Lets load the libraries required for this assignment

library(tidyverse)

## -- Attaching packages ------------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.2.1 v purrr 0.3.2  
## v tibble 2.1.3 v dplyr 0.8.3  
## v tidyr 0.8.3 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.4.0

## -- Conflicts ---------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(maps)

##   
## Attaching package: 'maps'

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

Question 1.

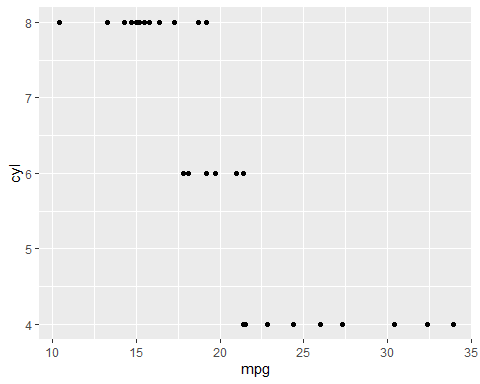
?mtcars

## starting httpd help server ... done

The gear variable in the data set for mtcars describes the number of forward gears.

Question 2.

ggplot(data = mtcars) + geom\_point(mapping = aes(x=mpg, y=cyl))+labs(x="mpg",y="cyl")



Question 3.

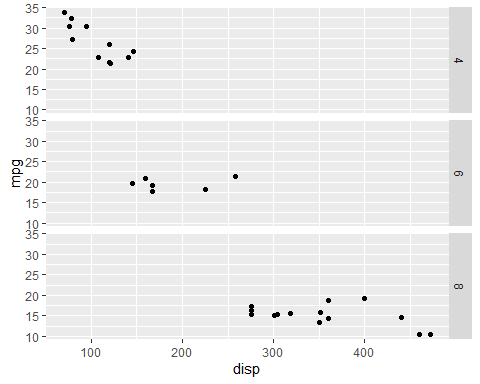
?mtcars  
str(mtcars)

## 'data.frame': 32 obs. of 11 variables:  
## $ mpg : num 21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...  
## $ cyl : num 6 6 4 6 8 6 8 4 4 6 ...  
## $ disp: num 160 160 108 258 360 ...  
## $ hp : num 110 110 93 110 175 105 245 62 95 123 ...  
## $ drat: num 3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...  
## $ wt : num 2.62 2.88 2.32 3.21 3.44 ...  
## $ qsec: num 16.5 17 18.6 19.4 17 ...  
## $ vs : num 0 0 1 1 0 1 0 1 1 1 ...  
## $ am : num 1 1 1 0 0 0 0 0 0 0 ...  
## $ gear: num 4 4 4 3 3 3 3 4 4 4 ...  
## $ carb: num 4 4 1 1 2 1 4 2 2 4 ...

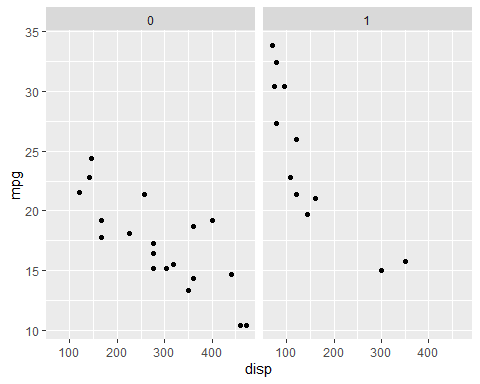
cyl, vs, am, gear, carb are categorical variables in mtcars. mpg, disp, hp, drat, wt, qsec are continuous variables. Variables which are mentioned as categorical above have a fixed set of numbers and can be distinguishable. Continuous variables are in the form of integers or decimals and do not have a fixed set of distinguishable numbers.?mtcars gives us the information and help related to mtcars. str(mtcars) will give details on what every variable of the data frame.

Question 4. We can facet the plots by using the argument facet\_grid(rows~columns). Refer below for detailed explanation on its usage.

ggplot(data = mtcars) + geom\_point(mapping = aes(x=disp, y=mpg)) + facet\_grid(cyl ~ .)

 The above plot with facet will subset the mtcars dataset with the cyl(number of cylinder) variable distinguised in a row format

ggplot(data = mtcars) + geom\_point(mapping = aes(x=disp, y=mpg)) + facet\_grid(. ~ am)

 The above plot with facet will subset the mtcars dataset with am (transmission) variable distinguised in a column format. If a . is provided then the plot will only be facetted in either the row or column depending on which variable is mentioned where in the syntax

Question 5

?facet\_wrap

nrow will control and set the number of rows in the plot. ncol will control and set the number of columns in the plot. nrow and ncol is used to set the aesthetic and the spread of the plot for better visualization.

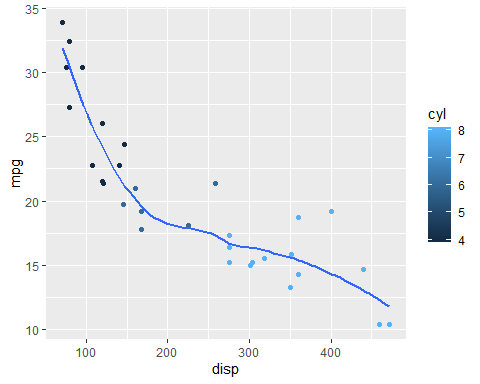
We can change the levels of the underlying factor in order to change the order in which panels appear. Also, we can use the ‘scales’ argument to allow scales to vary across the panels. Also, strip.position can be used to display the facet label at the side of your choice. We can also use shrink, switch, drop, dir to control the layout of the plot.

facet\_grid() is used when we have to distinguish the plot using combination of 2 variables. facet\_grid() will create a plot based on the two variables that are mentioned in the syntax. Therefore a nrow or ncol won’t work with facet\_grid() since facet\_grid() will pull the value from the variable in the syntax.

Question 6 Prediction for the below code is that it will plot a scatterplot for mtcars dataset with disp on X axis and mpg on Y axis with all points colored based on cyl variable. Diﬀerent number of cyl will have a diﬀerent color shades. It will also give a smooth line with the same shade of color which will best ﬁt all the points in the dataset but without the Conﬁdence Interval.

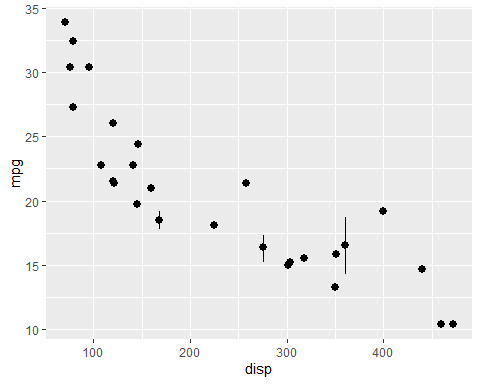
ggplot(data = mtcars, mapping = aes(x=disp, y=mpg, color = cyl)) +  
 geom\_point()+  
 geom\_smooth(se=FALSE)

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'



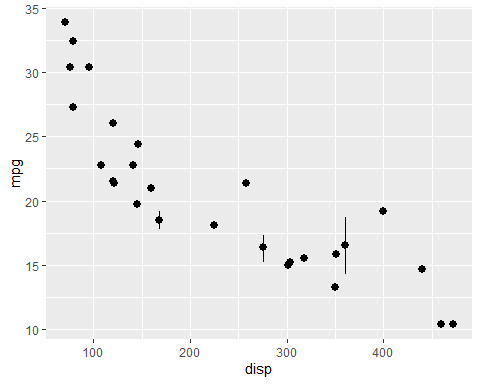
Question 7 The default geom associated with stat\_summary() is geom\_pointrange()

ggplot(data = mtcars)+  
 stat\_summary(mapping = aes(x=disp, y=mpg),  
 fun.ymin = min,  
 fun.ymax = max,  
 fun.y = median)



The above code can be written using geom\_pointrage() as below:

ggplot(data = mtcars)+  
 geom\_pointrange(mapping = aes(x=disp, y=mpg),  
 fun.ymin = min,  
 fun.ymax = max,  
 fun.y = median,  
 stat = "summary")

 We use the argument of stat=“summary” to use stat\_summary()

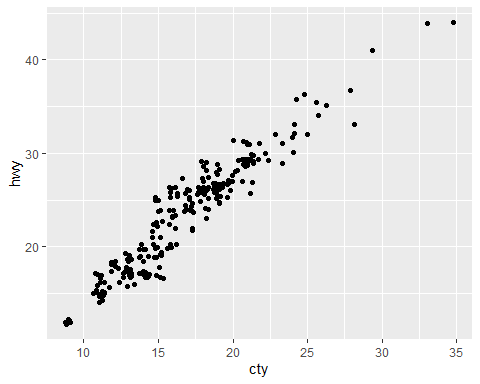
Question 8 Plotting using geom\_point()

ggplot(data = mpg, mapping = aes(x=cty, y=hwy)) +  
 geom\_point()



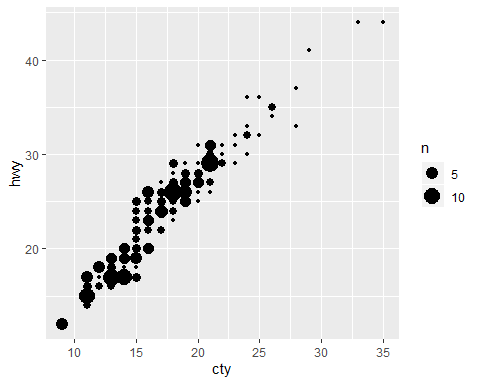
We will use geom\_jitter() in the above code

ggplot(data = mpg, mapping = aes(x=cty, y=hwy))+  
 geom\_jitter()



We will use geom\_count() in the above code

ggplot(data = mpg, mapping = aes(x=cty, y=hwy))+  
 geom\_count()

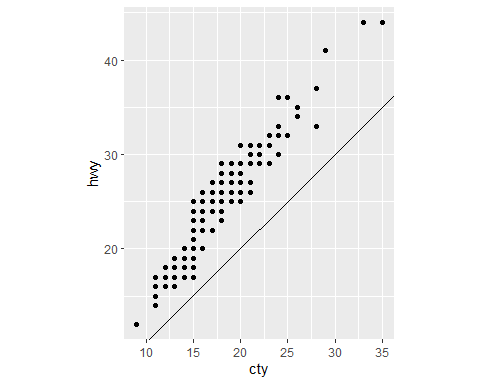


geom\_jitter() will add a small amount of noise to each point. This adds randomness to the plot to make it revealing at large scales. This is done to reduce overplotting, where several points overlap each other.

geom\_count() willincrease the size of the points based on the frequency of the data pointsin the variable. If diﬀerent observations have the same value then the points will have larger size as compared to other points. This can create overplotting. Usually geom\_count() is less readable than geom\_jitter().

Question 9

ggplot(data = mpg, mapping = aes(x=cty, y=hwy))+  
 geom\_point()+  
 geom\_abline()+  
 coord\_fixed()



The above plot shows a linear positive relation between cty and hwy variables. coord\_fixed() forces a specified ratio between the physical representation of data units on the axes. The default ration is 1 and it ensures that one unit on X axis is equal to one unit on Y axis. If ratio is greater than 1, units on Y axis are longer.Also, coord\_fixed() ensures that the line generated by geom\_abline() is at 45 degrees.This is ensured because the coordinated are fixed and therefore the line drawn has to be at 45 degrees. This plays an important role to study the relationship between variables.

geom\_abline() adds reference lines depending on the slope and intercept. It can draw horizontal, vertical or inclined lines.In the above code if coord\_fixed() is not included then the plot does not show a 45 degree line.

Question 10

mpg

## # A tibble: 234 x 11  
## manufacturer model displ year cyl trans drv cty hwy fl class  
## <chr> <chr> <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>  
## 1 audi a4 1.8 1999 4 auto~ f 18 29 p comp~  
## 2 audi a4 1.8 1999 4 manu~ f 21 29 p comp~  
## 3 audi a4 2 2008 4 manu~ f 20 31 p comp~  
## 4 audi a4 2 2008 4 auto~ f 21 30 p comp~  
## 5 audi a4 2.8 1999 6 auto~ f 16 26 p comp~  
## 6 audi a4 2.8 1999 6 manu~ f 18 26 p comp~  
## 7 audi a4 3.1 2008 6 auto~ f 18 27 p comp~  
## 8 audi a4 q~ 1.8 1999 4 manu~ 4 18 26 p comp~  
## 9 audi a4 q~ 1.8 1999 4 auto~ 4 16 25 p comp~  
## 10 audi a4 q~ 2 2008 4 manu~ 4 20 28 p comp~  
## # ... with 224 more rows

This will load the entire mpg dataset.

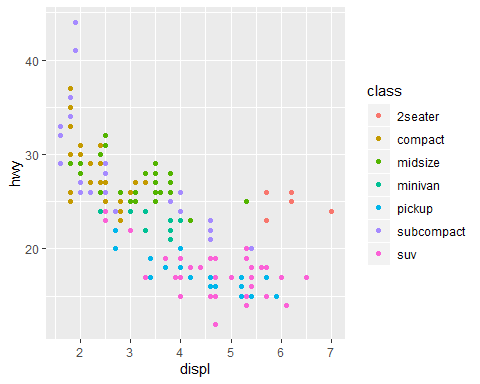
Creating a ggplot for displ VS hwy

ggplot(data = mpg) + geom\_point(mapping = aes(x=displ, y=hwy))



\*\*\*\*\*Aesthetic Mappings in GGPlot 1: Color

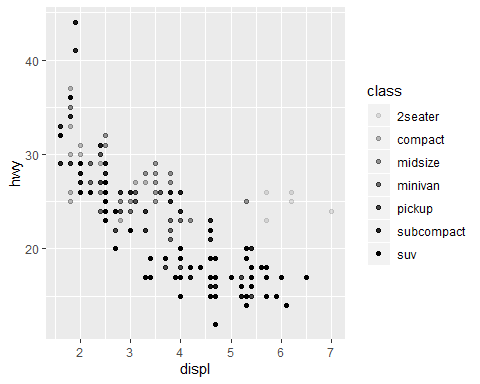
ggplot(data = mpg) + geom\_point(mapping = aes(x=displ, y=hwy, color = class))

 This code will plot the same scatterplot but will color the points with respect to class variable

2: Alpha

ggplot(data = mpg) + geom\_point(mapping = aes(x=displ, y=hwy, alpha = class))

## Warning: Using alpha for a discrete variable is not advised.

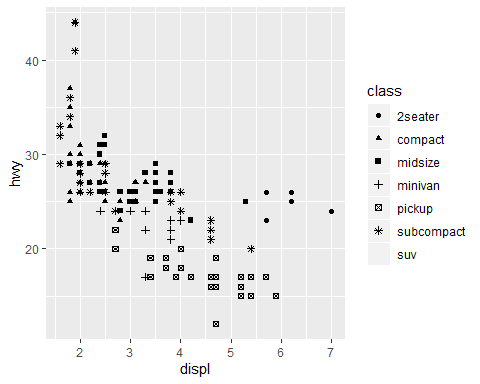
 This code will plot the scatterplot and alpha aesthetic will control the transparency of the points.

3: Shape

ggplot(data = mpg) + geom\_point(mapping = aes(x=displ, y=hwy, shape = class))

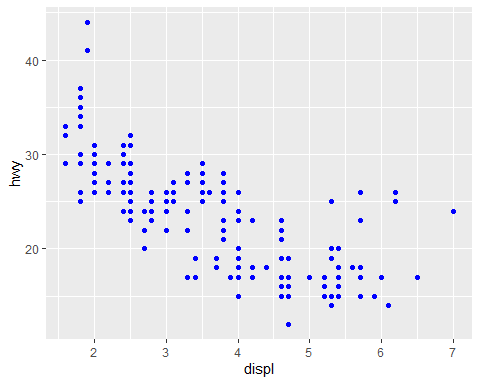
## Warning: The shape palette can deal with a maximum of 6 discrete values  
## because more than 6 becomes difficult to discriminate; you have 7.  
## Consider specifying shapes manually if you must have them.

## Warning: Removed 62 rows containing missing values (geom\_point).

 This code will plot the scatterplot and shape aesthetic will give different shapes to different class variable. But ggplot2 will only use 6 shapes at a time.

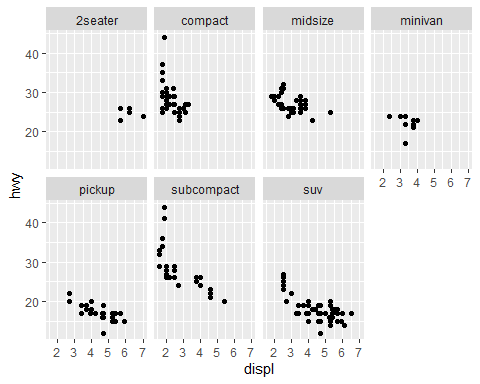
4: Color all points

ggplot(data = mpg) + geom\_point(mapping = aes(x=displ, y=hwy), color = "blue")



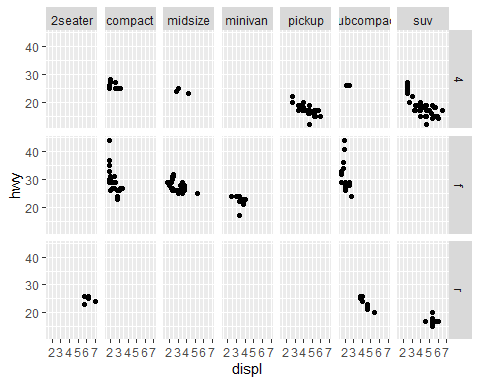
\*\*\*\*\*Facets in GGPlot 1. facet\_wrap()

ggplot(data = mpg) + geom\_point(mapping = aes(x=displ, y=hwy))+  
 facet\_wrap(~class, nrow = 2)

 facet\_wrap() is used to facet the plot by single variable. We can adjust. The number of rows and columns using nrow or ncol. The variable used to pass facet\_wrap should be discrete

1. facet\_grid()

ggplot(data = mpg) + geom\_point(mapping = aes(x=displ, y=hwy))+  
 facet\_grid(drv~class)

 facet\_grid() will facet the plot based on the combination of 2 variables. Before ~ is shown in rows and after ~ is shown in columns.

\*\*\*\*\*Geometric Objects 1. geom\_point()

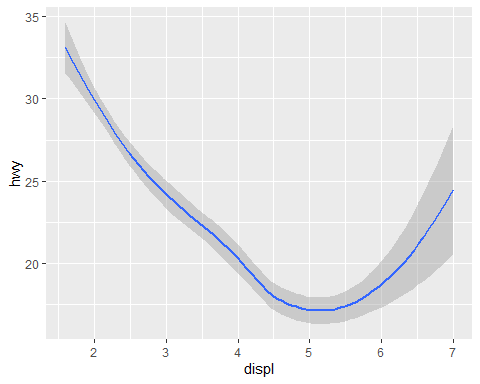
ggplot(data = mpg) + geom\_point(mapping = aes(x=displ, y=hwy))

 This will plot scatterplot

1. geom\_smooth()

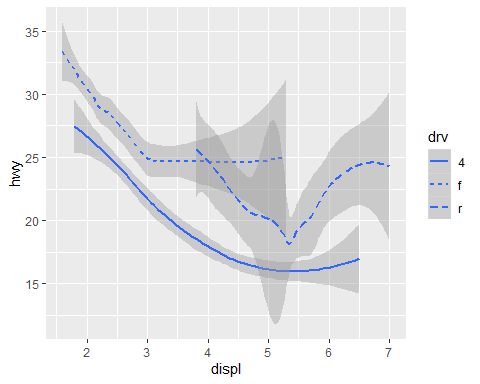
ggplot(data = mpg) + geom\_smooth(mapping = aes(x=displ, y=hwy))

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

 This will plot the best fit line with its confidence interval

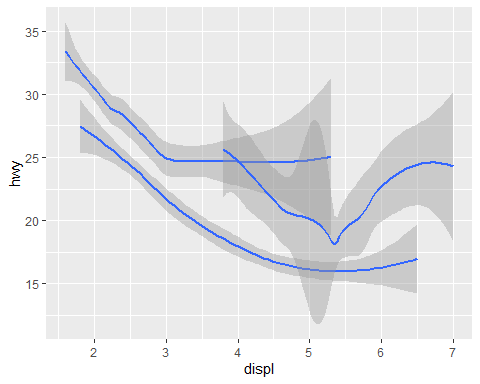
ggplot(data = mpg) + geom\_smooth(mapping = aes(x=displ, y=hwy, linetype = drv))

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

 This will plot the best fit line by distinguising the drv variable by its linetype

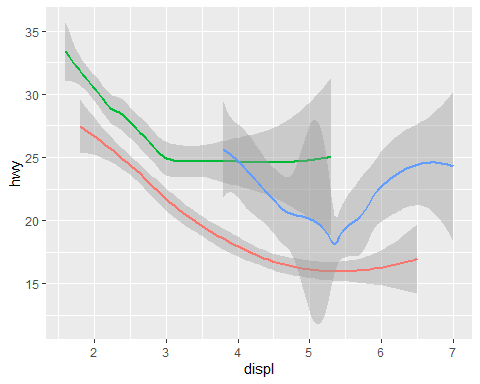
ggplot(data = mpg) + geom\_smooth(mapping = aes(x=displ, y=hwy, group=drv))

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

 This will create the best fit lines by grouping the drv variable

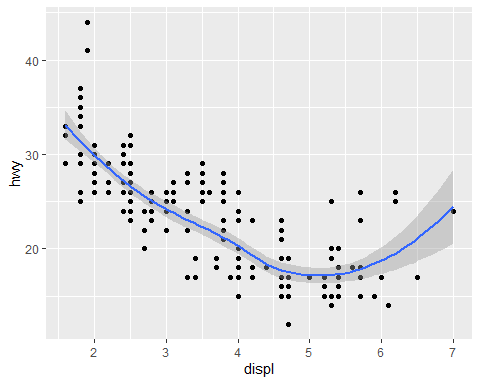
ggplot(data = mpg) + geom\_smooth(mapping = aes(x=displ, y=hwy, color=drv), show.legend = FALSE)

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

 This will create the best fit line by distinguisng the drv variable in color format

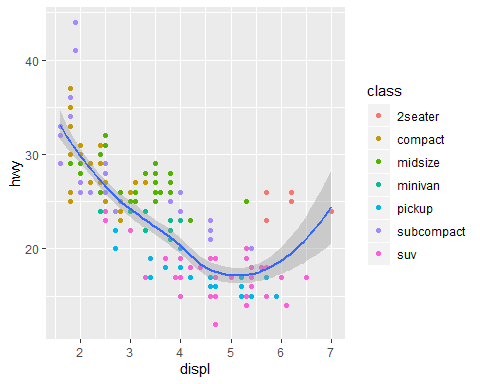
ggplot(data = mpg, mapping = aes(x=displ, y=hwy))+  
 geom\_point()+geom\_smooth()

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

 The above code will create a scatterplot along with the best fit line. This code is easier if we have to do any modifications since we have to change variables only in ggplot() statement

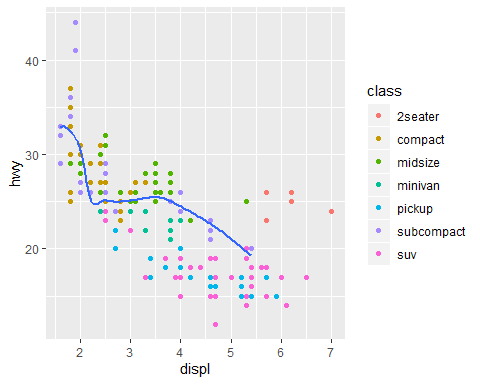
ggplot(data = mpg, mapping = aes(x=displ, y=hwy))+  
 geom\_point(mapping = aes(color=class))+geom\_smooth()

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

 This plot will create a scatterplot and all points will be colored based on different class variable. Also a best fit line for the data is plotted

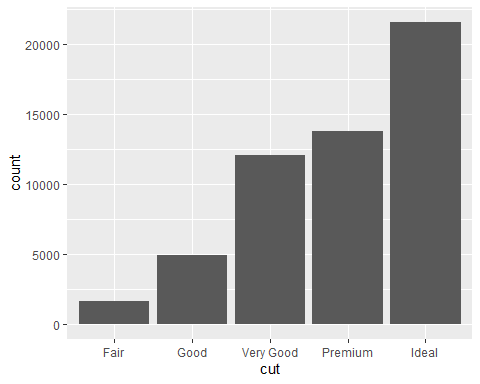
ggplot(data = mpg, mapping = aes(x=displ, y=hwy))+  
 geom\_point(mapping = aes(color=class))+  
 geom\_smooth(data = filter(mpg, class == "subcompact"), se = FALSE)

## `geom\_smooth()` using method = 'loess' and formula 'y ~ x'

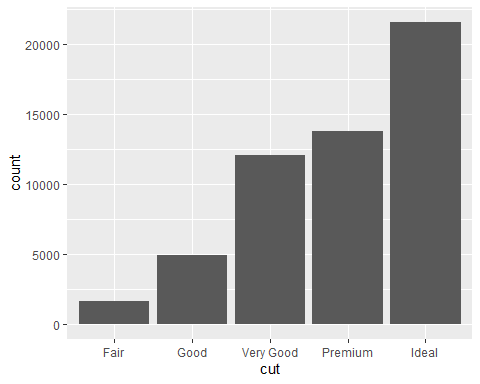
 This code will create a scatterplot with all points colored differently based on class variable. It will also create a best fit line but only for the subcompact class

\*\*\*\*\*Statistical Transformations

ggplot(data = diamonds) + geom\_bar(mapping = aes(x=cut))

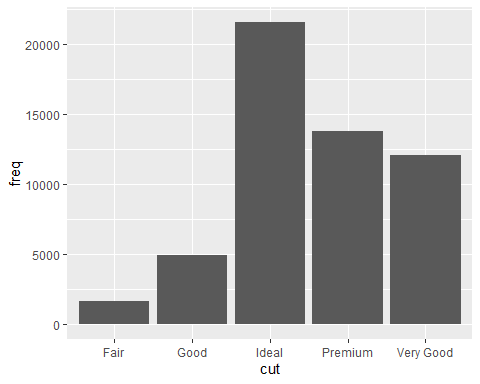
 The above code will create a bar chart for the diamonds data set grouped by the cut variable on the X axis

ggplot(data = diamonds) + stat\_count(mapping = aes(x=cut))

 This will create the same plot but by using the stat argument. stat\_count corresponds to geom\_bar

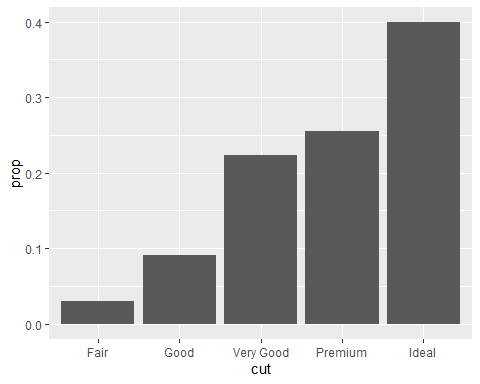
1. To change the stat from default to something else Here stat\_count is changed to identity. This will map the height of the bars to the raw values of Y variable

demo <- tribble(  
 ~cut, ~freq,  
 "Fair", 1610,  
 "Good", 4906,  
 "Very Good", 12082,  
 "Premium", 13791,  
 "Ideal", 21551)  
ggplot(data = demo) +  
 geom\_bar(mapping = aes(x=cut, y=freq), stat = "identity")



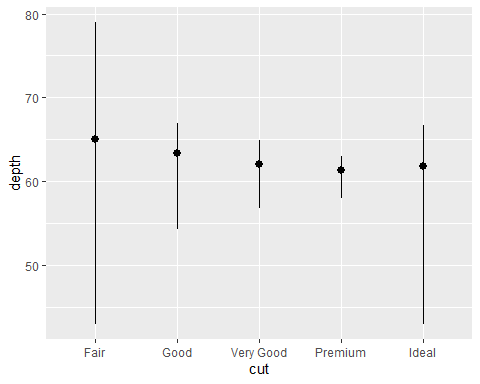
1. To display a bar chart of proportion

ggplot(data = diamonds) +  
 geom\_bar(mapping = aes(x=cut, y=..prop.., group=1))

 This will plot the bar chart but proportion on Y axis

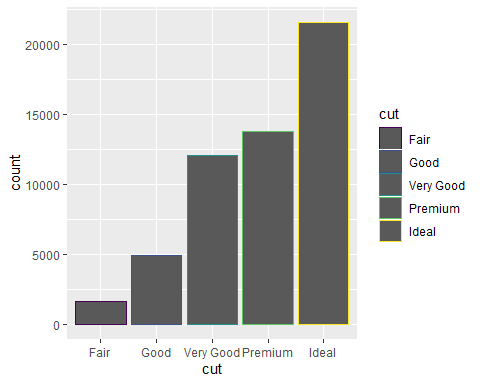
1. Using statistical transformation

ggplot(data = diamonds)+  
 stat\_summary(  
 mapping = aes(x=cut,y=depth),  
 fun.y = median,  
 fun.ymax = max,  
 fun.ymin = min  
 )



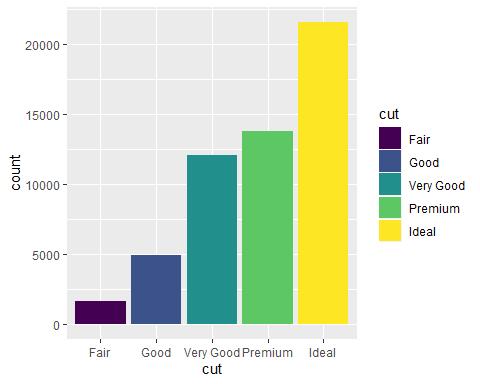
\*\*\*\*\*POSITION ADJUSTMENTS 1. To color the border of bar chart based on a variable

ggplot(data = diamonds) +  
 geom\_bar(mapping = aes(x=cut, color=cut))



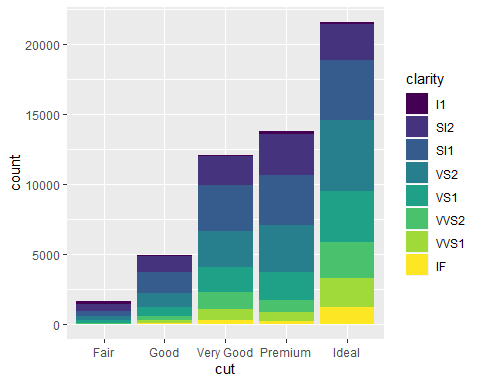
1. To fill the bars with colors

ggplot(data = diamonds) +  
 geom\_bar(mapping = aes(x=cut, fill=cut))



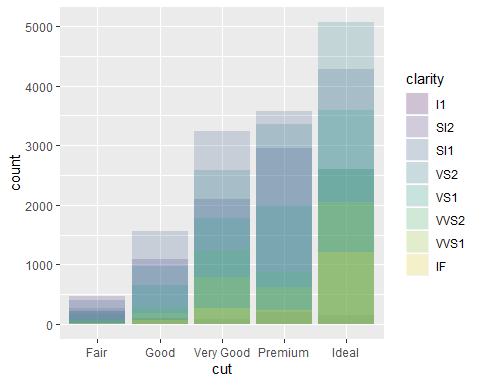
1. To create a stacked bar chart

ggplot(data = diamonds) +  
 geom\_bar(mapping = aes(x=cut, fill=clarity))

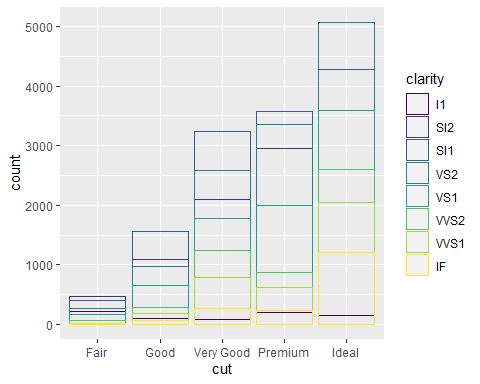
 This is done because another variable is assigned to the fill argument

1. If we do not need stacking we can use the position argument to avoid it as follows: 4.1 position=identity

ggplot(data = diamonds, mapping = aes(x=cut, fill=clarity))+  
 geom\_bar(alpha=1/5, position = "identity")

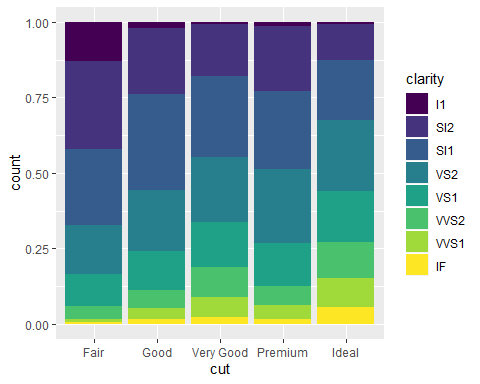


ggplot(data = diamonds, mapping = aes(x=cut, color=clarity))+  
 geom\_bar(fill=NA, position = "identity")

 The above two graphs will place each object excatly where it falls in the context of the graph. aplha will create transparency in color and second code will only create a colored border for each object

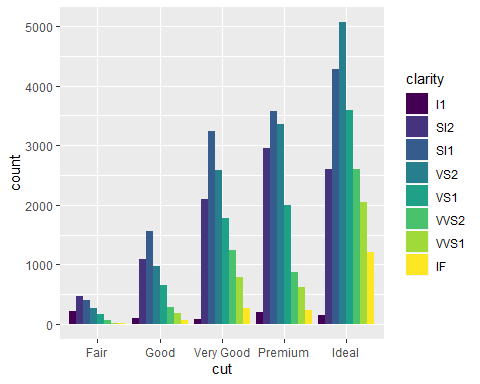
4.2 position=fill

ggplot(data = diamonds) +  
 geom\_bar(mapping = aes(x=cut, fill=clarity), position = "fill")

 The above code will create a bar chart with each set of stacked bar chart the same height. Y axis will be proportion

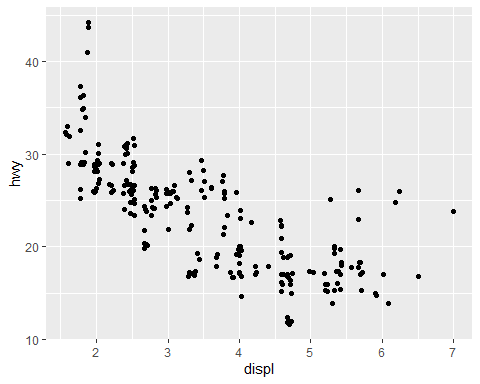
4.3 position=dodge

ggplot(data = diamonds) +  
 geom\_bar(mapping = aes(x=cut, fill=clarity), position = "dodge")

 The above code will place overlapping objects next to each other

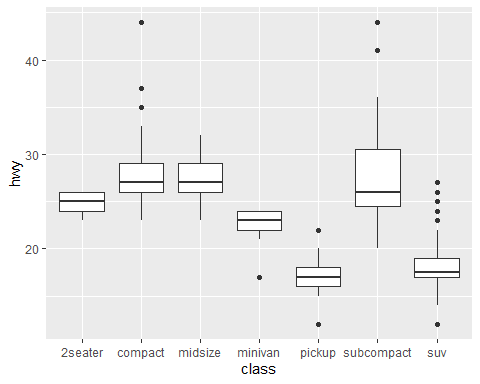
\*\*\*\*\*Jitter

ggplot(data = mpg)+  
 geom\_point(mapping = aes(x=displ,y=hwy),position = "jitter")

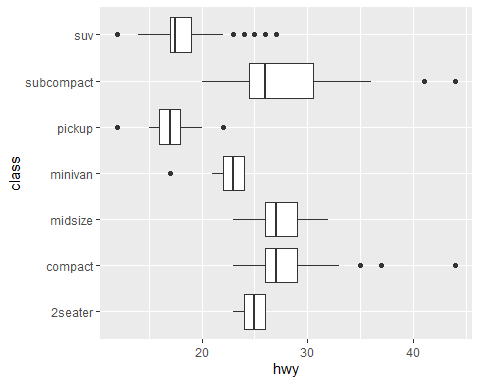
 This code will add small random noise to each point. This avoids overplotting.

\*\*\*\*\*COORDINATE SYSTEM 1. coord\_flip()

ggplot(data = mpg, mapping = aes(x=class,y=hwy)) +  
 geom\_boxplot()

 This will create a boxplot for every class variable VS hwy

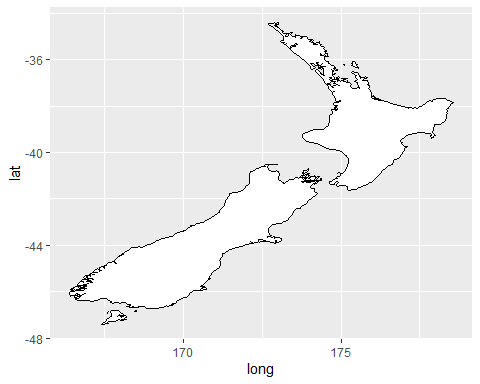
ggplot(data = mpg, mapping = aes(x=class,y=hwy)) +  
 geom\_boxplot()+coord\_flip()

 This code will flip the X and Y axis as compared to the previous code

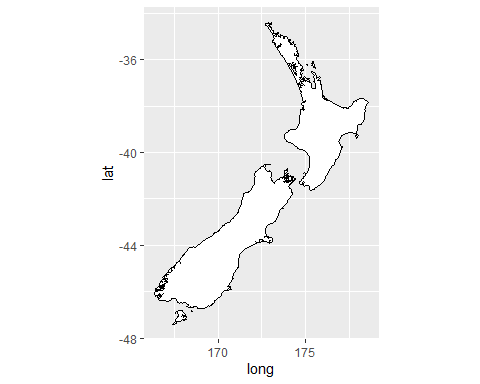
coord\_quickmap()

## <ggproto object: Class CoordQuickmap, CoordCartesian, Coord, gg>  
## aspect: function  
## backtransform\_range: function  
## clip: on  
## default: FALSE  
## distance: function  
## expand: TRUE  
## is\_free: function  
## is\_linear: function  
## labels: function  
## limits: list  
## modify\_scales: function  
## range: function  
## render\_axis\_h: function  
## render\_axis\_v: function  
## render\_bg: function  
## render\_fg: function  
## setup\_data: function  
## setup\_layout: function  
## setup\_panel\_params: function  
## setup\_params: function  
## transform: function  
## super: <ggproto object: Class CoordQuickmap, CoordCartesian, Coord, gg>

nz <- map\_data("nz")  
ggplot(nz, aes(long, lat, group=group))+  
 geom\_polygon(fill="white", color="black")



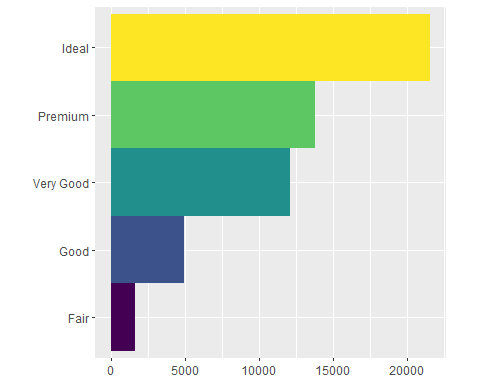
nz <- map\_data("nz")  
ggplot(nz, aes(long, lat, group=group))+  
 geom\_polygon(fill="white", color="black")+ coord\_quickmap()

 This function is used to set the aspect ratio for maps.

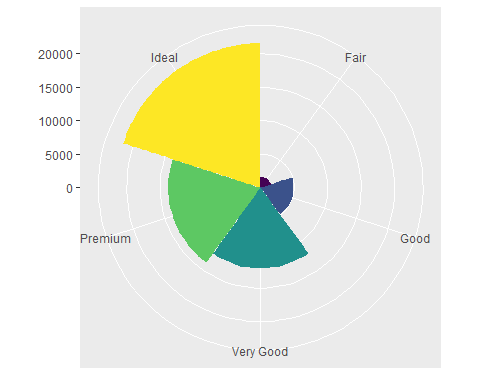
coord\_polar()

## <ggproto object: Class CoordPolar, Coord, gg>  
## aspect: function  
## backtransform\_range: function  
## clip: on  
## default: FALSE  
## direction: 1  
## distance: function  
## is\_free: function  
## is\_linear: function  
## labels: function  
## modify\_scales: function  
## r: y  
## range: function  
## render\_axis\_h: function  
## render\_axis\_v: function  
## render\_bg: function  
## render\_fg: function  
## setup\_data: function  
## setup\_layout: function  
## setup\_panel\_params: function  
## setup\_params: function  
## start: 0  
## theta: x  
## transform: function  
## super: <ggproto object: Class CoordPolar, Coord, gg>

bar <- ggplot(data = diamonds)+  
 geom\_bar(mapping = aes(x=cut, fill=cut), show.legend = FALSE , width = 1)+  
 theme(aspect.ratio = 1)+  
 labs(x=NULL, y=NULL)  
bar+coord\_flip()



bar+coord\_polar()

 The above code initially creates a bar chart for variable cut from dataset diamonds. Then the bar chart is flipped using coord\_fli(). This flipped barchart is converted to a pie chart by using coord\_polar()