty, y = hwy, color=class,
                        shape= displ>4)) +
  geom_point()+ scale_color_discrete("Class") +
  scale_shape_discrete("Displacement > 4")
plot + facet_grid(vars(drv), vars(cyl))+
  ggtitle("City MPG Vs Highway MPG different Classes")+
  theme(plot.title = element_text(hjust = 0.5))+
  xlab("City MPG") + ylab("Highway MPG")
```

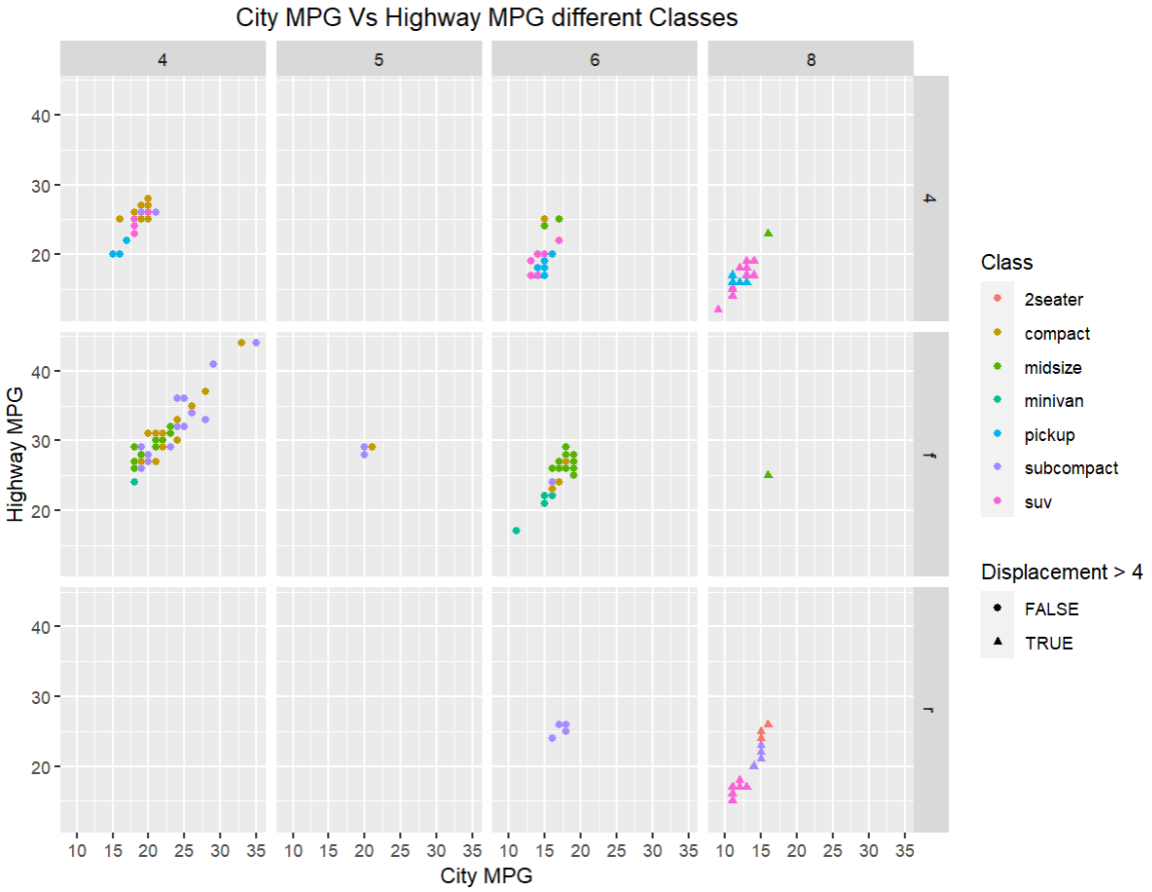

Figure 13: City MPG Vs Highway MPG different Classes

In the plot above, we have presented the City and Highway MPG based on the different classes of the vehicle (2-seaters, compact, midsize etc) and the displacement size (different shapes based on displacement size). For the first question (best MPG in both City and Highway?), the best type of car would be the **Front-Wheel Drive Subcompact type with 4 cylinders and a small Engine size (<4)**. In our plot, we can see that the Subcompact type of vehicle (purple points on our plot) has a high overall mileage in both the city and the highway. Even for large engine sizes, they show a fairly good performance compared to others.

For the second part (high-engine size vehicle for highway), I would choose **2-seater vehicle class with rear-wheel drive and 8 cylinders**. From our plot, we can see that for larger displacement values (displ>4), that combination has the higher range of mpg(around 25) in highway.