ggplot2

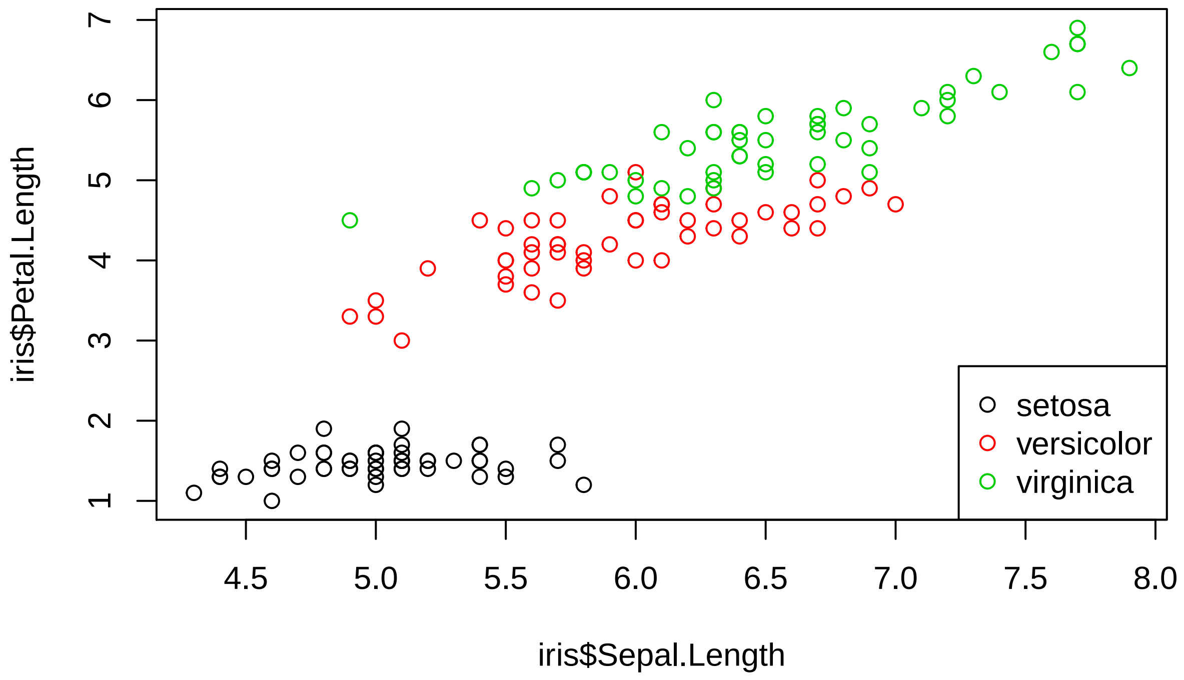
Today we'll compare R basic plotting to ggplot2 using the preloaded dataset *iris*. First let's look at *iris*:

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Say you wanted to plot sepal length versus petal length for each species. In base R graphics you would do something like:

```
plot(iris$Sepal.Length,iris$Petal.Length,col=iris$Species)
legend("bottomright",legend = unique(iris$Species),
       col=1:3,pch=1)
```



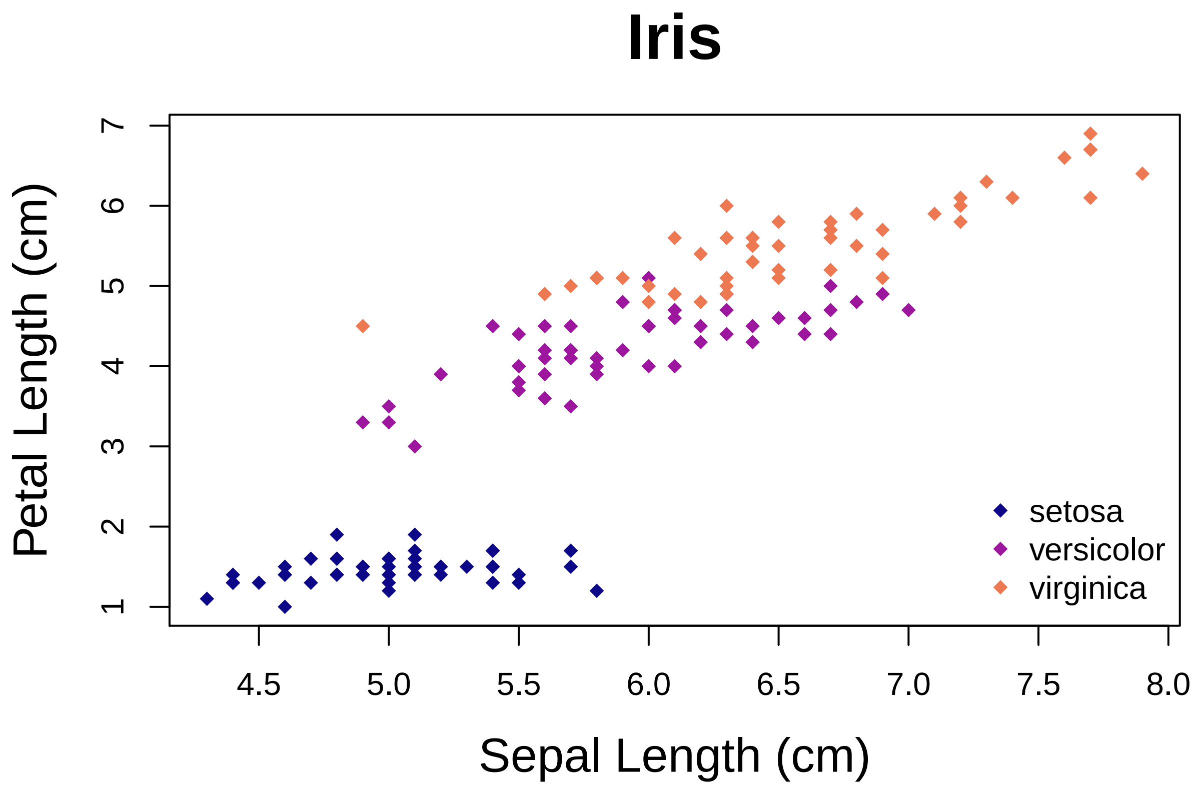
To add the legend, you'd need first to visually inspect the plot and find an area that has no data points and fits the legend. Remember that if you change the font size or save this image as a pdf, this position may not work as the aspect ratio of the plot may not work.

Now I'll use all of my tricks to make this pretty for a presentation.

```
require(viridis)  #this is a package that contains
```

```
## Loading required package: viridis
## Loading required package: viridisLite
```

```
                    #pretty palettes
par(cex.lab=1.5,cex.main=2)  #changes overall parameters
palette(viridis_pal(option = "C")(4))  #changes the pallete
plot(iris$Sepal.Length,iris$Petal.Length,col=iris$Species,
     xlab="Sepal Length (cm)", ylab="Petal Length (cm)",
     main="Iris", pch=18)
legend("bottomright",legend = unique(iris$Species),
       col=1:3,pch=18,bty='n')
```



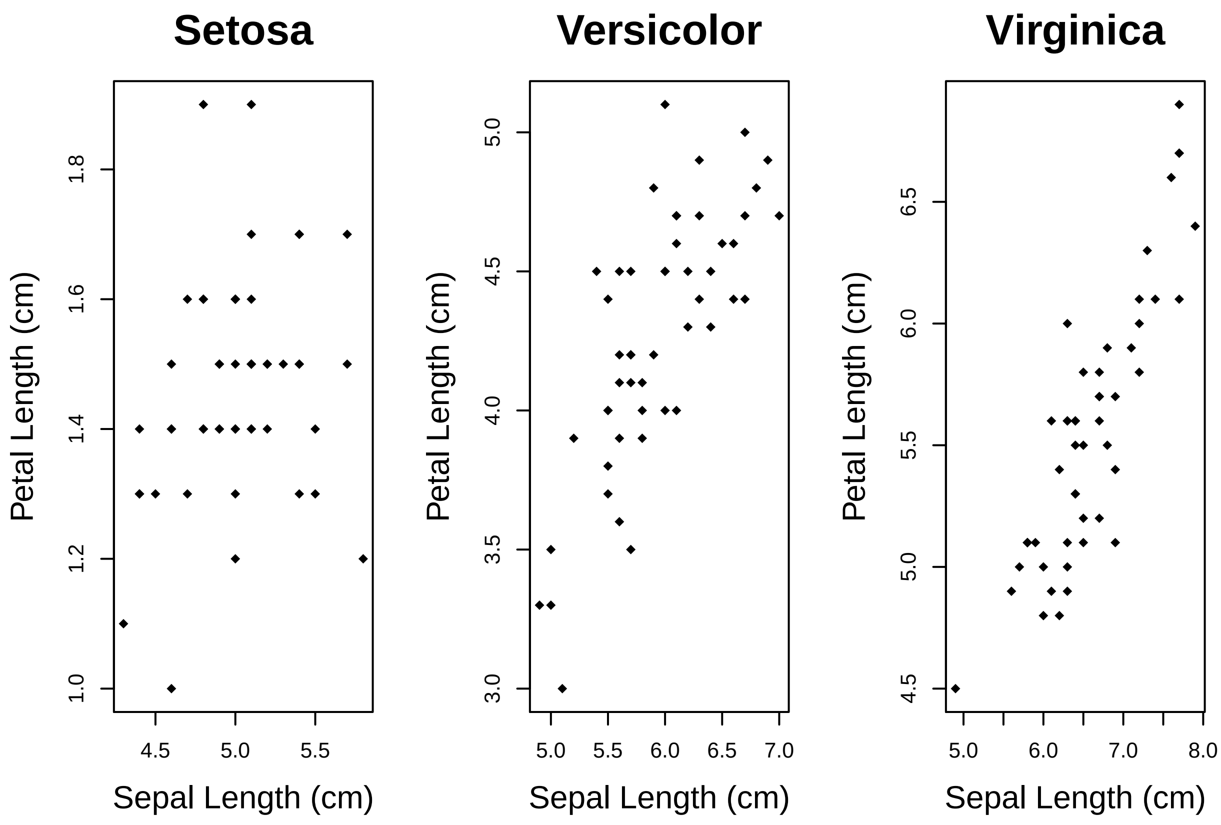
The legend still looks a bit uncomfortable, but it is *very* hard to put the legend outside of the plotting area using R basic graphics.

Now let's say I want separate plots for each species. This time you don't need the color legend but everything else needs to be repeated 3 times, each time subsetting the dataset to one species only.

```
par(cex.lab=1.5,cex.main=2)  #changes overall parameters
setosa=subset(iris, iris$Species=="setosa")
versicolor=subset(iris, iris$Species=="versicolor")
virginica=subset(iris, iris$Species=="virginica")

par(mfrow=c(1,3))

plot(setosa$Sepal.Length,setosa$Petal.Length,
     xlab="Sepal Length (cm)", ylab="Petal Length (cm)",
     main="Setosa", pch=18)
plot(versicolor$Sepal.Length,versicolor$Petal.Length,
     xlab="Sepal Length (cm)", ylab="Petal Length (cm)",
     main="Versicolor", pch=18)
plot(virginica$Sepal.Length,virginica$Petal.Length,
     xlab="Sepal Length (cm)", ylab="Petal Length (cm)",
     main="Virginica", pch=18)
```



Notice how many small changes need to happen: I need to subset correctly, I need to know the species names and I have to manually change each title. There are many opportunities to make a small mistake that would go unnoticed, such as plotting virginica against setosa or mislabeling the titles. Also, notice that the axes lengths now don't match which makes comparison difficult. This is easy to fix however. And the axis labels are needlessly repeated and the x axis label is cropped. This is very hard to fix without simply making the axis label size smaller and harder to read.

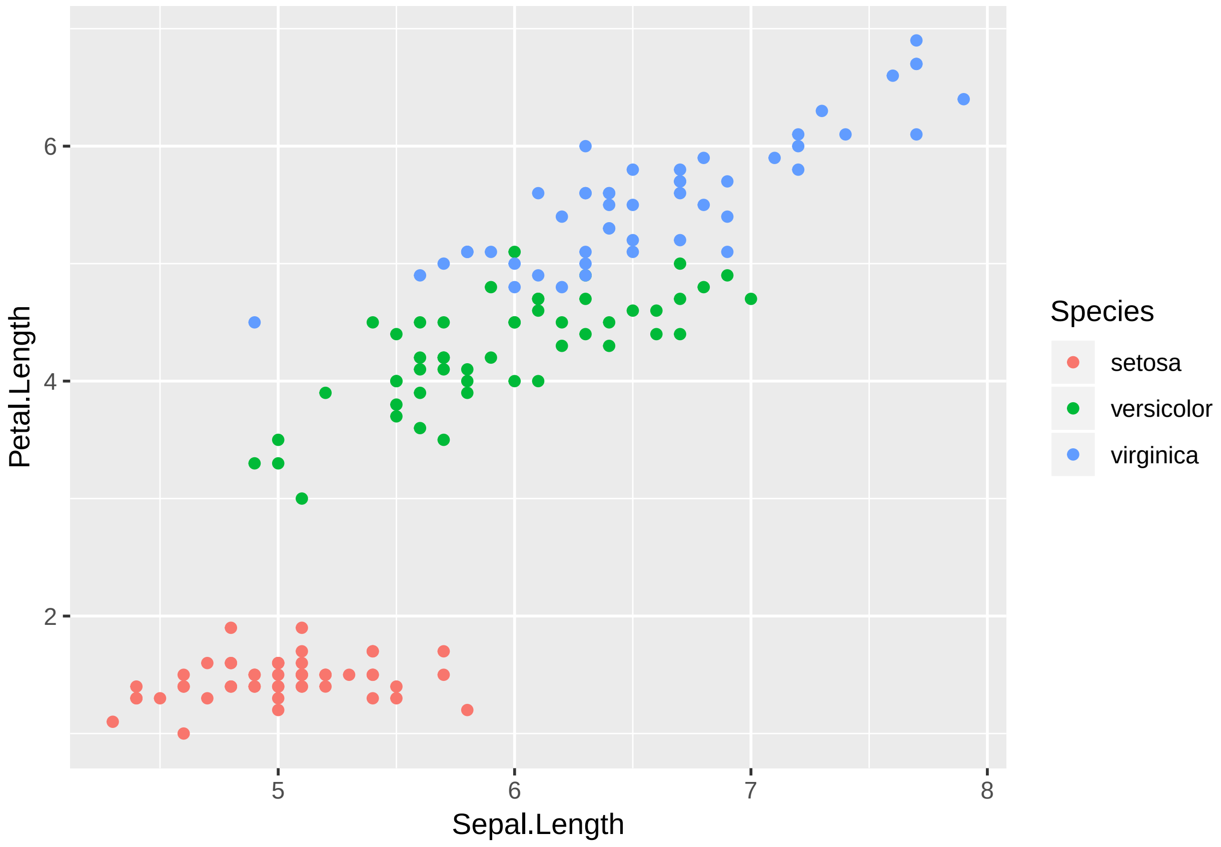
Now let's try with *ggplot2*. This is the basic:

```
require(ggplot2);require(dplyr)
```

```
FALSE Warning: package 'ggplot2' was built under R version 3.5.2

FALSE Warning: package 'dplyr' was built under R version 3.5.2
```

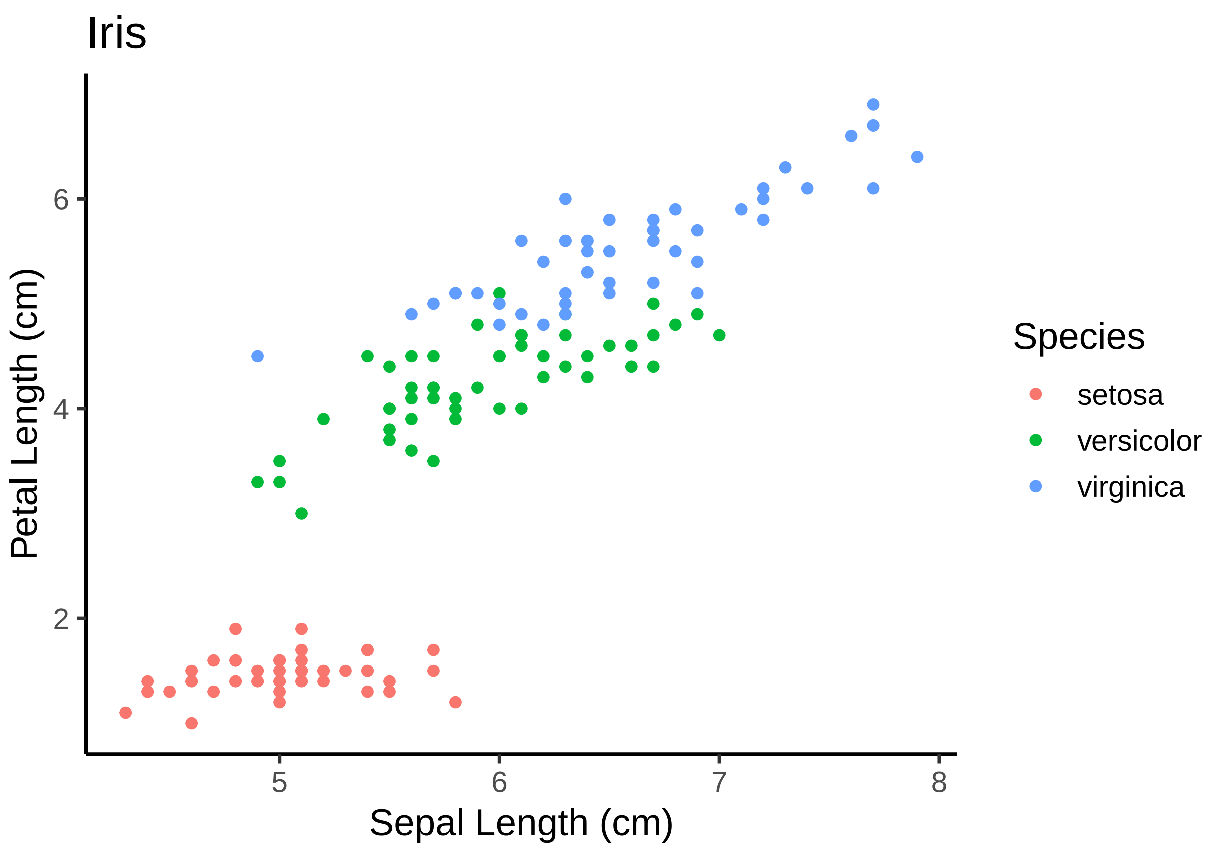
```
ggplot(data=iris)+
    geom_point(aes(x=Sepal.Length,y=Petal.Length,
                   color=Species))
```



The legend is automatic. The default colors are pretty good. You don't need a ton of $ operators: ggplot knows that since data=iris, what you are plotting is iris columns.

Now I'll do my tricks to make it better for a presentation.

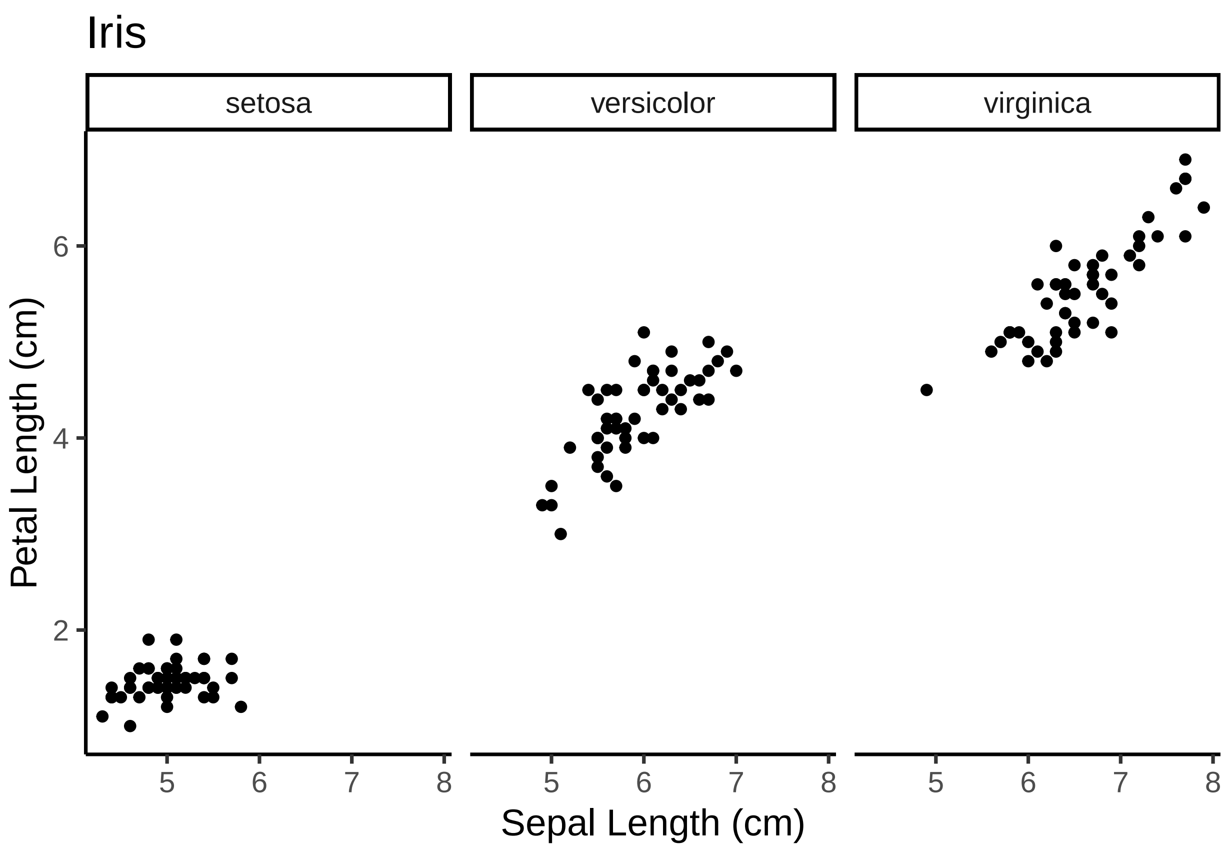
```
ggplot(data=iris)+
    theme_classic(base_size=14)+
    ggtitle("Iris")+
    xlab("Sepal Length (cm)")+
    ylab("Petal Length (cm)")+
    geom_point(aes(x=Sepal.Length,y=Petal.Length,
                   color=Species))
```



theme_classic(base_size=14) removes the grey background with white gridlines, which I honestly don't like very much. The argument also sets writing to be 14pt, which is the font size convention that we are all used to, unlike base R which uses a weird convention. Also, you don't need to set the size for the title and the labels separately, it just rescales everything.

What about the 3 separate plots?

```r
require(ggplot2)
ggplot(data=iris)+
    theme_classic(base_size=14)+
    ggtitle("Iris")+
    xlab("Sepal Length (cm)")+
    ylab("Petal Length (cm)")+
    geom_point(aes(x=Sepal.Length,y=Petal.Length))+
    facet_wrap(~Species) #this is the only thing that changed
```



The subtitles are automatic, the axis look as they should, and you don't need to subset anything.

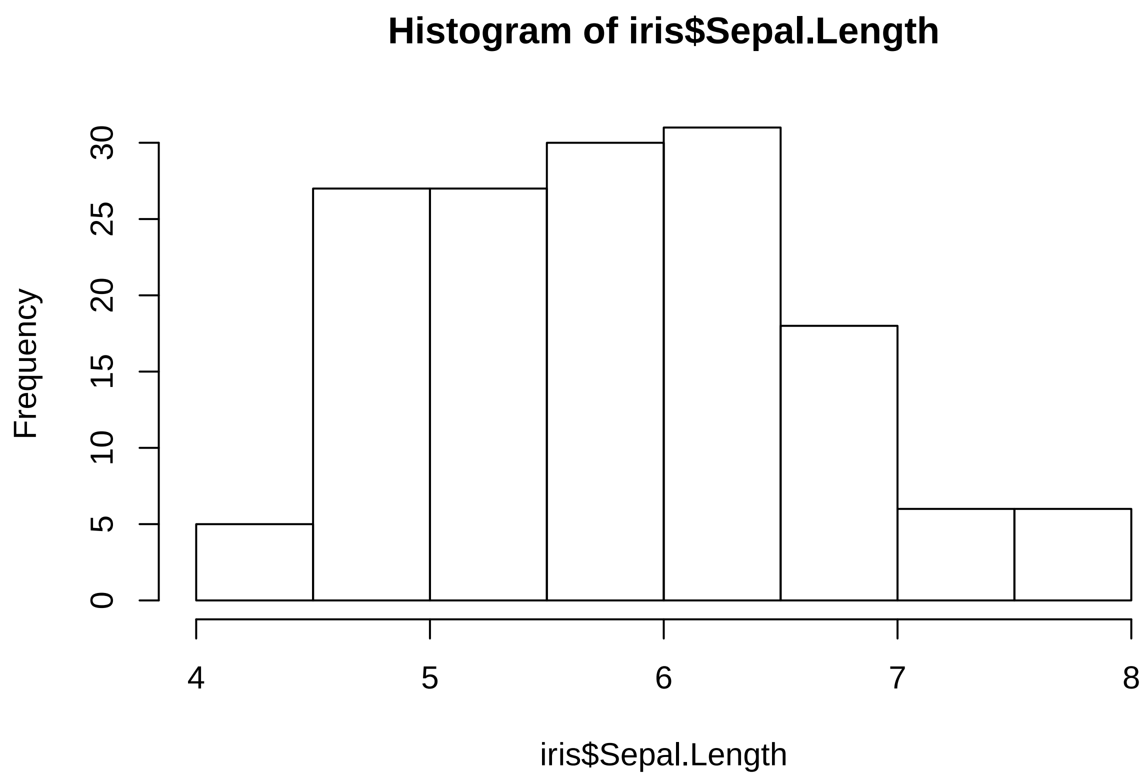
Everything in ggplot roughly works like this:

```
ggplot(data = yourdataframe, aes(x=columnWithXvalues,
                                  y=columnWithYvalues,
                                  color=columnWithColorIDs))+
    geom_*type of plot you want*()
```

There are geoms for all sorts of plots: points, lines, densities, polygons, heatmaps, histograms, regression lines, violin plots, etc. You can find a list of geoms by loading ggplot2, typing geom_ and pushing *tab*.

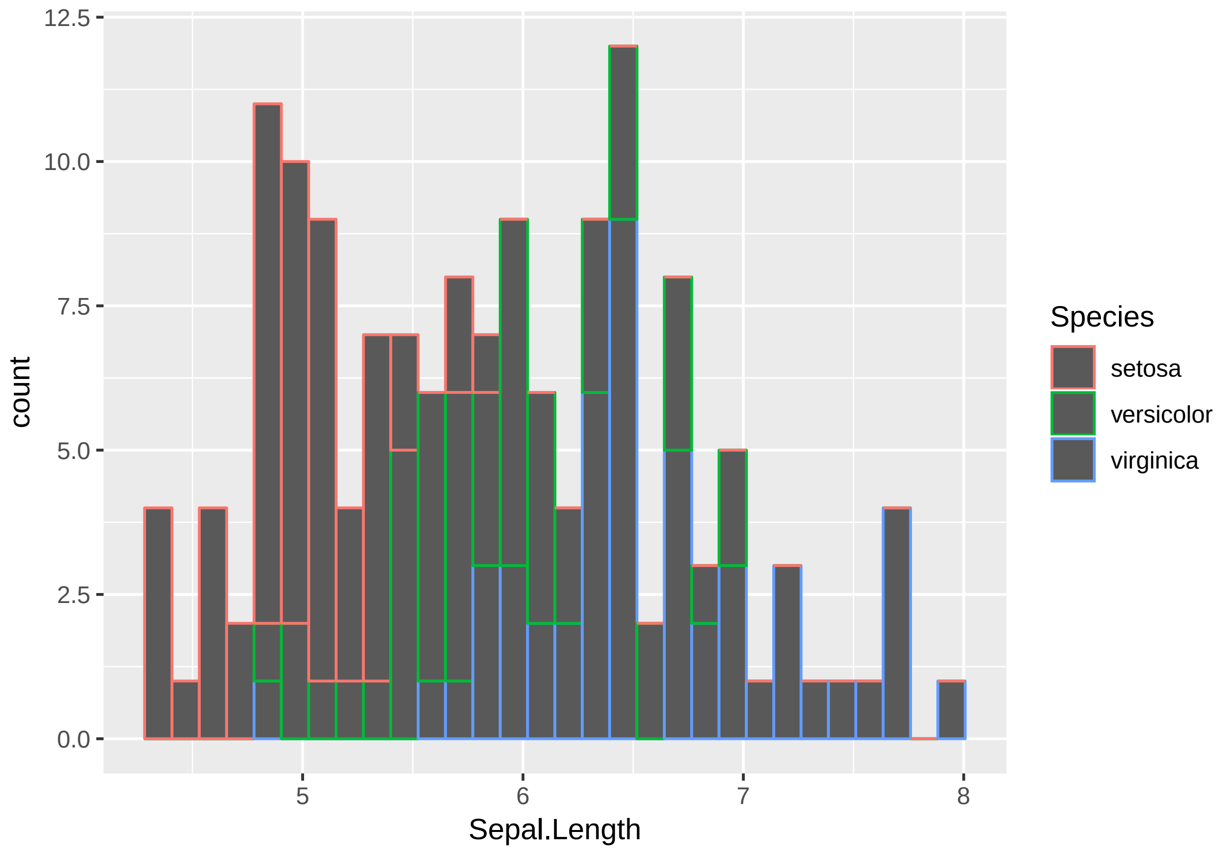
R base graphics can do most, but not all, of the same types of plots but they each require a different function that can work very differently from plot(). For example, look at a base graphics and a ggplot histogram of iris sepal lengths.

```
#The very basic plots in R base are ok, probably easier than ggplot.
hist(iris$Sepal.Length)
```



```r
ggplot(iris,aes(Sepal.Length,color=Species))+
    geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Exercise: make the above histogram look good.

Challenge:

Using this dataset and also this dataset try to reproduce this figure.

Hint: you will need geom_tile, the viridis magma palette, and facet wrap. You may also want to use the function month() from package lubridate, just so you don't have to type the names of each month.