MB5370\_Module\_04\_JC

James Coon

2025-03-19

# Module 4, Data Science in R

### Load Packages

#install.packages("tidyverse", 'ggplot2')  
library(tidyverse, ggplot2)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.1 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.4 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

### Load Data

mpg

## # A tibble: 234 × 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 quattro 1.8 1999 4 manu… 4 18 26 p comp…  
## 9 audi a4 quattro 1.8 1999 4 auto… 4 16 25 p comp…  
## 10 audi a4 quattro 2 2008 4 manu… 4 20 28 p comp…  
## # ℹ 224 more rows

### Create first plot

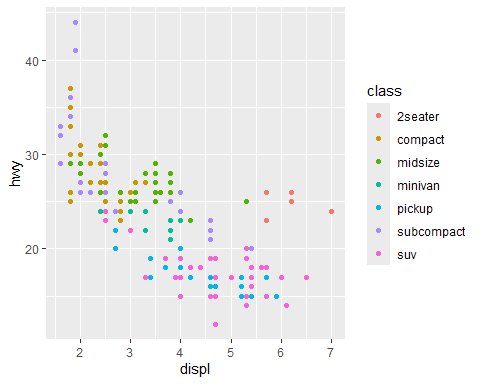
ggplot(data = mpg) +   
 geom\_point(mapping = aes(x = displ, y = hwy))



### Graphing Template:

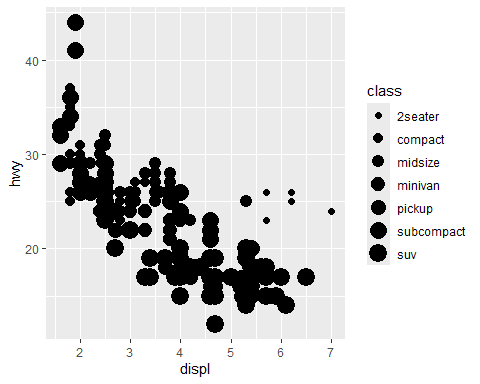
ggplot(data = ) + (mapping = aes())

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



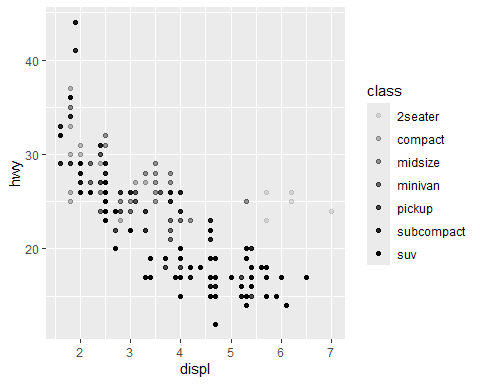
ggplot(data = mpg) +   
 geom\_point(mapping = aes(x = displ, y = hwy, size = class)) #point size by class!

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



ggplot(data = mpg) +   
 geom\_point(mapping = aes(x = displ, y = hwy, alpha = class)) #alpha is transparency, thus transparency by class!

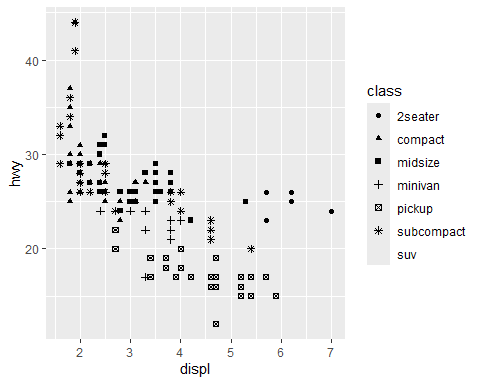
## Warning: Using alpha for a discrete variable is not advised.



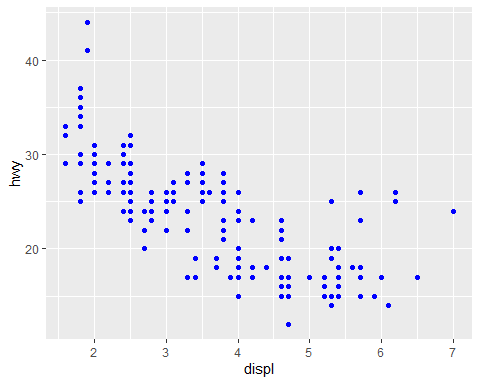
ggplot(data = mpg) +   
 geom\_point(mapping = aes(x = displ, y = hwy, shape = class)) #point shape by class!

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

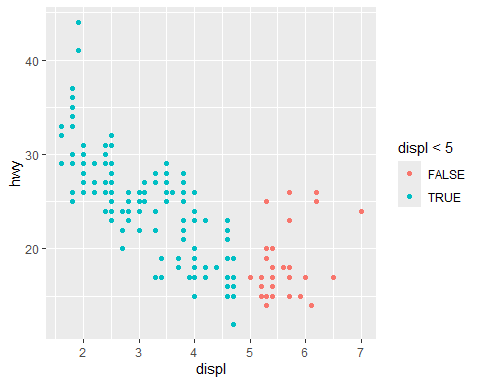
## Warning: Removed 62 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



ggplot(data = mpg) +   
 geom\_point(mapping = aes(x = displ, y = hwy), color = "blue") #can manually make all points whatever color you want!

 What happens if aes(colour = displ < 5)? It creates a legend which displays values over a 5 dspl in turquoise and under in red.

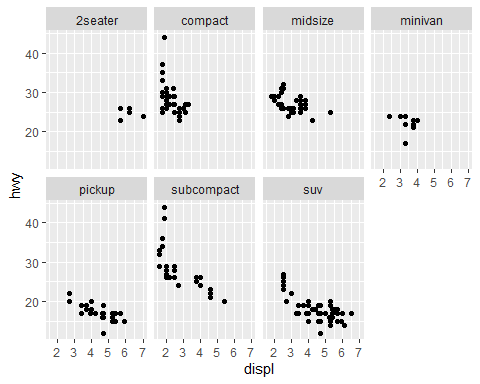
ggplot(data = mpg) +   
 geom\_point(mapping = aes(x = displ, y = hwy, color = displ < 5))



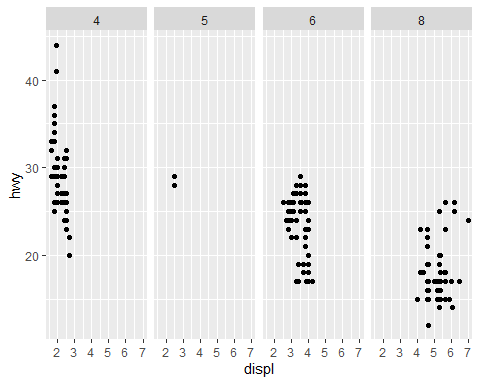
### Facets and Panel Plots

Creates plots in a grid format, where it’s easier to compare them.

ggplot(data = mpg) +   
 geom\_point(mapping = aes(x = displ, y = hwy)) +   
 facet\_wrap(~ class, nrow = 2) #Creates 2 rows, regardless of amount of variables/plots



ggplot(data = mpg) +   
 geom\_point(mapping = aes(x = displ, y = hwy)) +   
 facet\_grid(. ~ cyl) #If showing more than 1 variable as a row, use facet\_grid instead of facet\_wrap.

 Exercise-

What does ncol and nrow do? They choose the number of columns and rows respectively.

What other options control the layout of the individual panels? Reorder and as.table

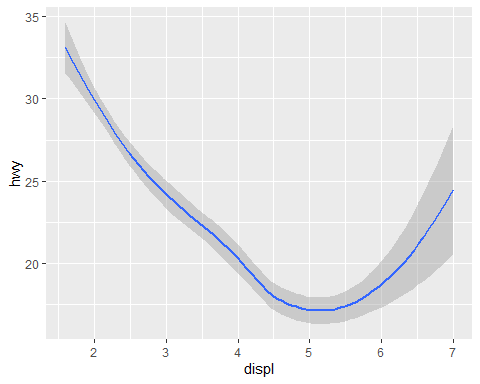
## Fitting Simple Lines

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



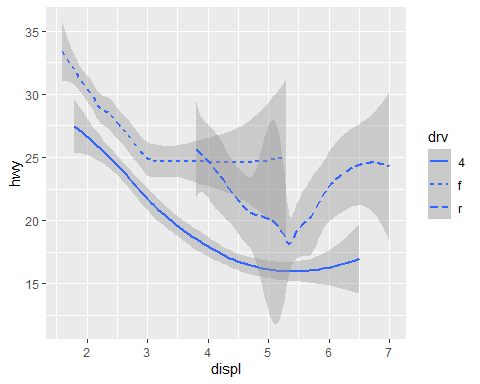
ggplot(data = mpg) +   
 geom\_smooth(mapping = aes(x = displ, y = hwy)) #Plot as smooth line! geom\_smooth

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



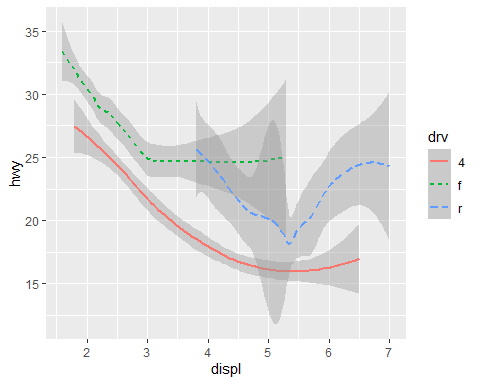
ggplot(data = mpg) +   
 geom\_smooth(mapping = aes(x = displ, y = hwy, linetype = drv)) #Can change plot line by class!

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



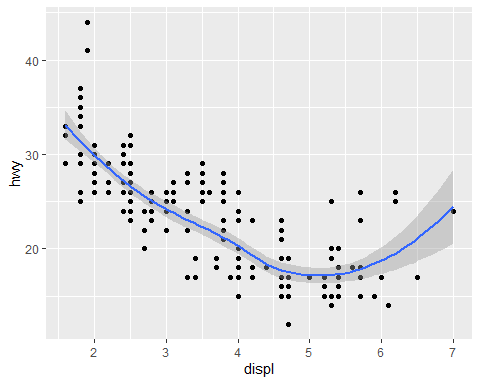
ggplot(data = mpg) +  
 geom\_smooth(mapping = aes(x = displ, y = hwy, group = drv, color=drv, linetype = drv)) #Change some aesthetic based on drv

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



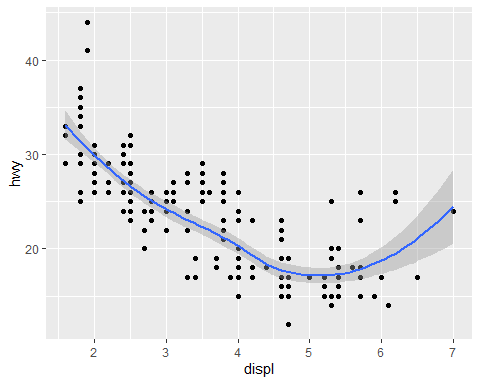
ggplot(data = mpg) +   
 geom\_point(mapping = aes(x = displ, y = hwy)) +  
 geom\_smooth(mapping = aes(x = displ, y = hwy)) #Plots both line and points. Must just be used additively

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



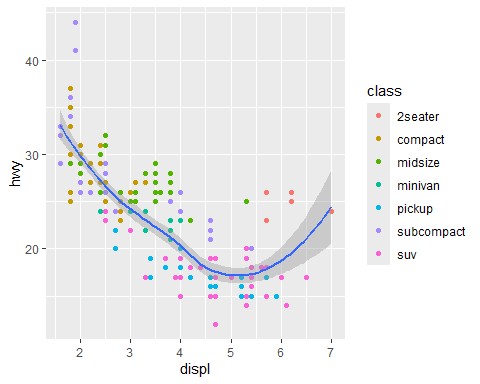
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +   
 geom\_point() +   
 geom\_smooth() #Same as above, but more efficient

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



ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +   
 geom\_point(mapping = aes(color = class)) +   
 geom\_smooth() #Style each point individually based upon many variables

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

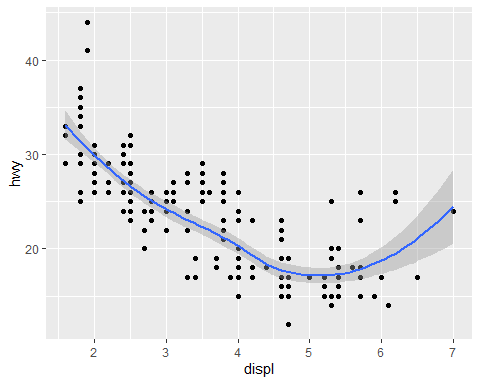


#### Exercise

1. What geom would you use to draw a line chart? A boxplot? A histogram? An area chart? line cart- geom\_smooth boxplot- geom\_boxplot histogram- geom\_histogram or geom\_bar
2. Run this code in your head and predict what the output will look like. Then, run the code in R and check your predictions.

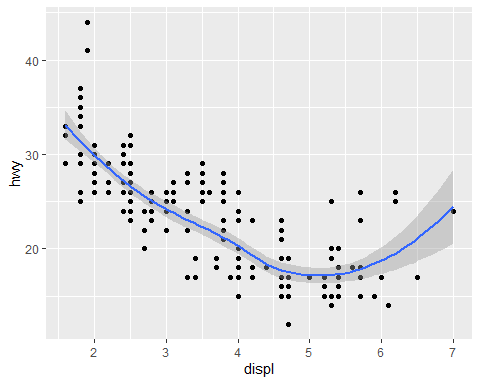
#I predict both graphs will be the same.  
  
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +   
 geom\_point() +   
 geom\_smooth()

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



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

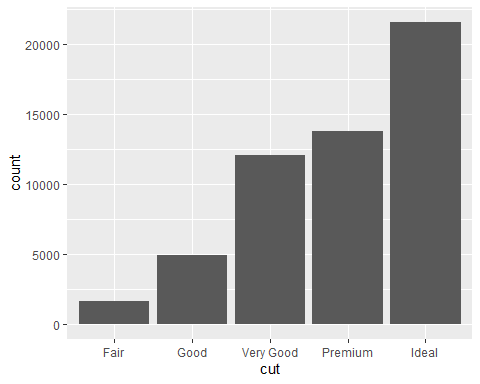
## `geom\_smooth()` using method = 'loess' and formula = 'y ~ x'



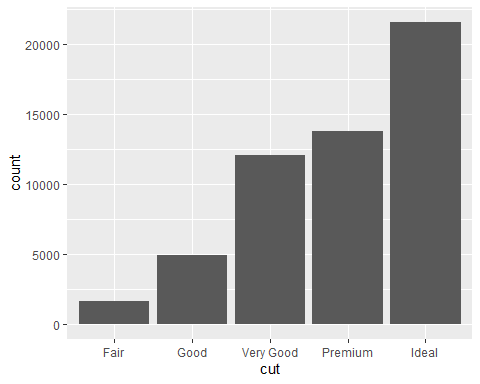
1. Will these two graphs look different? Why/why not? No, because the specifications made in the second plot are listed in the first plot. The first plot is simply more efficient because the specifications only have to be typed once.

## Transformations and Stats

ggplot(data = diamonds) + #data found in ggplot2  
 geom\_bar(mapping = aes(x = cut))



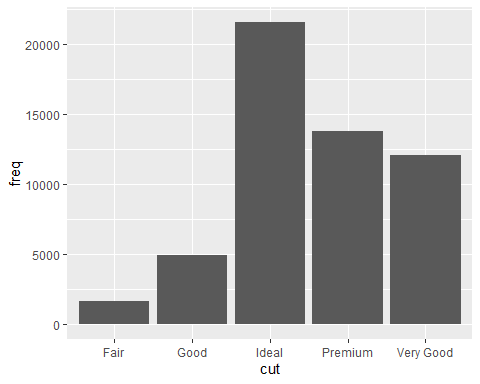
ggplot(data = diamonds) +   
 stat\_count(mapping = aes(x = cut)) #can often use stat and geom interchangeably

 #### Overriding Defaults

demo <- tribble( #tribble is basically a dataframe  
 ~cut, ~freq,  
 "Fair", 1610,  
 "Good", 4906,  
 "Very Good", 12082,  
 "Premium", 13791,  
 "Ideal", 21551  
)  
demo

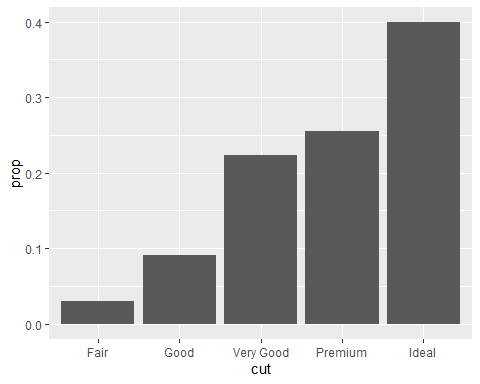
## # A tibble: 5 × 2  
## cut freq  
## <chr> <dbl>  
## 1 Fair 1610  
## 2 Good 4906  
## 3 Very Good 12082  
## 4 Premium 13791  
## 5 Ideal 21551

ggplot(data = demo) +  
 geom\_bar(mapping = aes(x = cut, y = freq), stat = 'identity') #default stat is count



ggplot(data = diamonds) +   
 geom\_bar(mapping = aes(x = cut, y = stat(prop), group = 1)) # y is a proportion rather than raw count

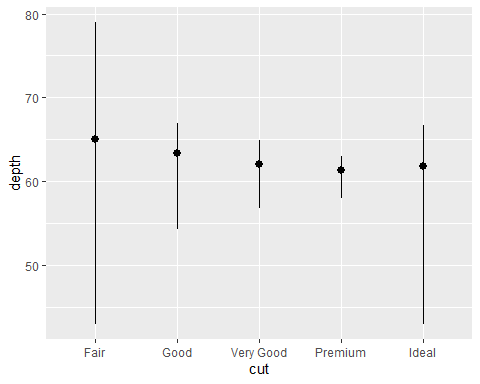
## Warning: `stat(prop)` was deprecated in ggplot2 3.4.0.  
## ℹ Please use `after\_stat(prop)` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.



#### Plotting Statistical Details

Showing statistical summaries is a good ideal for transparency of your work

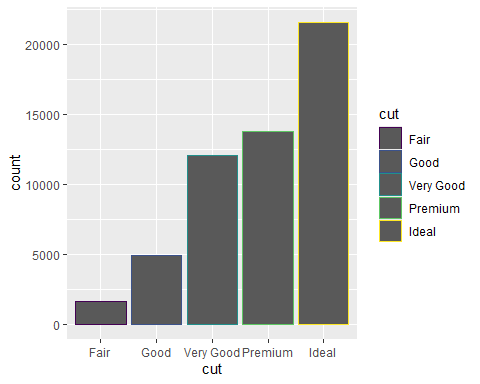
ggplot(data = diamonds) +   
 stat\_summary( #Shows error bars when specified  
 mapping = aes(x = cut, y = depth),  
 fun.min = min,  
 fun.max = max,  
 fun = median)



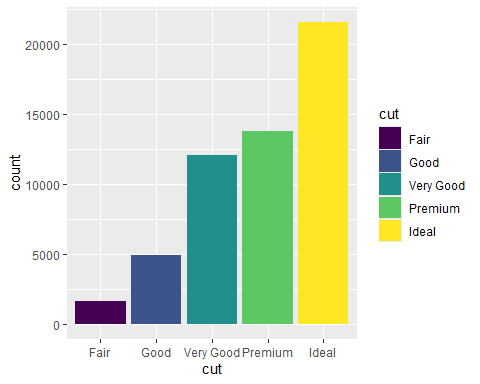
## Aesthetic Adjustments

You can use commands such as color or fill to customize bars.

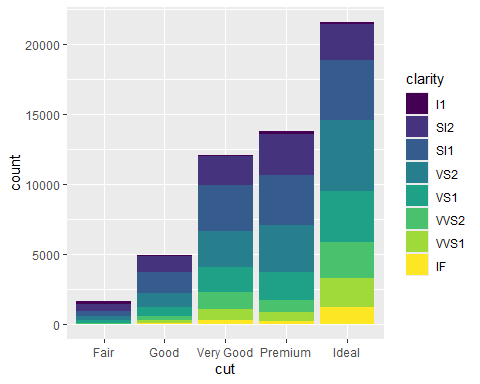
ggplot(data = diamonds) +   
 geom\_bar(mapping = aes(x = cut, colour = cut)) #only outline is colored



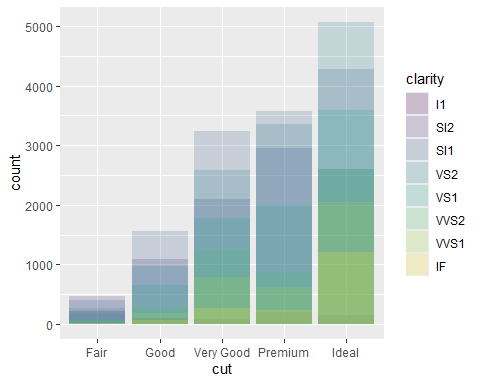
ggplot(data = diamonds) +   
 geom\_bar(mapping = aes(x = cut, fill = cut)) #Fills bars



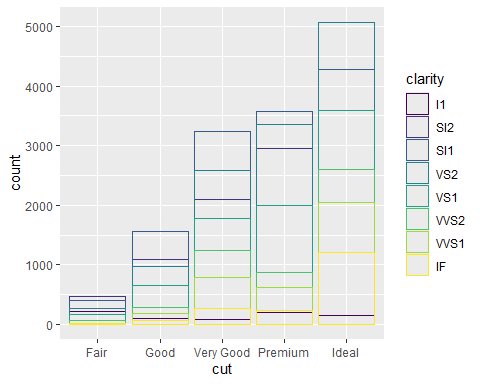
ggplot(data = diamonds) +   
 geom\_bar(mapping = aes(x = cut, fill = clarity)) #Stacks clarity coloration into existing bars!



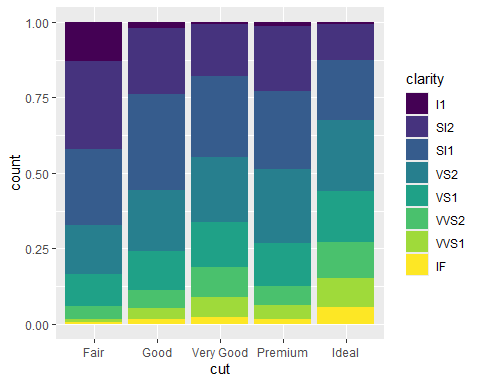
ggplot(data = diamonds) +  
(mapping = aes(x = cut, fill = clarity)) +   
 geom\_bar(alpha = 1/5, position = "identity") #Alpha alters transparency



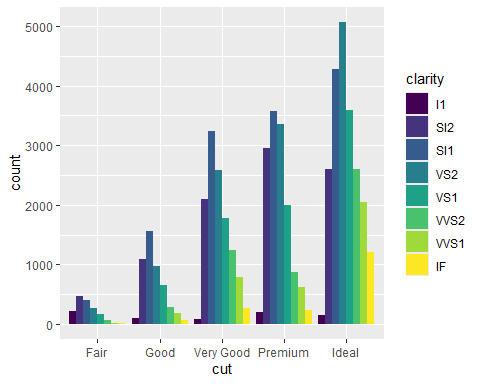
ggplot(data = diamonds) +   
 (mapping = aes(x = cut, colour = clarity)) +   
 geom\_bar(fill = NA, position = "identity") #Transparent bars and colored outline.



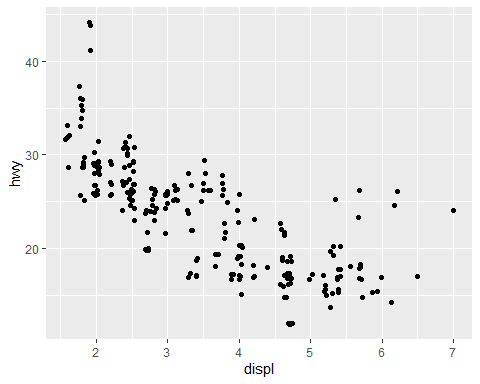
ggplot(data = diamonds) +   
 geom\_bar(mapping = aes(x = cut, fill = clarity), position = "fill") #Similar to stacking, but makes all the bars the same height.



ggplot(data = diamonds) +   
 geom\_bar(mapping = aes(x = cut, fill = clarity), position = "dodge") #Dodge puts overlapping objects next to eachother instead of overlapping.



ggplot(data = mpg) +   
 geom\_point(mapping = aes(x = displ, y = hwy), position = "jitter") #Jitter makes random noise in the plot to slightly and randomly space data out and make it mroe visible.



##The layered grammar of graphics

An updated plot template:

ggplot(data = ) + ( mapping = aes(), stat = , position = ) +

## Workshop 1 Assignment

IMPORT DATASET

library(readxl)  
API\_ER\_FSH\_AQUA\_MT\_DS2\_en\_excel\_v2\_967 <- read\_excel("C:/Users/jacoo/Downloads/API\_ER.FSH.AQUA.MT\_DS2\_en\_excel\_v2\_967.xls",   
 sheet = "Sheet2")  
View(API\_ER\_FSH\_AQUA\_MT\_DS2\_en\_excel\_v2\_967)  
JA\_Fisheries <- API\_ER\_FSH\_AQUA\_MT\_DS2\_en\_excel\_v2\_967

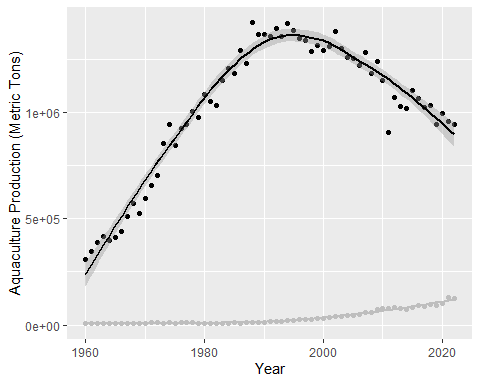
Alter the plot!

JA\_Fisheries$Year<- as.numeric(JA\_Fisheries$Year) #This held us up for a half hour lol. Need to assign to the data sheet, not just as.numeric(data)  
JA\_Fisheries <- na.omit(JA\_Fisheries)  
str(JA\_Fisheries$Year)

## num [1:63] 1960 1961 1962 1963 1964 ...

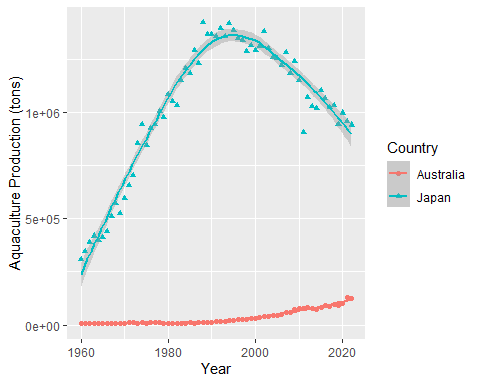
ggplot(data= JA\_Fisheries) +   
 geom\_point(mapping = aes(x= Year, y= Japan), color="black") +  
 geom\_point(mapping = aes(x= Year, y= Australia), color="grey") +  
 geom\_smooth(aes(x= Year, y= Japan), color="black", method='loess') +  
 geom\_smooth(aes(x= Year, y= Australia), color="grey", method='loess') +  
labs(y = "Aquaculture Production (Metric Tons)", x = "Year", color = 'Country') #This coloration is better, but not for color blindness.

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



ggplot(data = JA\_Fisheries) +   
 geom\_point(aes(x = Year, y = Japan, color = "Japan", shape = "Japan")) +  
 geom\_point(aes(x = Year, y = Australia, color = "Australia", shape = "Australia")) +  
 geom\_smooth(aes(x = Year, y = Japan, color = "Japan"), method = "loess") +  
 geom\_smooth(aes(x = Year, y = Australia, color = "Australia"), method = "loess") +  
 labs(y = "Aquaculture Production (tons)", x = "Year", color = "Country", shape= "Country") # Legend title, and symbology is much better for the color blind, though color may not be ideal.

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

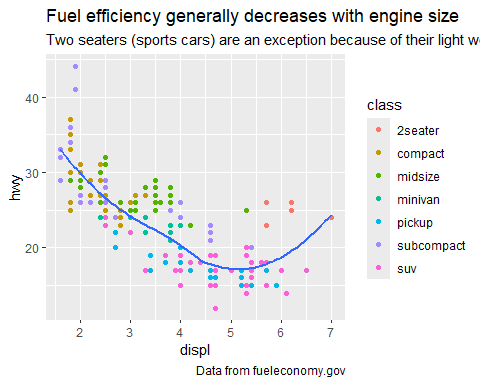
 Caption- The aquaculture production in Australia is much lower than in Japan. However, there is a noticeable increase in Australia’s production recently (the 1990s) while Japan has been declining consistently since about 1995. Data provided by World Data Bank.

#Workshop 2- GGplot2 for Communication

## Labels

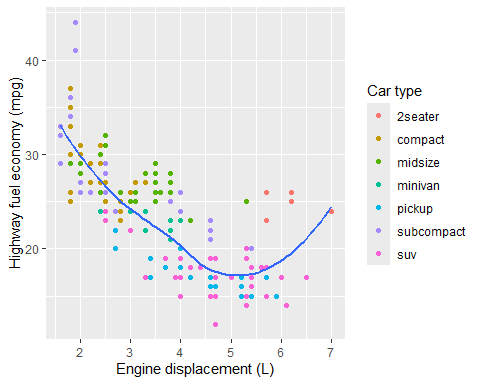
ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(color = class)) +  
 geom\_smooth(se = FALSE) +  
 labs(  
 title = "Fuel efficiency generally decreases with engine size", #Main title label  
 subtitle = "Two seaters (sports cars) are an exception because of their light weight", #smaller title label  
 caption = "Data from fueleconomy.gov" #by default appears on the bottom of page  
 )

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



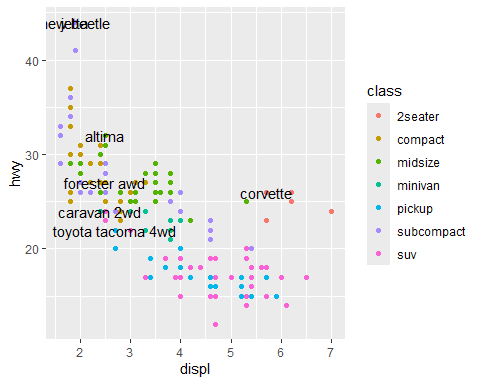
ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(colour = class)) +  
 geom\_smooth(se = FALSE) +  
 labs(  
 x = "Engine displacement (L)", #axis titles  
 y = "Highway fuel economy (mpg)",  
 colour = "Car type" #legend title  
 )

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



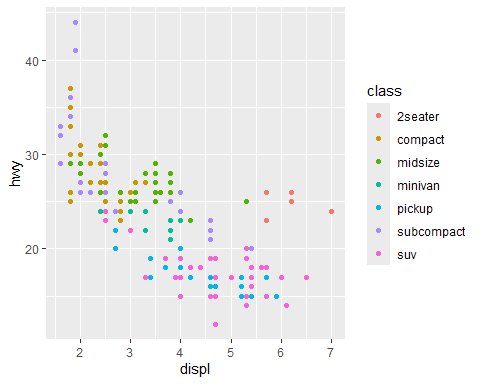
## Annotations

best\_in\_class <- mpg %>%  
 group\_by(class) %>%  
 filter(row\_number(desc(hwy)) == 1) #This bit of code filters out the best mpg car for each car class!  
  
ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(colour = class)) +  
 geom\_text(aes(label = model), data = best\_in\_class) #annotates each best car model for the mpg we selected earlier! Annotation overlapping can be resolved with nudge()

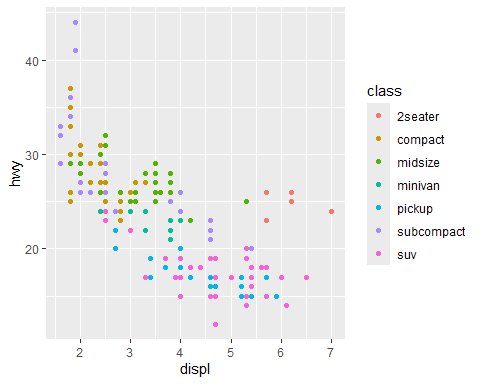


## Scales

ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(colour = class)) #Automatically selects scales for you, but you can tweak them if you need

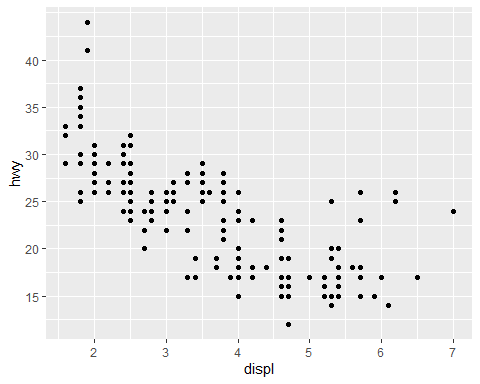


ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(colour = class)) +  
 scale\_x\_continuous() +  
 scale\_y\_continuous() +  
 scale\_colour\_discrete() #Can also provide ylims and xlims!

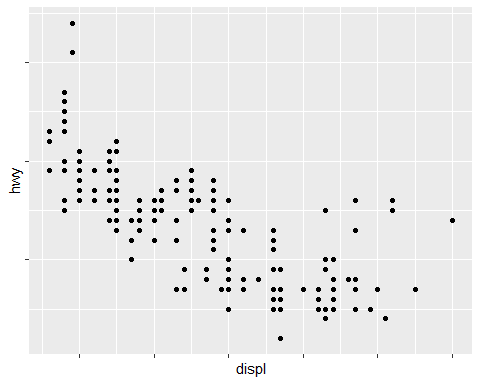


## Axis Ticks

ggplot(mpg, aes(displ, hwy)) +  
 geom\_point() +  
 scale\_y\_continuous(breaks = seq(15, 40, by = 5)) #Seq sets limits and selects ticks by the interval selected within the limits.

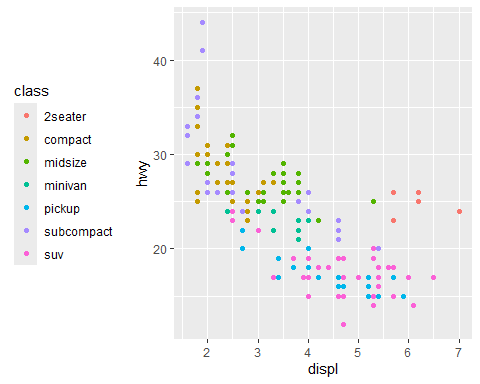


ggplot(mpg, aes(displ, hwy)) +  
 geom\_point() +  
 scale\_x\_continuous(labels = NULL) +  
 scale\_y\_continuous(labels = NULL) #Set to null to rid of labels all together!

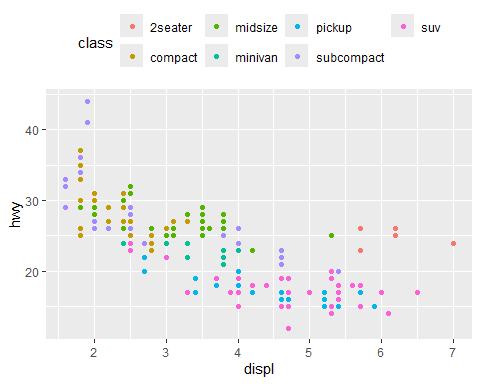


## Legends and Color Schemes

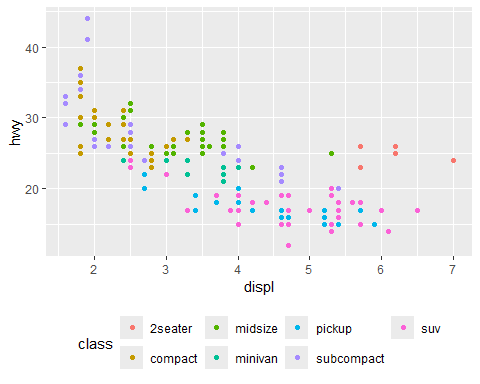
base <- ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(colour = class))  
  
base + theme(legend.position = "left")



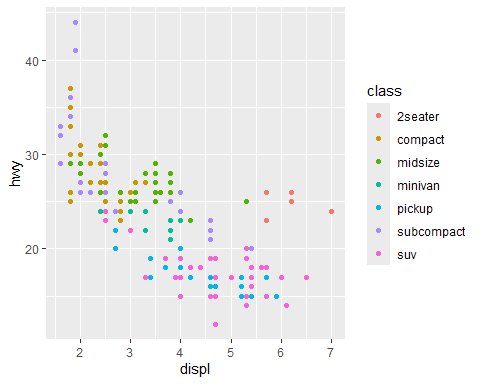
base + theme(legend.position = "top")



base + theme(legend.position = "bottom")

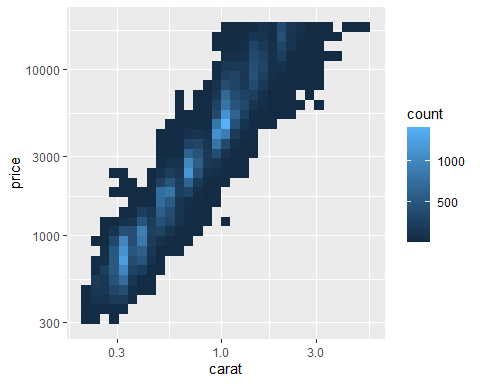


base + theme(legend.position = "right") #Simply changes the position of the legend in relation to the plot

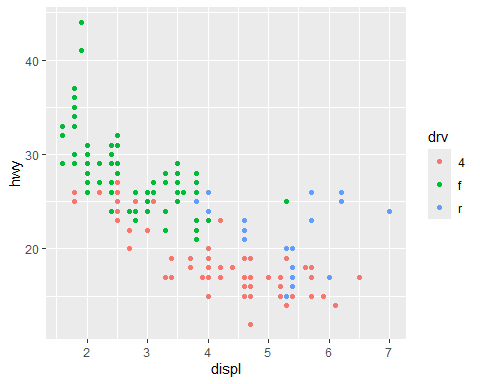


## Replacing a Scale

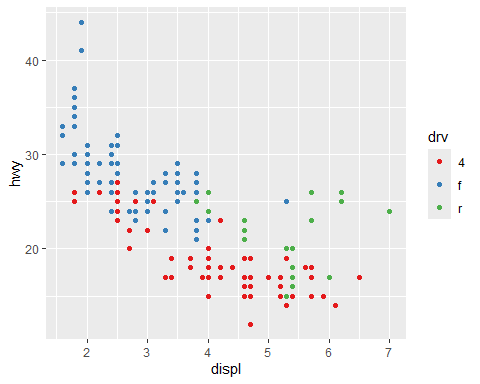
ggplot(diamonds, aes(carat, price)) +  
 geom\_bin2d() +   
 scale\_x\_log10() +   
 scale\_y\_log10() #Scale transformations can show important information! Remember to say they were transformed for transparency.



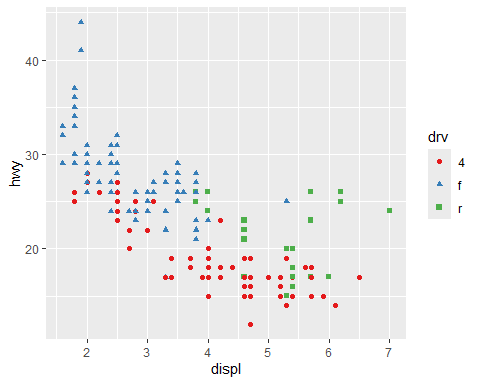
ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(color = drv)) #default color scale



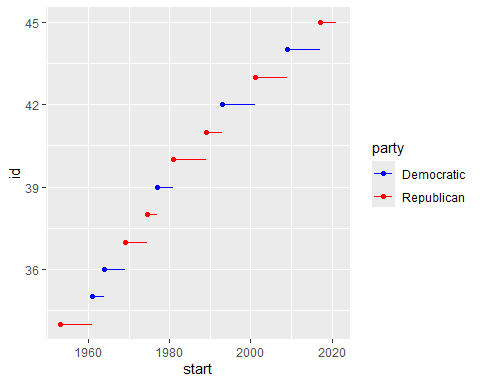
ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(color = drv)) +  
 scale\_colour\_brewer(palette = "Set1") #Creates own color palette!



ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(color = drv, shape = drv)) + #adding shape makes graph interpretable when black and white as well! good for color blindness and cheap printing.  
 scale\_colour\_brewer(palette = "Set1")



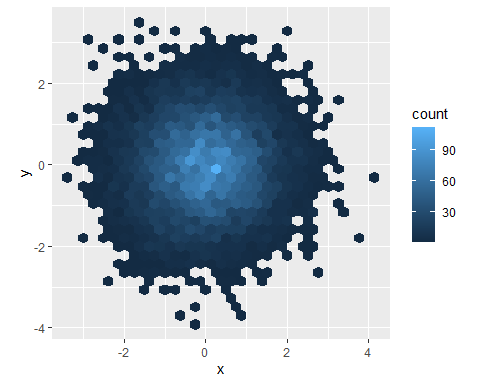
presidential %>%  
 mutate(id = 33 + row\_number()) %>%  
 ggplot(aes(start, id, colour = party)) +  
 geom\_point() +  
 geom\_segment(aes(xend = end, yend = id)) +  
 scale\_colour\_manual(values = c(Republican = "red", Democratic = "blue")) #manually selects colors if needed to be pre defined



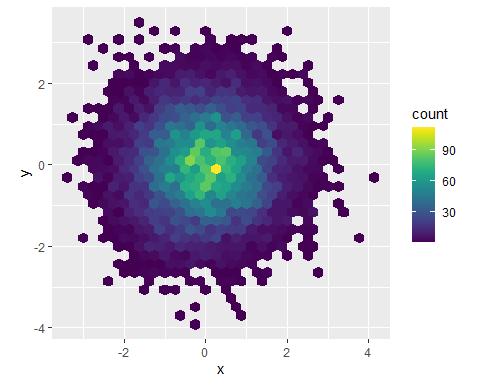
#install.packages('viridis') #Sweet packages  
#install.packages('hexbin') #Another one  
library(viridis)

## Loading required package: viridisLite

library(hexbin)  
  
df <- tibble( # note we're just making a fake dataset so we can plot it  
 x = rnorm(10000),  
 y = rnorm(10000)  
)  
ggplot(df, aes(x, y)) +  
 geom\_hex() + # a new geom!  
 coord\_fixed()



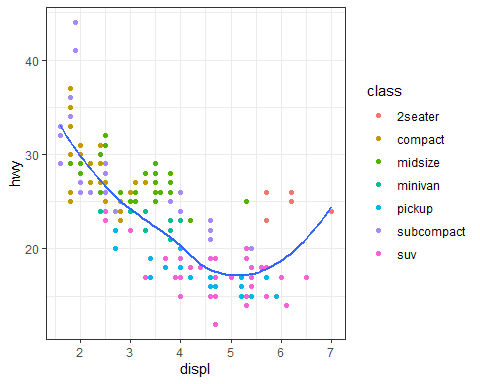
ggplot(df, aes(x, y)) +  
 geom\_hex() +  
 viridis::scale\_fill\_viridis() +  
 coord\_fixed()



## Themes

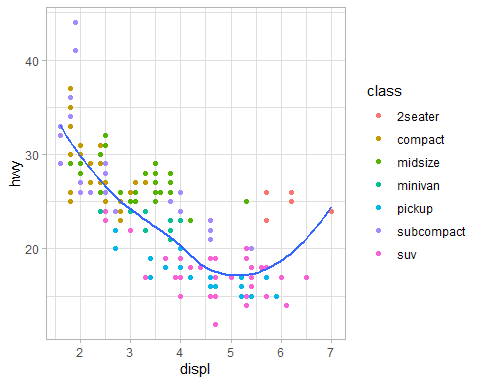
ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(color = class)) +  
 geom\_smooth(se = FALSE) +  
 theme\_bw() #black and white theme

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



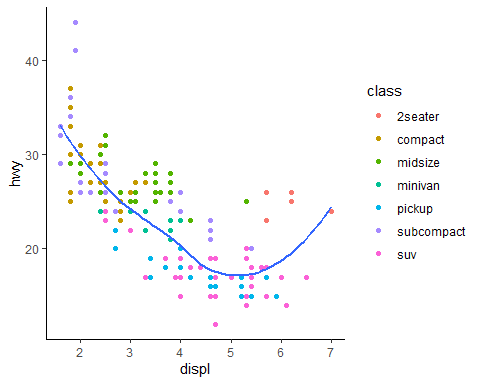
ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(color = class)) +  
 geom\_smooth(se = FALSE) +  
 theme\_light() #lighter graph borders

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



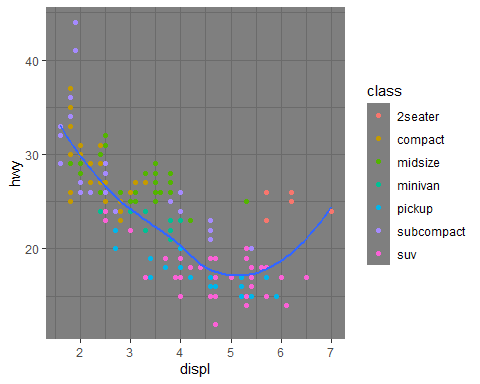
ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(color = class)) +  
 geom\_smooth(se = FALSE) +  
 theme\_classic() #no grid lines, only axis borders

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



ggplot(mpg, aes(displ, hwy)) +  
 geom\_point(aes(color = class)) +  
 geom\_smooth(se = FALSE) +  
 theme\_dark() #Darkened graph background, difficult to read nless lighter data points

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



theme (panel.border = element\_blank(),  
 panel.grid.minor.x = element\_blank(),  
 panel.grid.minor.y = element\_blank(),  
 legend.position="bottom",  
 legend.title=element\_blank(),  
 legend.text=element\_text(size=8),  
 panel.grid.major = element\_blank(),  
 legend.key = element\_blank(),  
 legend.background = element\_blank(),  
 axis.text.y=element\_text(colour="black"),  
 axis.text.x=element\_text(colour="black"),  
 text=element\_text(family="Arial")) #Can develop yuor own themes by setting every variable yourself

## List of 12  
## $ text :List of 11  
## ..$ family : chr "Arial"  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size : NULL  
## ..$ hjust : NULL  
## ..$ vjust : NULL  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin : NULL  
## ..$ debug : NULL  
## ..$ inherit.blank: logi FALSE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ axis.text.x :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : chr "black"  
## ..$ size : NULL  
## ..$ hjust : NULL  
## ..$ vjust : NULL  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin : NULL  
## ..$ debug : NULL  
## ..$ inherit.blank: logi FALSE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ axis.text.y :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : chr "black"  
## ..$ size : NULL  
## ..$ hjust : NULL  
## ..$ vjust : NULL  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin : NULL  
## ..$ debug : NULL  
## ..$ inherit.blank: logi FALSE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ legend.background : list()  
## ..- attr(\*, "class")= chr [1:2] "element\_blank" "element"  
## $ legend.key : list()  
## ..- attr(\*, "class")= chr [1:2] "element\_blank" "element"  
## $ legend.text :List of 11  
## ..$ family : NULL  
## ..$ face : NULL  
## ..$ colour : NULL  
## ..$ size : num 8  
## ..$ hjust : NULL  
## ..$ vjust : NULL  
## ..$ angle : NULL  
## ..$ lineheight : NULL  
## ..$ margin : NULL  
## ..$ debug : NULL  
## ..$ inherit.blank: logi FALSE  
## ..- attr(\*, "class")= chr [1:2] "element\_text" "element"  
## $ legend.title : list()  
## ..- attr(\*, "class")= chr [1:2] "element\_blank" "element"  
## $ legend.position : chr "bottom"  
## $ panel.border : list()  
## ..- attr(\*, "class")= chr [1:2] "element\_blank" "element"  
## $ panel.grid.major : list()  
## ..- attr(\*, "class")= chr [1:2] "element\_blank" "element"  
## $ panel.grid.minor.x: list()  
## ..- attr(\*, "class")= chr [1:2] "element\_blank" "element"  
## $ panel.grid.minor.y: list()  
## ..- attr(\*, "class")= chr [1:2] "element\_blank" "element"  
## - attr(\*, "class")= chr [1:2] "theme" "gg"  
## - attr(\*, "complete")= logi FALSE  
## - attr(\*, "validate")= logi TRUE

## Save and Export Your Figures

ggplot(mpg, aes(displ, hwy)) + geom\_point()



ggsave("my-plot.pdf")

## Saving 5 x 4 in image

#> Saving 7 x 4.32 in image to your working directory

##QFish Assignment

library(readxl)  
export <- read\_excel("C:/Users/jacoo/Downloads/export.xlsx")  
View(export) #Import Dataset  
QFish <- export

#Data is fairly clean, need to remove last year (2024) because it is incomplete.  
#QFish <- QFish %>%  
 #slice(-n())  
#view(QFish)  
#commented out as to not over-use the command and lose data

QFish$Tonnes <- as.numeric(QFish$Tonnes)  
QFish$Days <- as.numeric(QFish$Days)  
QFish$CalendarYear <- as.numeric(QFish$CalendarYear)

## Warning: NAs introduced by coercion

ggplot(data= QFish) +   
 geom\_point(aes(x = CalendarYear, y = Days, color = "Days"), alpha = 0.7) +  
 geom\_point(aes(x = CalendarYear, y = Tonnes, color = "Tonnes"), alpha = 0.7) +  
 geom\_smooth(aes(x = CalendarYear, y = Days, color = "Days"), method = 'loess', se= FALSE) +  
 geom\_smooth(aes(x = CalendarYear, y = Tonnes, color = "Tonnes"), method = 'loess', se= FALSE) +  
 scale\_color\_manual(values = c("black", "red")) +  
 labs(y = "Fishing Effort (Days) or Yield (Tonnes)", x = "Year", color = "Measure") +  
 theme\_bw() +  
 theme(panel.grid = element\_blank()) #Use minimal theme for clarity

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

## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_smooth()`).

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

## Warning: Removed 2 rows containing non-finite outside the scale range  
## (`stat\_smooth()`).

## Warning: Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).  
## Removed 2 rows containing missing values or values outside the scale range  
## (`geom\_point()`).

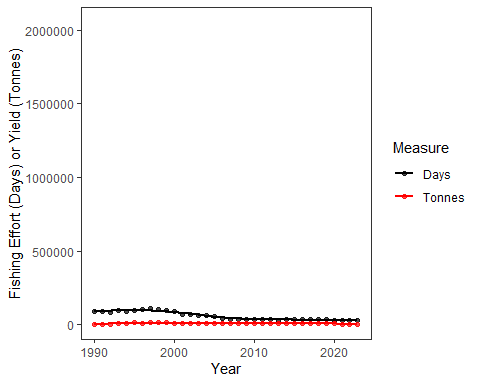
 Caption- Otter trawl efforts have decreased quite quickly since the 1990s while yields have been decreasing very slightly ever since. This could be due to an increase in technology efficiency and overfishing depleted populations. (Data from QFish.fisheries.qld.gov)

table1 #Cases and population are displayed by country and by year in the same row.

## # A tibble: 6 × 4  
## country year cases population  
## <chr> <dbl> <dbl> <dbl>  
## 1 Afghanistan 1999 745 19987071  
## 2 Afghanistan 2000 2666 20595360  
## 3 Brazil 1999 37737 172006362  
## 4 Brazil 2000 80488 174504898  
## 5 China 1999 212258 1272915272  
## 6 China 2000 213766 1280428583

table2 #Cases and population are shown in different rows, differentiated by column "type"

## # A tibble: 12 × 4  
## country year type count  
## <chr> <dbl> <chr> <dbl>  
## 1 Afghanistan 1999 cases 745  
## 2 Afghanistan 1999 population 19987071  
## 3 Afghanistan 2000 cases 2666  
## 4 Afghanistan 2000 population 20595360  
## 5 Brazil 1999 cases 37737  
## 6 Brazil 1999 population 172006362  
## 7 Brazil 2000 cases 80488  
## 8 Brazil 2000 population 174504898  
## 9 China 1999 cases 212258  
## 10 China 1999 population 1272915272  
## 11 China 2000 cases 213766  
## 12 China 2000 population 1280428583

table3 #Cases and population are condensed into a single column labelled "rate", showing the rate of cases in the population.

## # A tibble: 6 × 3  
## country year rate   
## <chr> <dbl> <chr>   
## 1 Afghanistan 1999 745/19987071   
## 2 Afghanistan 2000 2666/20595360   
## 3 Brazil 1999 37737/172006362   
## 4 Brazil 2000 80488/174504898   
## 5 China 1999 212258/1272915272  
## 6 China 2000 213766/1280428583

#From table1, select cases and population by year and country using a pipe function and pivot\_longer. Then, use the mutate function to create and store the correct values back in their respective place.

stocks <- tibble(  
 year = c(2015, 2015, 2016, 2016),  
 half = c( 1, 2, 1, 2),  
 return = c(1.88, 0.59, 0.92, 0.17)  
)  
stocks %>%   
 pivot\_wider(names\_from = year, values\_from = return) %>%   
 pivot\_longer(`2015`:`2016`, names\_to = "year", values\_to = "return")

## # A tibble: 4 × 3  
## half year return  
## <dbl> <chr> <dbl>  
## 1 1 2015 1.88  
## 2 1 2016 0.92  
## 3 2 2015 0.59  
## 4 2 2016 0.17

#They are symmetrical in the actual hard data, just not the exact order. It seems to be sorting the data based on the column "half".   
  
#The code fails because there is no year 1999 or 2000 in the existing data.  
  
#The table should be made longer. the variables are 'Gender', 'Pregnant', and 'count'.

read\_csv("a,b\n1,2,3\n4,5,6") #Each row has 3 values, but there's only 2 columns.

## Warning: One or more parsing issues, call `problems()` on your data frame for details,  
## e.g.:  
## dat <- vroom(...)  
## problems(dat)

## Rows: 2 Columns: 2  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## dbl (1): a  
## num (1): b  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

## # A tibble: 2 × 2  
## a b  
## <dbl> <dbl>  
## 1 1 23  
## 2 4 56

read\_csv("a,b,c\n1,2\n1,2,3,4") #There are 3 columns, but rows have 2 and 4 values. Each must have 3 values to match the columns.

## Warning: One or more parsing issues, call `problems()` on your data frame for details,  
## e.g.:  
## dat <- vroom(...)  
## problems(dat)

## Rows: 2 Columns: 3  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## dbl (2): a, b  
## num (1): c  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

## # A tibble: 2 × 3  
## a b c  
## <dbl> <dbl> <dbl>  
## 1 1 2 NA  
## 2 1 2 34

read\_csv("a,b\n\"1") #Quotes issue messes with the way the computer reads the code.

## Rows: 0 Columns: 2  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (2): a, b  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

## # A tibble: 0 × 2  
## # ℹ 2 variables: a <chr>, b <chr>

read\_csv("a,b\n1,2\na,b") #Has issue with character vs numeric type in certain row.

## Rows: 2 Columns: 2  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (2): a, b  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

## # A tibble: 2 × 2  
## a b   
## <chr> <chr>  
## 1 1 2   
## 2 a b

read\_csv("a;b\n1;3") #Uses ; when read.csv expects commas.

## Rows: 1 Columns: 1  
## ── Column specification ────────────────────────────────────────────────────────  
## Delimiter: ","  
## chr (1): a;b  
##   
## ℹ Use `spec()` to retrieve the full column specification for this data.  
## ℹ Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

## # A tibble: 1 × 1  
## `a;b`  
## <chr>  
## 1 1;3