Programming Fundamentals

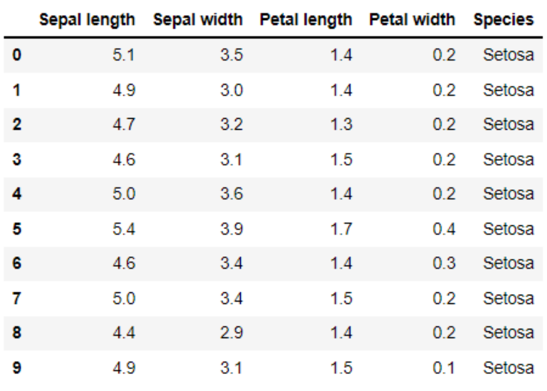
Raheel Shaikh

Part 1. Solution to Python portfolio tasks

Task 1 solution:

```
# importing necessay libraries
import pandas as pd
# Reading the 'iris.csv' file and creating a data frame. The 'iris.csv' is
saved in the working directory
df = pd.DataFrame(data=pd.read_csv("iris.csv"), index=None)
# Displaying the first 10 row.
df.head(10)
```

output:



	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 row of data frame

```
# Creating a dataframe
df = pd.DataFrame(data=data, index=None)

# Getting the petal width for each species.
petalWidth_setosa = df.loc[df['Species'] == "Setosa",'Petal width']
petalWidth_versicolor = df.loc[df['Species'] == "Versicolor",'Petal width']
petalWidth_virginica = df.loc[df['Species'] == "Virginica",'Petal width']

# Mean & Median of Setosa
setosa_mean = petalWidth_setosa.mean()
setosa_median = petalWidth_setosa.median()

# Mean & Median of Versicolor
versicolor_mean = petalWidth_versicolor.mean()
versicolor_median = petalWidth_versicolor.median()

# Mean & Median of Virginica
virginica_mean = petalWidth_virginica.mean()
virginica_median = petalWidth_virginica.median()

# Printing the result
print("Measure of Central Tendency:")
print(f'Versicolor: Mean = {round(versicolor_mean,3)}  Median =
{versicolor_median}')
```

```
print(f'Setosa: Mean = {round(setosa_mean,3)}  Median = {setosa_median}')
print(f'Virginica: Mean = {round(virginica_mean,3)}  Median =
{virginica_median}')

std_setosa = round(petalWidth_setosa.std(),3)
std_versicolor = round(petalWidth_Versicolor.std(),3)
std_virginica = round(petalWidth_Virginica.std(),3)

# Computing the range by using .max() - .min()
range_setosa = petalWidth_setosa.max()-petalWidth_setosa.min()
range_versicolor = petalWidth_Versicolor.max()-petalWidth_Versicolor.min()
range_virginica = petalWidth_Virginica.max()-petalWidth_Virginica.min()

# Printing the results
print("Measure of Dispersion:")
print(f'Setosa: Standard deviation = {std_setosa}  Range = {range_setosa}')
print(f'Versicolor: Standard deviation = {std_versicolor}  Range =
{range_versicolor}')
print(f'Virginica: Standard deviation = {std_virginica}  Range =
{range_virginica}')
```

output:

```
Measure of Central Tendency:
Versicolor: Mean = 1.326  Median = 1.3
Setosa: Mean = 0.244  Median = 0.2
Virginica: Mean = 2.026  Median = 2.0

Measure of Dispersion:
Setosa: Standard deviation = 0.107 and Range = 0.5
Versicolor: Standard deviation = 0.198 and Range = 0.8
Virginica: Standard deviation = 0.275 and Range = 1.1
```

Task 2 solution:

Box plot also as known as the whisker plot is one of the best plots to represent one dimension data. This plot di splay as summary of the data. Properties like minimum and maximum data points, mean, median, first quartile and thirds quartile are easily identifiable. The graph is usually plotted between the 1st and 3rd quartile. While, a line passes through the median of the graph. The x-axis denoted the data that is being plotted while the y-axi s denoted the frequency of this data (Government of Canada & Canada,2021).

```
# importing necessay libraries
import seaborn as sb

# Getting the desired columns
petalWidth_virginica = df.loc[df['Species'] == "Virginica", ['Petal
width','Species']]

# onverting the data into a dataframe
dataset = pd.DataFrame(data=petalWidth_virginica)

# Using boxplot to visualize the data
sb.boxplot(x="Species",y="Petal width", data=dataset,
whis=3.0).set(title='Plot for petal width of Virginica Species');
```

output:

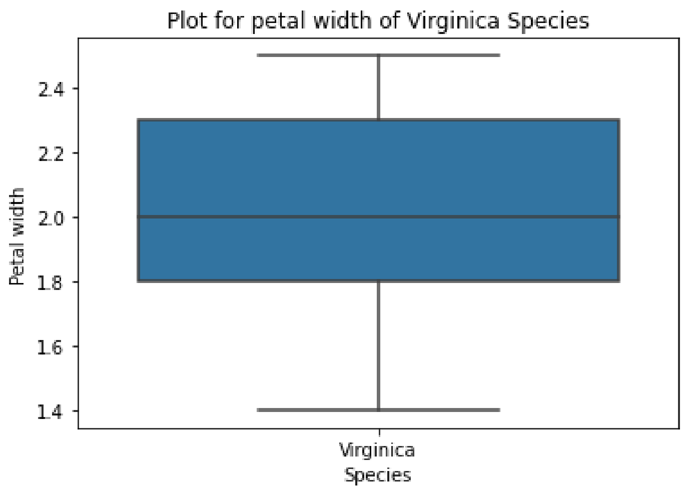


Figure 2: Box plot of Petal width vs Virginica Species

From the above graph we can identify that the median of the petal width for the Virginica species is 2 cm and the minimum and maximum value is 1.4 cm and 2.5 cm respectively. We can also identify the first quantile i.e., 1.8 cm and the third quantile i.e., 2.3 cm.

Task 3 solution:

```
# importing necessay libraries
import matplotlib.pyplot as plt

# using pandas plotting.parallel_coordinates
# setting alpha = 0.2 to adjust the transparency of the plot and identify
overlapping.
pd.plotting.parallel_coordinates(df, "Species", color=["#EF2648",
"#5A8A91","#FBC800"],alpha=0.4);

# Adding a title to the graph
plt.title("IRIS Flowers Parallel Coordinates Plot")

# Adding label to the x and y axis
plt.xlabel("Iris Flower attributes")
plt.ylabel("Measure of length and width")
```

output:

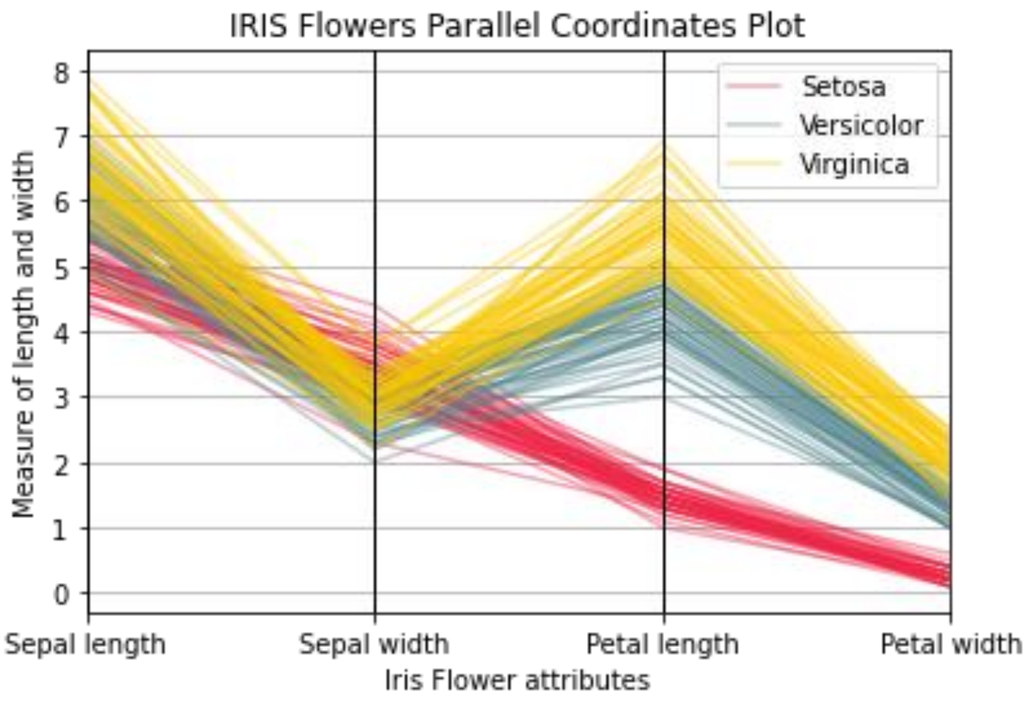


Figure 3: Parallel Coordinates plot of 'IRIS' data set

We see in the plot there is dense overlapping of data points for all three species of flowers in the case of sepal length. Similarly, we can see a considerable amount of coinciding point for sepal width. Whereas, in the case of petal length and petal width there is very less overlapping of data point for all three species and hence best suited for classification. To conclude Petal length is the best property for all three species that can be used to carry out accurate classification since the data points have very litter or in case ‘setosa’ zero overlapping.

Part 2. Solution to R portfolio tasks

```
# Importing necessary libaries in the start
library("DescTools")
library("dplyr")
library("ggplot2")
```

Task 1 solution:

1.
   ```
   who1 = who %>% pivot_longer(cols=c('new_sp_m014': 'newrel_f65')
   names_to='key',values_to='cases',values_drop_na = TRUE)
   ```

2.
   ```
   who2$key  <- stringr::str_replace(who1$key,'newrel','new_rel')
   ```

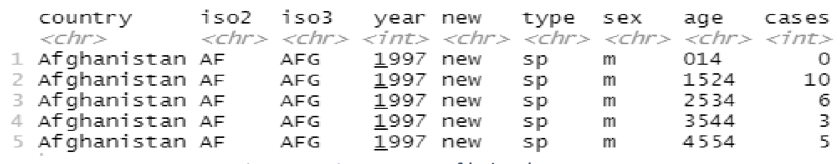
3.
   ```
   who3 = who2 %>% separate(key, c("new", "type", "sexage"), sep = "_")
   ```

   '%>%' know as the pipe operator belongs to the 'magrittr' package. What his operator does is that it passes the value or the result of function/expression into another function/expression. Though the effectiveness of this may not be very evident here but it is very useful when we have multiple function call and each function uses the out of the previous function as its parameter(s) (Boehmke,2016).

4.
   ```
   who4 <-who3 %>% separate(sexage, c("sex", "age"),sep = 1)
   ```

5.
   ```
   head(who4,5)
   ```

   output:

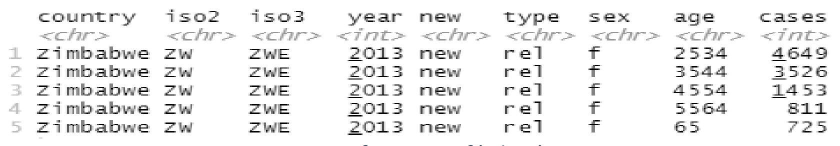


```
  country     iso2  iso3  year  new   type  sex   age   cases
  <chr>       <chr> <chr> <int> <chr> <chr> <chr> <chr> <int>
1 Afghanistan AF    AFG    1997 new   sp    m     014       0
2 Afghanistan AF    AFG    1997 new   sp    m     1524     10
3 Afghanistan AF    AFG    1997 new   sp    m     2534      6
4 Afghanistan AF    AFG    1997 new   sp    m     3544      3
5 Afghanistan AF    AFG    1997 new   sp    m     4554      5
```

Figure 4: First 5 rows of 'who4'

   ```
   tail(who4,5)
   ```

   output:



```
  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 5: Last five rows of 'who4'

6.
   ```
   write.csv(who4,"who4.csv", row.names = FALSE)
   ```

Task 2 solution:

1.

```
# Using the in built
mean()
Nile_mean = mean(Nile)
       print(Nile_mean)
```
output:
```
[1] 919.35
```

```
# Using the in built
median()
Nile_median =
median(Nile,na.rm =
FALSE)
print(Nile_median)
```
output:
```
[1] 893.5
```

```
# Using Mode() from
DescTools
Nile_mode = Mode(Nile)
print(Nile_mode)
```
output:
```
[1]  845 1020 1100 1160
attr(,"freq")
[1] 3
```

```
# Using Var() for
variance
Nile_var = var(Nile)
print(Nile_var)
```
output:
```
[1] 28637.95
```

```
#standard deviation
using StdDev()
Nile_std =
StdDev(Nile)
print(Nile_std)
```
output:
```
[,1]   StdDev 169.2275
```

2.

```
# Using min()
min_nile = min(Nile)
print(min_nile)
```
output:
```
[1] 456
```

```
# Using max()
max_nile = max(Nile)
print(max_nile)
```
output:
```
[1] 1370
```

```
# Range max - min
range_nile = max_nile -
min_nile
print(range_nile)
```
output:
```
[1] 914
```

3.

```
# Using in built IQR()
IQR(Nile,na.rm = FALSE)
```
output:
```
[1] 234
```

```
# Using in built quantile()
quantile(Nile,na.rm = TRUE)
```
output:
```
   0%    25%    50%    75%   100%
456.0  798.5  893.5 1032.5 1370.0
```

The Interquartile range (IQR) is the measure of difference between the upper (75%) and lower (25%) quartile, i.e., it tells is how far apart the middle portion of data spreads in value. Whereas, the quantile function gives us all the quantiles in not specified.

4.
```
# Using the in built hist() to plot a histogram
hist(Nile,main ="Measurements of the annual flow of the river Nile",xlab
= "volume of water(10^8 Cubic meter)",col=' #2ADFC6')
```
output:

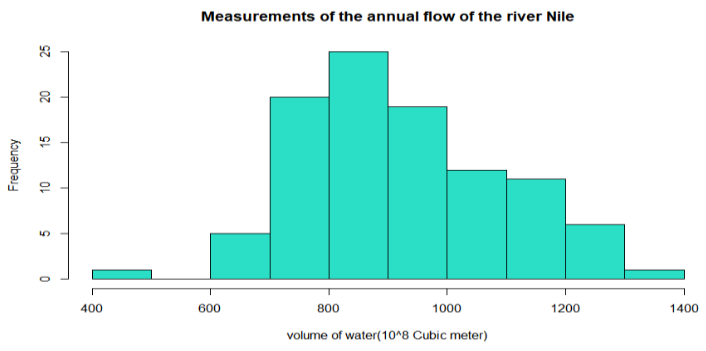


Figure 6: Histogram of the Nile dataset

From the above graph we can see that the highest volume of water recorded for the river Nile was between 800-900 cubic meters. Whereas the lowest is between 400-500 cubic meters.

5.

```r
# Using qqnorm to plot the QQ normal plot of values in y
qqnorm(Nile)
# Using qqline to draw a line to “theoretical” a by default normal line
qqline(Nile,col = "red",lwd=2)
```

output:

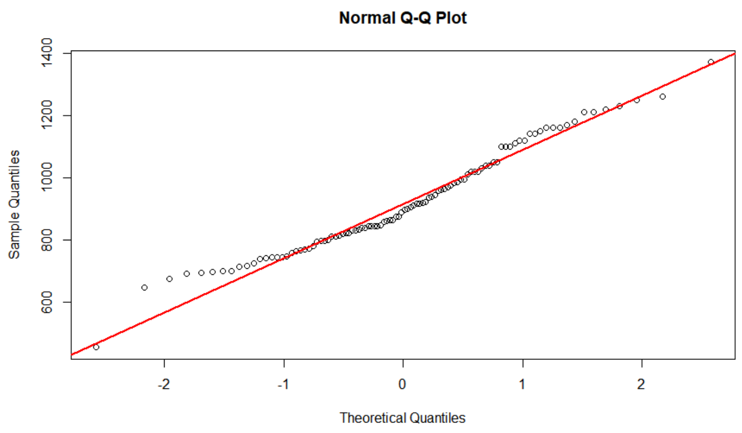


Figure 7: Normality plot for the Nile dataset

The Q-Q plots help us to find out if a data set comes from some theoretical distribution such as normal distribution. Two sets of quantiles are plotted against one another to generate a scatterplot known as a Q-Q plot. And a theoretical red line is drawn which is by default normal. From the above graph we can infer that the data is not normally distributed and as the scatter plot doesn’t not follow the normal line.

6.

```r
plot(Nile,main="Measurements of the annual flow of the river Nile",xlab =
"Years",ylab = "Volume of water(10^8 cubic meter)")
```

output:

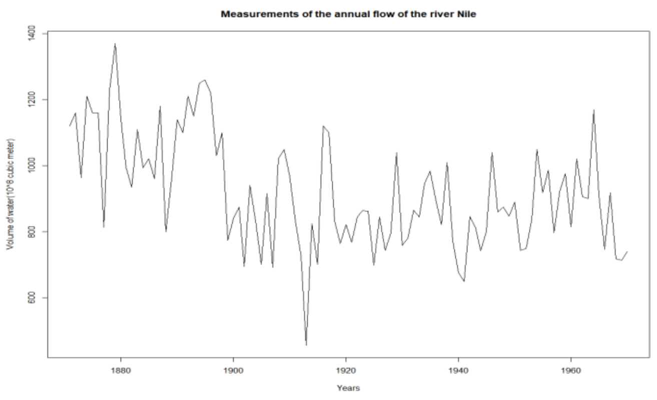


Figure 8: Plot of the Nile Data set (Volume of water vs Years)

As we can see there a lot of variation in the flow of the river Nile. From the above plot we can infer that in the year 1800 the flow of the river Nile was the highest little less than 1400 cubic meters and around the year 1914 the flow of the river was the lowest i.e., under 500 cubic meters.

Task 3 solution:

1.

```
data = ggplot2::mpg
# Calculating the average mpg for each vehicle across both city and highway
data$Avg_mpg = rowMeans(cbind(data$cty,data$hwy),na.rm = TRUE)
View(data)
# Calcualting the average mpg for each brand
AvgBrand_mpg = data %>%
  group_by(manufacturer) %>%
  summarise(Avg_mpg = mean(Avg_mpg))
# Plotting a bar chart to best represent our findings
ggplot(data=AvgBrand_mpg, aes(x=manufacturer, y=Avg_mpg)) +
  geom_bar(stat="identity")+
  ggtitle("Vehicle brand efficiency in mpg across both city and highway")+
labs(y=" Average MPG (city and highway)",x="Manufacturer")
```

output:

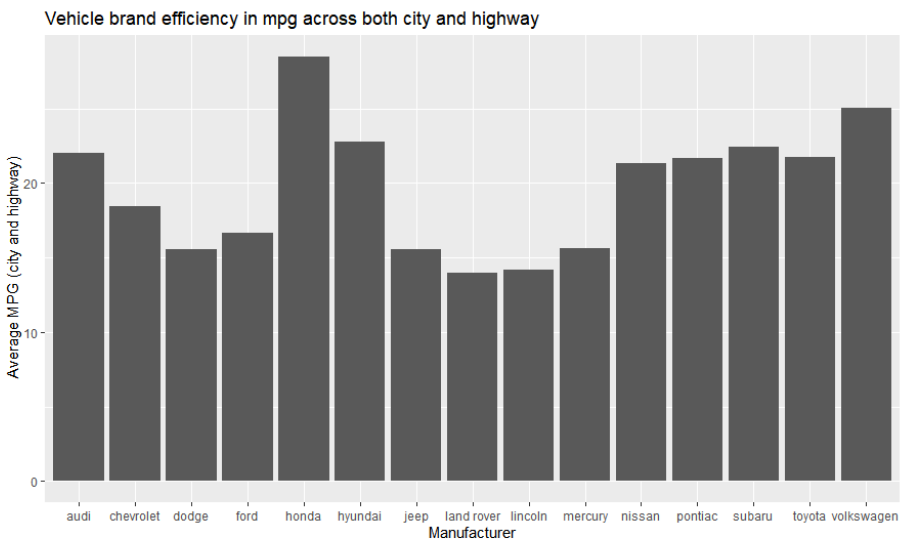


Figure 9: Bar chart showing the mpg efficiency of each vehicle brand

For the above plot we can infer that 'Honda' had the highest average mpg efficiency, and therefore we can conclude that it offers the best mpg in both city as well as highway.

2.

```
# using ggplot with facet_wrap to plot mpg in city of vehicle with
different engine sizes.
ggplot(data = data) +
  geom_point(mapping = aes(x = displ, y = cty, color=class)) +
  facet_wrap(~ class, nrow = 2)+
  ggtitle("Engine size vs mpg in city - (categorised based on Vehicle type
)")+
  labs(y="MPG in city",x="Engine size")
```

output:

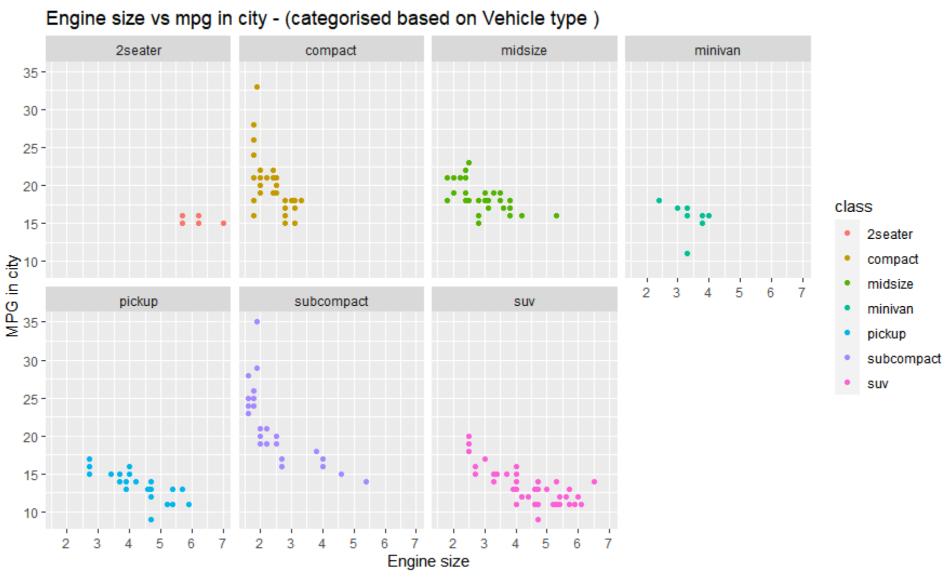


Figure 10: Scatter plot measuring mpg of different vehicle in the city with different engine sizes

From the plot we can tell that the vehicle type "SUV" has the lowest mpg in the city since, the pink dots in the "SUV" category are closely clustered to the lower end of the y-axis and higher end of the x-axis. Thus, indicating lower mpg and lager engine size.

3.

```
# using ggplot with facet_grid to plot mpg in city and highway of vehicle
with different engine sizes and number of cylinders.
  ggplot(data = data) +
  geom_point(mapping = aes(x = cty, y = hwy , color=displ)) +
  facet_grid(cyl ~ drv )+
  scale_color_gradient(low="red", high='green')+
  ggtitle("mpg in city vs mpg on highway  - (categorised based on drive
type,No.of cylinders and engine size )")+
  labs(y="MPG on highway",x="MPG in city")
```

output:

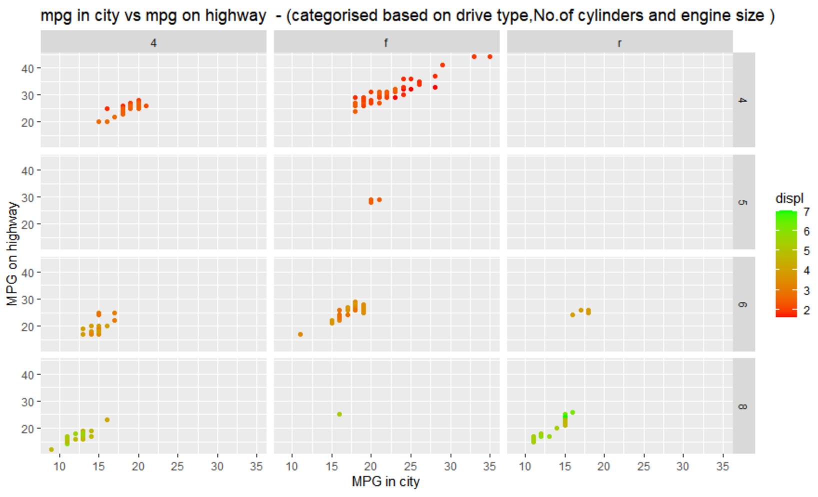


Figure 11: Scatter plot measuring mpg of different vehicle in the city with different engine sizes and cylinder

From the above plot we can infer that the cars with '4 cylinders' and 'front wheel' drive have the best mpg in both the city and the highway. The best vehicle that one can choose for highway with a high litre engine is a rear wheel drive with 8 cylinders.

References

Boehmke, B. C. (2016). Simplify your code with %>%. In *Use R!* (pp. 199–207). Springer International Publishing.

Government of Canada, & Canada, S. (2021, September 2). *4.5.2 Visualizing the box and whisker plot*. Statcan.Gc.Ca. https://www150.statcan.gc.ca/n1/edu/power-pouvoir/ch12/5214889-eng.htm