Data Visualiation in Tidyverse

Monika Baloda

January 10, 2023

This R-markdown file uses *tidyverse* package in R to do data visualization exercise. In particular, it does the following:

Part I : Using `dimond` price dataset, we use some advance plots

1. Heatmaps and Changing colors of heatmap
2. Histogram, Boxplot, Frequency Polygon, and Violin plots of tidyverse.

Part II : We use earthquake data `quakes` of R to visualize it.

1. Plotting the distribution of earthquake magnitudes using tidyverse. In particular, we plot
   i) histogram ii) boxplot iii) empirical cdf iv)Q-Q plot.
2. We plot earthquakes point on top of a map layer.

Part III : We use `mpg` dataset to use cool visualization functions of tidyverse

1. Aesthetic mapping of `color`
2. Different type of smoothing curves and colored points using group identity of a variable.
3. Using *Facets* function to add additional variable(s) to a 2D plot.
4. Playing with `stat` function.
5. Position adjustment options for 'geom_bar()

Part IV : In-depth visualization of `mpg` dataset

1. Barplot, Coxcomb, Pie Charts
2. How highway mileage varies across drive train type : ordered median and coordination flips to compare.

Load necessary packages

```
# install the tidyverse package first if you have not done it yet.
#install.packages("tidyverse") # you can comment out this line after you have installed `tidyverse`

library(tidyverse) # for the `ggplot2` package
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.1     v readr     2.1.4
```

```
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.0     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

For rendering in PDF If you don't want to render (knit) the file in PDF format, you can ignore this block of code.If you face problem in rendering, please refer to debugging on https://yihui.org/tinytex/r/#debugging .

```
#tinytex::reinstall_tinytex()
```

Main Analysis Starts From Here

***Part I: Visualization the `diamonds` data set** This data set contains the prices and other attributes of almost 54,000 diamonds.

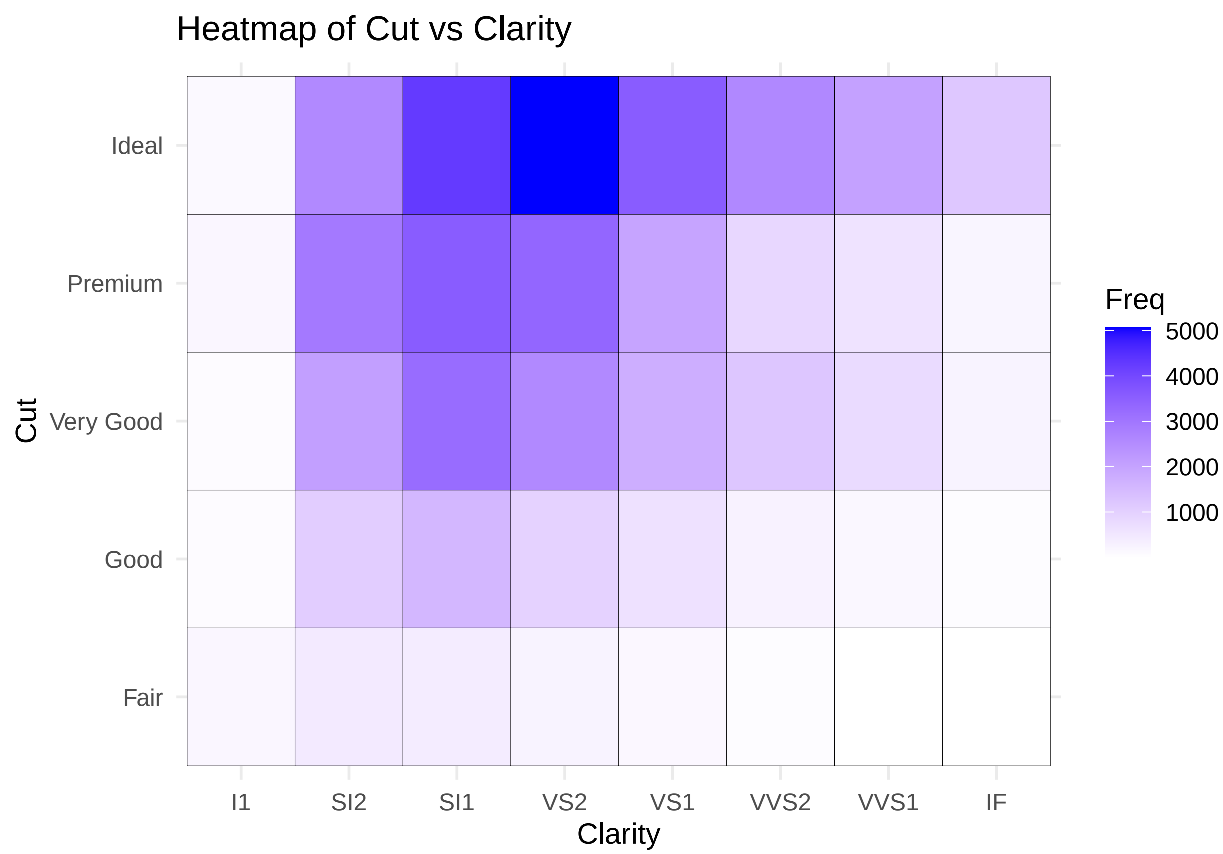
A quick look at the dataset I'm commenting most of it because my rendered PDF is getting too long.

```
#?diamonds
dim(diamonds)  # dimension of the table
## [1] 53940    10
#diamonds # print/view diamonds
#str(diamonds)  # list the structures in diamonds
#glimpse(diamonds) # get a glimpse of the data
```

a) Heatmap of `cut` vs `clarity`

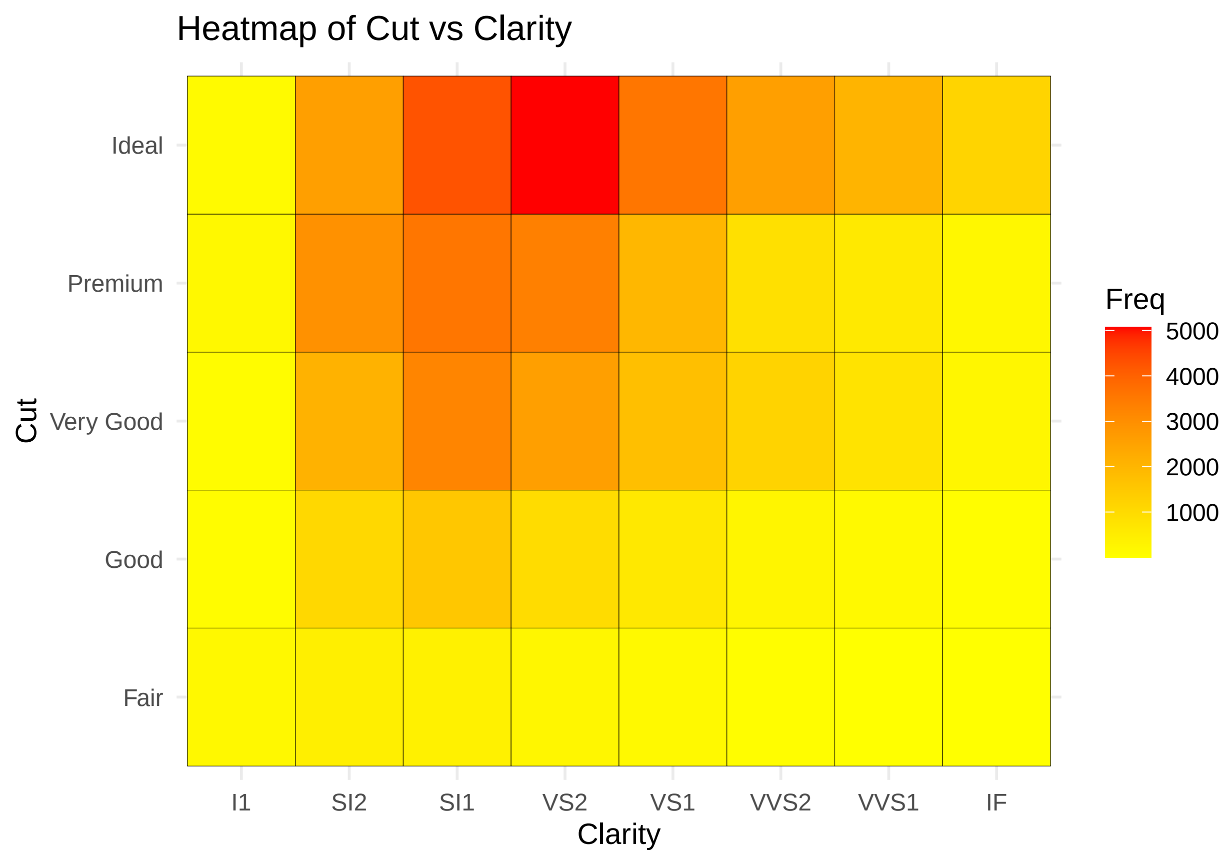
(i) Using the `geom_tile()` function to make a heatmap to visualize the number of diamonds in each `cut` and `clarity` combination.

```
# to calculate the frequency of each combination of cut and clarity
cut_clarity_freq <- with(diamonds, table(cut, clarity))

# Plotting the heatmap
ggplot(as.data.frame(cut_clarity_freq), aes(clarity, cut, fill = Freq)) +
geom_tile(color = "black") +
scale_fill_gradient(low = "white", high = "blue") +
labs(x = "Clarity", y = "Cut", title ="Heatmap of Cut vs Clarity") +
theme_minimal()
```



(ii) Change the color palette of your heatmap.

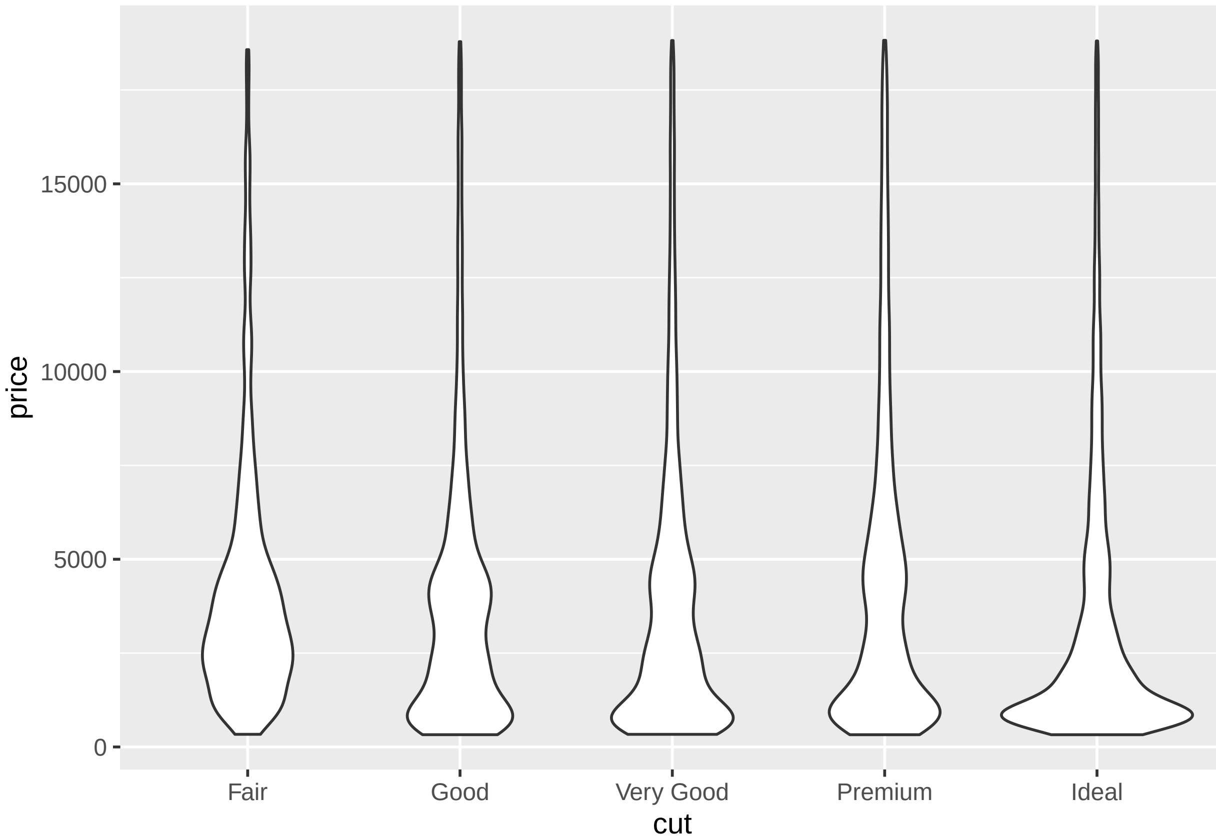
```
ggplot(as.data.frame(cut_clarity_freq),aes(clarity, cut, fill = Freq))+
geom_tile(color = "black") + scale_fill_gradient(low = "yellow", high = "red") +
labs(x = "Clarity", y = "Cut", title = "Heatmap of Cut vs Clarity") +
  theme_minimal()
```



b) Visualize the distribution of diamond price

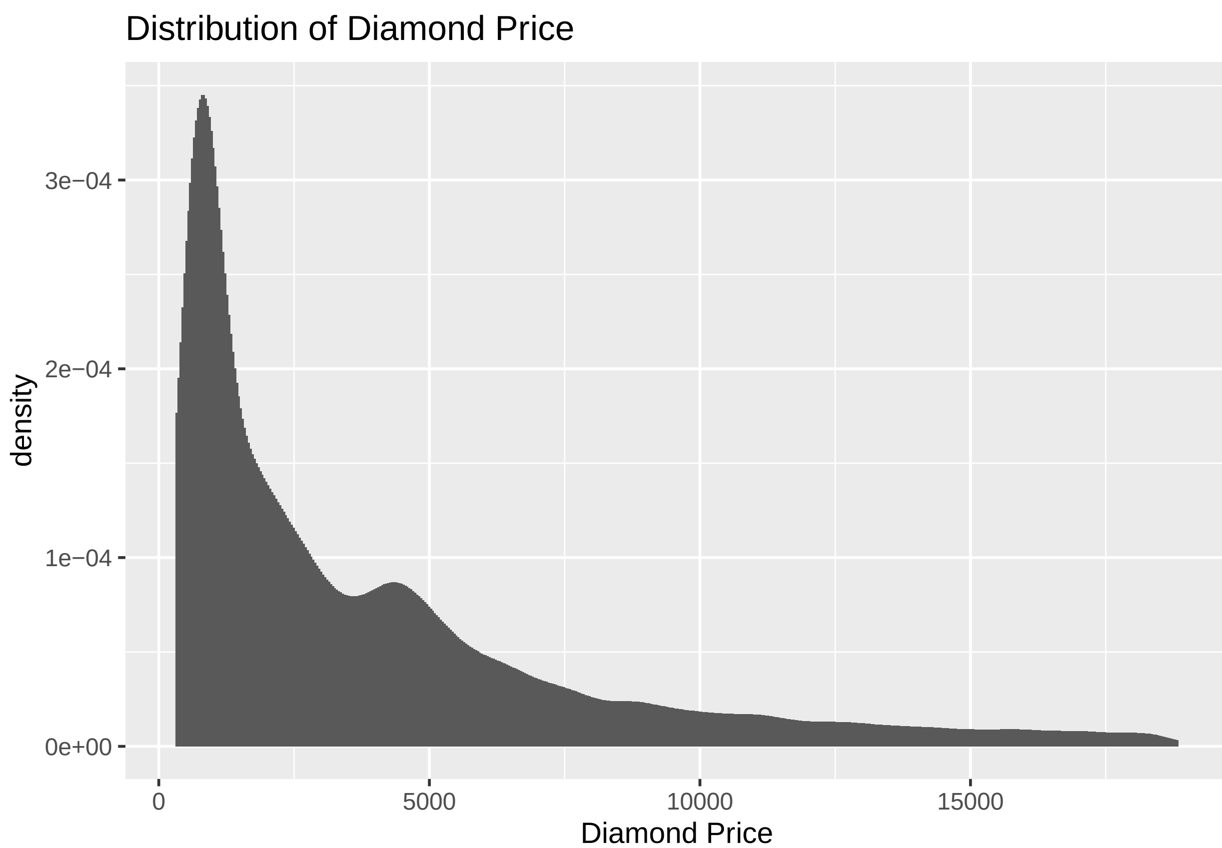
(i) Using the `geom_violin()` function to compare the distribution of `price` across different `cut`.

```
library(ggplot2) #library is loaded
ggplot(diamonds, aes(x = cut, y = price)) +geom_violin() #creates a violin plot(diamonds, is the data f
```



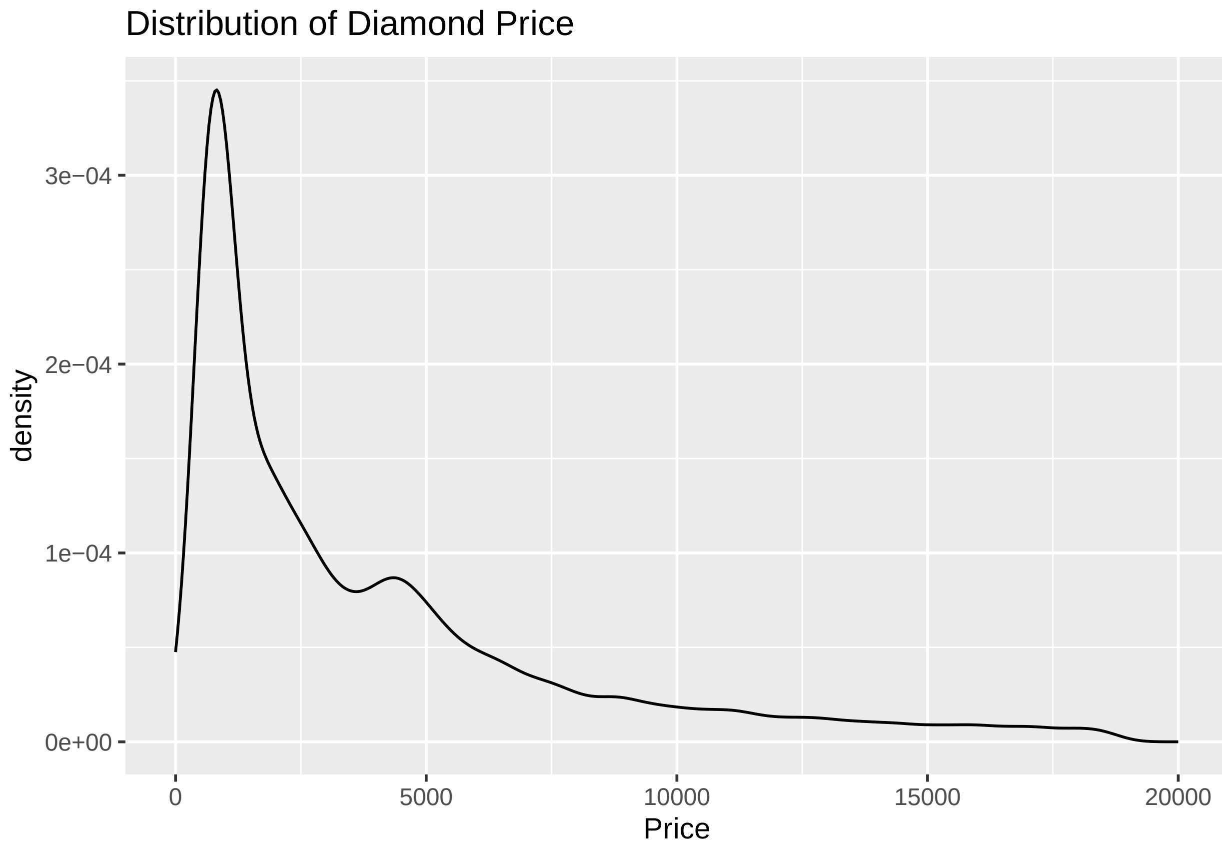
(ii) Using the `geom_histogram()` function to compare the distribution of `price` across different `cut`. Change the y-axis to density, and use the `dodge` position adjustment.

```
ggplot(diamonds, aes(price)) + geom_histogram(binwidth = 500, stat='density', position = 'dodge') +
labs(x = "Diamond Price", y = "density") +
  ggtitle("Distribution of Diamond Price")
## Warning in geom_histogram(binwidth = 500, stat = "density", position =
## "dodge"): Ignoring unknown parameters: `binwidth`, `bins`, and `pad`
```



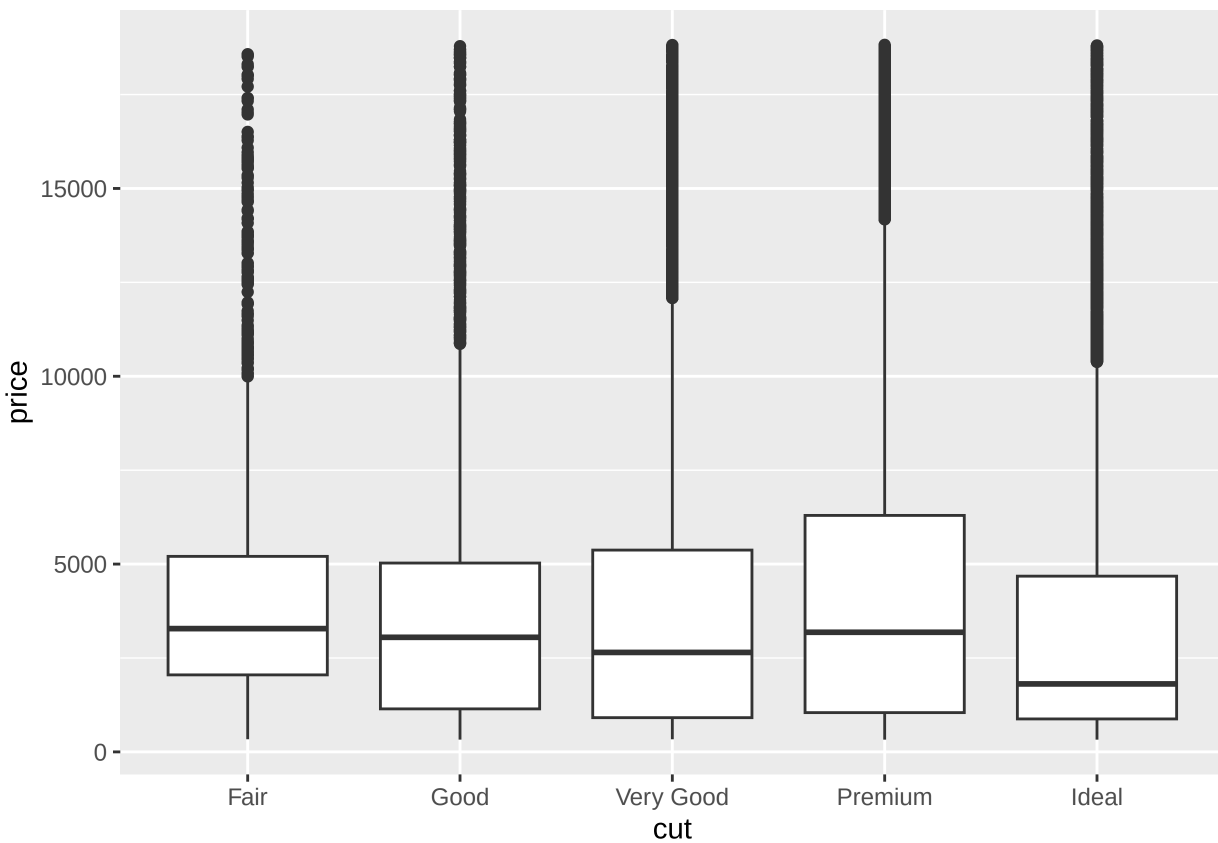
(iii) Using the `geom_freqpoly()` function to compare the distribution of `price` across different `cut`. Change the y-axis to density.

```
ggplot(diamonds, aes(price)) + geom_freqpoly(binwidth = 500, stat='density') +
xlim(c(0, 20000)) + xlab("Price") + ylab("density")+
ggtitle("Distribution of Diamond Price")
## Warning in geom_freqpoly(binwidth = 500, stat = "density"): Ignoring unknown
## parameters: `binwidth`
```



(iv) Using the `geom_boxplot()` function to compare the distribution of `price` across different `cut`.

```
ggplot(diamonds, aes(x = cut, y = price)) +
  geom_boxplot()
```



Question What observations can you make from the above plots? Which visualization function is your favorite? Explain your choice.

ANSWER In the plots mentioned above, we can observe the distribution of the diamond prices and how it varies depending on attributes like cut, clarity, and color. The histogram and frequency polygon gives us an idea of the density of the prices and how they are spread across various values. The boxplot give us median, quartiles and outliers of the data set, usually helps in comparing multiple datasets because easy comparison of range and median values can be done due to box shape. Violin plots have character of both box plot with kernel density plot to show us full density distribution of the data. In violin plot, shape of data distribution includes skewness and multiplicity with range and median values. So, box plot is good if we want to understand whole distribution of one dataset. It is convenient to use both plots to provide more information. At last, in my opinion, if we want quick summary of data distribution, then box plot is a good choice. However, if we need more detailed understanding of density and shape of distribution then violin plot is better. And if we have a large number of observation, then a histogram or frequency polygon can help us to understand overall distribution of the data.**

Part II: Visualization the `quakes` data set in tidyverse In our data visualization with Base R, we used the `quakes` data set contain the locations of 1000 seismic events of MB > 4.0. The events occurred in a cube near Fiji since 1964.

```
library(datasets)
?quakes
```

```
## starting httpd help server ... done
```

```
class(quakes)
## [1] "data.frame"
head(quakes, n=5) # print first 5 rows of quakes
##      lat   long depth mag stations
## 1 -20.42 181.62   562 4.8       41
## 2 -20.62 181.03   650 4.2       15
## 3 -26.00 184.10    42 5.4       43
## 4 -17.97 181.66   626 4.1       19
## 5 -20.42 181.96   649 4.0       11
dim(quakes)  # dimension of the table
## [1] 1000    5
names(quakes)  # list the variables in quakes
## [1] "lat"      "long"     "depth"    "mag"      "stations"
str(quakes)  # list the structures in quakes
## 'data.frame':    1000 obs. of  5 variables:
##  $ lat     : num  -20.4 -20.6 -26 -18 -20.4 ...
##  $ long    : num  182 181 184 182 182 ...
##  $ depth   : int  562 650 42 626 649 195 82 194 211 622 ...
##  $ mag     : num  4.8 4.2 5.4 4.1 4 4 4.8 4.4 4.7 4.3 ...
##  $ stations: int  41 15 43 19 11 12 43 15 35 19 ...
glimpse(quakes) # get a glimpse of the quakes data
## Rows: 1,000
## Columns: 5
## $ lat      <dbl> -20.42, -20.62, -26.00, -17.97, -20.42, -19.68, -11.70, -28.1~
## $ long     <dbl> 181.62, 181.03, 184.10, 181.66, 181.96, 184.31, 166.10, 181.9~
## $ depth    <int> 562, 650, 42, 626, 649, 195, 82, 194, 211, 622, 583, 249, 554~
## $ mag      <dbl> 4.8, 4.2, 5.4, 4.1, 4.0, 4.0, 4.8, 4.4, 4.7, 4.3, 4.4, 4.6, 4~
## $ stations <int> 41, 15, 43, 19, 11, 12, 43, 15, 35, 19, 13, 16, 19, 10, 94, 1~
```

a) Plotting the distribution of earthquake magnitudes

Writing `ggplot2` code to reproduce the following four subfigures in a 2-by-2 layout.

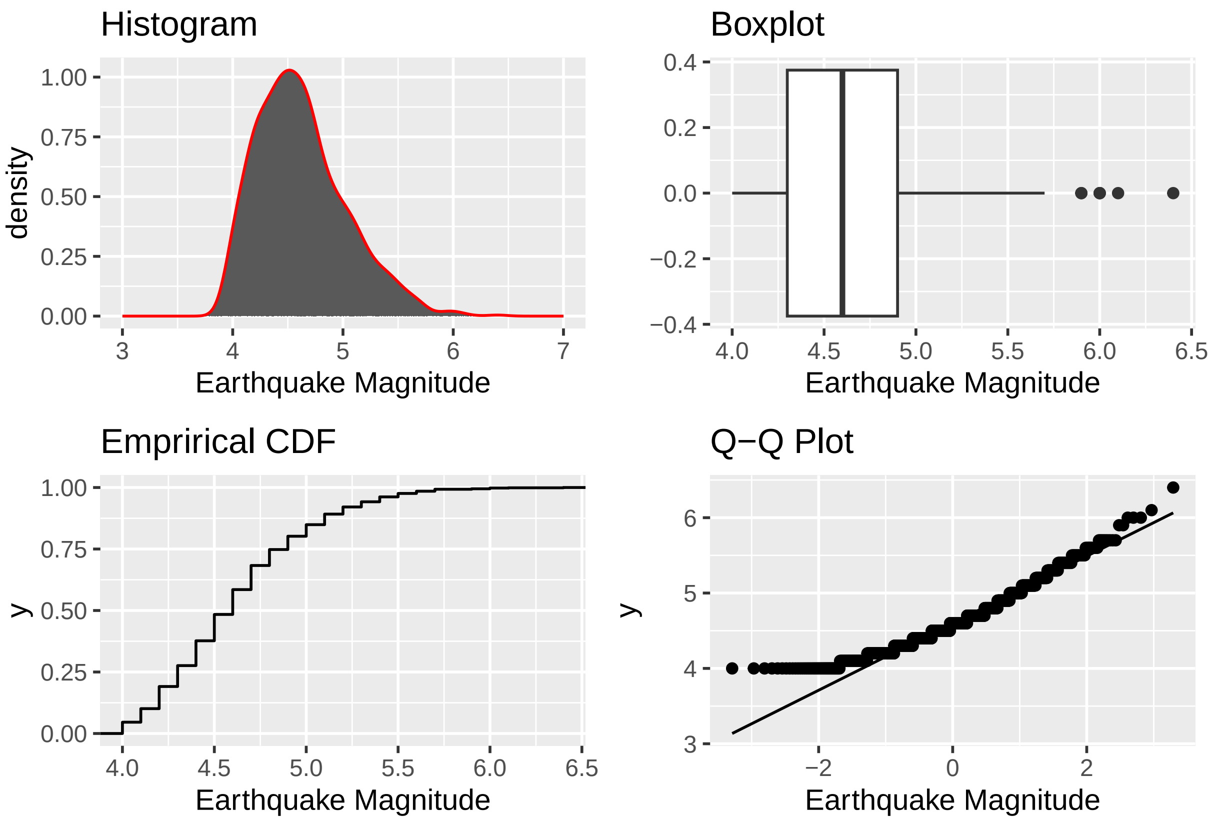
- subfigure #1: plot a density histogram of the earthquake magnitudes, and then plot the estimated probability density curve in red color in the same plot
- subfigure #2: plot a horizontal boxplot of the earthquake magnitudes
- subfigure #3: plot the empirical cdf of the earthquake magnitudes
- subfigure #4: make a Q-Q plot to compare the observed earthquake magnitudes distribution with the Normal distribution. Add a *thick* Q-Q line in blue color.

```
library(maps)
```

```
##
## Attaching package: 'maps'

## The following object is masked from 'package:purrr':
##
##     map
```

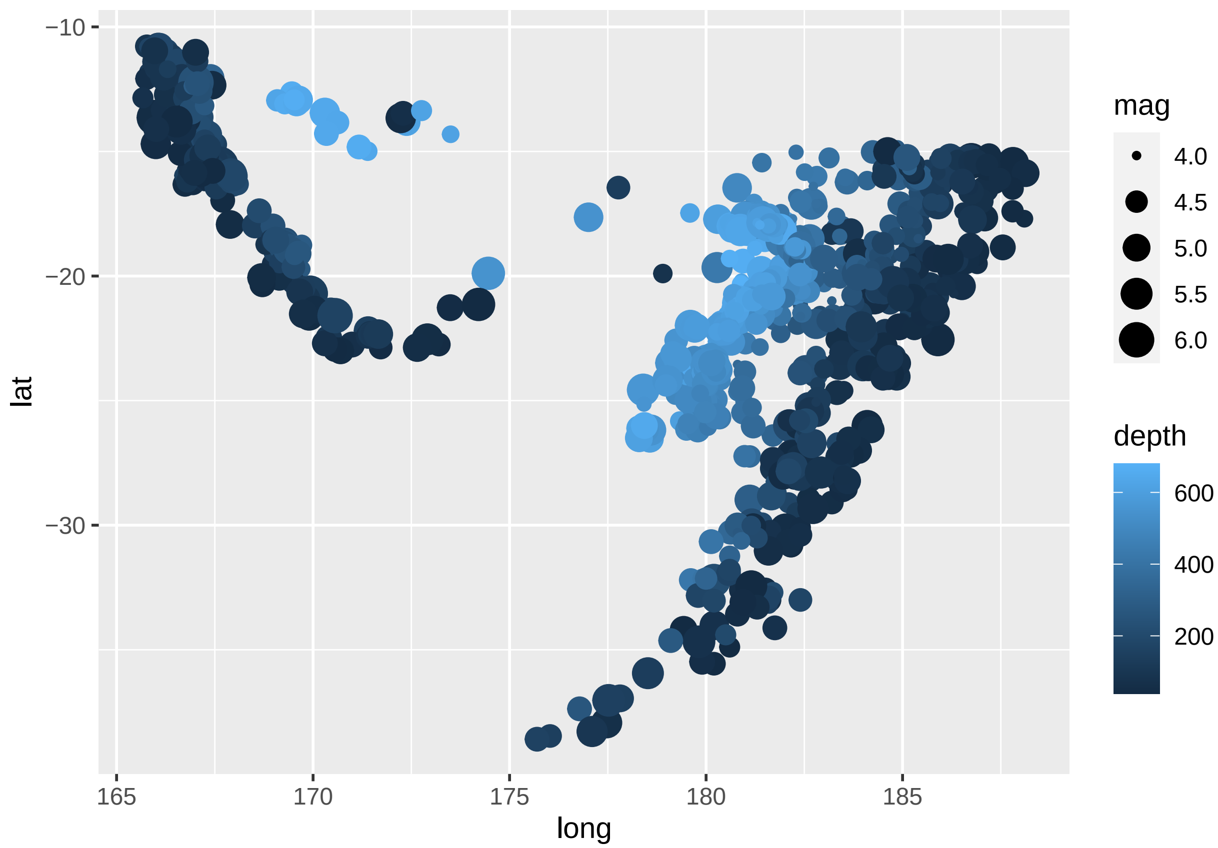
```
library(datasets)
library(patchwork)
```

```
par(mfrow=c(2,2))

#subfigure-1
plot1=ggplot(data=quakes)+
geom_histogram(aes(mag), stat="density")+
xlab("Earthquake Magnitude")+
ggtitle("Histogram")+
geom_density(aes(mag), col="red")+
xlim(c(3,7))


#subfigure-2
plot2=ggplot(data=quakes)+
geom_boxplot(aes(mag))+
xlab("Earthquake Magnitude")+
ggtitle("Boxplot")


#subfigure-3
plot3=ggplot(data=quakes)+
stat_ecdf(aes(mag))+
xlab("Earthquake Magnitude")+
ggtitle("Emprirical CDF")

#subfigure-4
plot4=ggplot(data=quakes,mapping=aes(sample=mag))+
geom_qq()+ geom_qq_line()+
xlab("Earthquake Magnitude")+
ggtitle("Q-Q Plot")

plot1+plot2+plot3+plot4
```



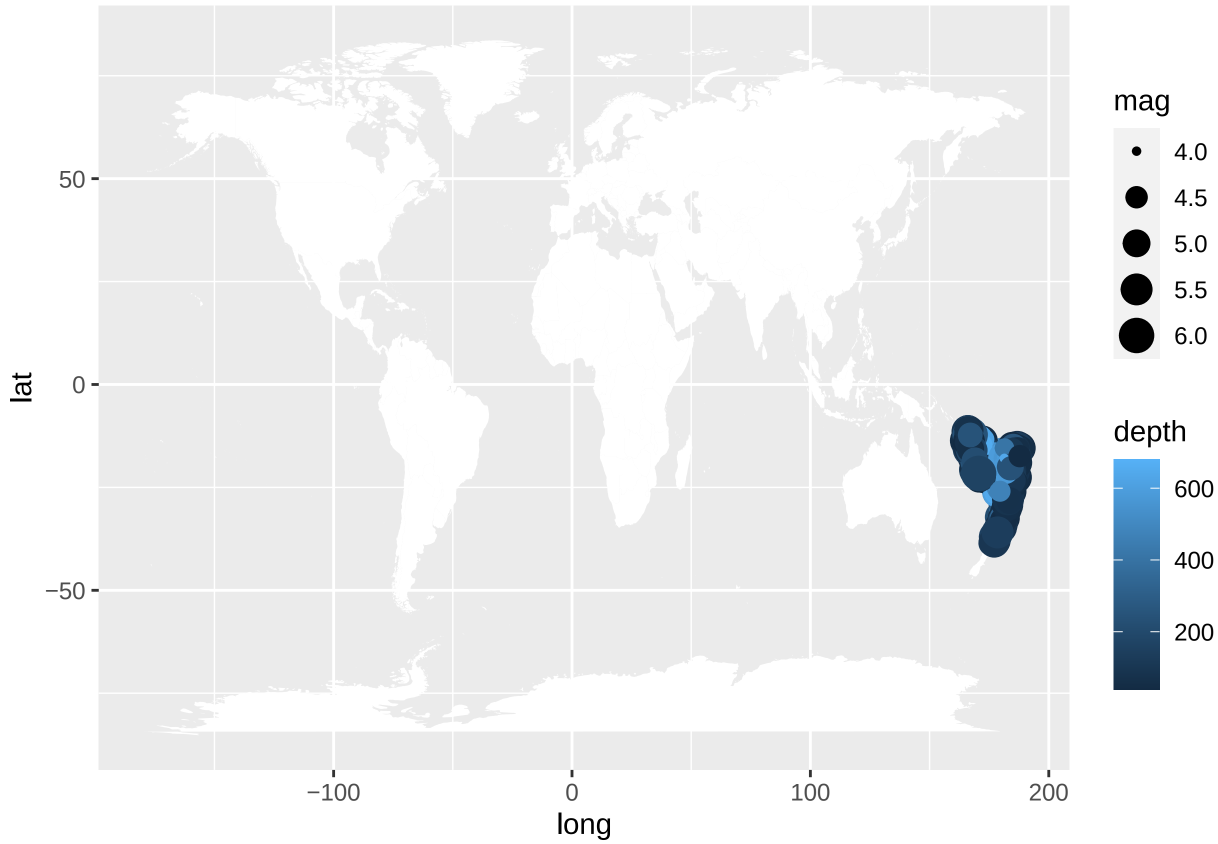
b) Earthquake location map Scatter plot of the earthquake locations. Use `long` as the x-axis and `lat` as the y-axis. Map `mag` to the size aesthetic and `depth` to the color aesthetic.

```
ggplot(data=quakes, mapping=aes(x = long, y = lat, colour=depth, size=mag))+
geom_point()
```



c) Plotting earthquakes point on top of a map layer

```
library(maps)
wc=map_data("world")

ggplot()+
geom_map(data=wc, map=wc, aes(long,lat, map_id=region), fill="white")+
geom_point(data=quakes, mapping=aes(x = long, y = lat, colour=depth, size=mag))
```



Part III : Visualization of the `mpg` data set This data set contains fuel economy data 1999 - 2008 for 38 popular car models (source: EPA http://fueleconomy.gov).

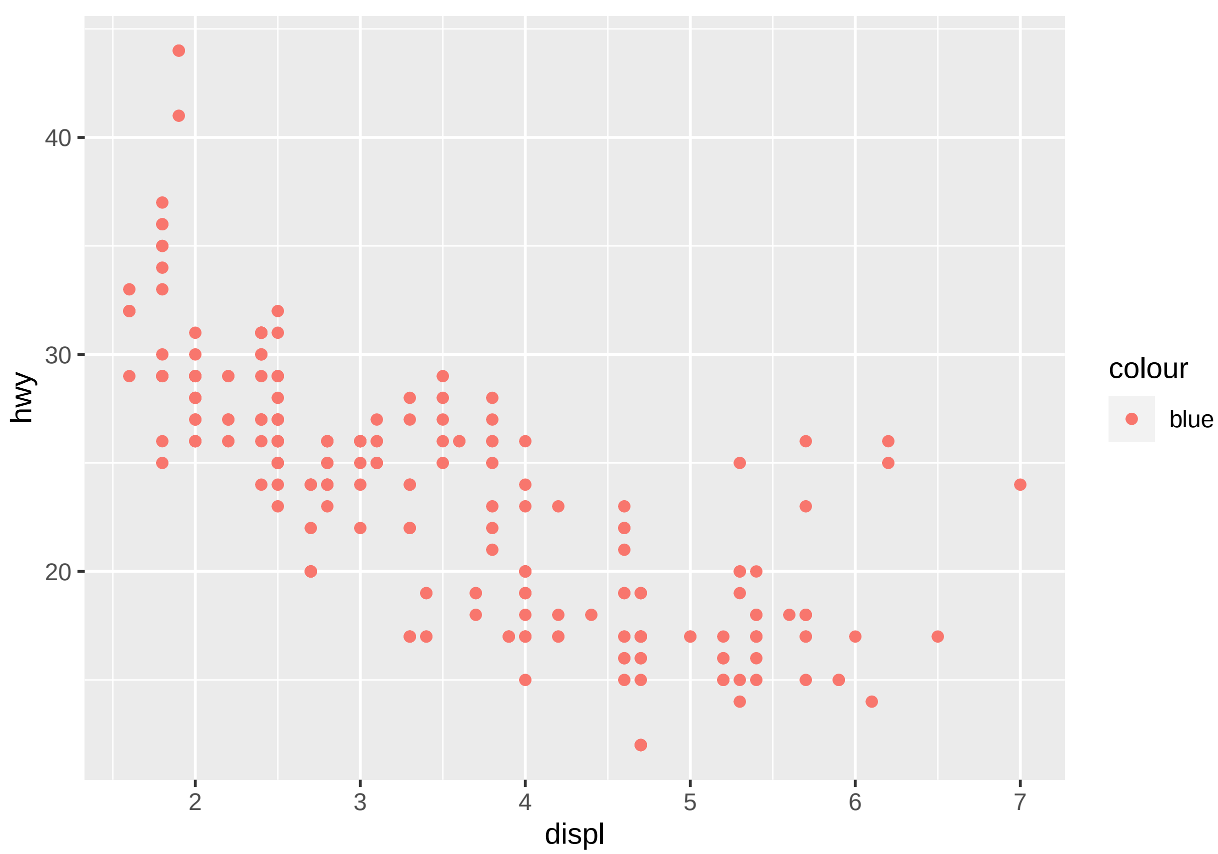
```
?mpg
dim(mpg)  # dimension of the table
## [1] 234  11
names(mpg)  # list the variables in mpg
##  [1] "manufacturer" "model"        "displ"        "year"         "cyl"
##  [6] "trans"        "drv"          "cty"          "hwy"          "fl"
## [11] "class"
str(mpg)  # list the structures in mpg
## tibble [234 x 11] (S3: tbl_df/tbl/data.frame)
##  $ manufacturer: chr [1:234] "audi" "audi" "audi" "audi" ...
##  $ model       : chr [1:234] "a4" "a4" "a4" "a4" ...
##  $ displ       : num [1:234] 1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
##  $ year        : int [1:234] 1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
##  $ cyl         : int [1:234] 4 4 4 4 6 6 6 4 4 4 ...
##  $ trans       : chr [1:234] "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
##  $ drv         : chr [1:234] "f" "f" "f" "f" ...
##  $ cty         : int [1:234] 18 21 20 21 16 18 18 18 16 20 ...
##  $ hwy         : int [1:234] 29 29 31 30 26 26 27 26 25 28 ...
##  $ fl          : chr [1:234] "p" "p" "p" "p" ...
##  $ class       : chr [1:234] "compact" "compact" "compact" "compact" ...
glimpse(mpg) # get a glimpse of the mpg data
## Rows: 234
## Columns: 11
```

```
## $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi", "audi", "audi", "~
## $ model        <chr> "a4", "a4", "a4", "a4", "a4", "a4", "a4", "a4 quattro", "~
## $ displ        <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.8, 1.8, 2.0, 2.0, 2.~
## $ year         <int> 1999, 1999, 2008, 2008, 1999, 1999, 2008, 1999, 1999, 200~
## $ cyl          <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6, 6, 6, 6, 6, 8, 8, ~
## $ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)", "auto(av)", "auto~
## $ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4", "4", "4", "4", "4~
## $ cty          <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 20, 19, 15, 17, 17, 1~
## $ hwy          <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 28, 27, 25, 25, 25, 2~
## $ fl           <chr> "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p", "p~
## $ class        <chr> "compact", "compact", "compact", "compact", "compact", "c~
```

a) Aesthetic mapping of `color`

i) The following codes does not show points in blue color. What's gone wrong with the following code? Why are the points not blue?

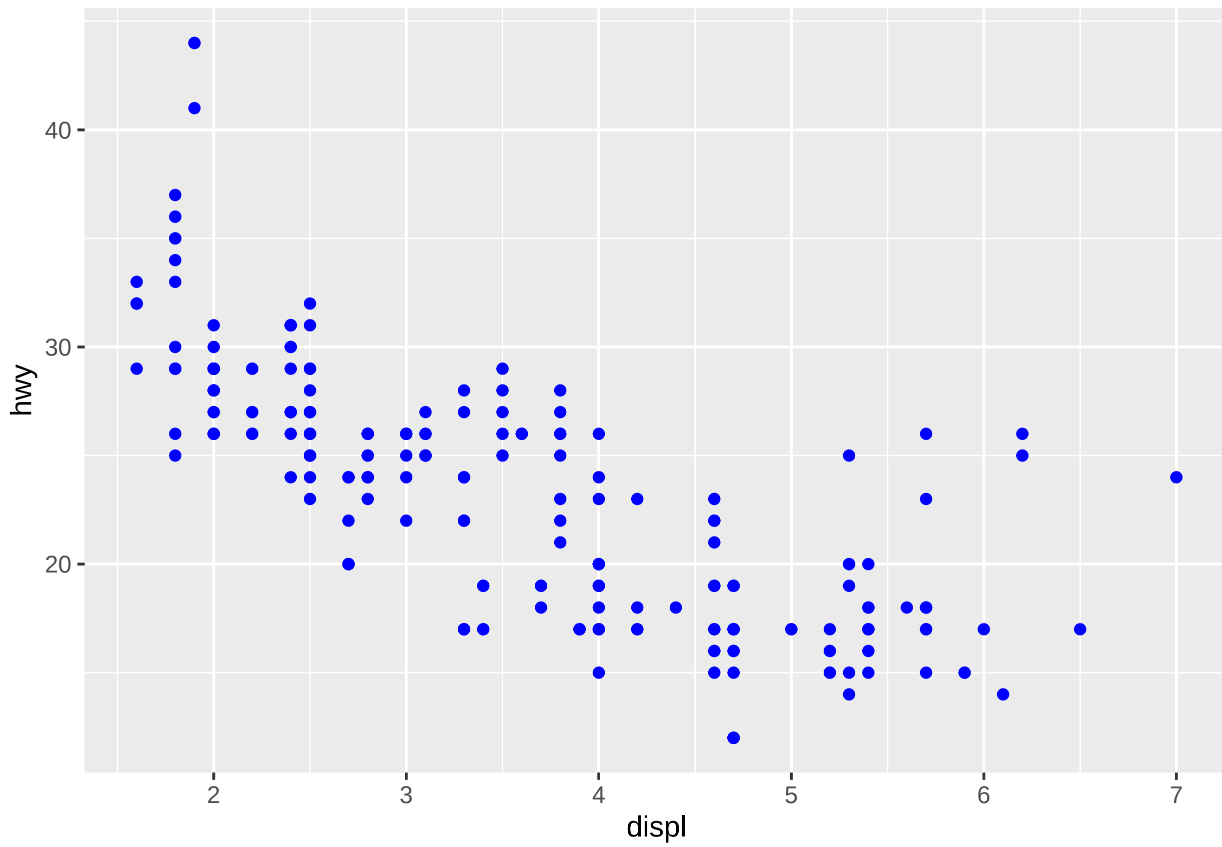
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = "blue"))
```



Answer: Points were not blue because color="blue" was written inside aes()/mapping which is not read as argument color by R, it is read as a vector c("blue") to map to an aesthetic, just like x and y variables displ and hwy. After writing color outside the mapping/aes, the R reads it as color**.

Task: Correct the code to plot blue points.

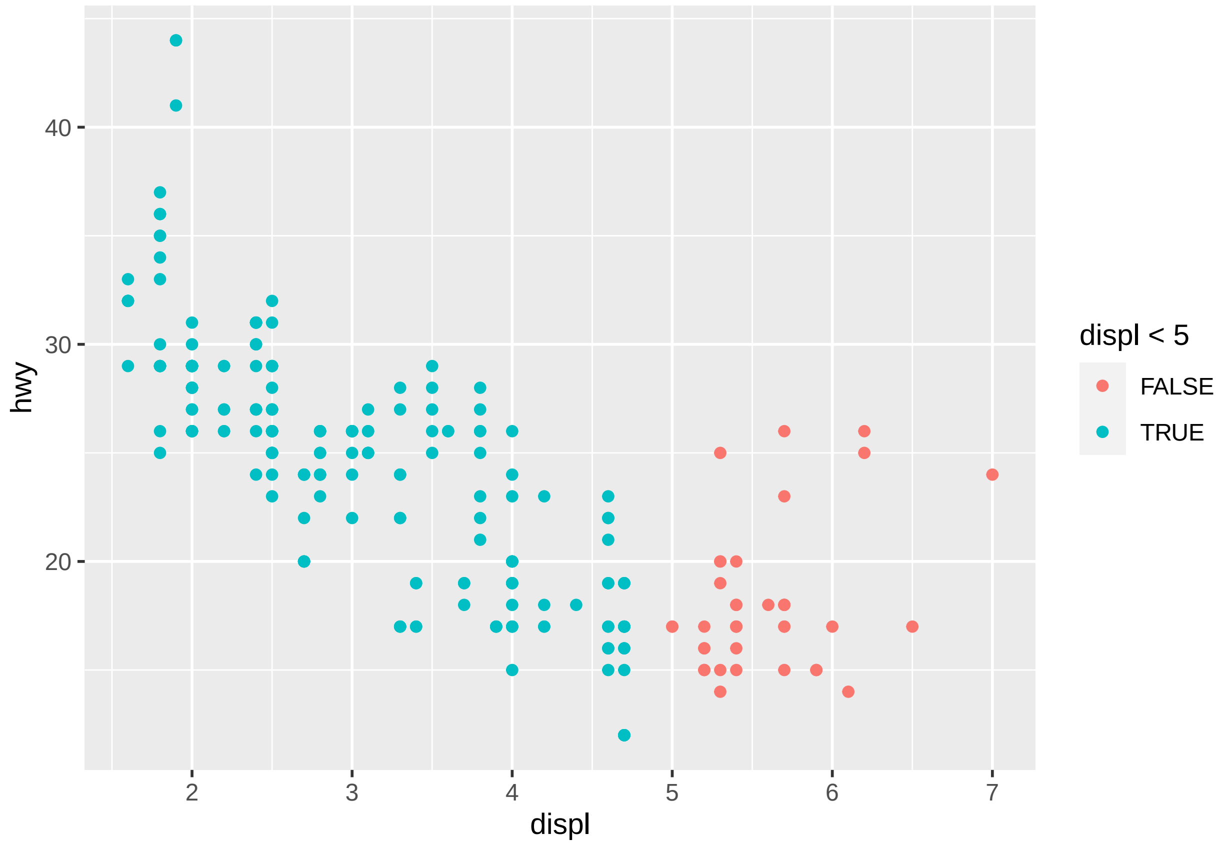
```
ggplot(data = mpg) +
  geom_point(aes(x = displ, y = hwy), color = "blue")
```



(ii) What happens if we map an aesthetic to something other than a variable name, like `aes(colour = displ < 5)`?

Answer If we map an aesthetic to something other than a variable name, like 'aes(colour= displ<5) then ggplot() function works like a temporary variable is added in the data with values equal to the result of expression. In our case, it takes values of 'TRUE' Or 'FALSE' because displ<5 is a logical variable.x and y are displ and hwy as before.

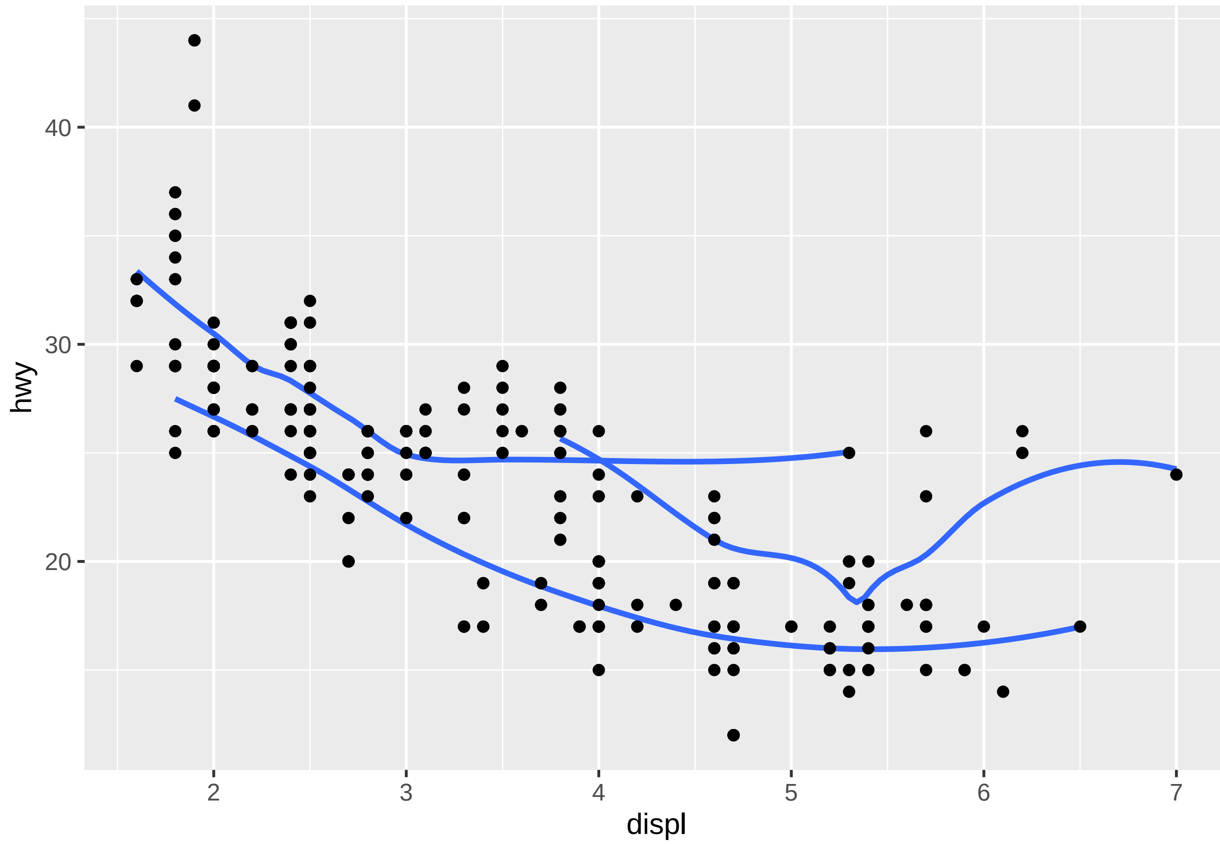
```
ggplot(mpg, aes(x = displ, y = hwy, colour = displ < 5)) +
  geom_point()
```



Part b) Some colorful plot using ggplot2 of tidyverse

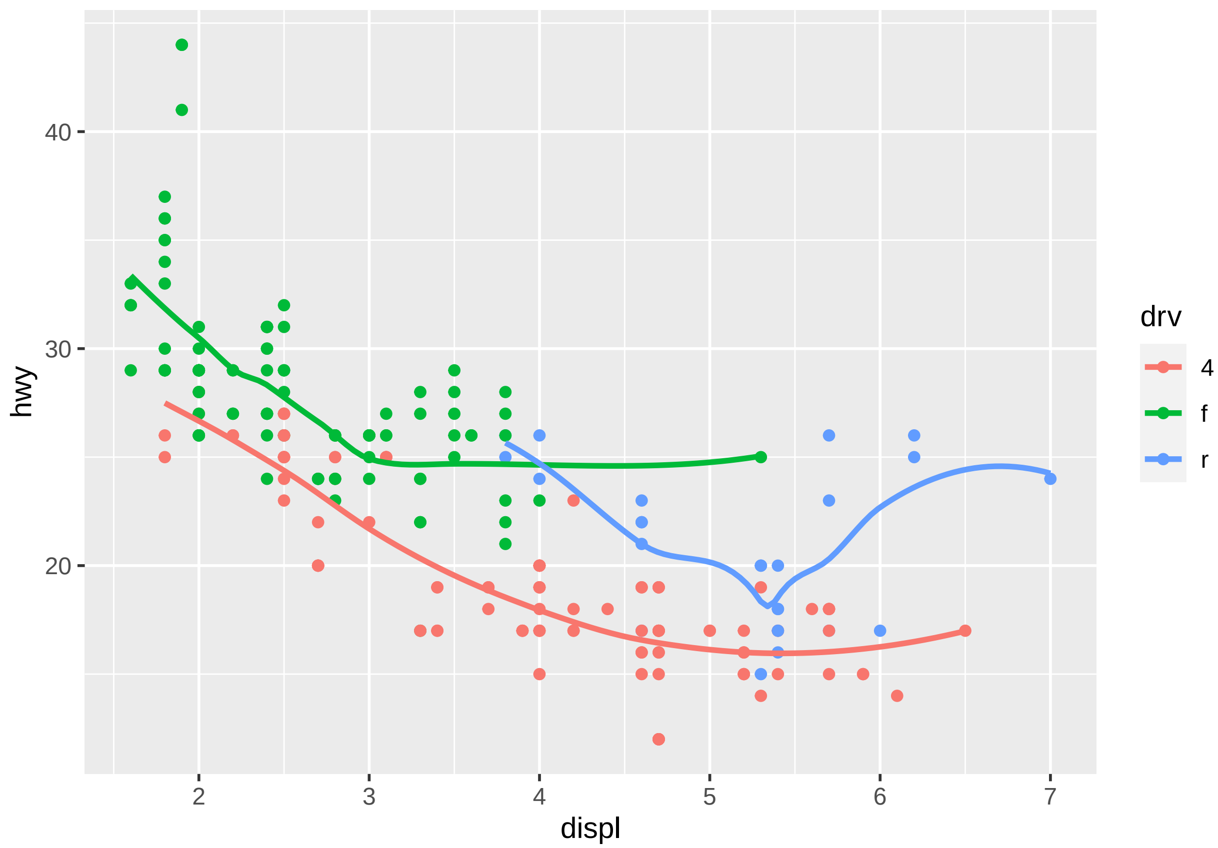
i) Smooth plots according to group identity but same color. Group is `drv` variable which is a categorical variable taking 3 values: {r,f,4}. Where `r` is for rear wheels, `f` is for front wheels, and `4` stands for all four wheels.

```
# Enter your code here
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(group = drv), se = FALSE) +
  geom_point()
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



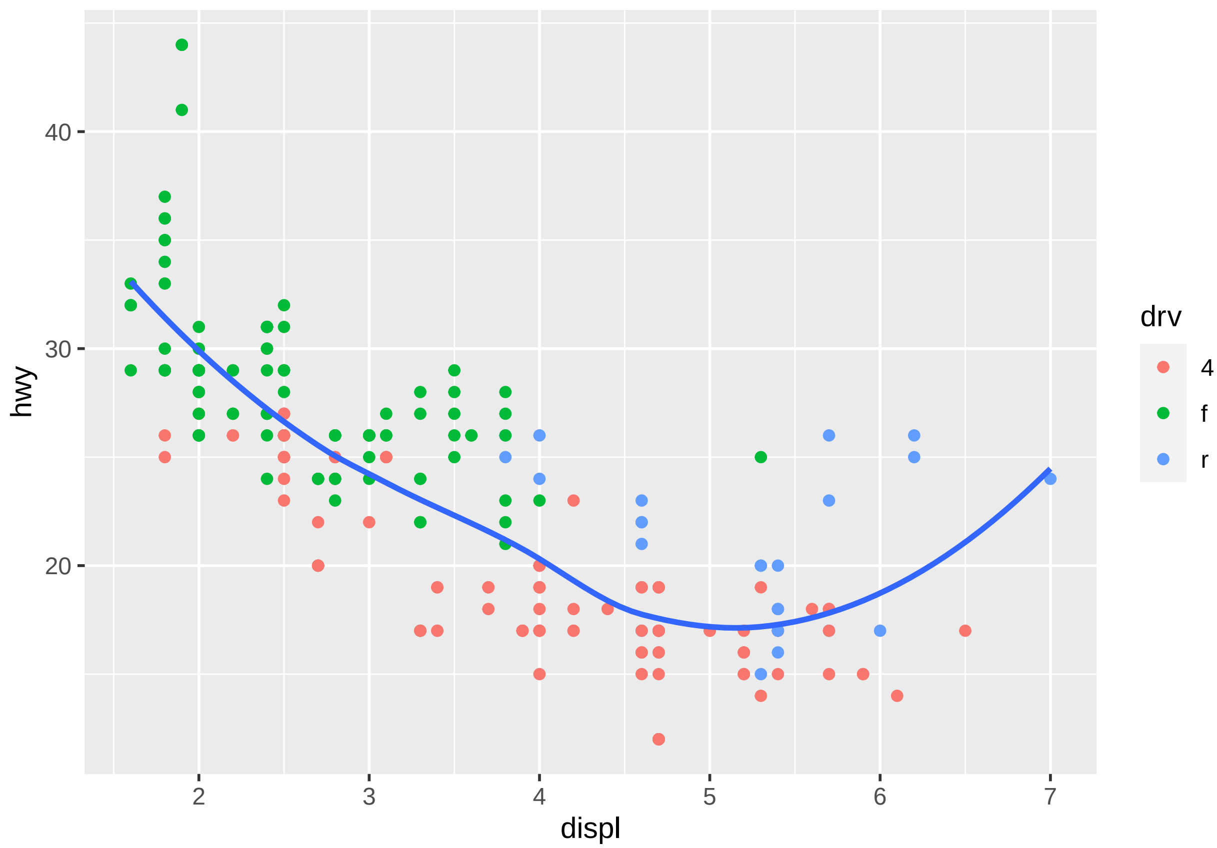
ii) Smooth plots and colors according to group identity. Group is `drv` variable which is a categorical variable taking 3 values: {r,f,4}. WHERE r IS for rear wheels,f is for front wheels, and 4 stands for all four wheels.

```
ggplot(mpg, aes(x = displ, y = hwy, colour = drv)) +
  geom_point() +
  geom_smooth(se = FALSE)
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



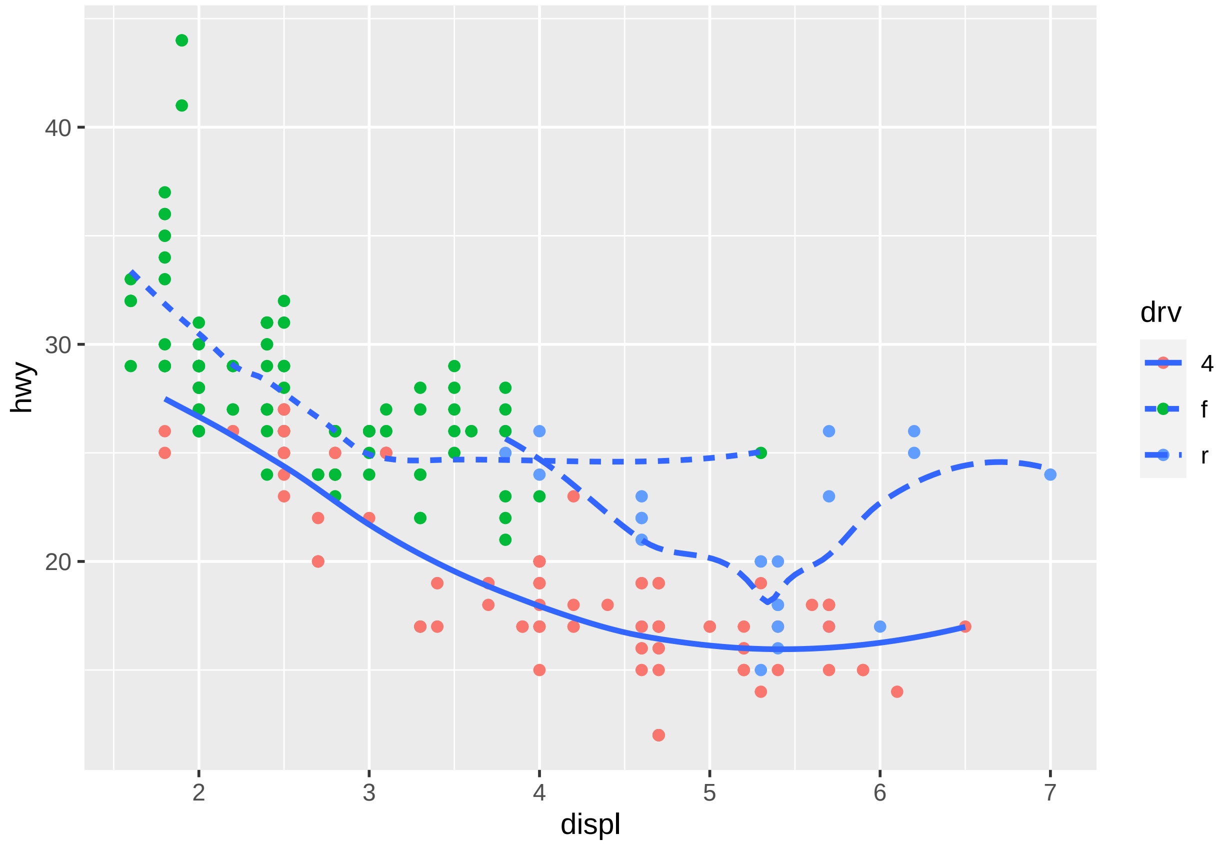
iii) Single smooth plot but data points being colored according to group identity. Group is `drv` variable which is a categorical variable taking 3 values: {r,f,4}. WHERE r IS for rear wheels,f is for front wheels, and 4 stands for all four wheels.

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(aes(colour = drv)) +
  geom_smooth(se = FALSE)
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



iv) Diffierent smooth plots with different linetype according to group identity. Group is `drv` variable.

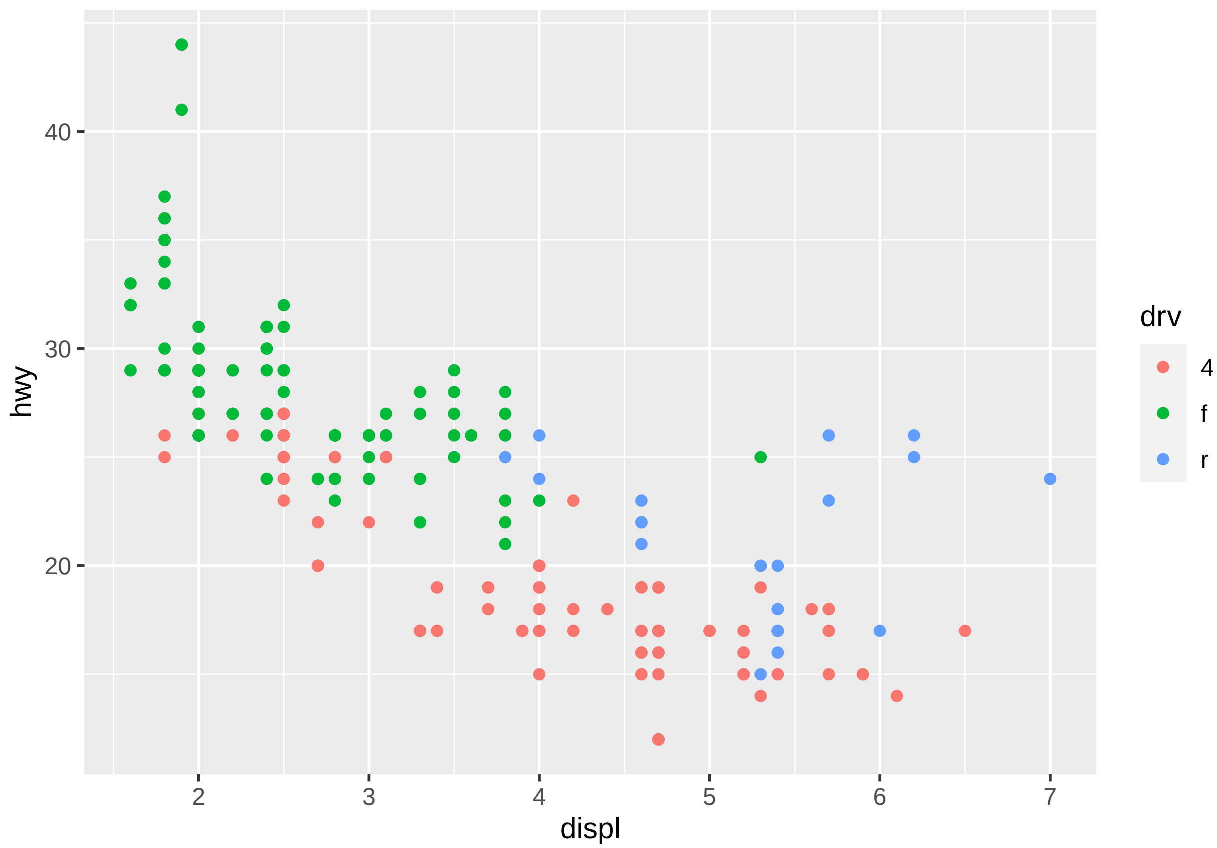
```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(aes(colour = drv)) +
  geom_smooth(aes(linetype = drv), se = FALSE)
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



c) Facets: to add additional variable(s) to a 2D plot There are two ways to add additional variable(s) to a 2D plot. One is using aesthetics, the other one is using facets.

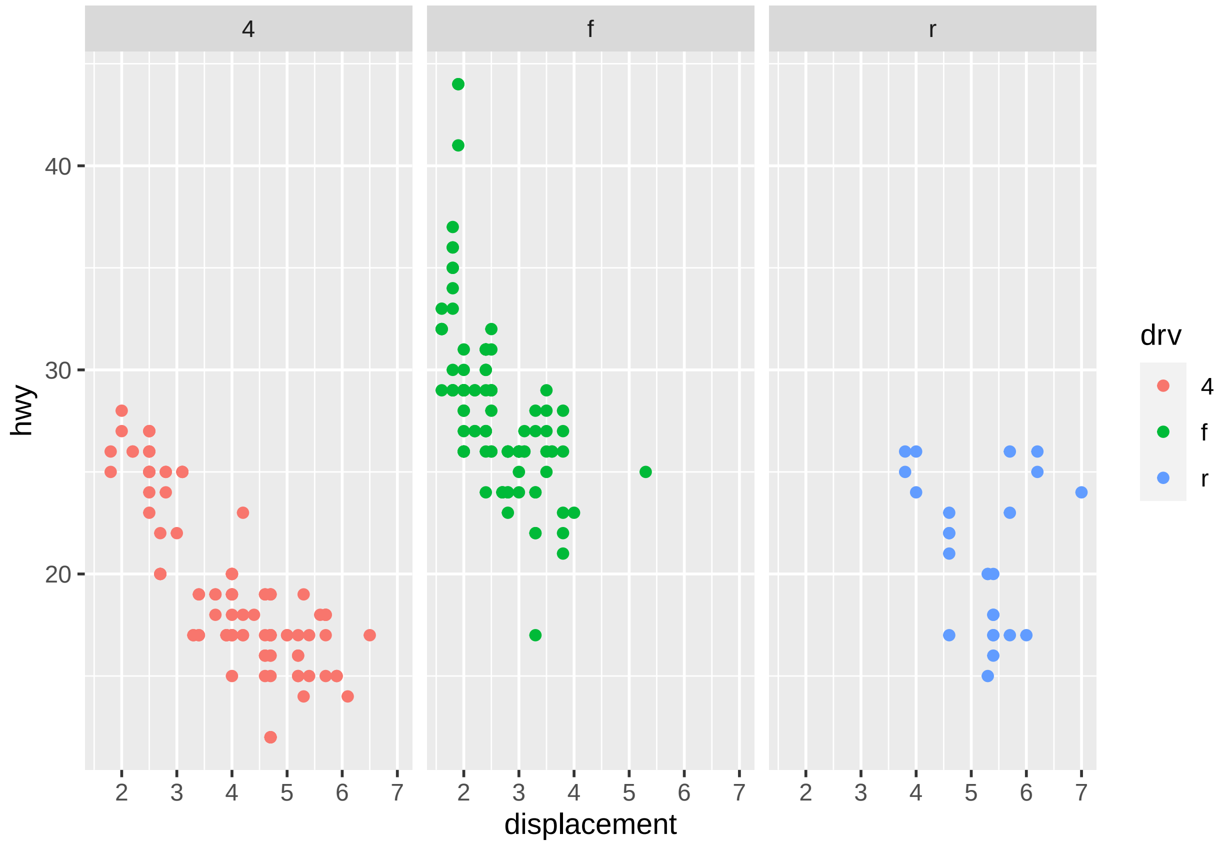
(i) Scatter plot : x-axis is `displ` and y-axis is `hwy`. We use different colors to distinguish `drv` types.

```
ggplot(data=mpg, mapping=aes(x = displ, y = hwy, colour=drv))+
  geom_point()
```



(ii) Facet `drv` into the rows. That is, makes several **rows** of subplots, one row for each `drv` type. Each subplot has `displ` mapped to the x-axis and `hwy` mapped to the y-axis. [*Note*: Use `nrow` or `ncol` to control the layout of the individual panels].

```
ggplot(data=mpg, aes(x=displ, y=hwy, color=drv))+
 geom_point()+
 labs(x="displacement", y="hwy")+
 scale_color_discrete(name="drv")+
 facet_wrap(~drv, nrow=1)
```



d) `stat` functions

Most `geom` functions and `stat` functions come in pairs that are almost always used in concert.

- every `geom` has a default `stat`
- every `stat` has a default `geom`

Look up the default `stat` functions for the `geom` functions listed in the following table. The variables computed by the default `stat` function (Reference: the **Computed variables** section in the R-documentation page).

`geom` function	default `stat` function	variables computed by the default `stat` function
`geom_bar()`	count	Frequency
`geom_histogram()`	bin	Frequency
`geom_density()`	density	Density, Count
`geom_point()`	identity	Sum of values
`geom_smooth()`	smooth	Smoothed or locally averaged value

Question: Some `geom` function has stat = "identity" as the default. What does that mean?

Answer : If there are 3 teams A,B, and c with equal occurrence. Then geom_bar function will create bar chart displaying the count of occurrence in the games, which is equal. But, If we use stat="identity" with geom_bar then bar chart will be created displaying sum of points scored by the teams in each game. (Additional Note: Table formatting are sometimes tricky using R Markdown. Table Generator is a handy tool if you need to make tables in the future.)

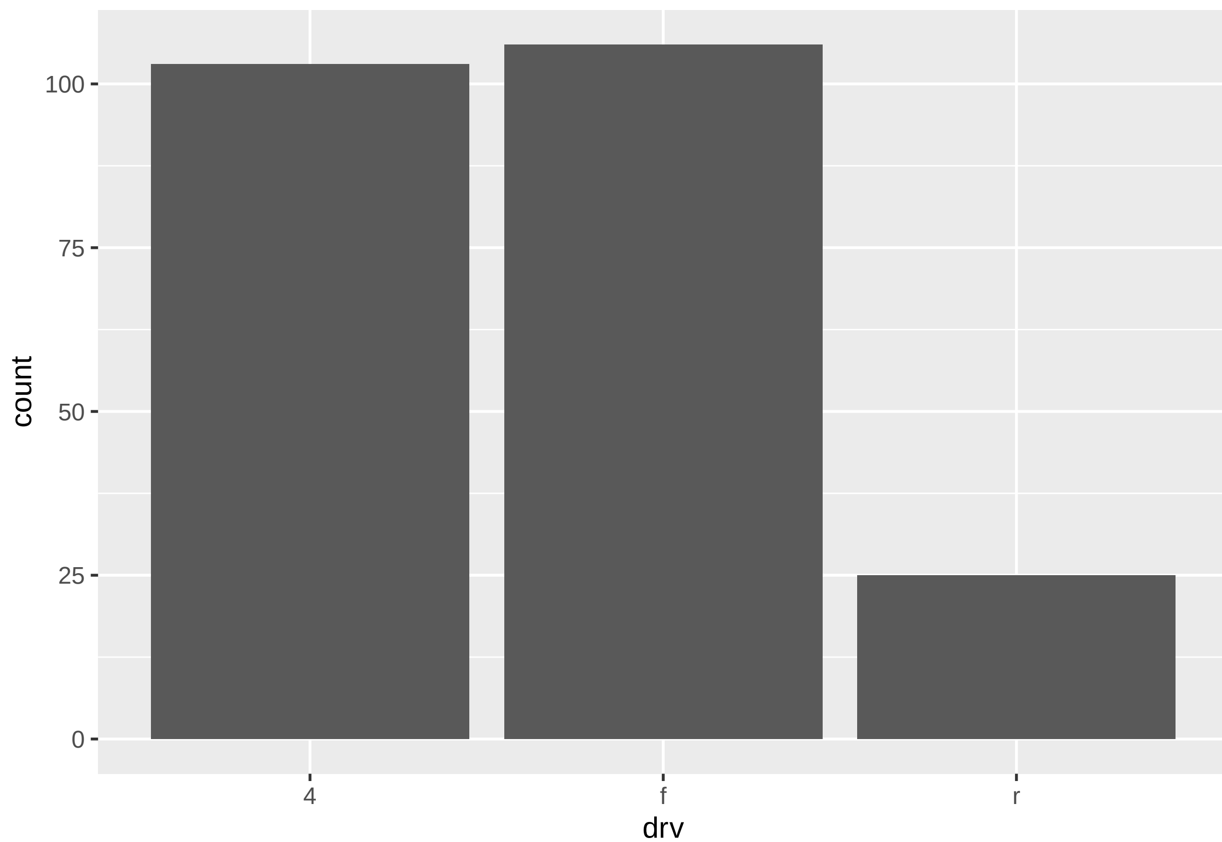
e) Position adjustment options for `geom_bar()`

Using two categorical variables from the `mpg` data set and to illustrate the following four position adjustment options for `geom_bar()`:

- **default**: position = "stack"
- position = "identity" will place each object exactly where it falls in the context of the graph.
- position = "fill" works like stacking, but makes each set of stacked bars the same height.
- position = "dodge" places overlapping objects directly beside one another. the bars are automatically stacked. Each colored rectangle represents a combination of cut and clarity.

i) position = "stack"

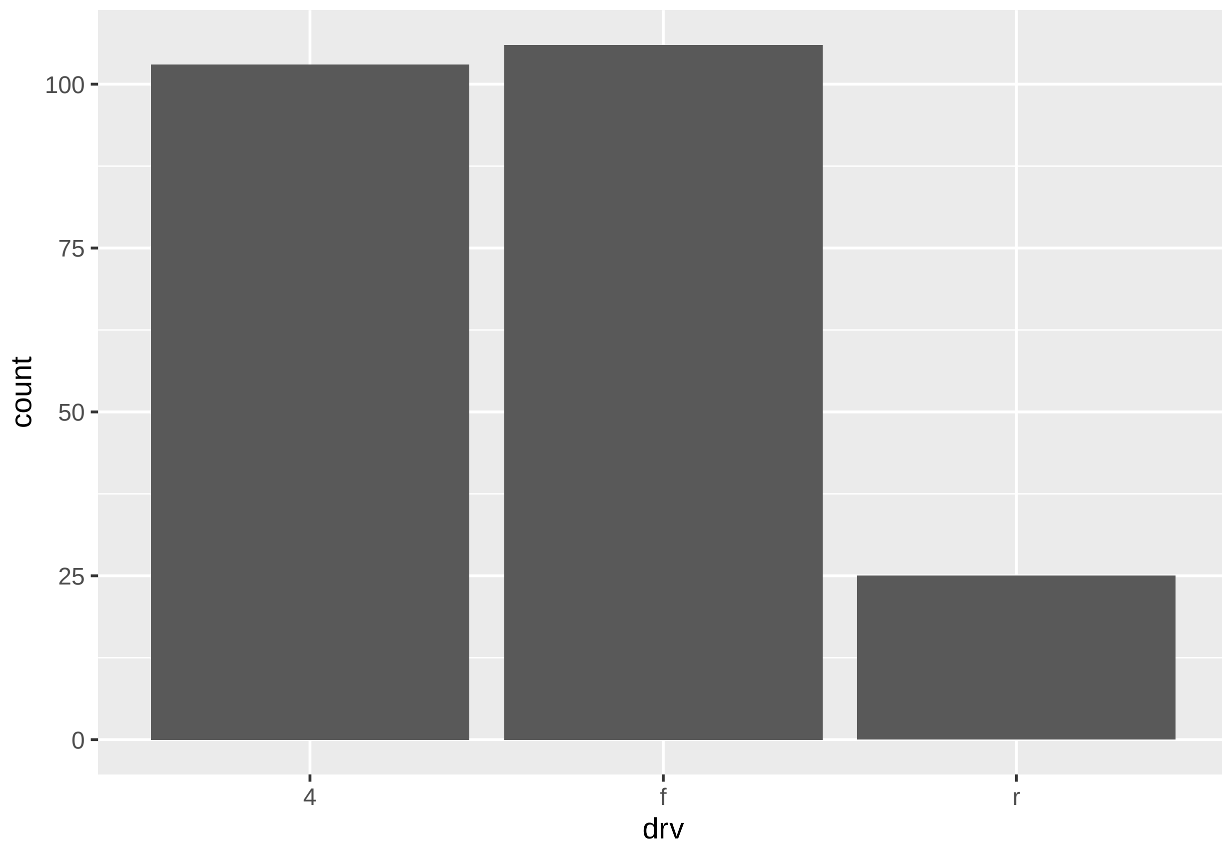
```
ggplot(data=mpg)+
geom_bar( mapping= aes(drv) , position = "stack")
```



ii) position = "identity"

It will place each object exactly where it falls in the context of the graph.

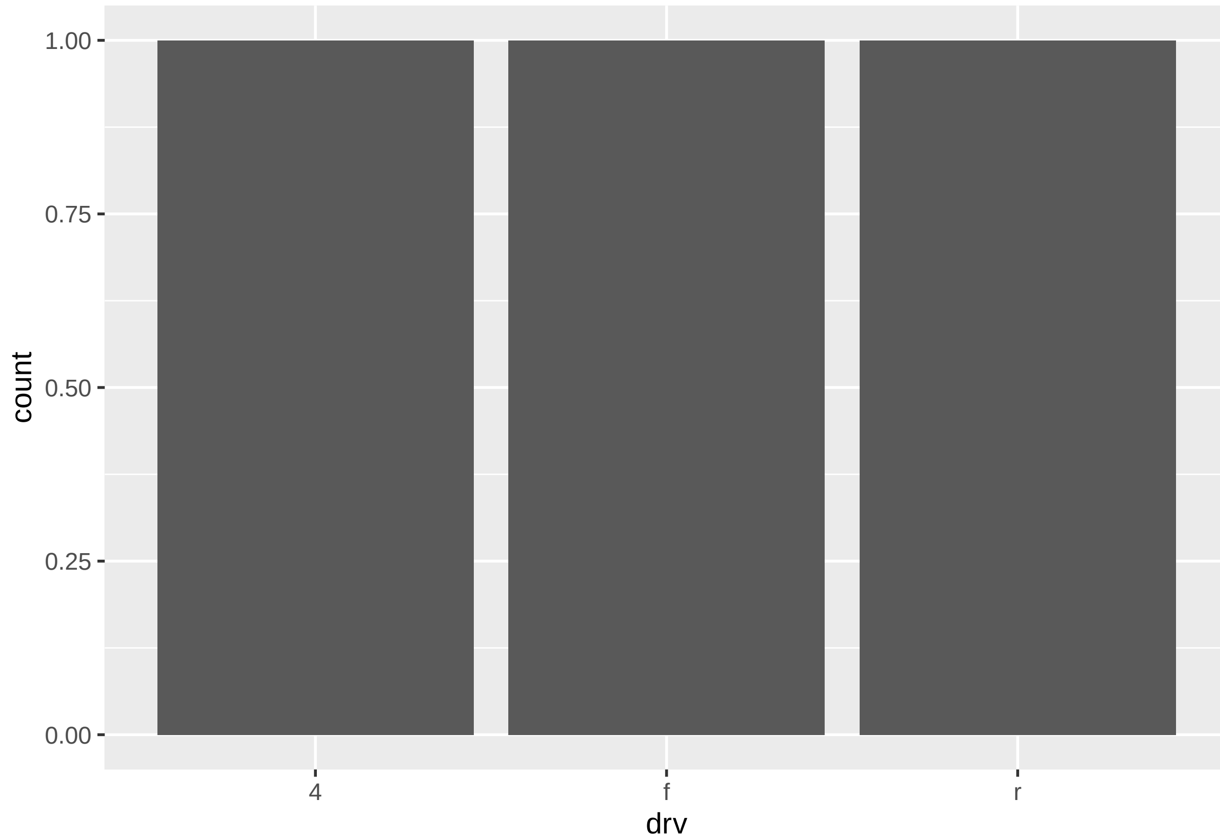
```
ggplot(data=mpg)+
geom_bar( mapping= aes(drv) , position = "identity")
```



iii) position = "fill"

It works like stacking, but makes each set of stacked bars the same height.

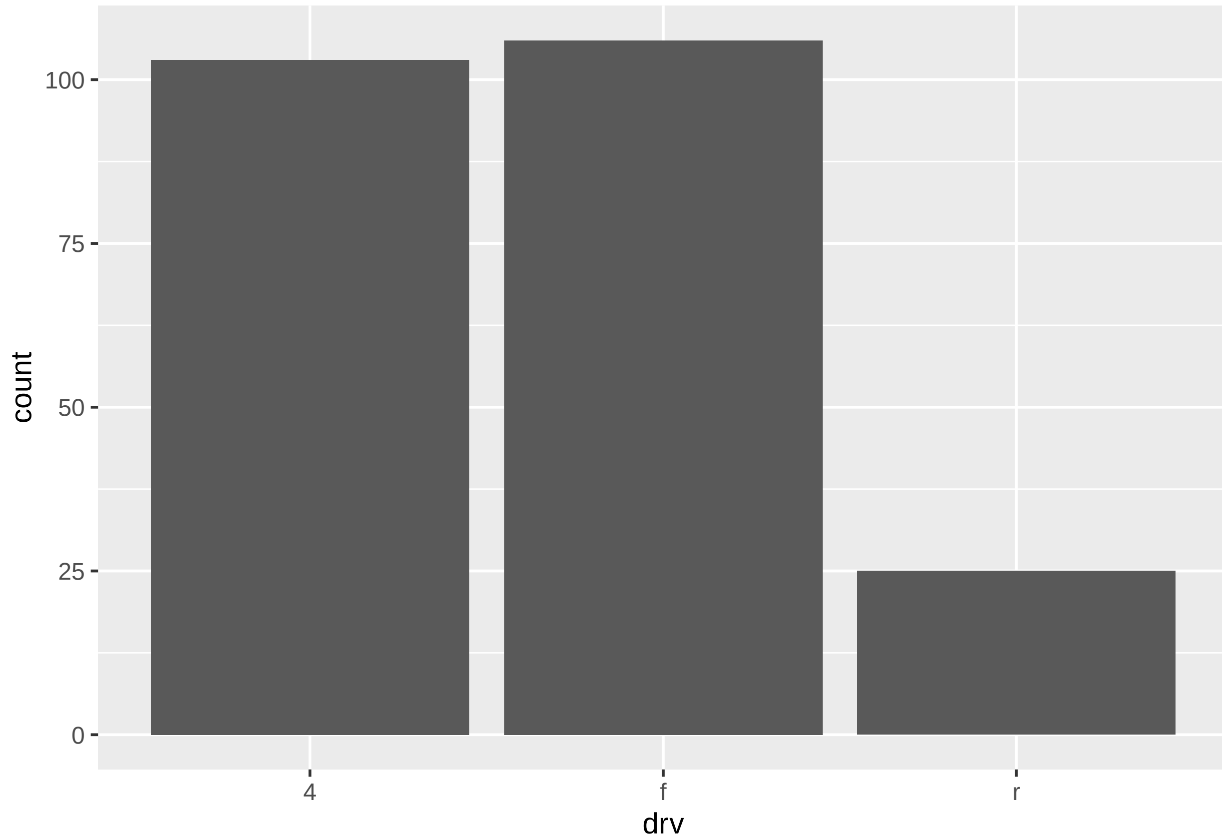
```
ggplot(data=mpg)+
geom_bar( mapping= aes(drv) , position = "fill")
```



iv) position = "dodge"

dodge places overlapping objects directly beside one another.

```
ggplot(data=mpg)+
geom_bar( mapping= aes(drv) , position = "dodge")
```



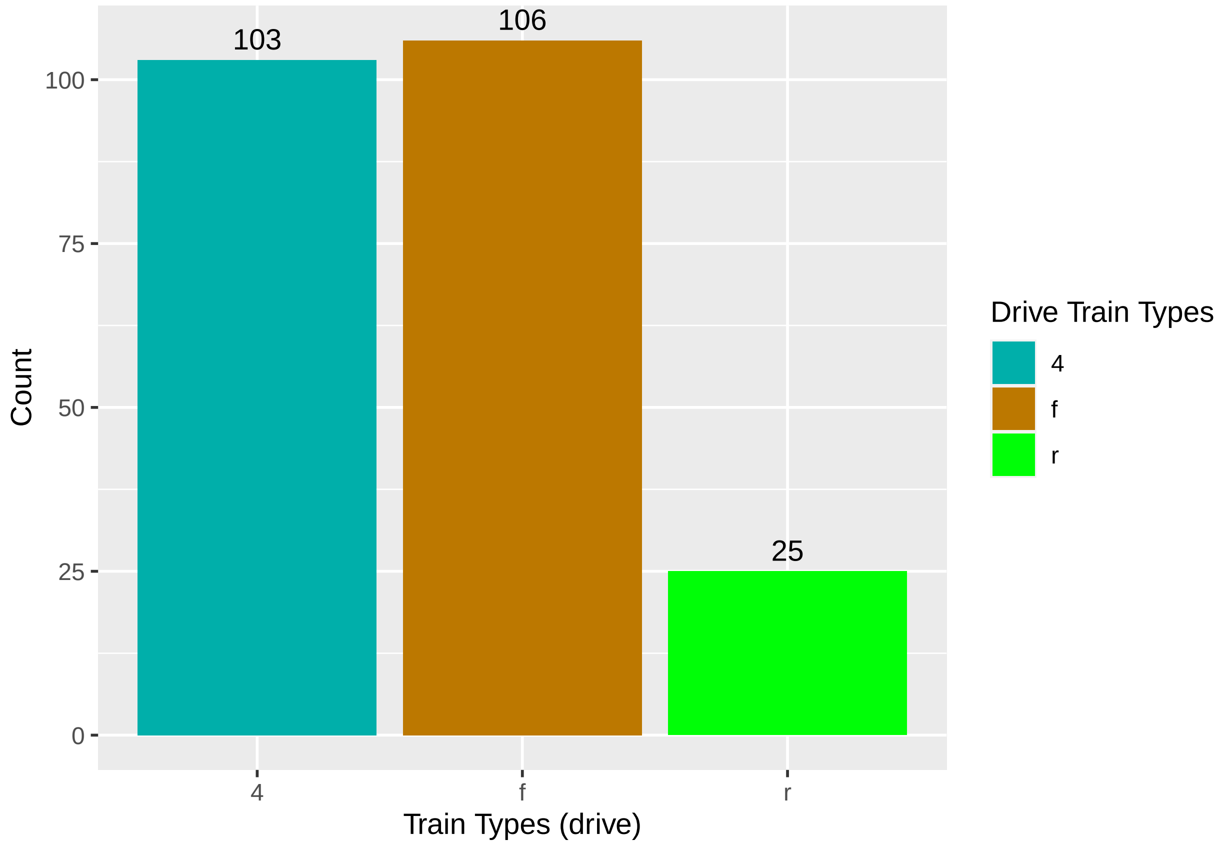
Question Which position option do you like most? What conclusions can you draw from your plot?

ANSWER : I liked position = "dodge" which places overlapping objects directly beside one another. Choosing right position argument is important part of making good plot. You may like other position, it is not really a right or wrong answer.

Part IV: Visualize the distribution of drive train types in mpg dataset

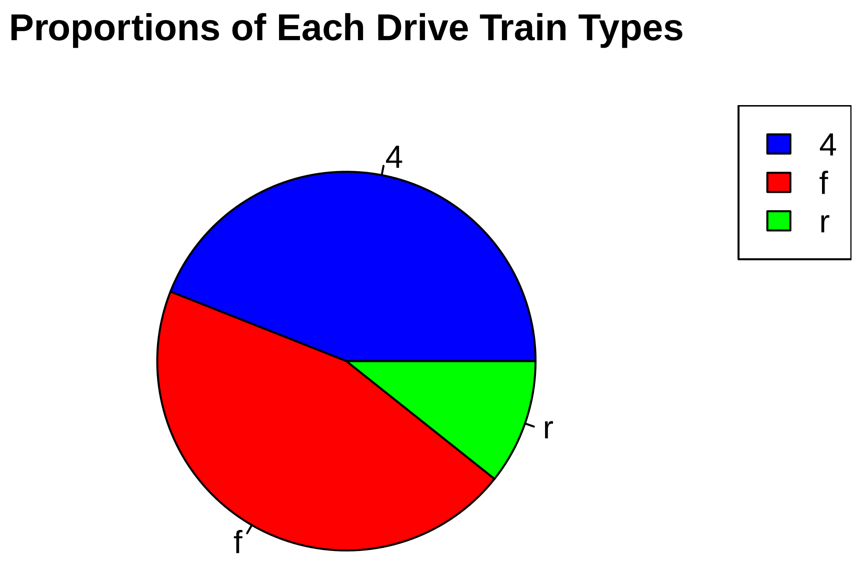
a) Barplot (frequency histogram) to display the distribution of `drv`, the type of drive train. Using different colors to distinguish different drive train types. Explicitly label the number of car models of each drive train type on top of the bars.

```
ggplot(mpg, aes(x = drv)) +
geom_bar(aes(fill = drv), position = "dodge")+ #create bar plot, graph that represents data using recta
geom_text(stat = "count", aes(label = ..count..), position =position_dodge(width = 0.1), vjust = -0.5) +
scale_fill_manual(values = c("#00AFAA", "#BC7800", "#00FE07")) +
labs(x = "Train Types (drive)", y = "Count", fill = "Drive Train Types")
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



b) Drawing a coxcomb or pie chart to display the proportions of each drive train types Plotting a pie chart to display the proportions of each drive train types

```
drv_counts= table(mpg$drv) #"drv" column of the "mpg" data frame is used as the categorical variable

pie(drv_counts, main="Proportions of Each Drive Train Types", labels=names(drv_counts), col=c("blue", "r
legend("topright", legend=names(drv_counts), fill=c("blue", "red", "green"))
```

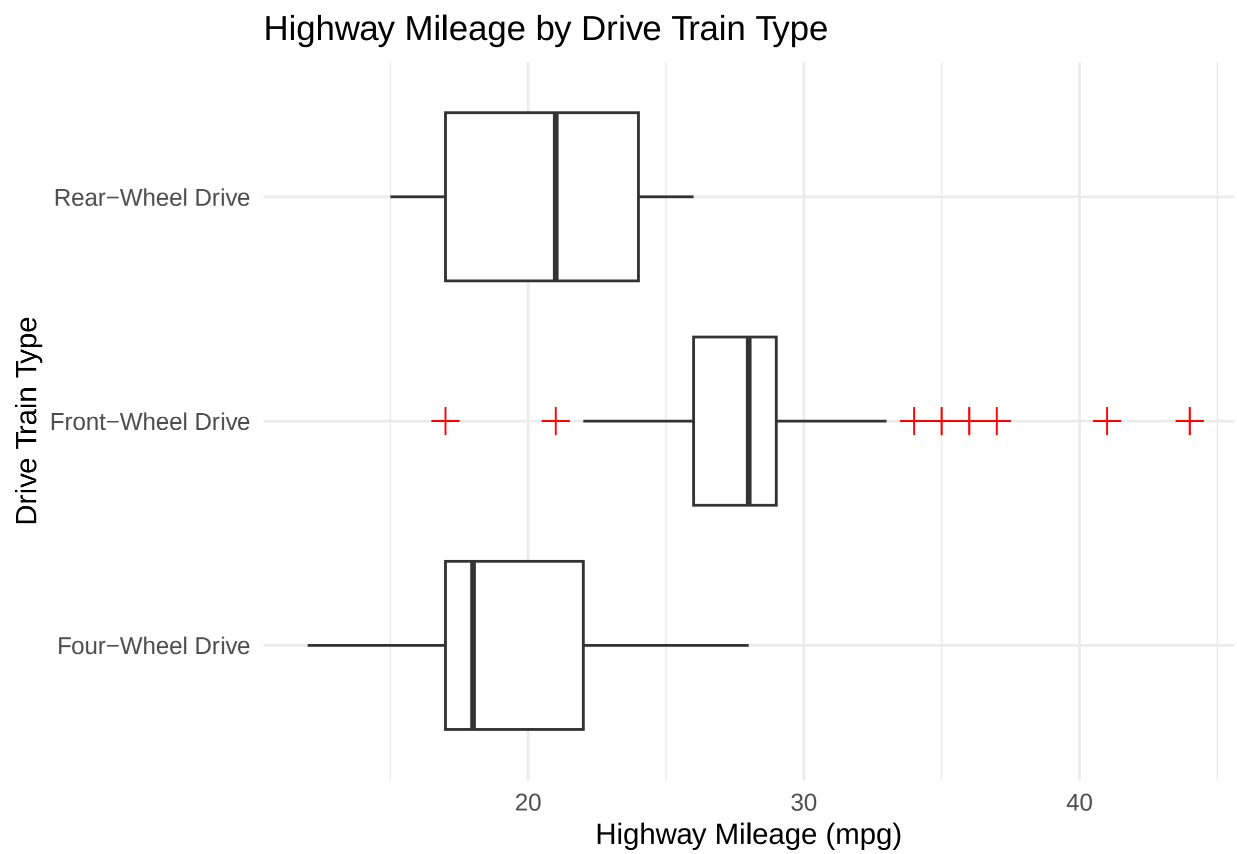


c) How highway mileage varies across drive train type?

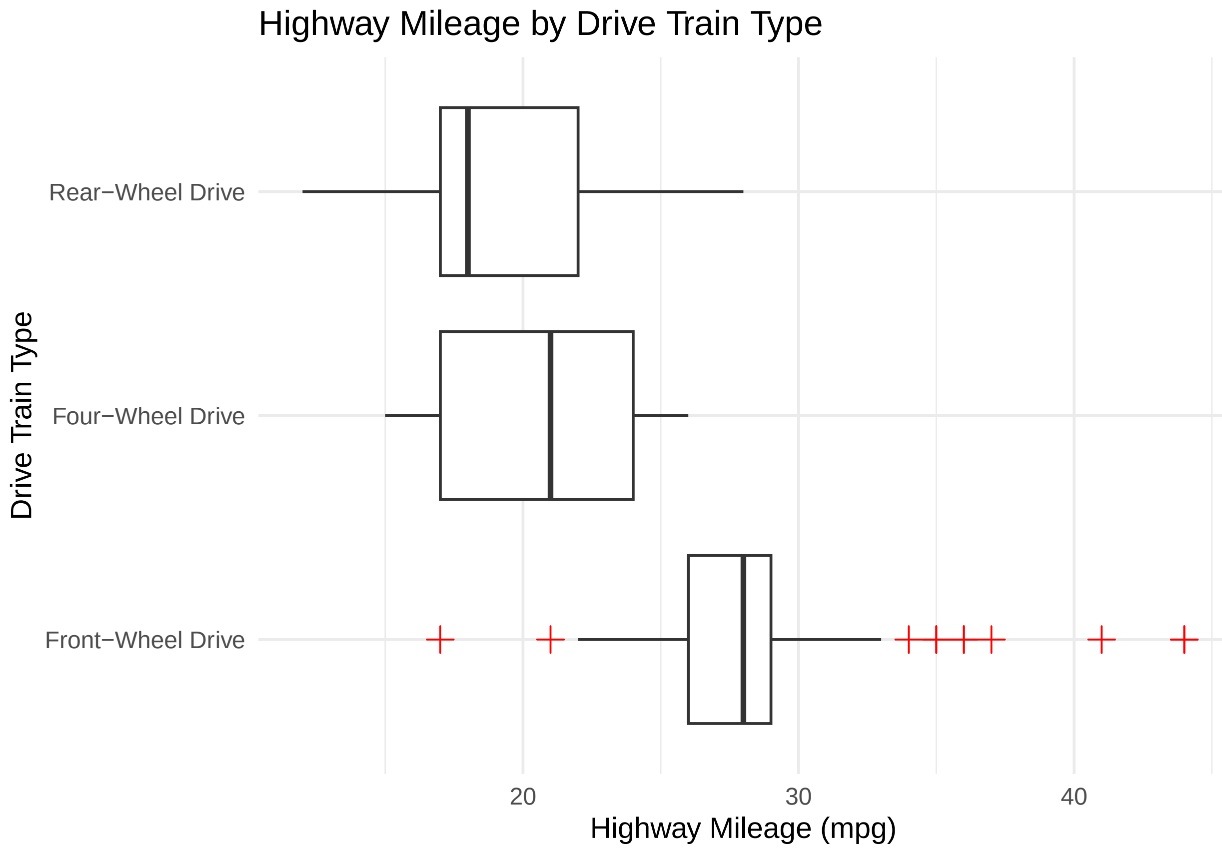
We generate a horizontal boxplot to compare the distribution of highway mileage across three different drive train types. Reorder the boxes by the median mileage values.

ANSWER The front wheel drive median highway mileage is high among other two. The least median mileage is of rear wheel drive. That means if we order all the vehicles of different categories the highway mileage of middle one is the highest in front-wheel drive. However, this category contains outliers unlike others.

```
ggplot(mpg, aes(x = drv, y = hwy)) + # part1-Creating a horizontal boxplot
  geom_boxplot(outlier.color = "red",outlier.shape = 3,outlier.size = 3) + #box plot with red triangle-
                                                                            #that are 3 times larger tha
  coord_flip()+ #"coord_flip()" function switches the orientation of the plot
  xlab("Drive Train Type") +
  ylab("Highway Mileage (mpg)") +
  ggtitle("Highway Mileage by Drive Train Type")+
  theme(plot.title =element_text(hjust = 0.5)) + #sets the justification of the plot title to be center
  theme(axis.text.x = element_text(angle =45, hjust = 1))+ #sets x-axis labels to be rotated by 45 degr
  theme(axis.title.x = element_blank()) +
  theme(axis.title.y = element_text(hjust = 0.5)) +
  scale_x_discrete(limits = c("4", "f", "r"), labels = c("Four-Wheel Drive", "Front-Wheel Drive", "Rear-
  theme(legend.title = element_blank(), legend.position = "none") +
  theme_minimal() #removes most of the plot decorations such as axis lines, gridlines, and background c
```




```
mpg_median <- mpg %>%  #part2-Reordeing the boxes
group_by(drv) %>% #specifies the variable(s) by which the data should be grouped.
summarize(median = median(hwy)) %>% #calculates median of "hwy"(highway miles per gallon)variabl fr eac
 arrange(desc(median)) #sort the data in descending order based on the median of the "hwy" variable

ggplot(mpg, aes(x = fct_reorder(drv, hwy, .fun = median), y = hwy)) +
geom_boxplot(outlier.colour = "red", outlier.shape = 3, outlier.size = 3)+
coord_flip() +
xlab("Drive Train Type") +
ylab("Highway Mileage (mpg)") +
ggtitle("Highway Mileage by Drive Train Type") +
  theme(plot.title = element_text(hjust = 0.5)) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  theme(axis.title.x = element_blank()) +
  theme(axis.title.y = element_text(hjust = 0.5)) +
  scale_x_discrete(limits = mpg_median$drv, labels = c("Front-Wheel Drive", "Four-Wheel Drive", "Rear-Wh
  theme(legend.title = element_blank(), legend.position = "none") +
  theme_minimal()
```



Acknowledgments

List of all the help I have received for completing this work. 1. I used rdocumentation.org website to learn more about the functions. 2. I used geeksforgeeks.com website to get help on tidyverse paradigm of R.