# Data Viz in R

Joshua Allen

2021-03-25

## Contents



## Introduction

R comes loaded with some stuff but you are going to load in "commands" to do things. I am what is known as a `tidyverse` person. The `tidyverse` is **massive** but one of the pros is that all the packages play well together and have similar logics! The `tidyverse` webiste is here but the user base is huge and the developers behind it are very helpful. For a great intro you can work through this **FREE** book.

Working in projects is the best practice! To open a new project in Rstudio click the second icon on the left. You can create a new directory or go to an existing folder. This is a fairly intuitive process. For now you can ignore the option at the very bottom, but it is useful later on to keep track of changes you made.

I wrote this in what is known as `Rmarkdown` which lets you write words and include code. However, this is a bit on the technical slide to learn so I would encourage you to start with `scripts` you can open a new source file in the top right hand corner. `#` is how you talk to yourself in `R` so if you copy and paste any code delete `#` for it to run.

#Getting Started To get the packages we want you first need to install them and then you need to load them! Like this:

```
#install.packages("tidyverse") installs stuff. The package is already on my computer so
# R got mad at me
library("tidyverse")
```

Now that we have loaded the packages into R we should load our data in! To do this we are going to use our first `tidyverse` function `read_csv` and assign it to an object. Objects come in all shapes and sizes but for the purpose of this example think of it as a dataset.

```
penguins <- read_csv("penguins.csv") %>% # don't worry about this for now
  na.omit() #tall I am doing is omitting missing values
```

Notice how in the top right pane it was initially empty but now it has our penguins dataset in what an object called penguins. You can press the play button or simply type

```
View(penguins)
```

## Fun Stuff

`ggplot` is the main thing we are going to use for this example. There are a ton of great packages that enhance its graphing capabilities and range of stuff stuff you can use. As and example we are going to load `patchwork`, `Manu`, `gameofthrones`, and some other friends to show you what it can do. `patchwork` makes it easier to combine plots, `Manu` is a color palette based on New Zealand birds, and `gameofthrones` is a color palette based on the books/show Game of Thrones.

```
# install.packages(c("Manu","gameofthrones","patchwork"))

library(Manu)
library(gameofthrones)
library(patchwork)
```

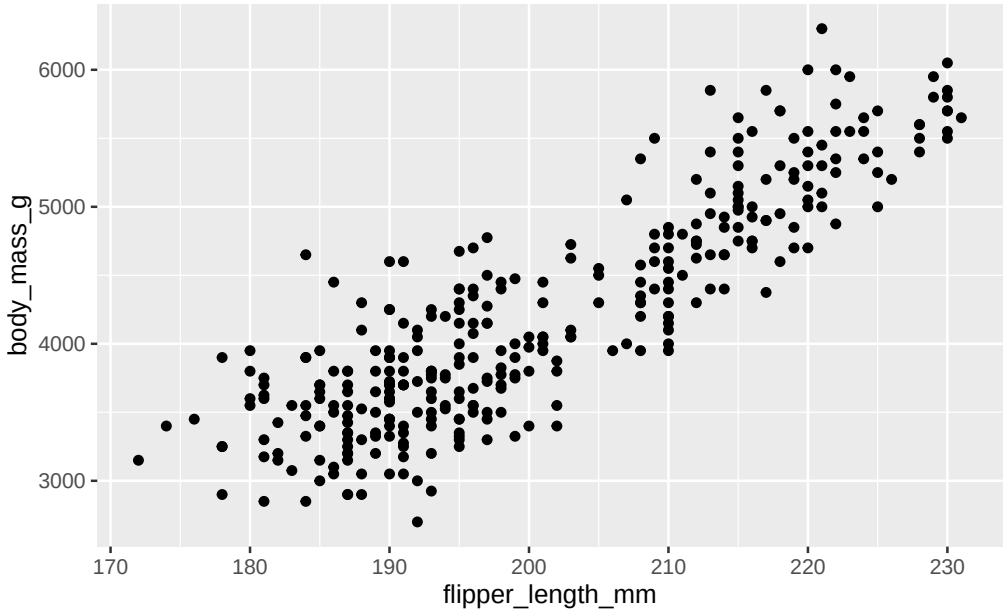
## Basics

The function we are going to be working with is `ggplot` there are a lot of components to it and it is a little overwhelming at first. But this is the basic logic of

- `data=` tells `R` what data we are using for this plot
- `aes` tells `R` what columns in the dataset we want graphed
- `geom_` tells `R` what kind of plot we want.

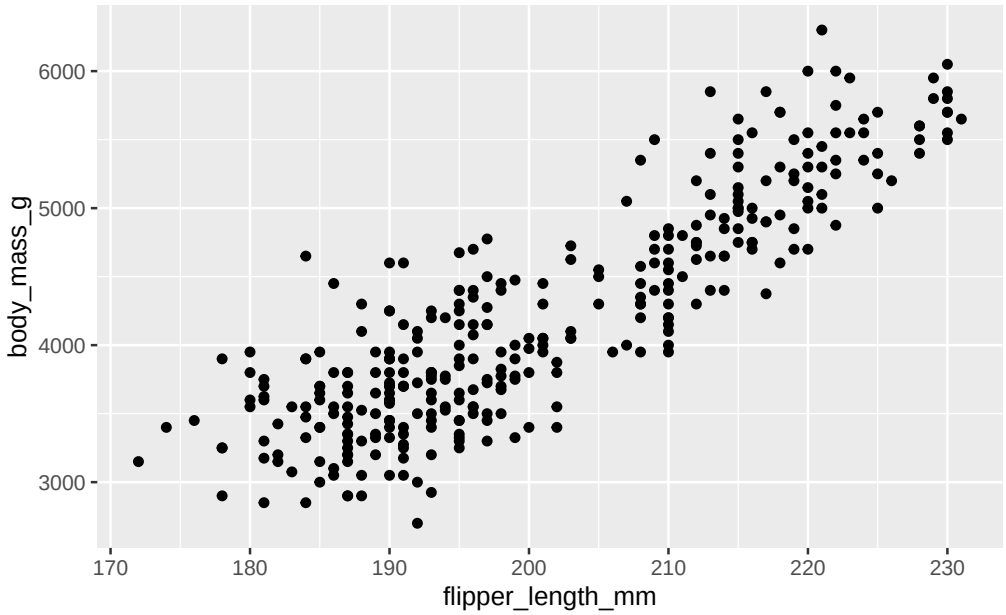
Here is a basic scatter plot in `ggplot`

```
ggplot(data = penguins) +
  geom_point(aes(x = flipper_length_mm, y = body_mass_g))
```



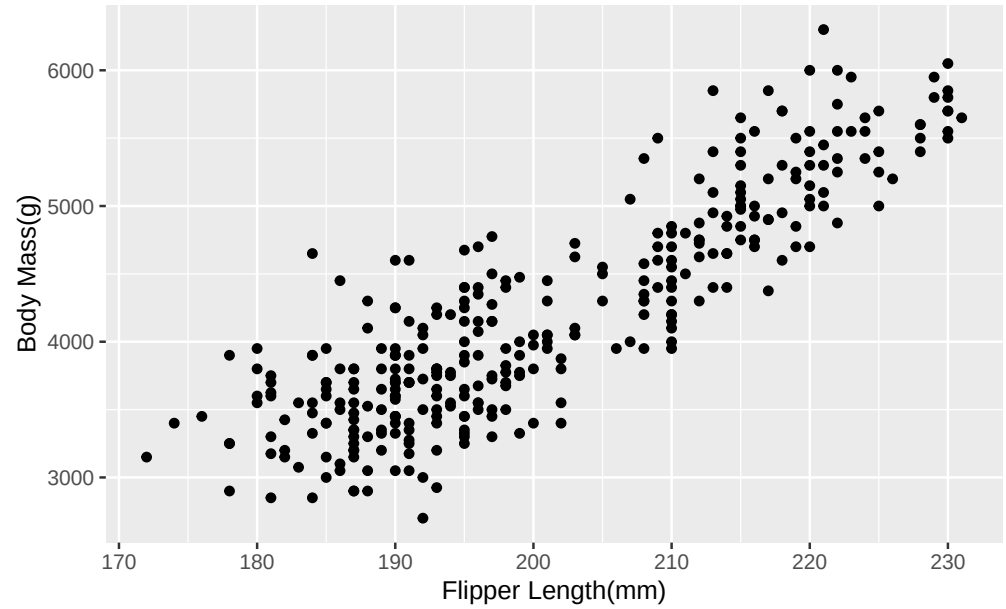

This is usually how I graph stuff. `R` for Datascience covers this better than I could.

```
ggplot(data = penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point()
```



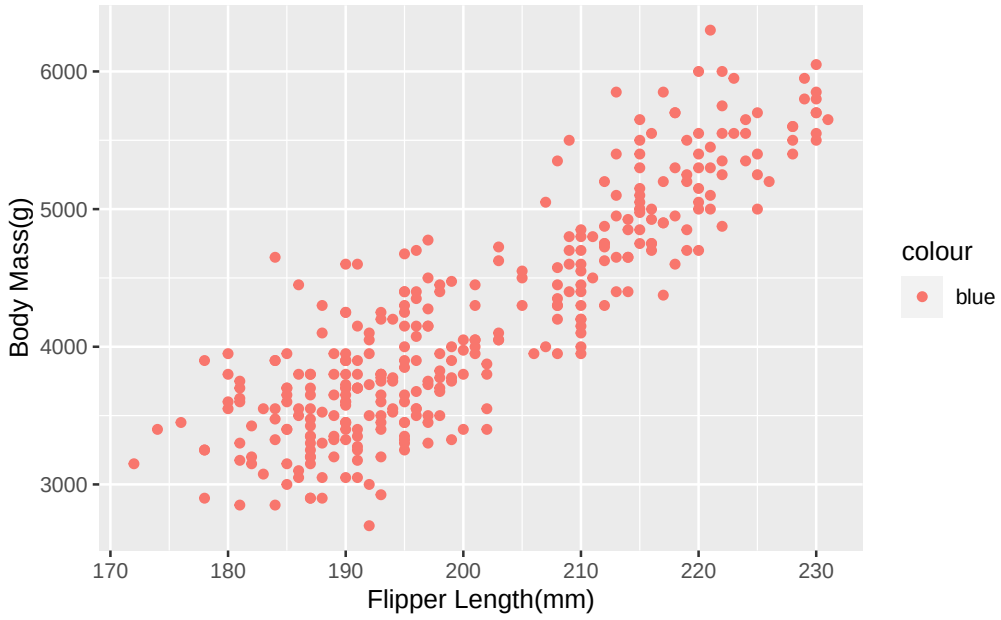

Like `Stata` you need to add stuff to customize plot when you add stuff instead of `,options` you use `+` and then add stuff. Here is an example of stuff you can add, but this is only the tip of the stuff iceberg.

```
ggplot(data = penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point() +
  labs(x = "Flipper Length(mm)", y = "Body Mass(g)")
```



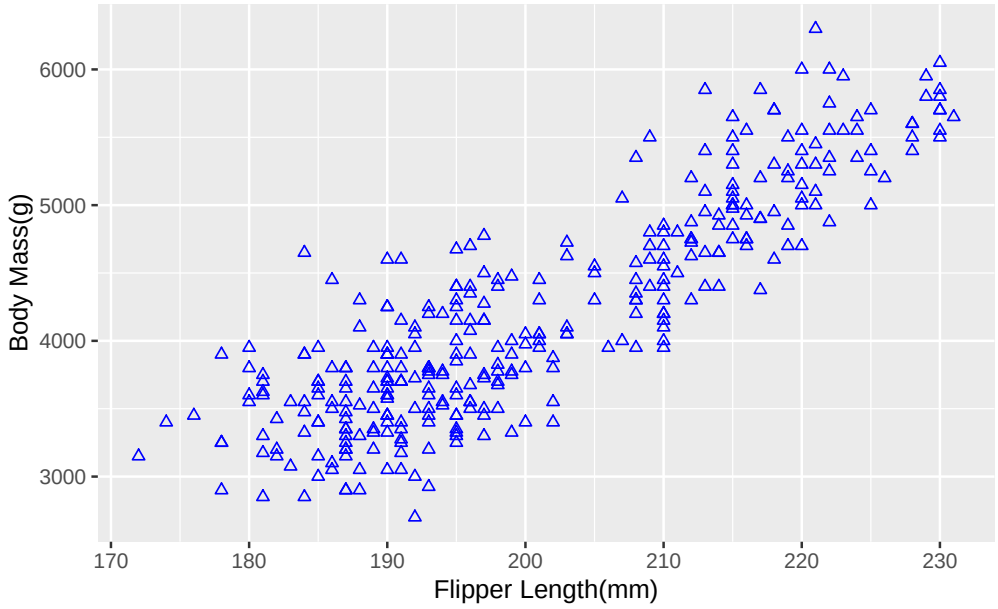

So far we have only done stuff working within `aes`. Here is one of the most common mistakes when working in `ggplot` I still do this!.

```
ggplot(data = penguins, aes(x = flipper_length_mm, y = body_mass_g,
                            color = "blue")) +
  geom_point() +
  labs(x = "Flipper Length(mm)", y = "Body Mass(g)")
```




I wanted to change the color of the dots. However I did this in the `aes()`. If you look back then you will remember that `aes` will look to put stuff in your plots based off of **columns** in your dataset. So to change the look of your plot you need to do it **outside** of `aes()`. So lets correct that and then change the shape of the points.

```
ggplot(data = penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(color = "blue", shape = 2) +
  labs(x = "Flipper Length(mm)", y = "Body Mass(g)")
```

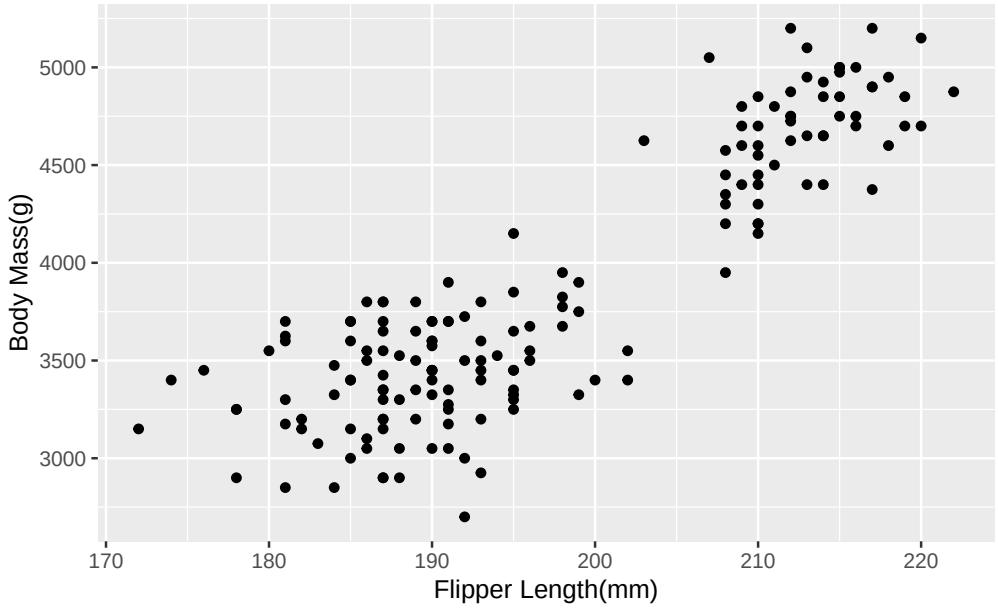



## Making subset plots

There are a ton of ways to do this but they get a little more complicated than in `Stata`. If you look at the code for that example all you had to do was `if sex == "male` in R it is a little more work and there are more ways to get at the right answer.

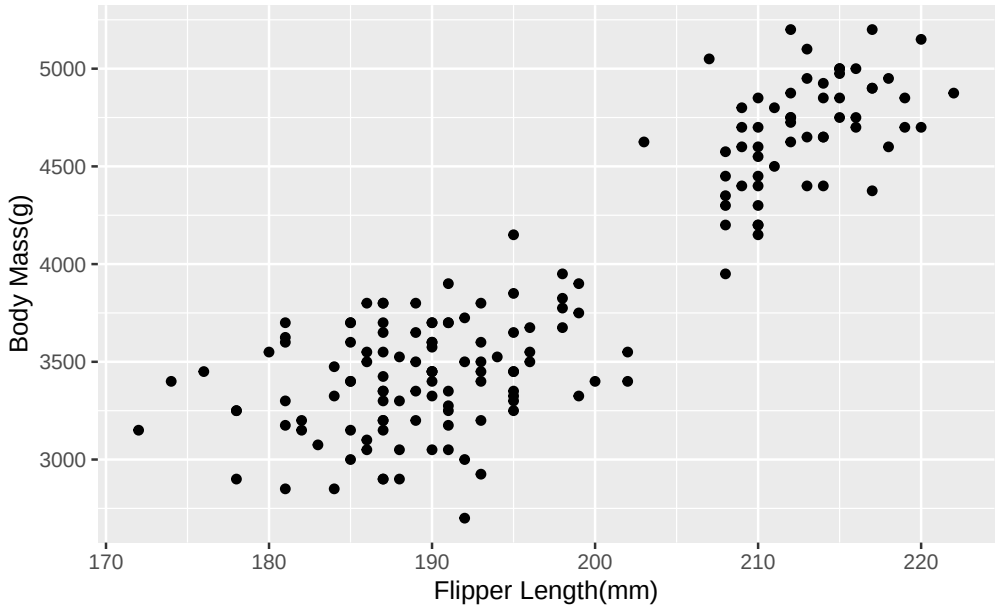
### Subsetting it by females

```
ggplot(penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(data = filter(penguins, sex == "female")) +
  labs(x = "Flipper Length(mm)", y = "Body Mass(g)")
```




This also works.

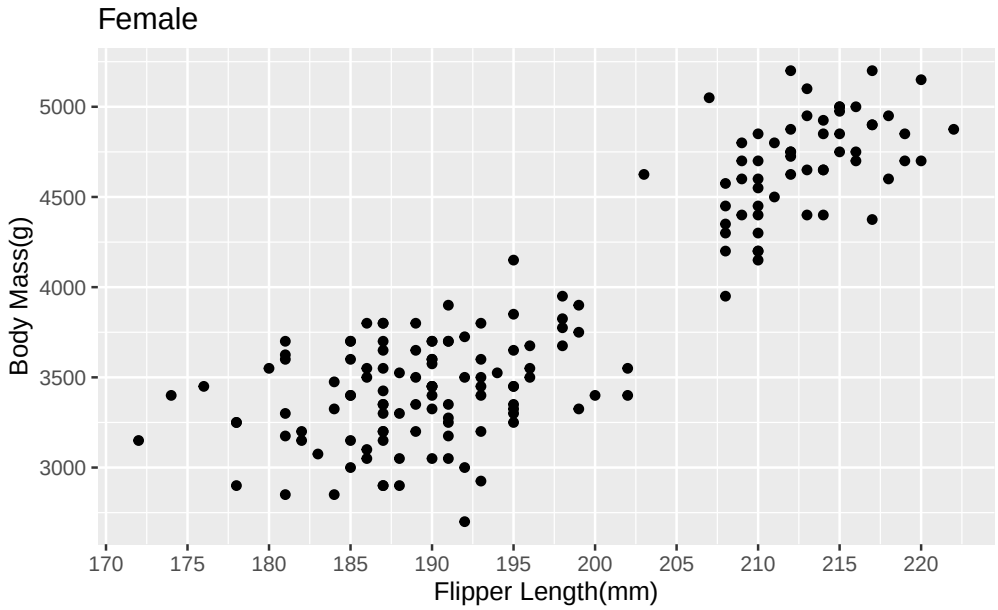
```
penguins %>%
  filter(sex == "female") %>%
  ggplot(aes (x = flipper_length_mm, y = body_mass_g)) +
  geom_point() +
  labs(x = "Flipper Length(mm)", y = "Body Mass(g)")
```




The difference is a matter of taste if we are being honest. I learned it the first way but tend to do the second way more often. So lets recreate the subset plot in the `Stata` example that way. To combine them we are going to assign it to an object than use `patchwork` to combine them.

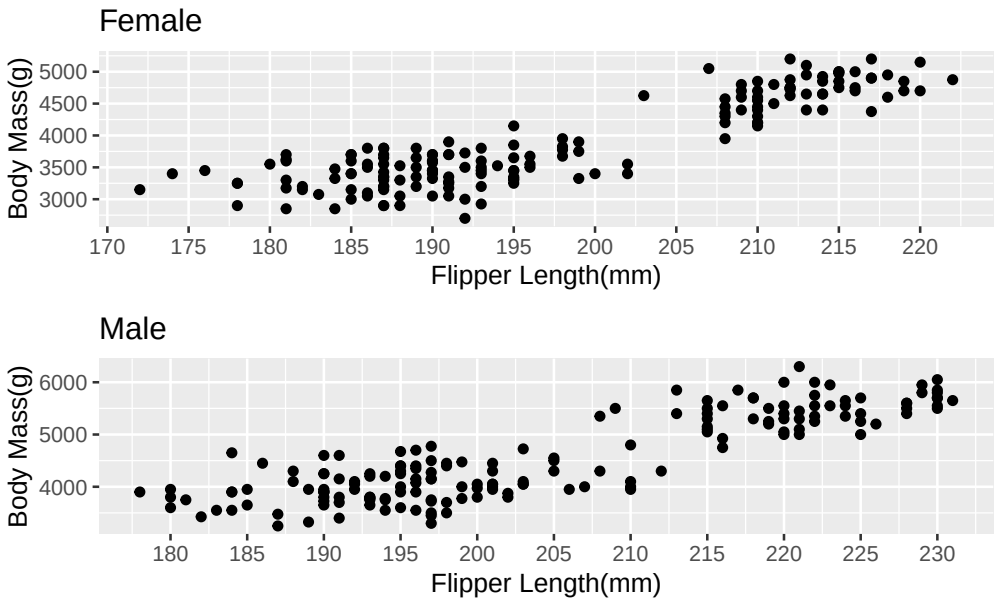
```
library(scales)
fem_plot<- penguins %>%
  filter(sex == "female") %>%
  ggplot(aes (x = flipper_length_mm, y = body_mass_g)) +
  geom_point() +
  labs(x = "Flipper Length(mm)", y = "Body Mass(g)", title = "Female") +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10))


fem_plot
```




```
male_plot<- penguins %>%
  filter(sex == "male") %>%
  ggplot(aes (x = flipper_length_mm, y = body_mass_g)) +
  geom_point() +
  labs(x = "Flipper Length(mm)", y = "Body Mass(g)", title = "Male") +
    scale_x_continuous(breaks = scales::pretty_breaks(n = 10))


fem_plot/male_plot
```




## Themes in ggplot

Themes in `ggplot` get pretty intense very quickly. Most people write their own themes but this gets scary when you are first learning ggplot we will use `ggthemes` first but as an example we will look at one I wrote for a class assignment.

```
covidmap<-ggplot(data = covid.map,mapping = aes(x=lon,y=lat,group=id,
fill=`14 day case rate`),alpha=1)+
  geom_polygon()+
  labs(title = "Georgia Covid Cases by County",color="white")+
  scale_fill_gradientn(
    colors =c("white","orange","red","brown"),
    values=c(0, 0.4, 0.8, 1),
    breaks=c(0, 200, 400, 600, 800),
    limits=c(0,1000),
    oob=scales::squish,
      guide=guide_colorbar(
        title.posititon="top",
        guide_legend(title = "Reported Cases per 100,000 in the last 14 days"),
        title.theme = element_text(
          size=12,
          face="italic",
          color="white"
        )))+
  theme_void()+
  theme(legend.position = "top", legend.box = "horizontal",
  legend.text = element_text(color="white"))+
  theme(plot.background=element_rect("darkblue"))+
  theme(plot.title = element_text(color="white"))
```

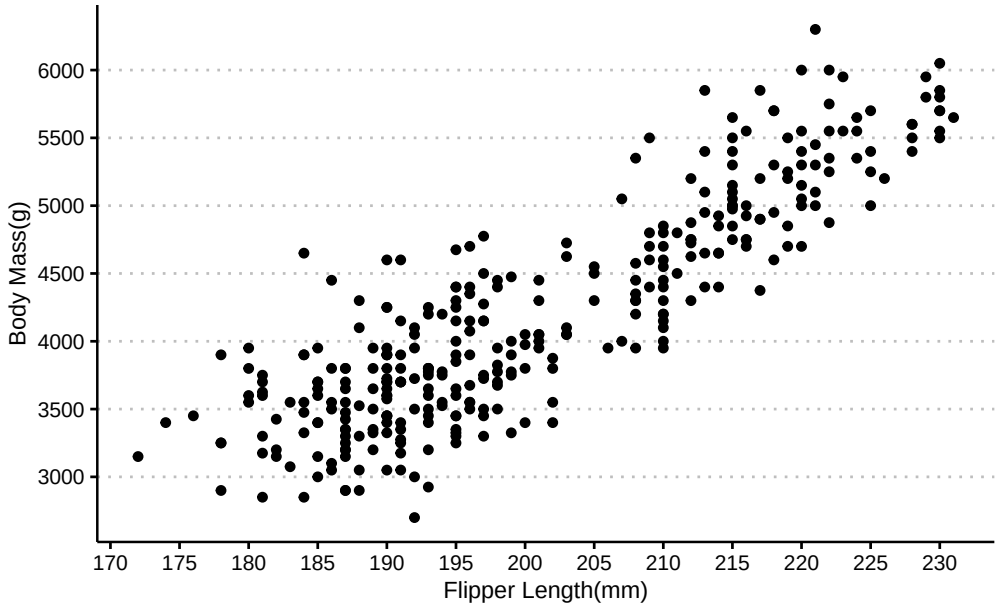
That is a bit intense but there are lots of pre-written schemes that are good. Here is a simple one that I use.

```
library(ggthemes)

theme_set(
  theme_clean() +
    theme(plot.background = element_blank(),
```

```
          legend.background = element_rect(color = "white"))
  )

ggplot(data = penguins, aes(x = flipper_length_mm, y = body_mass_g)) +
  geom_point(position = position_jitter(width = 0, height = 0.25, seed = 1234)) +
  labs(x = "Flipper Length(mm)", y = "Body Mass(g)") +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 10))
```



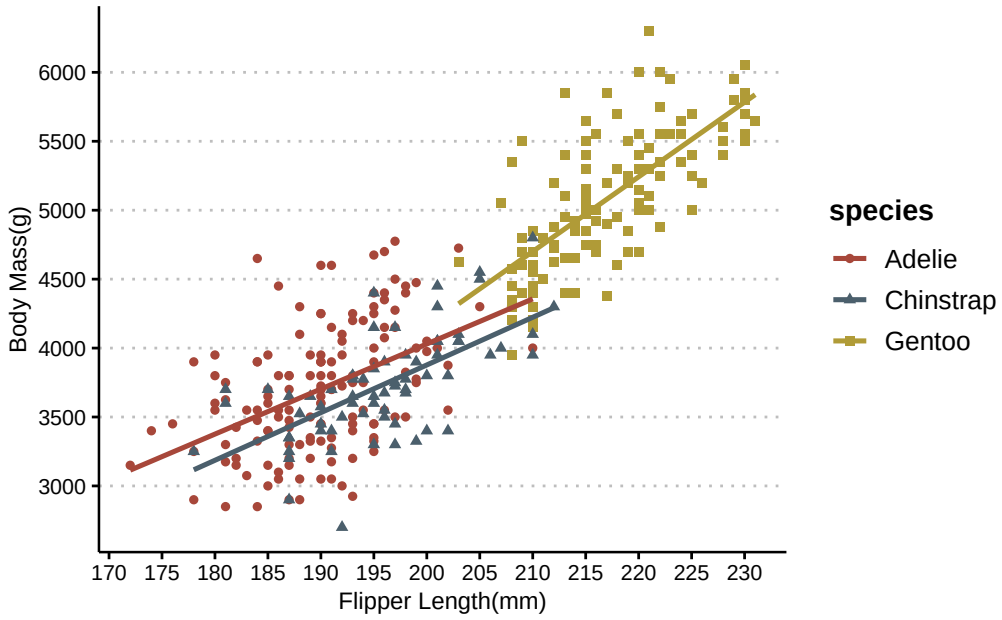

## Combining Plots and color schemes

We have done some super basic stuff but lets start making the plots convey more information. Then lets make them a little more fun.

```
color1<- get_pal("Kaka")

ggplot(data = penguins, aes(x = flipper_length_mm, y = body_mass_g , color = species,
                            shape = species)) +
  geom_point(position = position_jitter(width = 0, height = 0.25, seed = 1234)) +
  geom_smooth(method = "lm", se = FALSE) +
  labs(x = "Flipper Length(mm)", y = "Body Mass(g)", fill = "Species of Penguins") +
  scale_color_manual(values = color1) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 10))
```

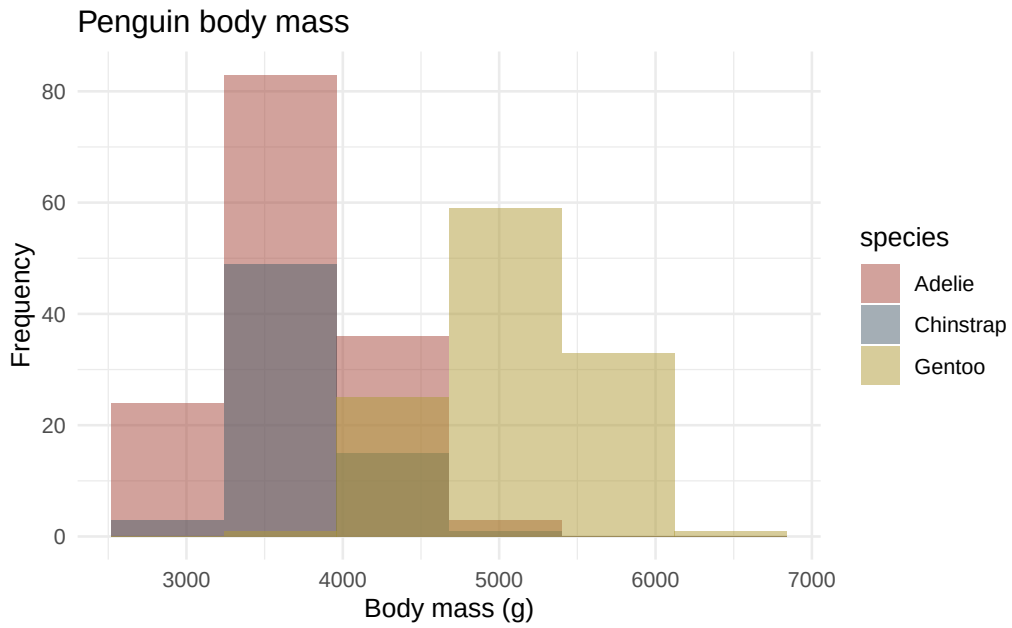
```
## `geom_smooth()` using formula 'y ~ x'
```




## Gentle introduction to amounts and Proportions

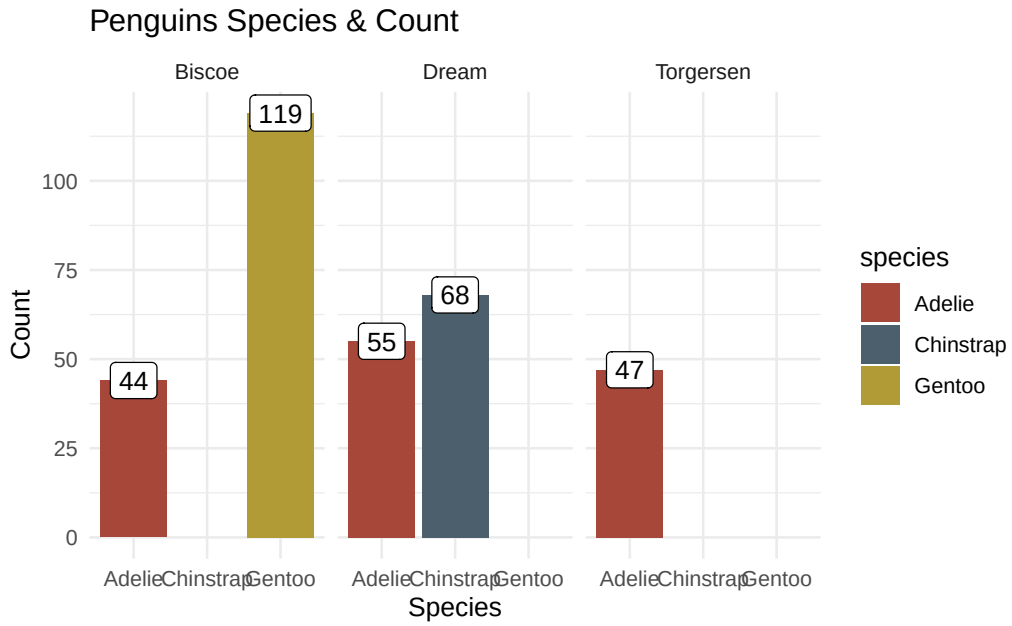
Here we are going to do an example from the palmer penguins webpage

```
ggplot(data = penguins, aes(x = body_mass_g)) +
  geom_histogram(aes(fill = species),
                 alpha = 0.5,
                 bins = 6,
                 position = "identity") +
  scale_fill_manual(values = color1) +
  theme_minimal() +
  labs(x = "Body mass (g)",
       y = "Frequency",
       title = "Penguin body mass")
```




What about a count of where they live?

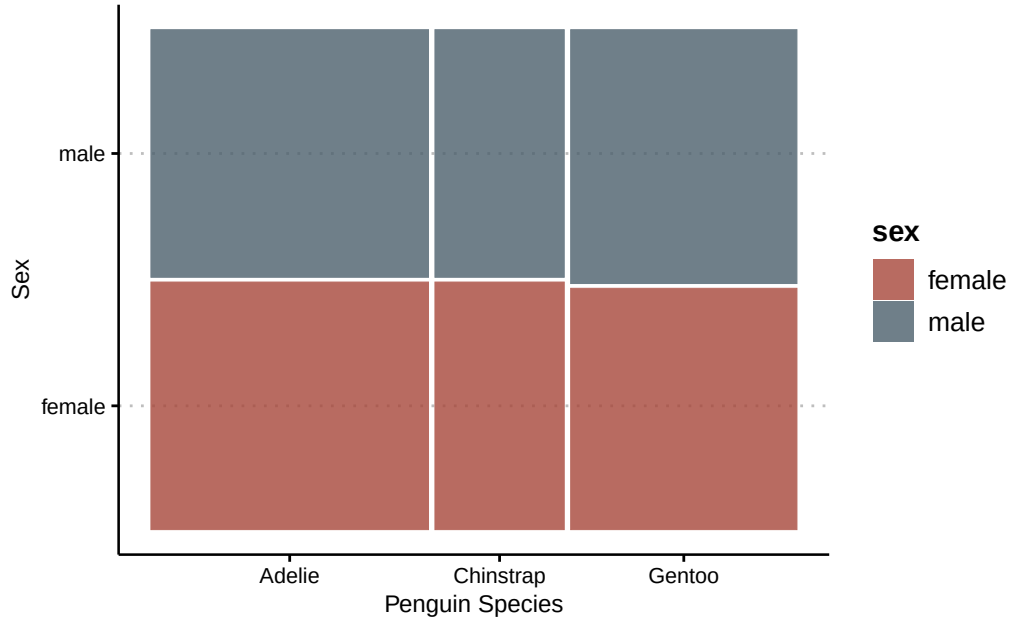
```
penguins %>%
count(species, island) %>%
ggplot() + geom_col(aes(x = species, y = n, fill = species)) +
geom_label(aes(x = species, y = n, label = n)) +
scale_fill_manual(values = color1 ) +
facet_wrap(vars(island)) +
theme_minimal() +
labs(title = 'Penguins Species & Count', y= "Count", x= "Species")
```




What about that mosaic plot you showed us?

```
library(ggmosaic)

ggplot(penguins) +
  geom_mosaic(aes(x = product(species), fill = sex)) +
  labs(y = "Sex", x = "Penguin Species") +
  scale_fill_manual(values = color1)
```



## What else can I do in ggplot?

The simple answer tons. Here is an example of something you can do with maps. We can recreate Minard's map. But you recreate the map of Middle Earth instead.

```
library(sf)
coastline <- read_sf("ME-GIS/Coastline2.shp")
contours <- read_sf("ME-GIS/Contours_18.shp")
rivers <- read_sf("ME-GIS/Rivers.shp")
lakes <- read_sf("ME-GIS/Lakes.shp")
forests <- read_sf("ME-GIS/Forests.shp")
mountains <- read_sf("ME-GIS/Mountains_Anno.shp")
placenames <- read_sf("ME-GIS/Combined_Placenames.shp")

places <- placenames %>%
  filter(NAME %in% c("Hobbiton", "Rivendell", "Edoras", "Minas Tirith"))

 ggplot() +
  geom_sf(data = contours, size = 0.15, color = "grey90") +
  geom_sf(data = coastline, size = 0.25, color = "grey50") +
  geom_sf(data = rivers, size = 0.2, color = "#0776e0", alpha = 0.5) +
  geom_sf(data = lakes, size = 0.2, color = "#0776e0", fill = "#0776e0") +
  geom_sf(data = forests, size = 0, fill = "#035711", alpha = 0.5) +
  geom_sf(data = mountains, size = 0.25) +
  geom_sf(data = places) +
  geom_sf_label(data = places, aes(label = NAME), nudge_y = 80000) +
  theme_void() +
  theme(plot.background = element_rect(fill = "#fffce3"))
```