Homework Set 2

Dr. Christopher Brown

December 23, 2018

Introduction

Welcome to the second homework set in the Data Analysis class. In the exercise sets in this class, you should expect to be learning as you work, so if you come to an exercise and you don't know what to do off the top of your head, well... that's what is supposed to happen sometimes!

My expectations for you in all homework assignments are:

- You'll use available resources to work through the exercises. Resources include the internet and each other.
- You'll try to learn techniques, concepts, and R commands as you progress through the exercises.
- You'll experiment a bit beyond the required exercises.

I've provided some R code examples in some exercises. As in the first homework set, this is not **the only** R code to carry out the tasks, it is **some workable** R code to carry out these tasks.

Summaries of Univariate Data

Exercise 1 Read the Wikipedia entry on the five-number summary, and write a brief synopsis. Make sure you understand how to compute and interpret this summary.

Exercise 2 Load the *iris* dataset with *data("iris")*. Use the *summarize* function to provide a summary of all of the *iris* variables:

```
data('iris')
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

How does *summary* summarize numerical variables? How does *summary* summarize categorical variables?

Exercise 3 In Exercise 2 we have produced a summary of all of the measurements, pooled together. But it is almost certainly better to produce summaries of the measurements separated by species. Select all the data in the dataframe for the *setosa* species and summarize it with *summary*, then do the same for each of the other two species.

Exercise 4 You can compute the range for numerical data by computing the difference between the maximum and minimum values of the data. For example,

```
max(iris$Sepal.Length)-min(iris$Sepal.Length)
```

```
## [1] 3.6
```

You can compute the **interquartile range** for numerical data by computing the difference between the 75th and 25th percentile values. In R, you can use the *IQR* function. For example,

```
IQR(iris$Sepal.Length)
```

```
## [1] 1.3
```

And, you can compute the **standard deviation** of numerical data by using the R function *sd*. For example,

```
sd(iris$Sepal.Length)
```

```
## [1] 0.8280661
```

The range, IQR, and standard deviation are all measures of variation for data. Okay, finally to the point of this exercise! We learn in intro statistics classes that whenever the data is normally distributed, we have

$$\text{IQR} \approx 0.35\sigma$$

and

$$\text{range} \approx 4\sigma$$

where σ denotes the standard deviation. Of course, we know that many data sets are non-normal, so these approximations may not be good. So, how good are these estimates for the *Sepal.Width* measurement in the iris dataset? Compute the IQR, range, and standard deviation values of *Sepal.Width* for each species and check against the stated estimates for normal data. Is this data normal?

Exercise 5 Moral of the story: Summaries are readable but they tend to obscure data. Appreciate them but be skeptical.

Suppose that someone gives you only the five number summary for some data: min=0, 25th percentile=1,median=2,75th percentile=3, and max=4. For example,

```
x <- seq(from=0,to=4,by=4/100)
summary(x)
```

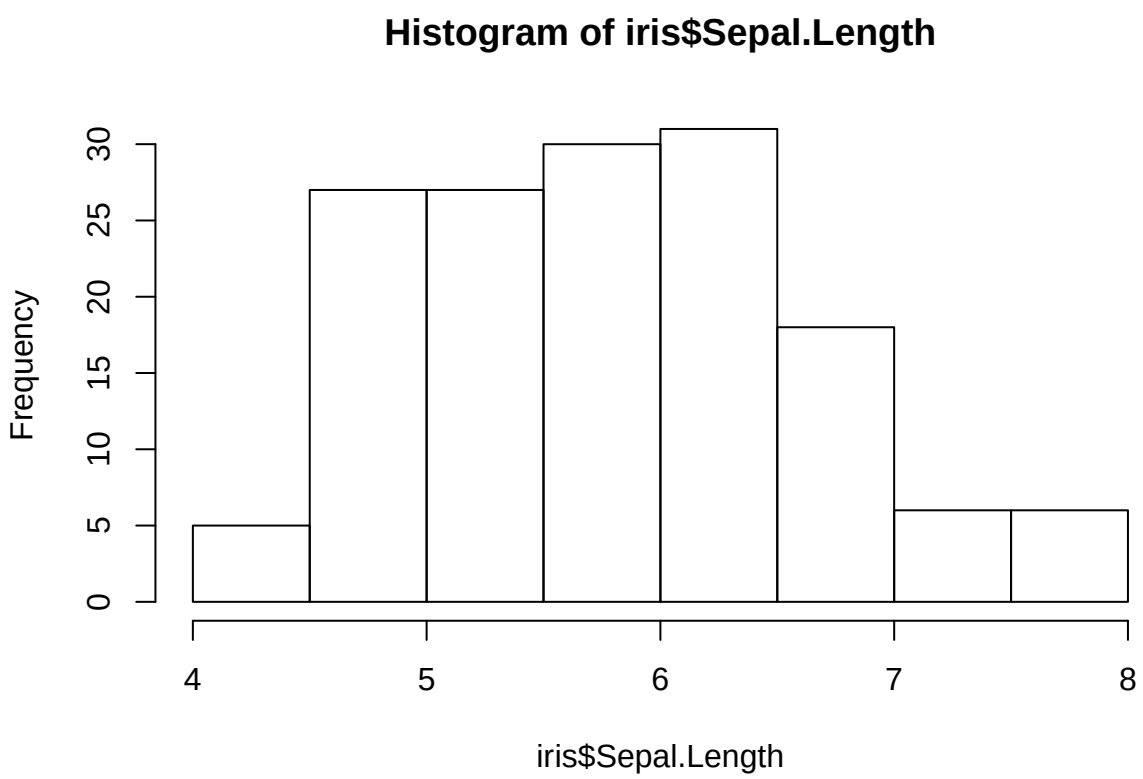
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0       1       2       2       3       4
```

How different can two data sets with this five number summary be? Describe a few very different data sets with this same five number summary. Be creative!

Graphical Presentation of Univariate Data

Exercise 6 Plot a histogram for the *Sepal.Length* variable in the *iris* dataset using

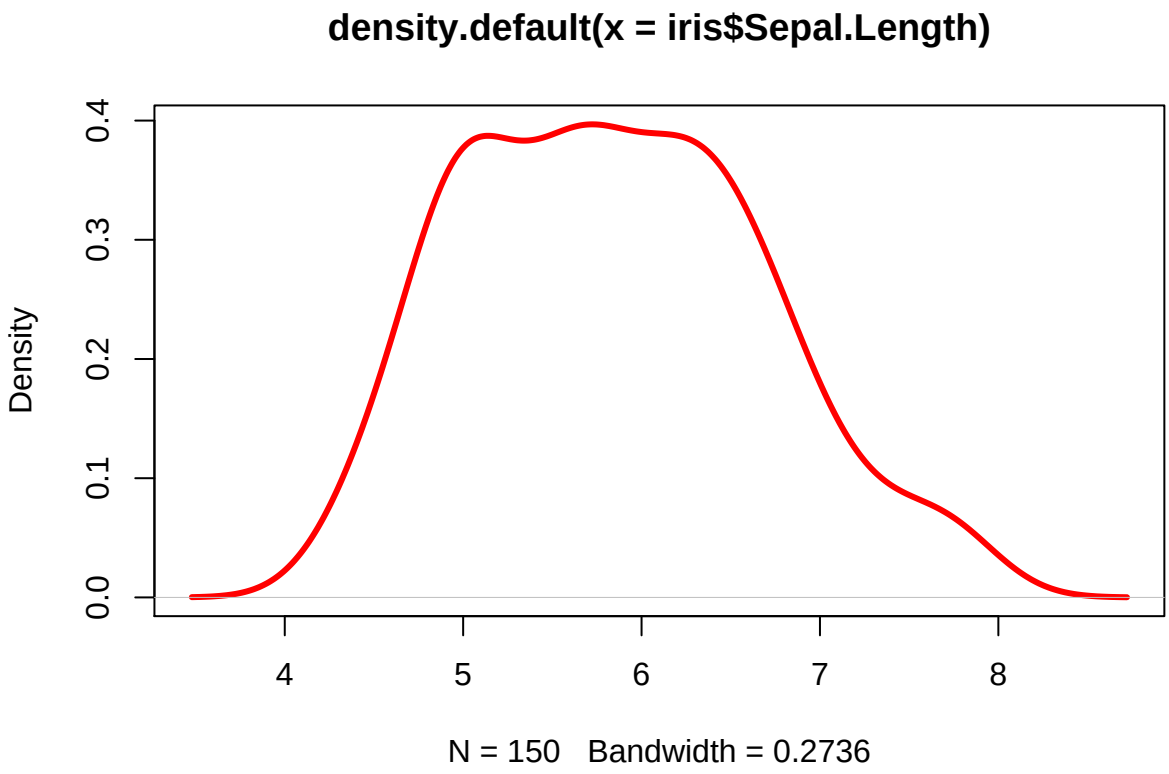
```
hist(iris$Sepal.Length)
```



Then, replot this using the *freq=FALSE* option. What changes? (Hint: axes.) Now try this for the *Sepal.Width* variable.

Exercise 7 Plot the density function for the *Sepal.Width* variable in the *iris* dataset using

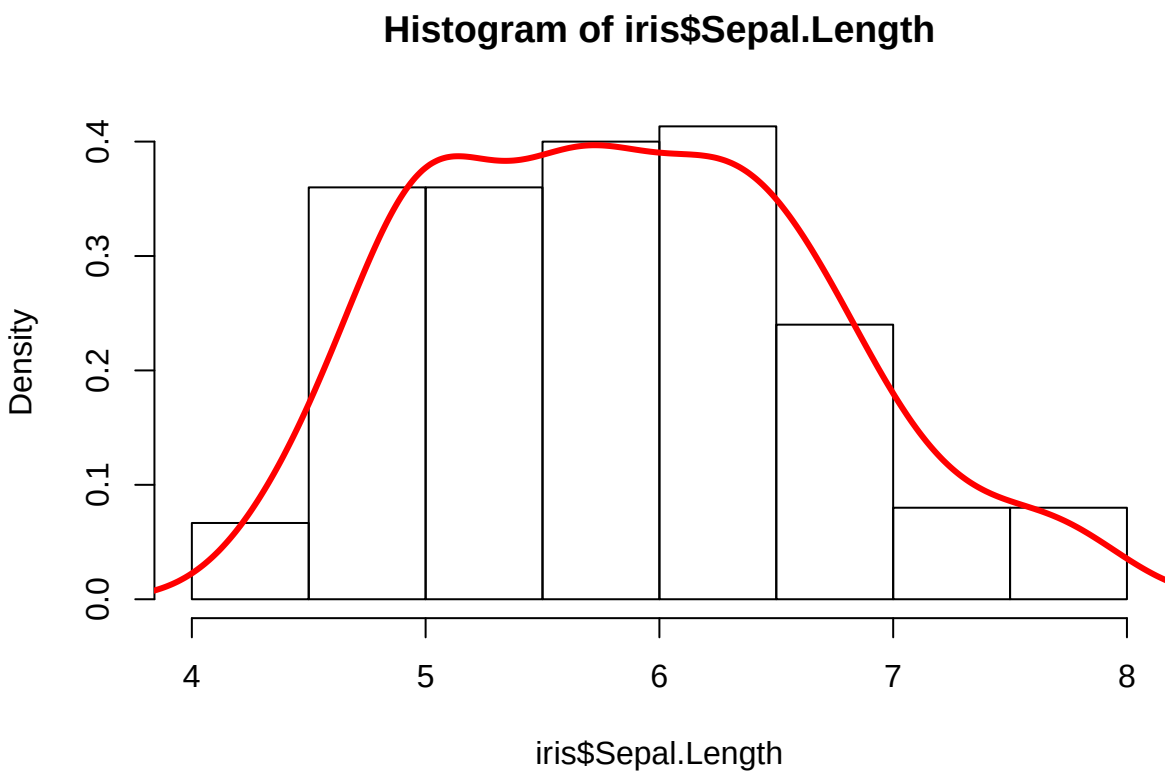
```
plot(density(iris$Sepal.Length),col="red",lwd=3)
```



I used some options to make the plot more visible. Now plot the density function for the *Sepal.Width* variable.

Exercise 8 You can plot the histogram and density functions together on the same axes using

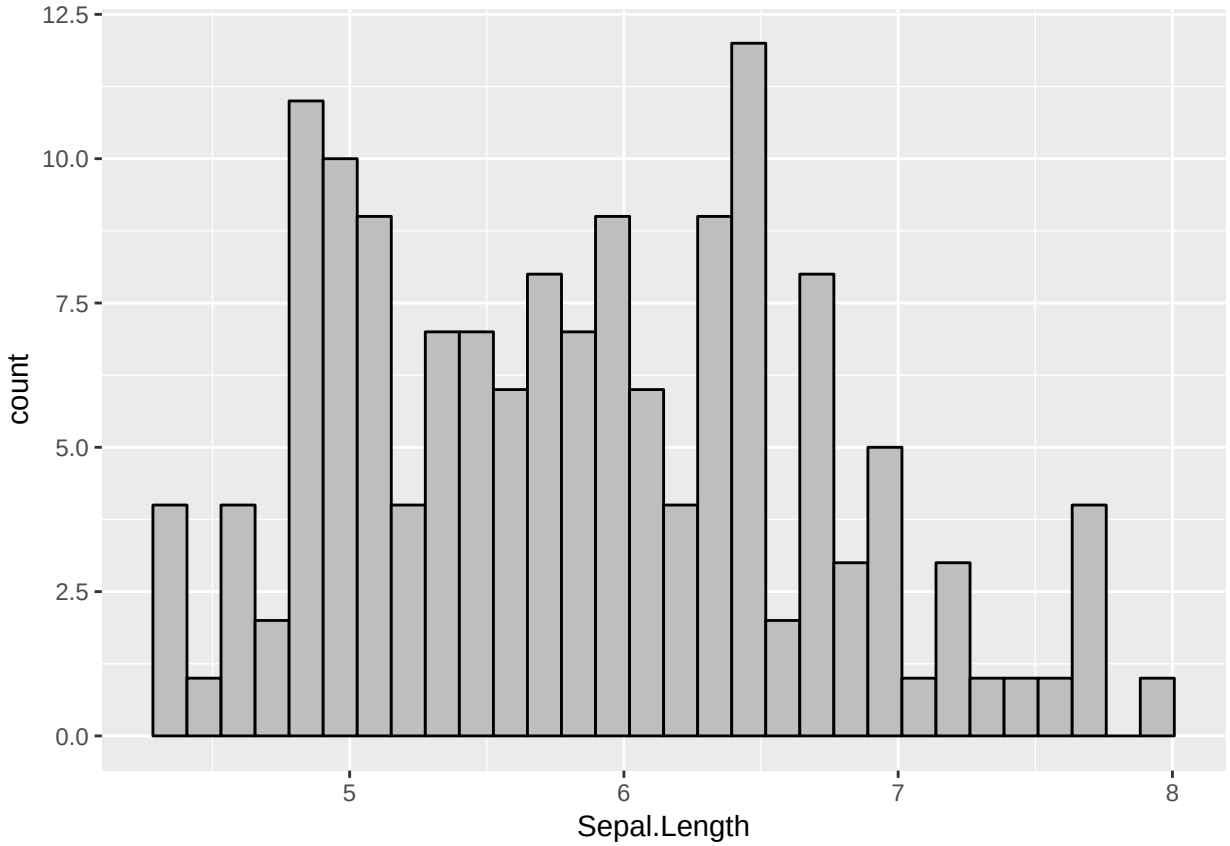
```
hist(iris$Sepal.Length,freq=FALSE)
points(density(iris$Sepal.Length),type='l',lwd=3,col="red")
```



Try repeating this for the other numeric variables in the *iris* dataset.

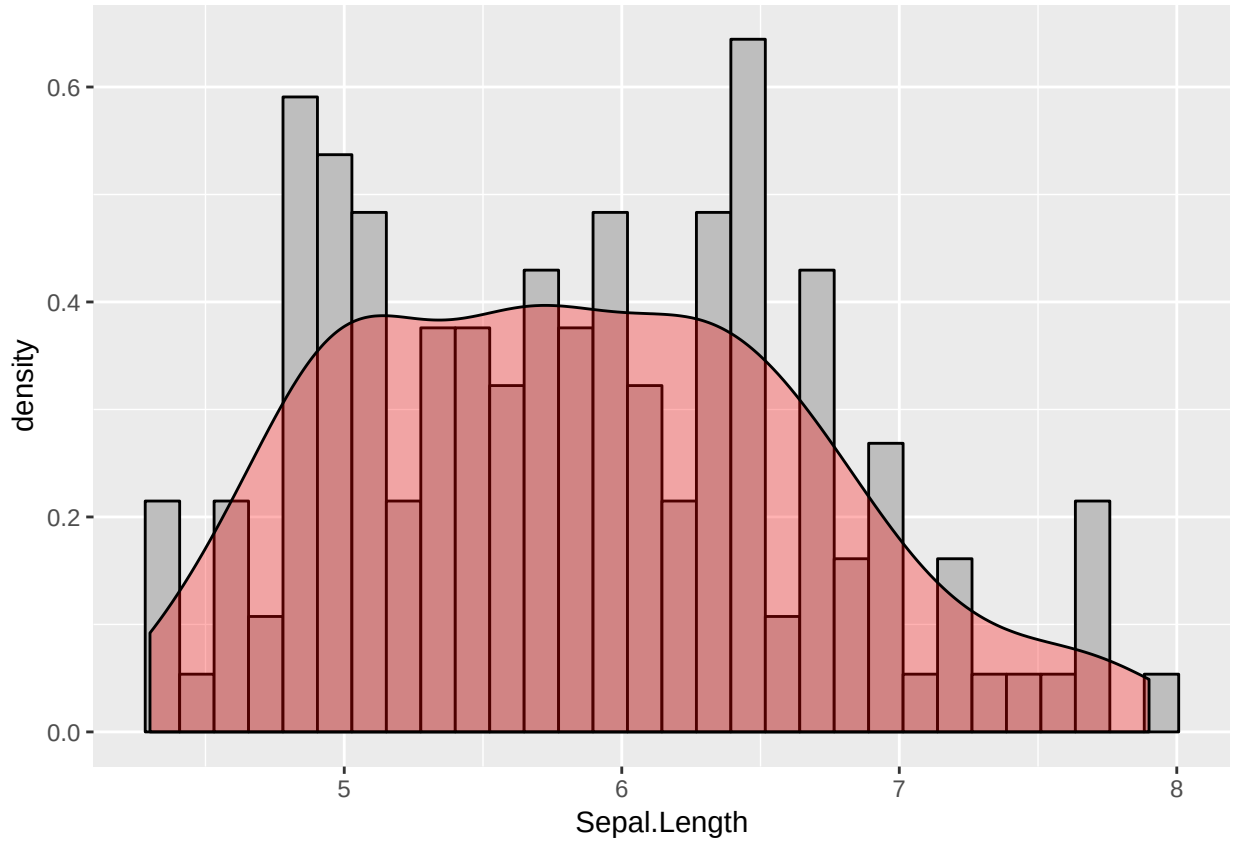
Exercise 9 You can produce similar graphics in the *ggplot2* library. The syntax is different, and takes a bit of acclimatization, but this library gives you tremendous flexibility in producing high-quality graphics. You can create a histogram with

```
library(ggplot2)
ggplot(data=iris,aes(x=Sepal.Length)) +
  geom_histogram(color="black",fill="gray")
```



You can then lay a density plot over this.

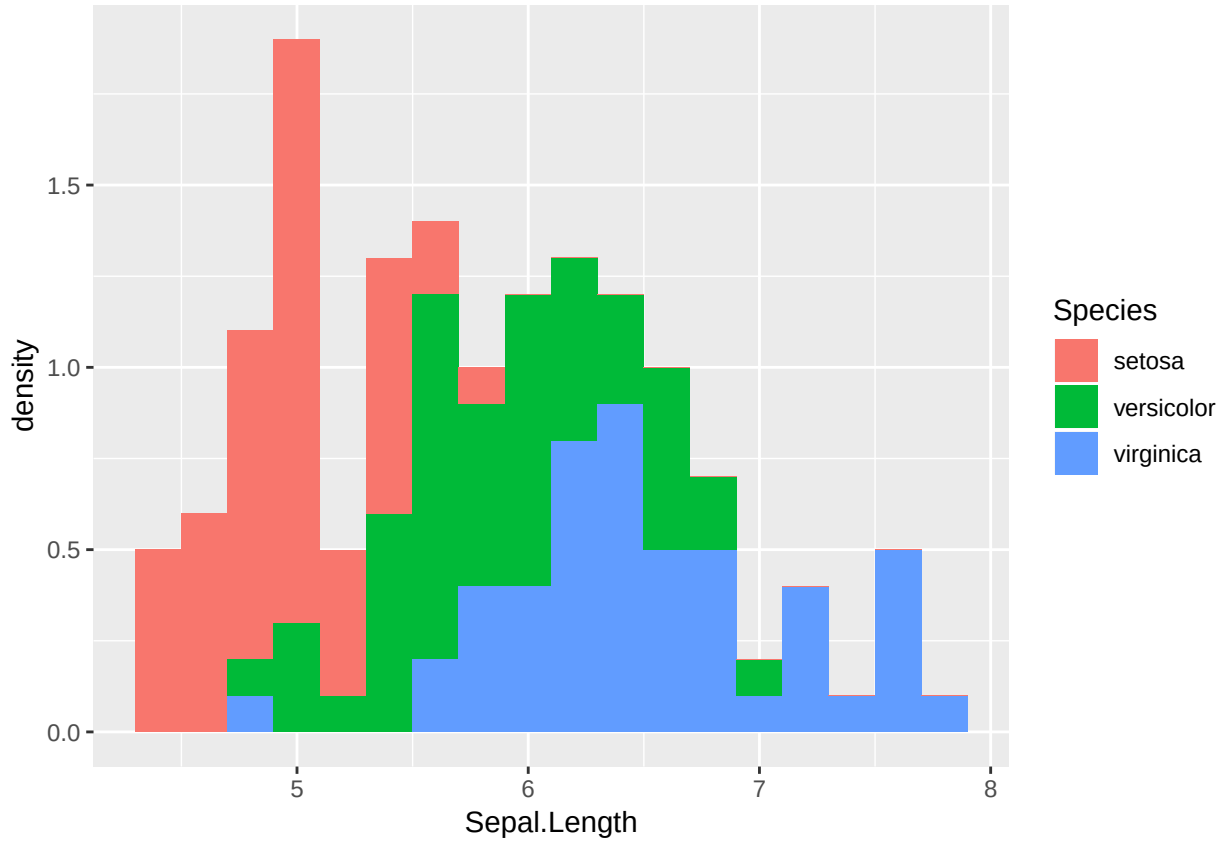
```
library(ggplot2)
ggplot(data=iris,aes(x=Sepal.Length)) +
  geom_histogram(aes(y=..density..),color="black",fill="gray") +
  geom_density(color='black',fill='red',alpha=0.3)
```



Try to produce this same graphic for the *Sepal.Width* variable. Try different combos of fill for the bars and the area under the density plot.

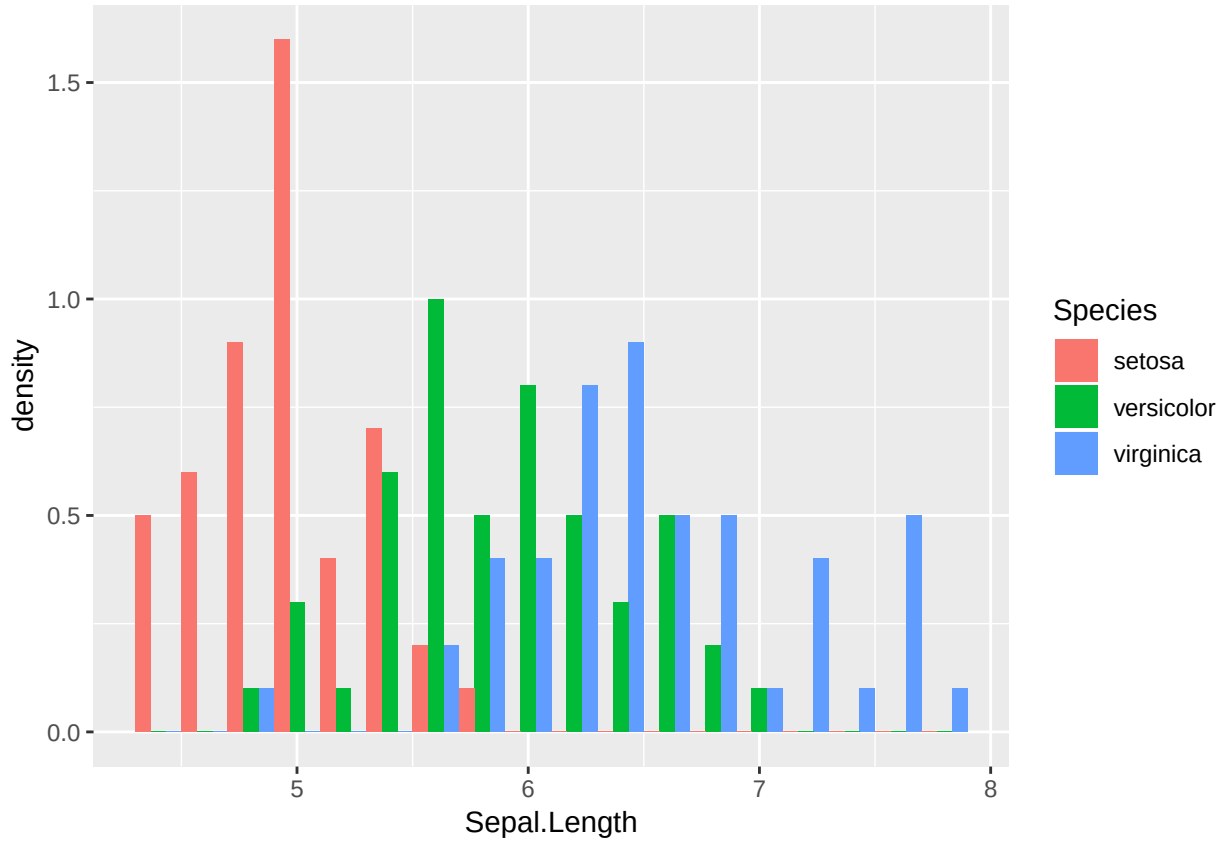
Exercise 10 In Exercises 6 through 9, our histograms and density functions did not separate the measurement by species. Let's see how to do this for histograms. This is much easier to do in the *ggplot2* library, so we'll use that again.

```
library(ggplot2)
ggplot(data=iris,aes(x=Sepal.Length,fill=Species)) +
  geom_histogram(aes(y=..density..),binwidth=0.2)
```



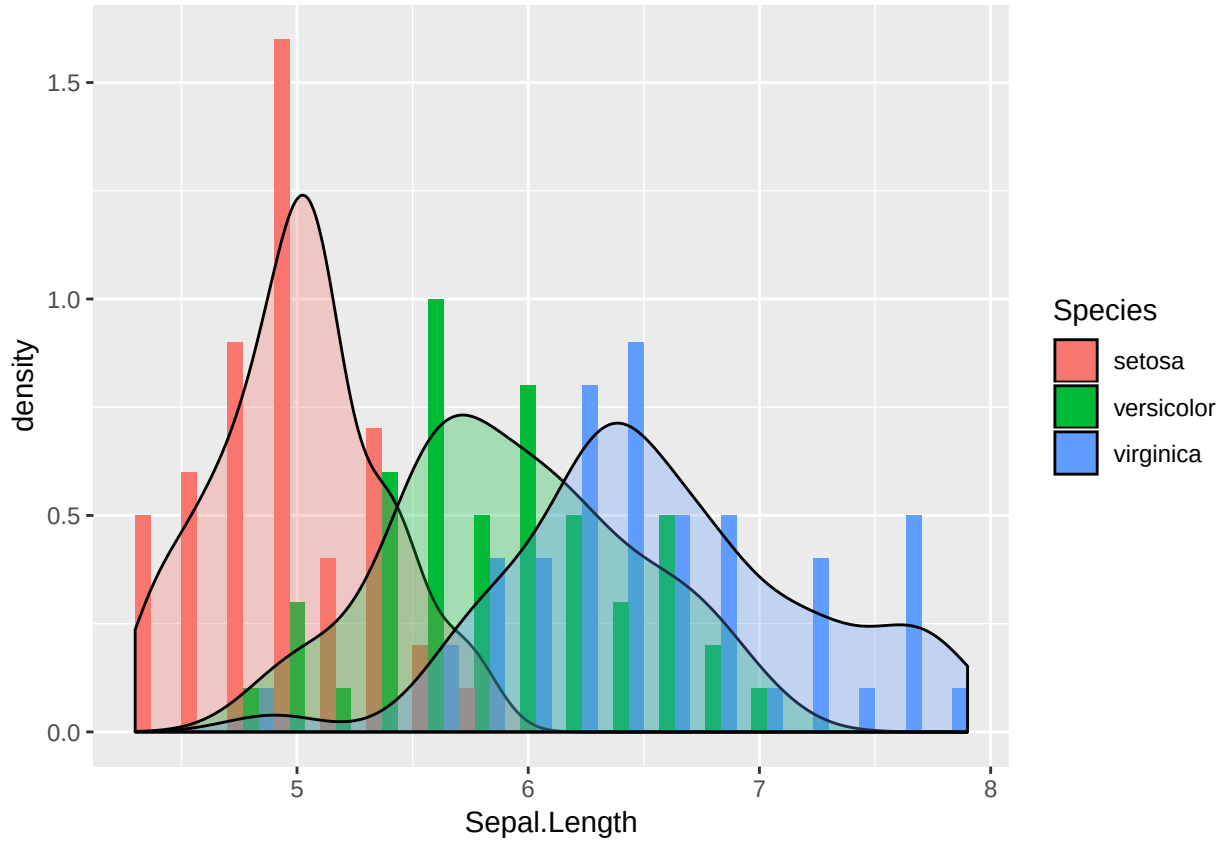
The overlapping histogram blocks here make this difficult to read. We can dodge them (shift them to the side) by using

```
library(ggplot2)
ggplot(data=iris,aes(x=Sepal.Length,fill=Species)) +
  geom_histogram(aes(y=..density..),position="dodge",binwidth=0.2)
```



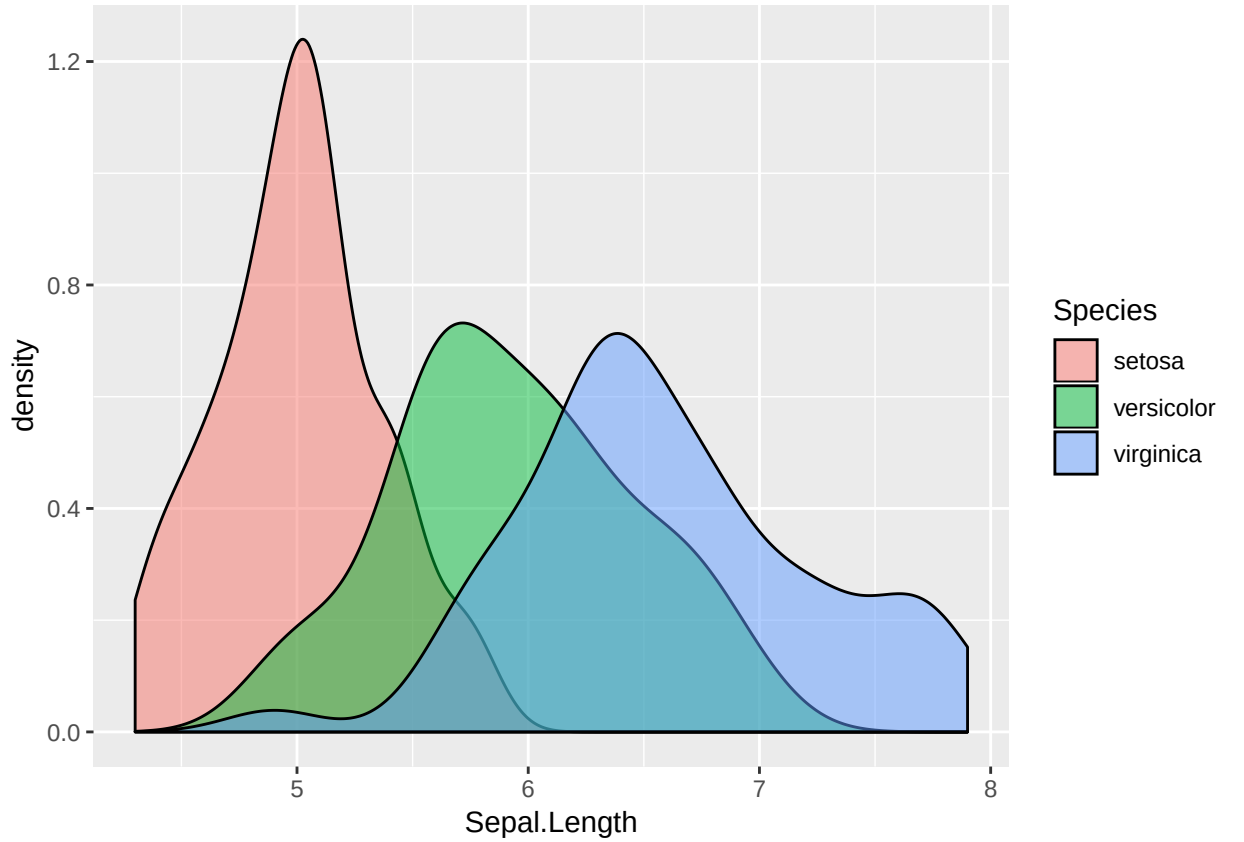
Finally, we can add density plots with

```
library(ggplot2)
ggplot(data=iris,aes(x=Sepal.Length,fill=Species)) +
  geom_histogram(aes(y=..density..),position="dodge",binwidth=0.2) +
  geom_density(alpha=0.3)
```



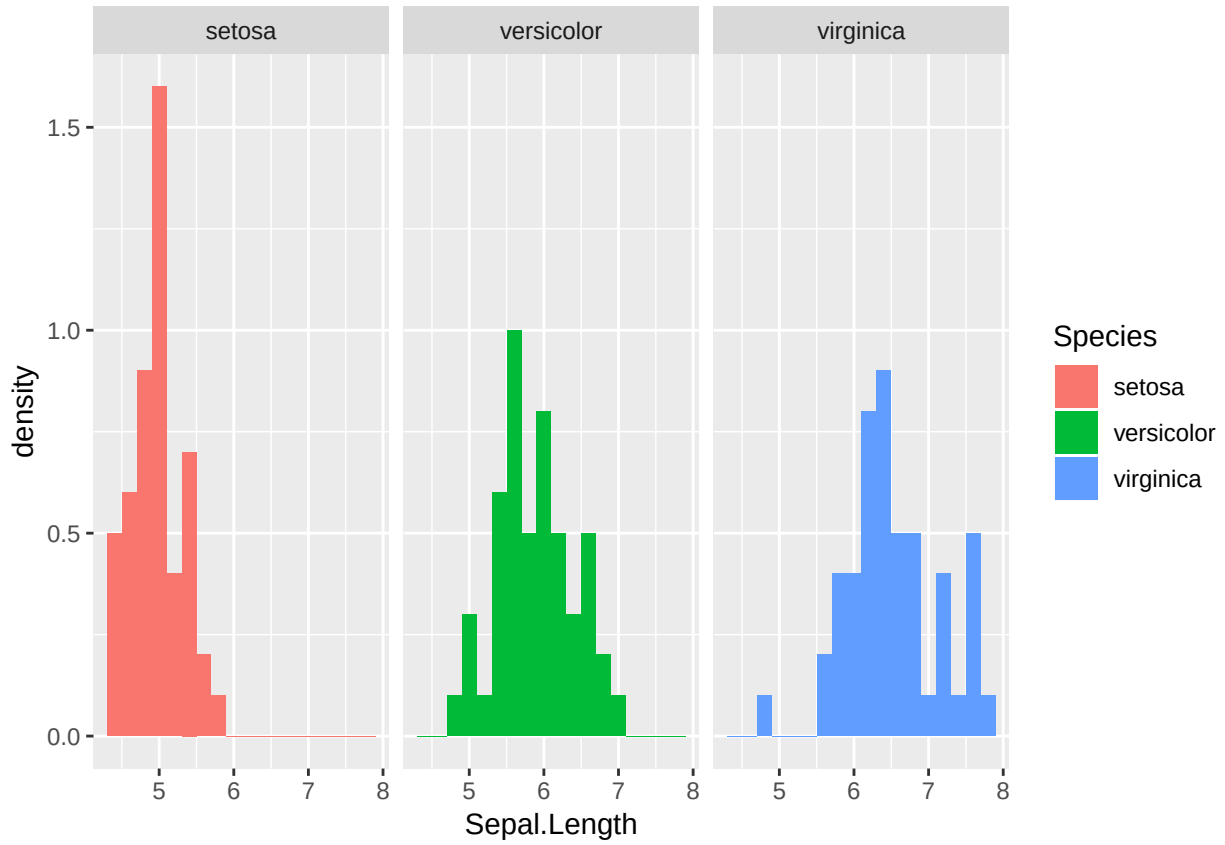
And of course if you just want the density plots,

```
library(ggplot2)
ggplot(data=iris,aes(x=Sepal.Length,fill=Species)) +
  geom_density(alpha=0.5)
```



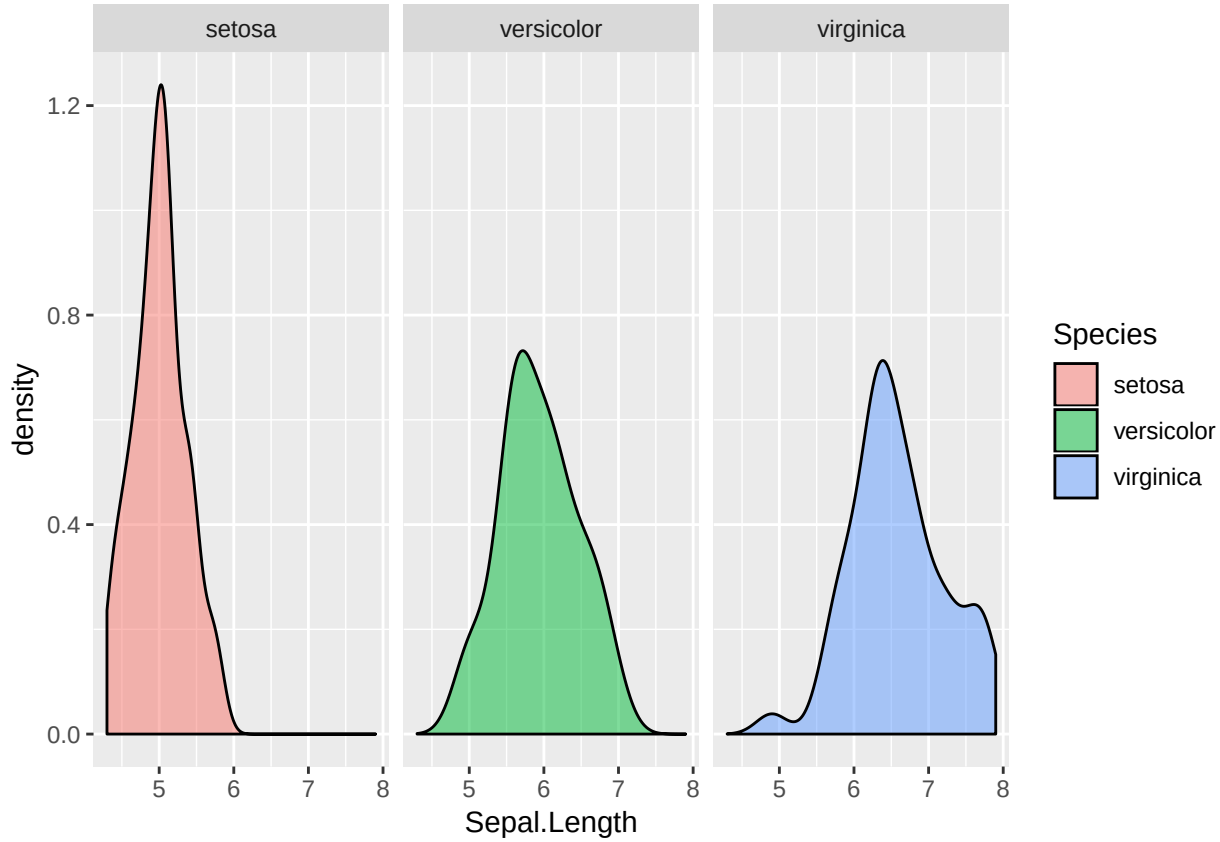
This often makes the most readable graphic. If you do not want the graphics overlaid but prefer them side-by-side, you can do that with

```
library(ggplot2)
ggplot(data=iris,aes(x=Sepal.Length,fill=Species)) +
  geom_histogram(aes(y=..density..),position="dodge",binwidth=0.2) +
  facet_grid(cols=vars(Species))
```



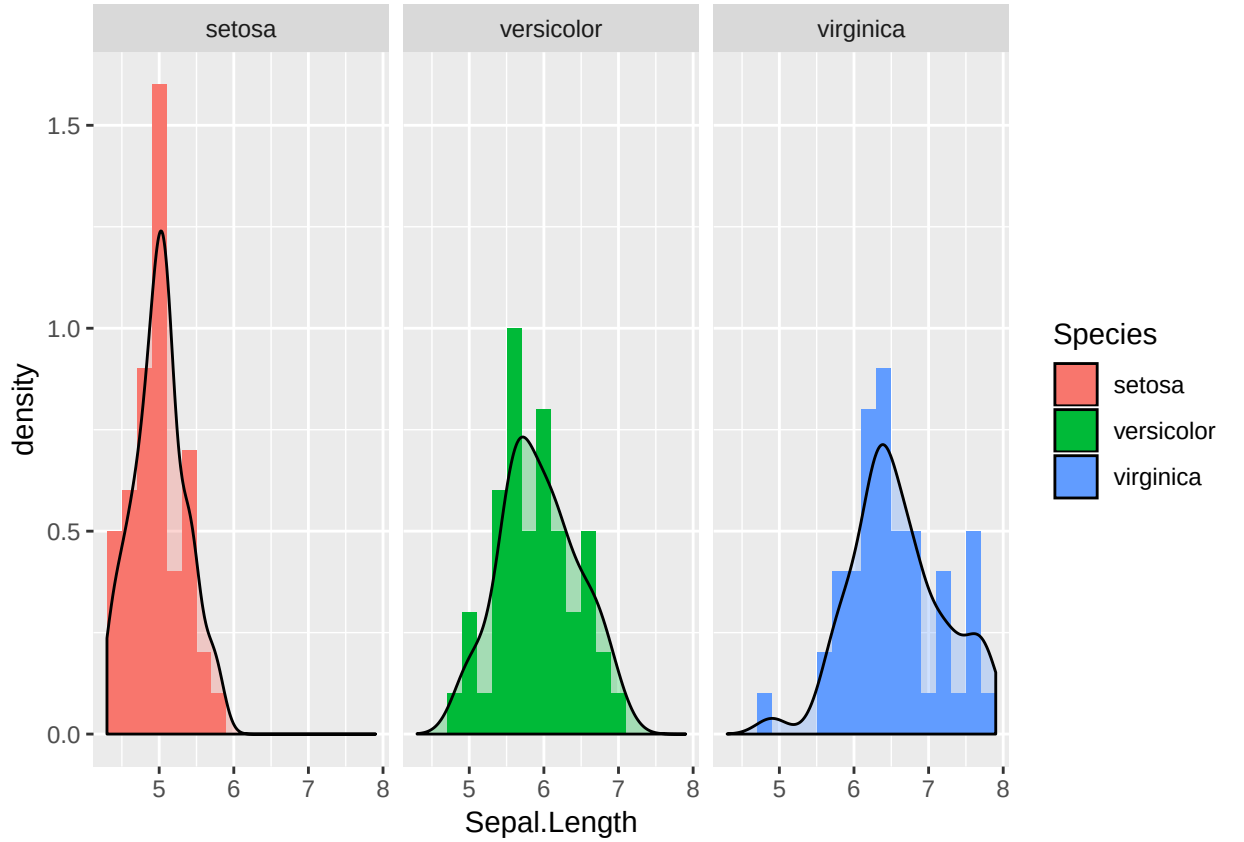
This works with the density plots

```
library(ggplot2)
ggplot(data=iris,aes(x=Sepal.Length,fill=Species)) +
  geom_density(alpha=0.5) +
  facet_grid(cols=vars(Species))
```



and with the combos.

```
library(ggplot2)
ggplot(data=iris,aes(x=Sepal.Length,fill=Species)) +
  geom_histogram(aes(y=..density..),binwidth=0.2) +
  geom_density(alpha=0.3) +
  facet_grid(cols=vars(Species))
```

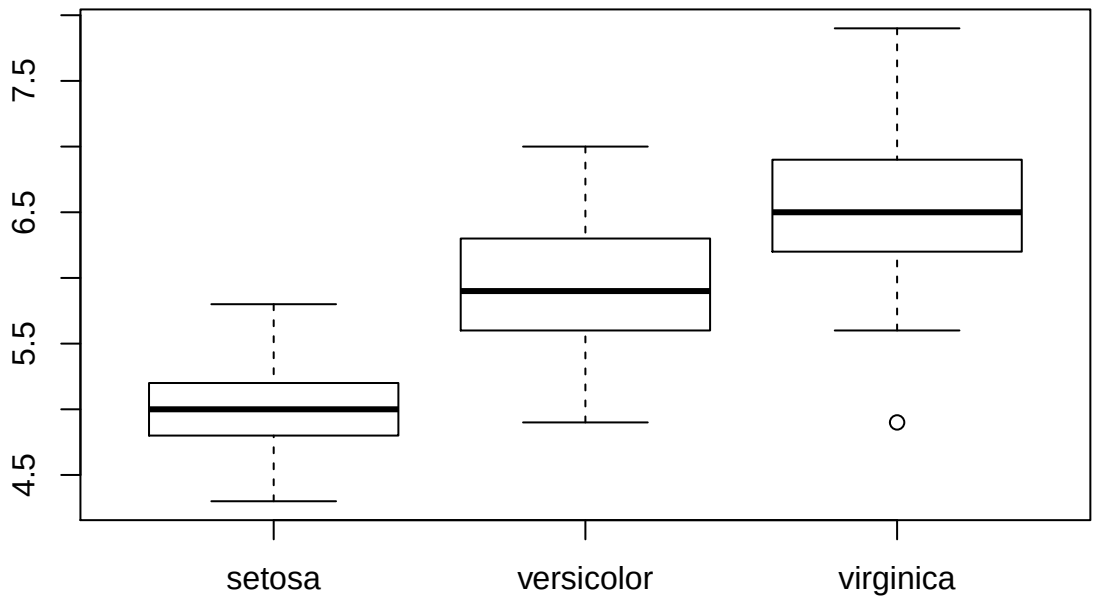


Create two density plots for the *Sepal.Width* variable. One should have the three densities for the three different species together in one plot, and the other should have the three separated into three separate plots (or facets).

Check out this page for more information on creating histograms and density plots using ggplot2. Check out this page for more information on laying out plots using *facet_grid*.

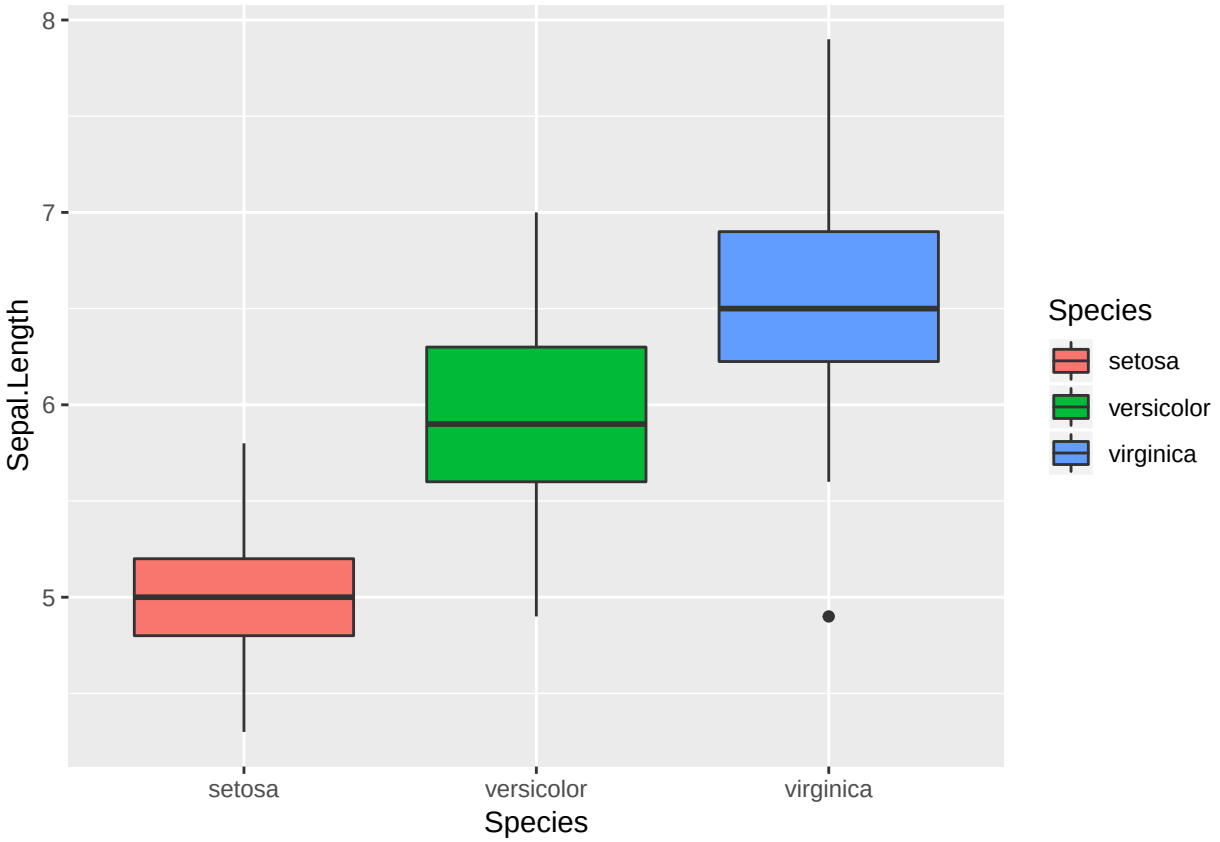
Exercise 11 Sometimes the information in grouped density plots can be overwhelming. Side-by-side boxplots can be much easier to read and much easier to create. For example,

```
boxplot(Sepal.Length ~ Species, data=iris)
```



In *ggplot2*, we have

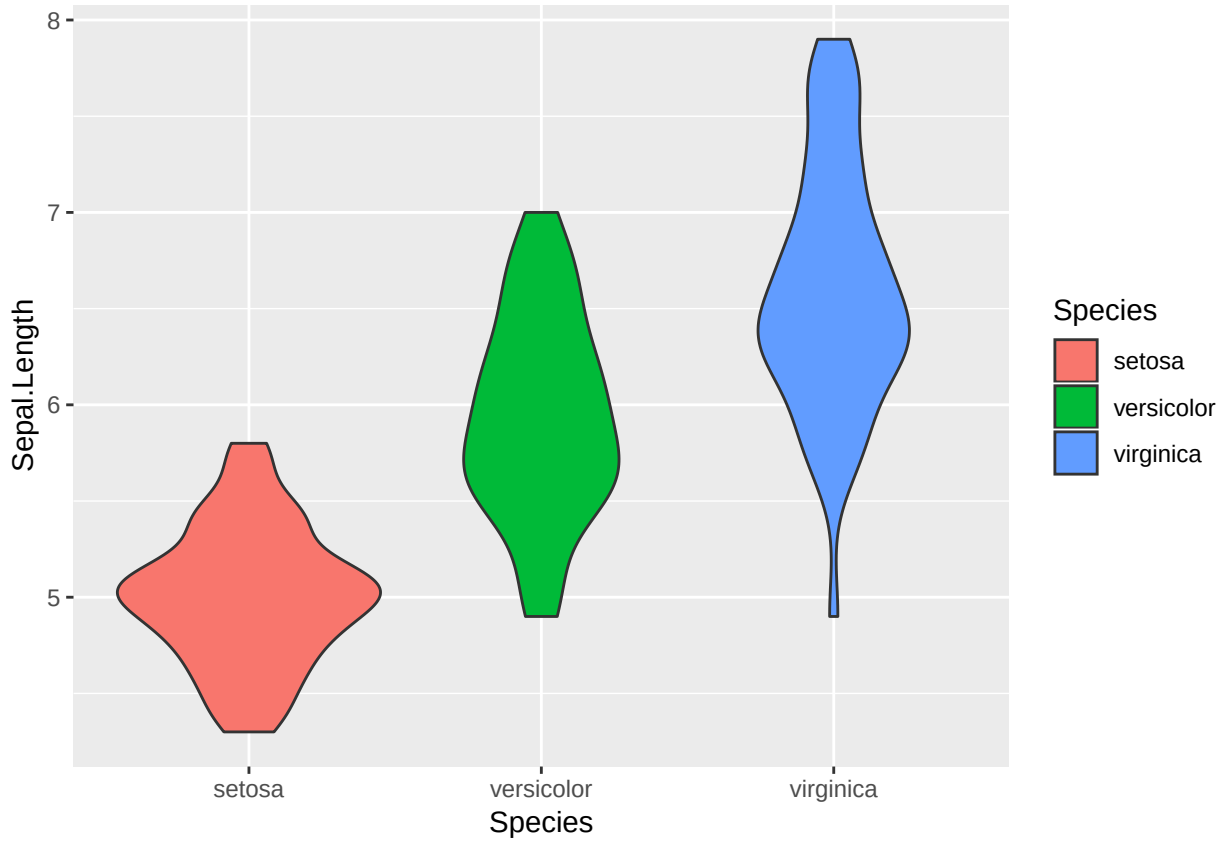
```
library(ggplot2)
ggplot(data=iris,aes(x=Species,y=Sepal.Length,fill=Species)) +
  geom_boxplot()
```



Create side-by-side boxplots for the *Sepal.Width* variable in the *iris* dataset.

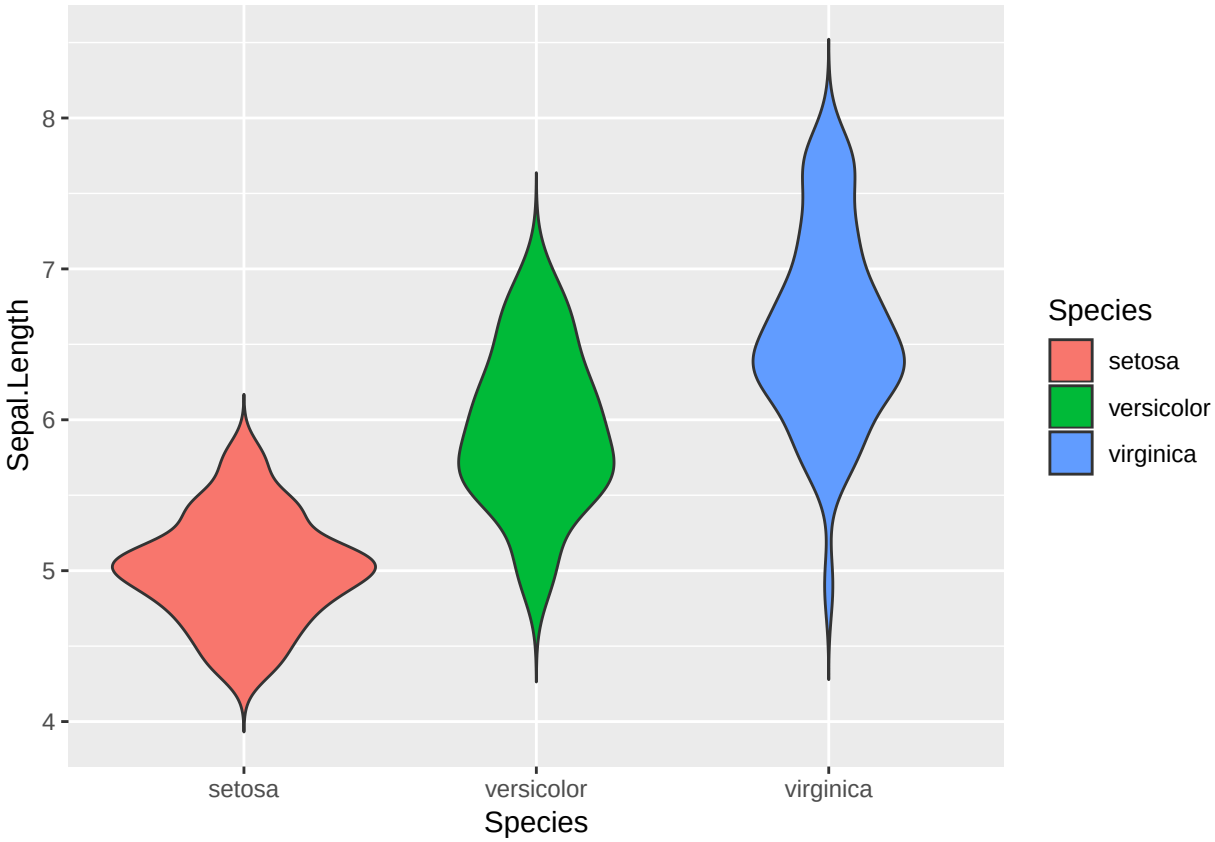
Exercise 12 As noted earlier, five number summaries can be misleading, and since boxplots are simply graphical representations of five-number summaries, we should be skeptical of them. A hybrid plot that combines the boxplot and the density plot is called the **violin plot**.

```
library(ggplot2)
ggplot(data=iris,aes(x=Species,y=Sepal.Length,fill=Species)) +
  geom_violin()
```



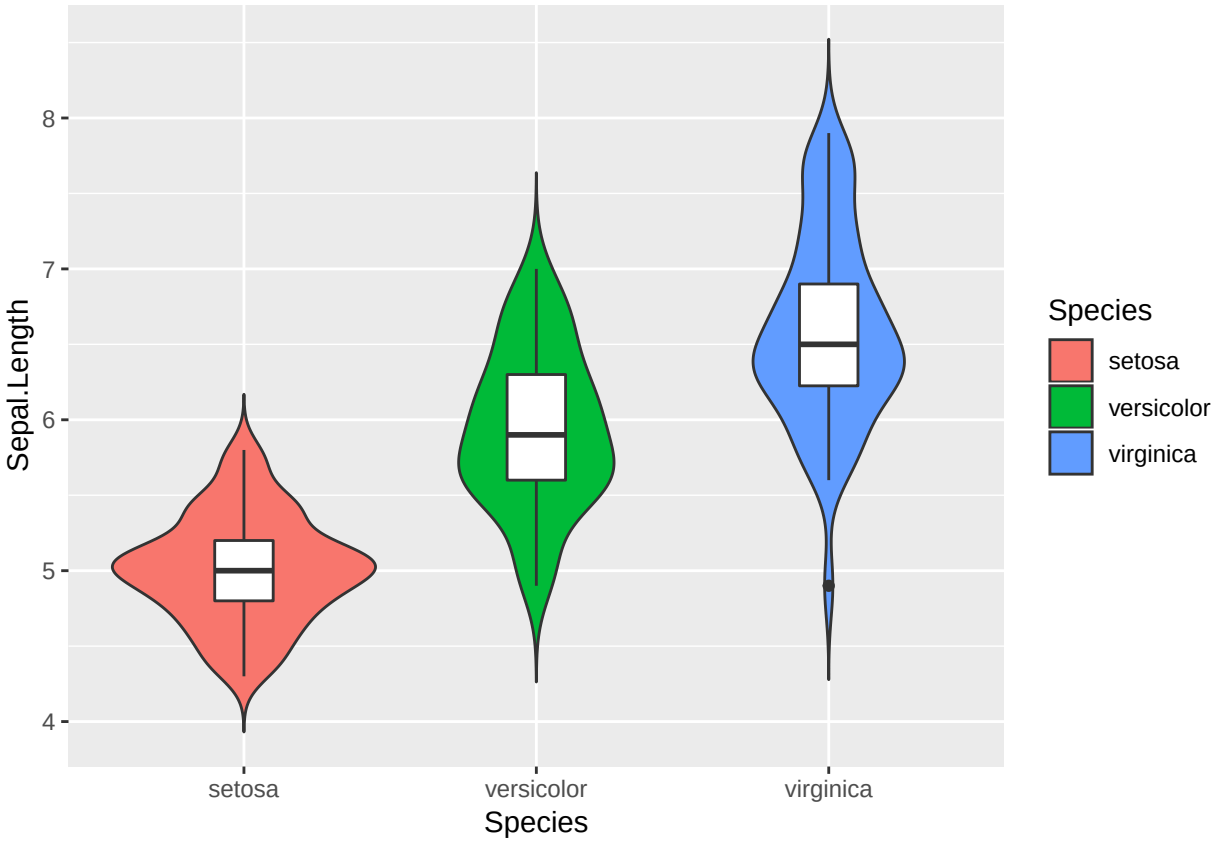
The tips of the violins (why didn't they call these paramecium plots?) look squared-off in some cases. That's because they are; some outlying data has been trimmed off. You can keep all the data (which I recommend) using the *trim=FALSE* option.

```
library(ggplot2)
ggplot(data=iris,aes(x=Species,y=Sepal.Length,fill=Species)) +
  geom_violin(trim=FALSE)
```



You can add a boxplot to your violin plot to get the best of both worlds.

```
library(ggplot2)
ggplot(data=iris,aes(x=Species,y=Sepal.Length,fill=Species)) +
  geom_violin(trim=FALSE) +
  geom_boxplot(width=0.2,fill="white")
```



You may have to adjust the width of your boxplot depending on the sizes of your violins. Create a boxplot, violin plot, and a combo of the two for the *Sepal.Width* variable in the *iris* dataset.

Note before we go on: The *ggplot2* library is rich, with many functions and many options. It takes a long time to learn it well enough to use it, and most people keep a cheatsheet handy. It's in your best interest to maintain a library of examples for *ggplot2* so you can refer to them later.

Tabular Summaries of Univariate Data

Exercise 13 You can create a frequency table for a categorical variable with the *table* command. For example,

```
table(iris$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

Of course, this is not very exciting for the iris dataset. There are two other situations in which you might have to "make" a categorical variable. We can see both in the *mtcars* dataset.

```
data("mtcars")
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

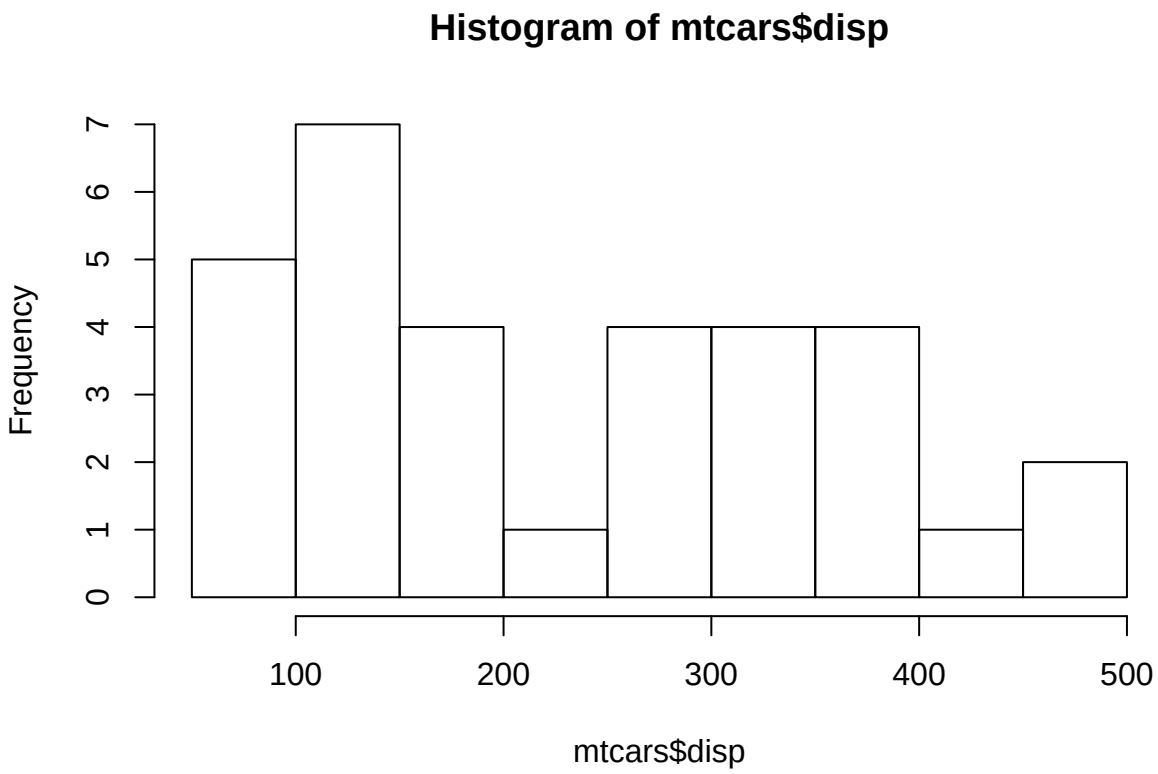
The *cyl* variable has values of only 4, 6, or 8. Despite the fact that it has been coded as numeric, it is really acting like a categorical variable. We can tabulate it as follows:

```
table(mtcars$cyl)
```

```
##
##  4  6  8
## 11  7 14
```

If we examine the histogram for the *disp* variable, we see that

```
hist(mtcars$disp)
```



we essentially have three categories, or bins: 0-200, 200-400, and above 400. We can bin the *disp* variable as follows:

```
breaks <- c(min(mtcars$disp),200,400,max(mtcars$disp)) # Create the endpoints of the bins
disp_freq <- cut(mtcars$disp,breaks=breaks) #Put data in the bins
table(disp_freq) # Tabulate, returning frequency counts in the bins
```

```
## disp_freq
## (71.1,200]  (200,400]  (400,472]
##         15         13          3
```

For the *mtcars* dataset, create a frequency table summarizing the *gear* variable, and bin the *wt* variable into a few bins and give a frequency table.

Summaries of Bivariate Data

Exercise 14 If we have two categorical variables we can tabulate them with the *table* command:

```
table(mtcars$cyl,mtcars$gear)
```

```
##
##      3  4  5
##   4  1  8  2
##   6  2  4  1
##   8 12  0  2
```

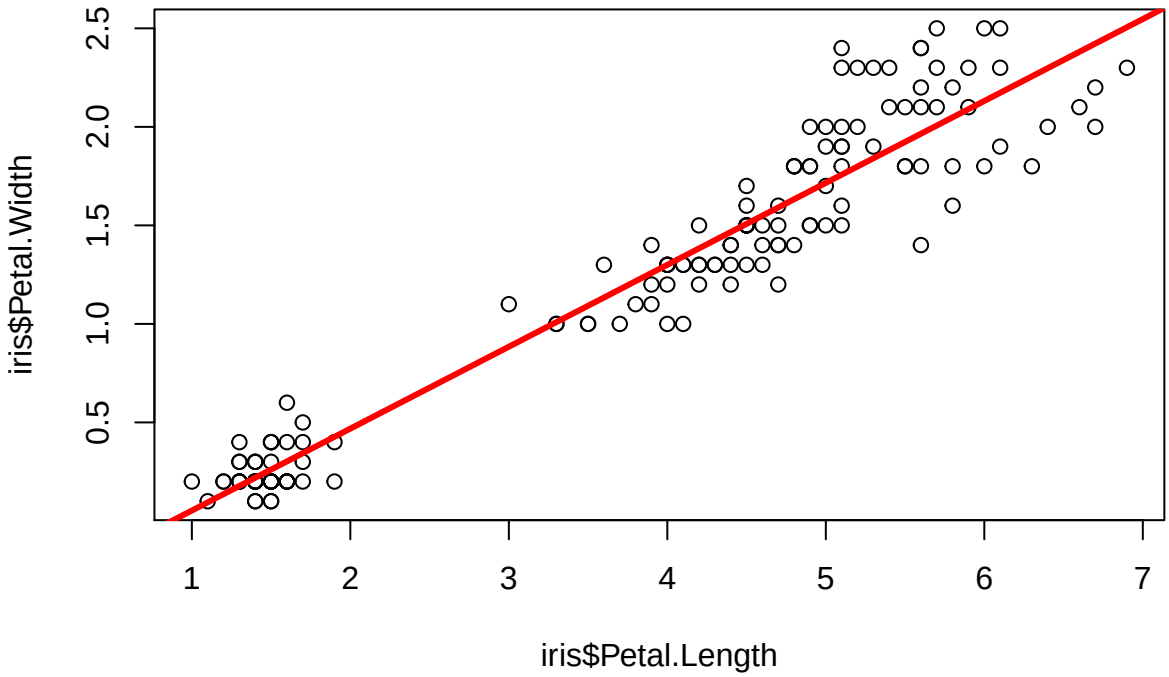
Thus, for example, there are 8 cars in the dataset with 4 cylinders and 4 gears. We can do the same for numerical variables, as long as we bin them first with the *cut* command. An easy way to do this is using quartile cuts, as in

```
table(cut(mtcars$disp,quantile(mtcars$disp)),
      cut(mtcars$hp,quantile(mtcars$hp)))
```

```
##
##               (52,96.5] (96.5,123] (123,180] (180,335]
##   (71.1,121]          4          2          0          0
##   (121,196]           2          5          1          0
##   (196,326]           0          2          5          1
##   (326,472]           0          0          2          6
```

This is not always incredibly readable, but can make for some quick initial analysis on the dataset. Create a tabular summary of the *Petal.Length* versus the *Petal.Width* variables in the *iris* dataset.

Exercise 15 In some cases there is a linear relationship between two variables. The **correlation** r tells us the strength of that relationship (how big is r^2?) and the direction. For example, the *Petal.Length* and *Petal.Width* variables in the *iris* dataset are reasonably linearly related, as we can see in the following scatterplot:



That is, the data clusters pretty well around the best-fit line through the data, plotted as a solid red line. We compute the correlation r with R's *cor* function.

```
cor(iris$Petal.Length,iris$Petal.Width)
```

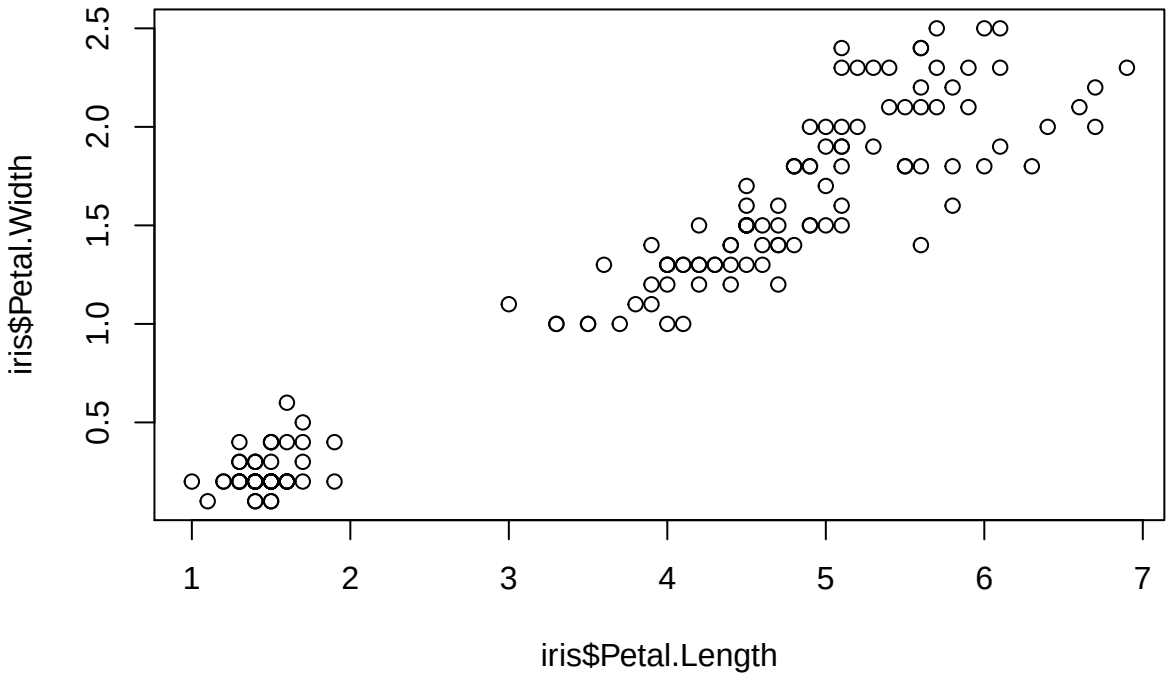
```
## [1] 0.9628654
```

A correlation near ± 1 tells us that the data is tightly clustered around the best-fit line; a correlation near 0 tells us that the data is not tightly clustered around its best-fit line at all. Find the correlation of the *Sepal.Length* and *Sepal.Width* variables, and the *Petal.Length* and *Sepal.Length* variables.

Graphical Presentations of Bivariate Data

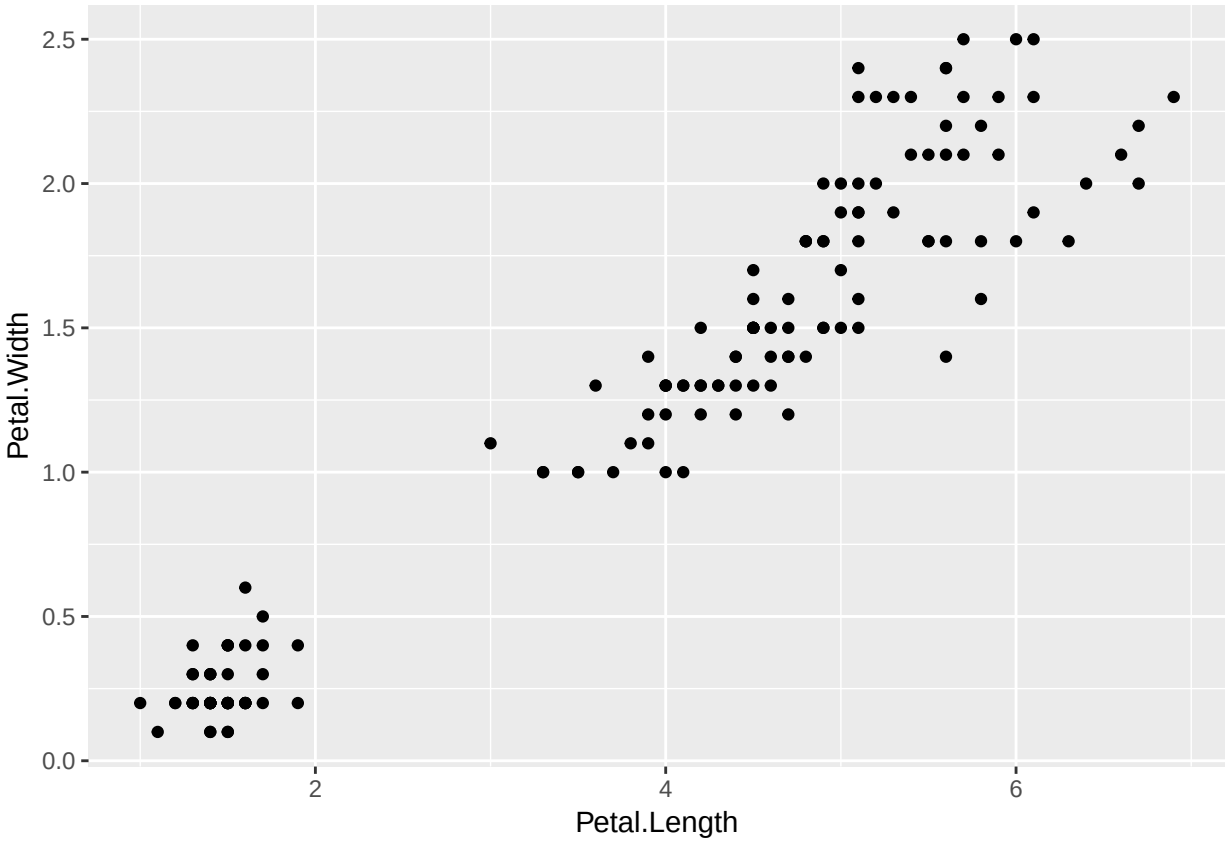
Exercise 16 The **scatterplot** is to bivariate data what the histogram/density plots are to univariate data; it is the basic workhorse that helps us see relationships between variables. To create a scatterplot, we can use

```
plot(iris$Petal.Length,iris$Petal.Width)
```



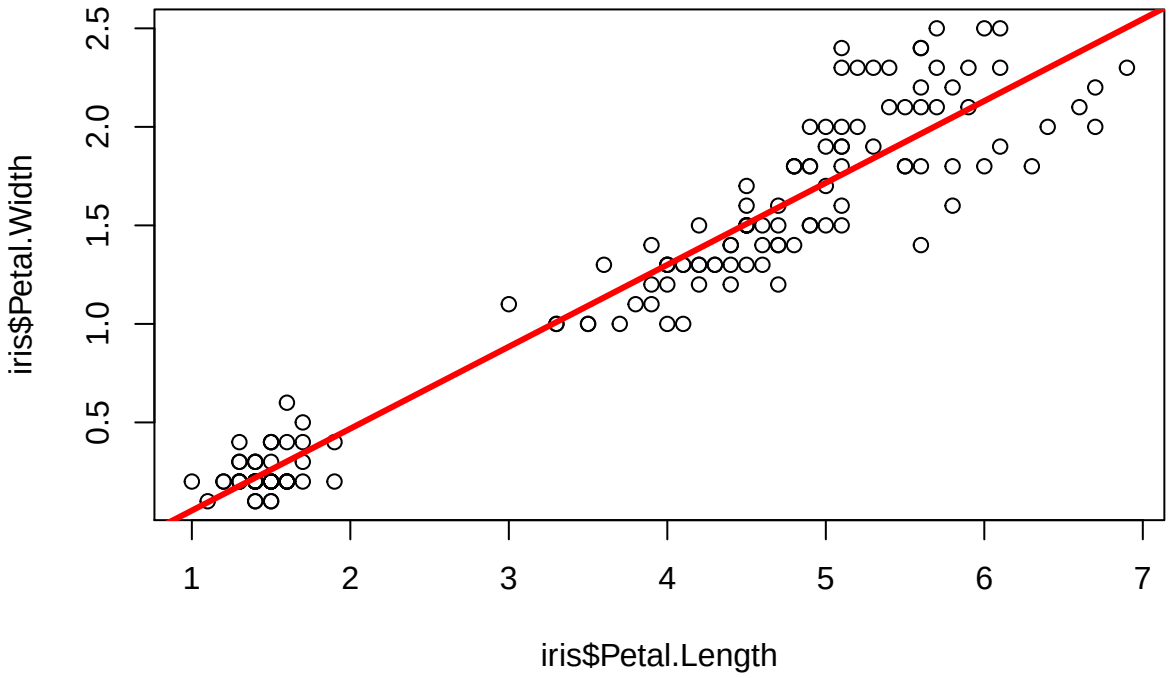
in base R, and

```
library(ggplot2)
ggplot(data=iris,aes(x=Petal.Length,y=Petal.Width)) +
  geom_point()
```



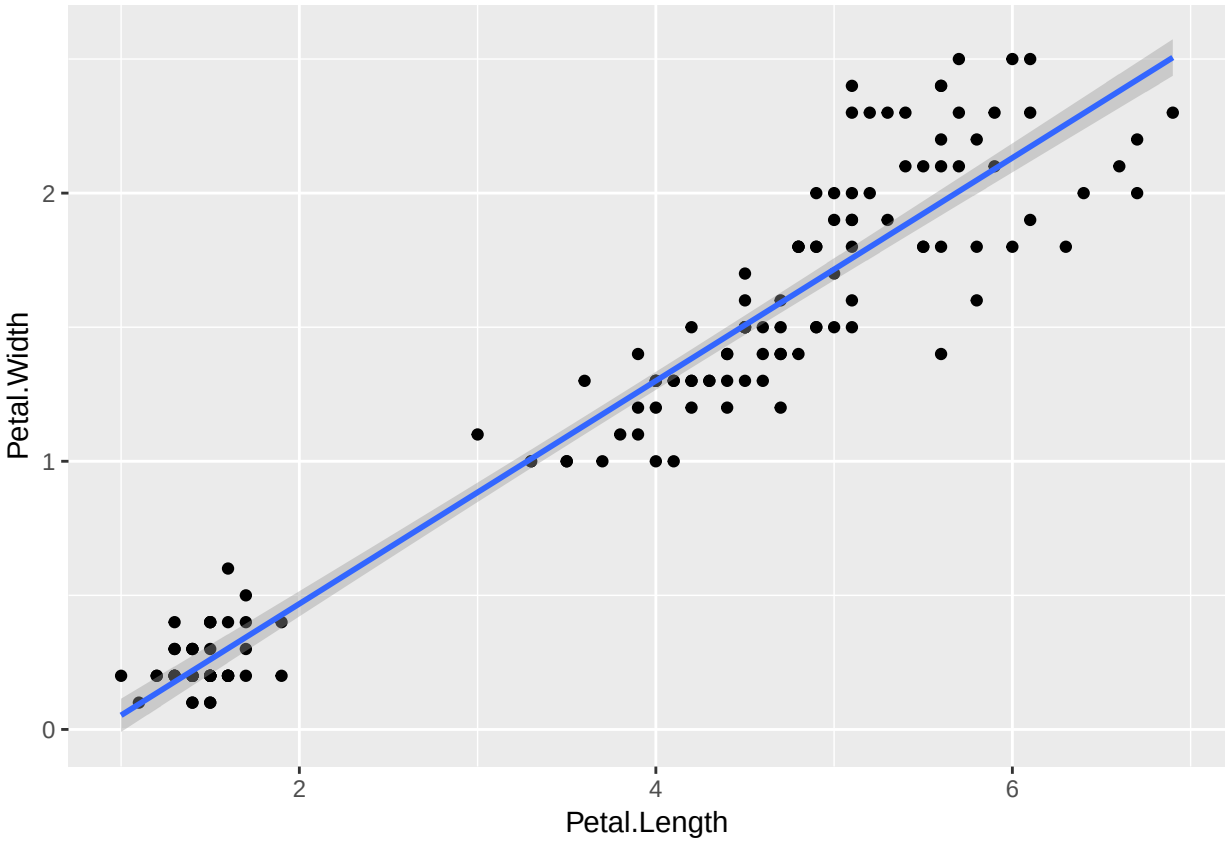
in *ggplot2*. To add the best-fit line in base R, we can use

```
plot(iris$Petal.Length,iris$Petal.Width)
abline(lm(Petal.Width ~ Petal.Length,data=iris),lwd=3,col="red")
```



and in *ggplot2* we can use

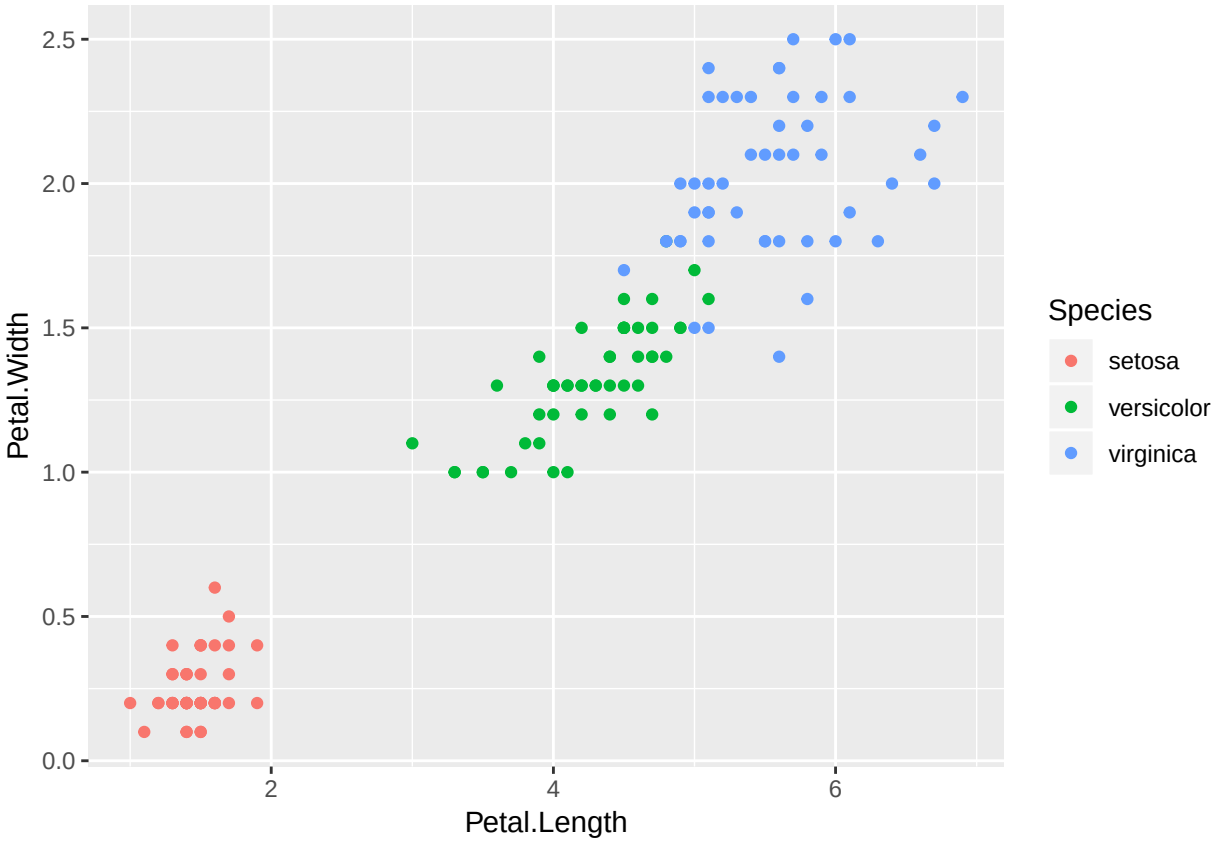
```
library(ggplot2)
ggplot(data=iris,aes(x=Petal.Length,y=Petal.Width)) +
  geom_point() +
  geom_smooth(method="lm")
```



We get a little more information with the *ggplot2* graphic; we'll discuss why the line has a cloud around it later in the class. Give a scatter plot for the *Sepal.Width* (y variable) versus the *Sepal.Length* (x variable) in the *iris* dataset, and include the best-fit line.

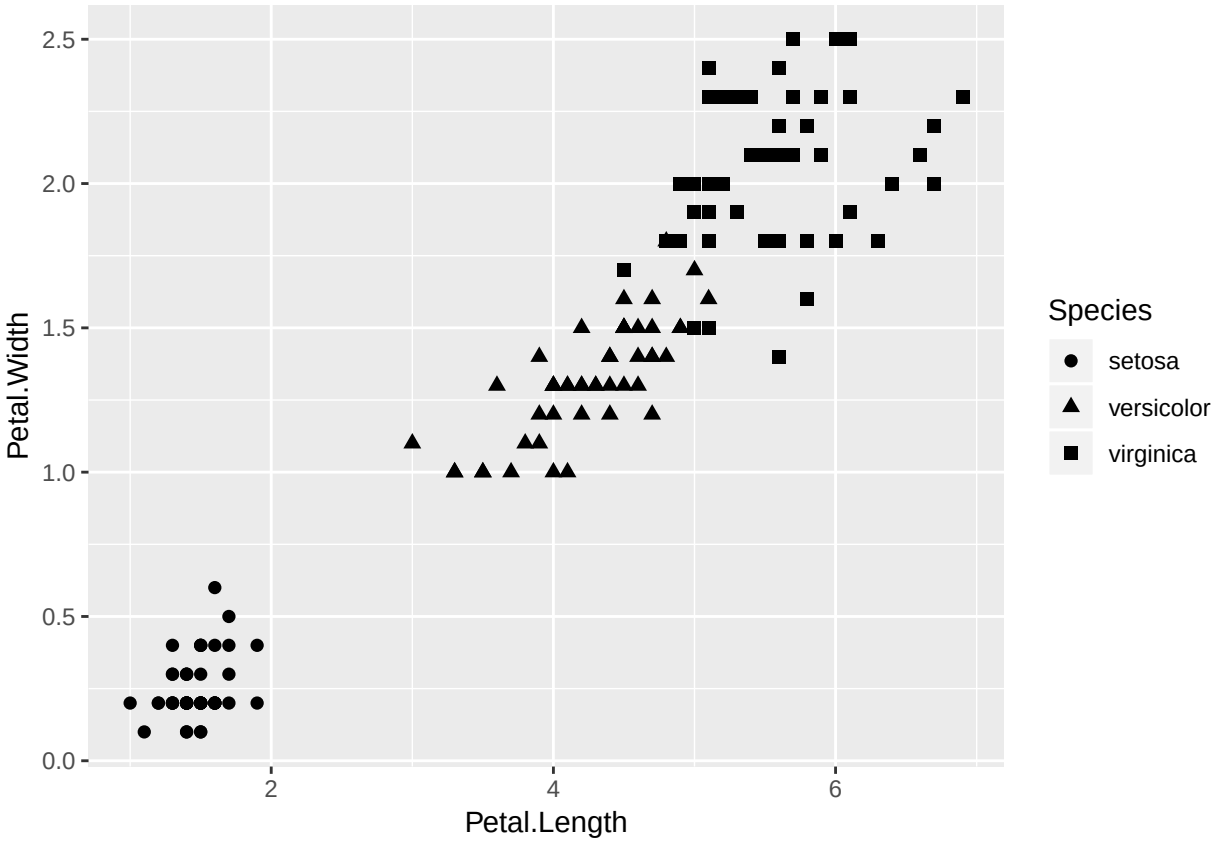
Exercise 17 It can be helpful to identify the class of points in a scatterplot with a shape or color. For example, the scatterplot

```
library(ggplot2)
ggplot(data=iris,aes(x=Petal.Length,y=Petal.Width,color=Species)) +
  geom_point()
```



is very similar to one we saw in the previous exercise, but coloring the points by the *Species* variable reveals a wealth of information. We can also change the shape of points as in

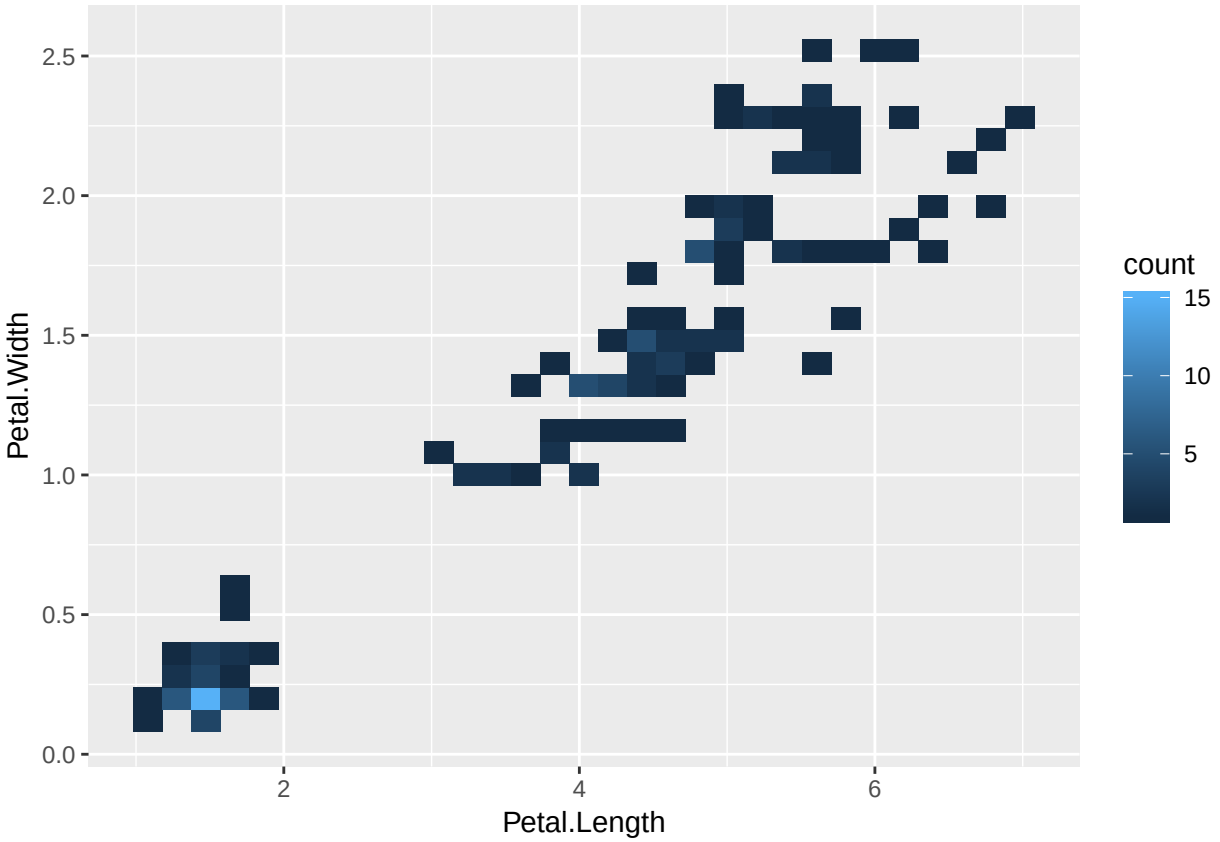
```
library(ggplot2)
ggplot(data=iris,aes(x=Petal.Length,y=Petal.Width,shape=Species)) +
  geom_point(size=2)
```



Create a scatterplot for the *Sepal.Width* (y variable) versus the *Sepal.Length* (x variable) in the *iris* dataset, and color the points by the *Species* variable. Then produce the same scatterplot with shapes differing instead of color. Can you do both?

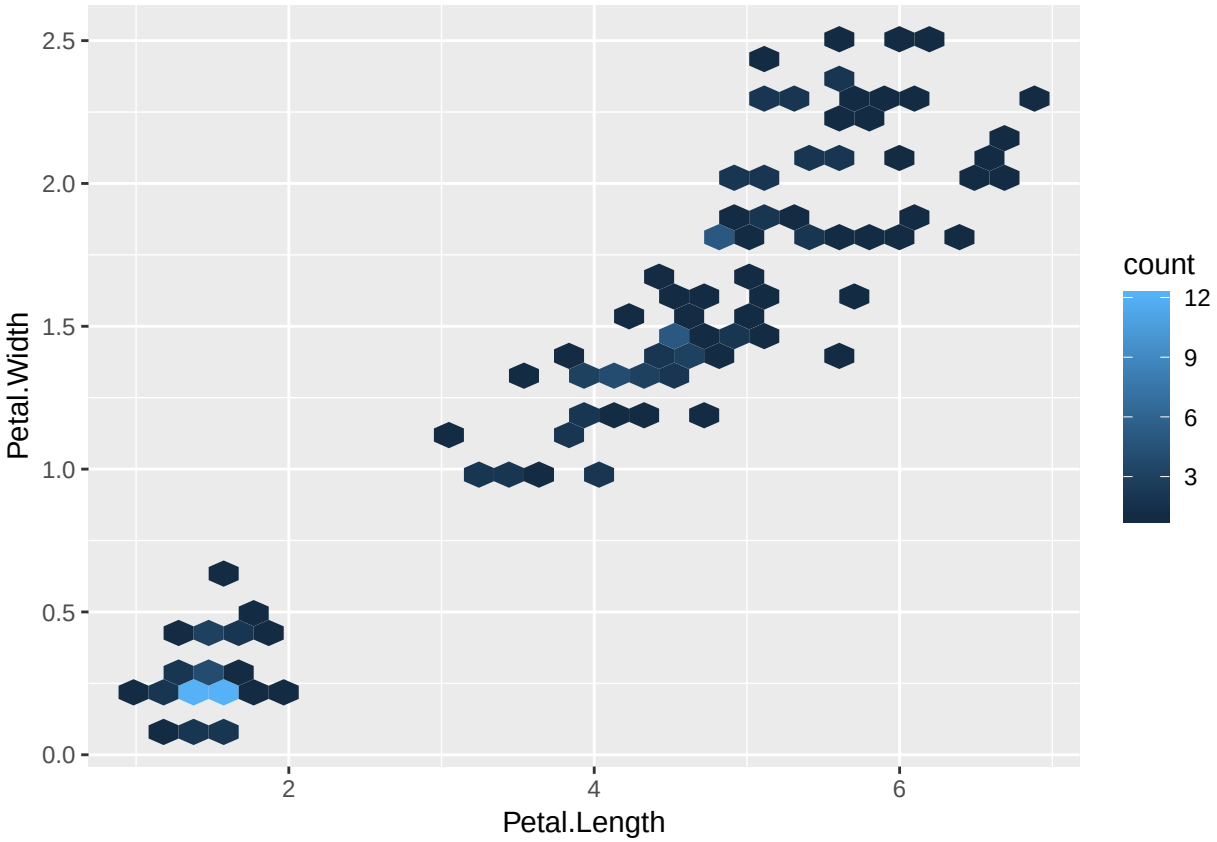
Exercise 18 What is the analog of the histogram for pairs of variables? Well, a 2-dimensional histogram! Because it is difficult to plot a 3-dimensional object on a 2-dimensional page, we represent the "heights" of the bars by colors rather than vertical bars. It can be helpful to use hexagonal bases for the bins, rather than rectangles. For example, with rectangles we have

```
library(ggplot2)
ggplot(data=iris,aes(x=Petal.Length,y=Petal.Width)) +
  stat_bin2d()
```



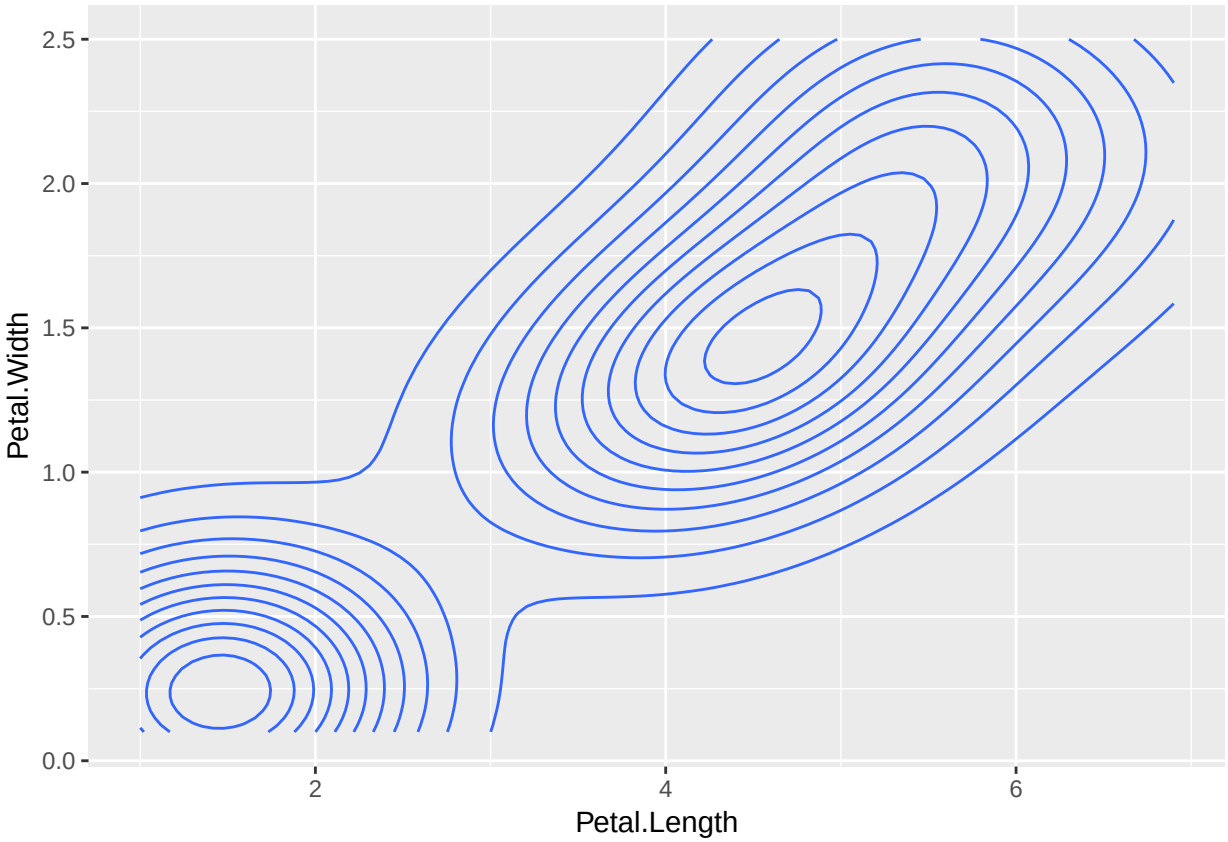
and with hexagons we have

```
library(ggplot2)
ggplot(data=iris,aes(x=Petal.Length,y=Petal.Width)) +
  stat_binhex()
```



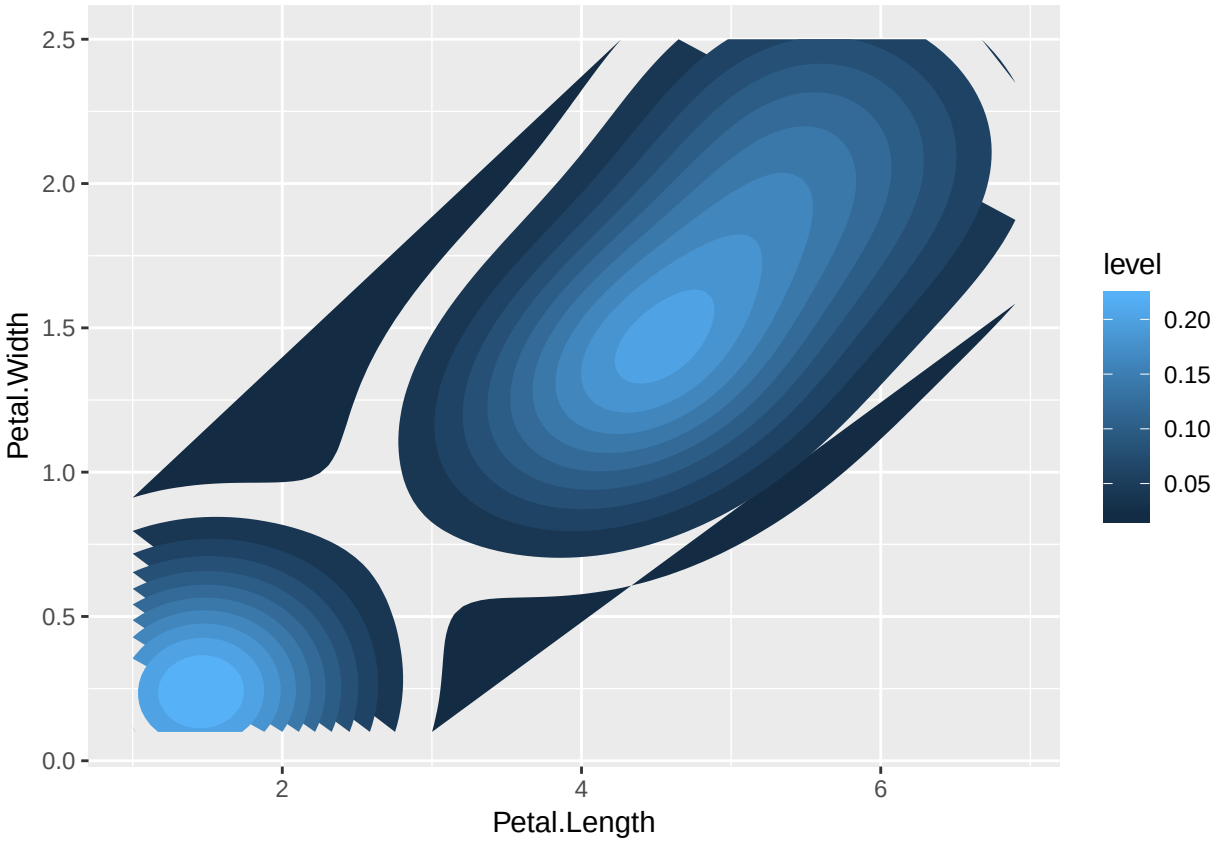
You can alternately create the level curves of the density function,

```
library(ggplot2)
ggplot(data=iris,aes(x=Petal.Length,y=Petal.Width)) +
  stat_density_2d()
```



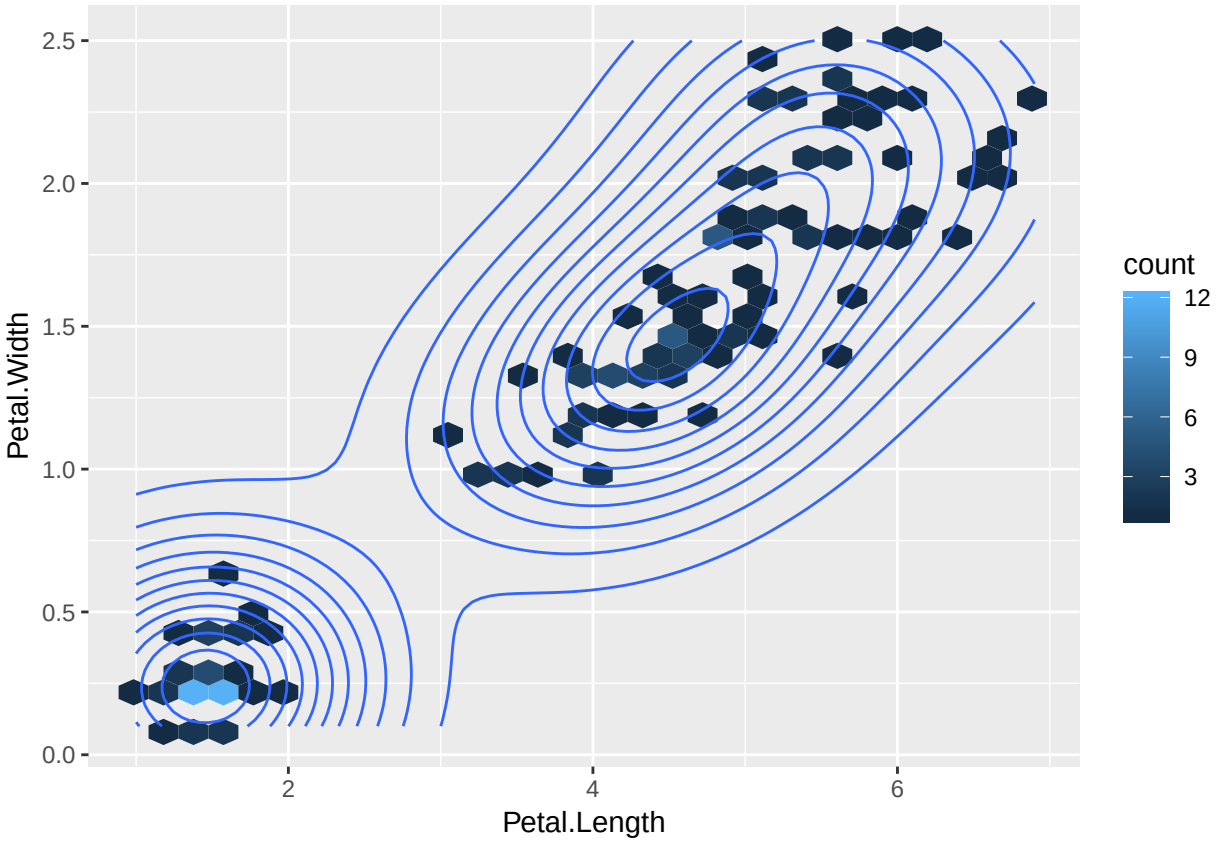
or even fill them in (which can often look funny, as we see below).

```
library(ggplot2)
ggplot(data=iris,aes(x=Petal.Length,y=Petal.Width)) +
  stat_density_2d(aes(fill=stat(level)),geom="polygon")
```



You can combine the histogram and the density plot.

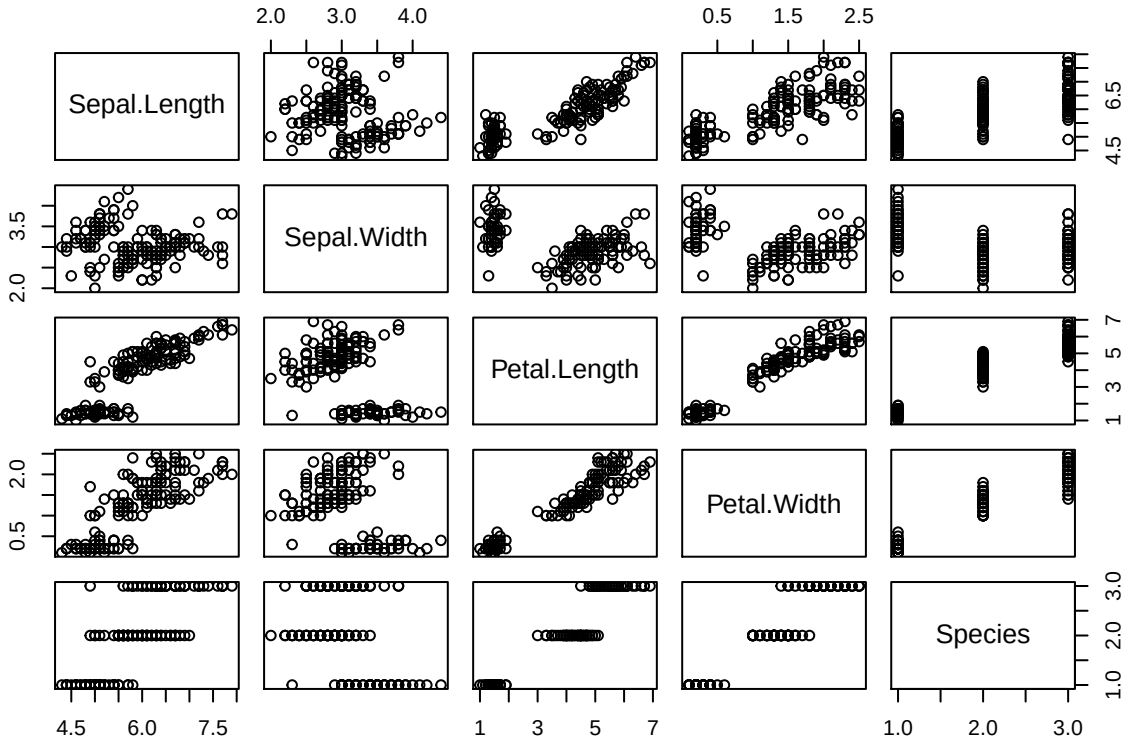
```
library(ggplot2)
ggplot(data=iris,aes(x=Petal.Length,y=Petal.Width)) +
  stat_binhex() +
  stat_density_2d()
```



and you can fill between curves if you like. Produce a 2-dimensional histogram with hexagonal bins and level curves for the density function for the *Sepal.Width* versus *Sepal.Length* variables in the *iris* dataset.

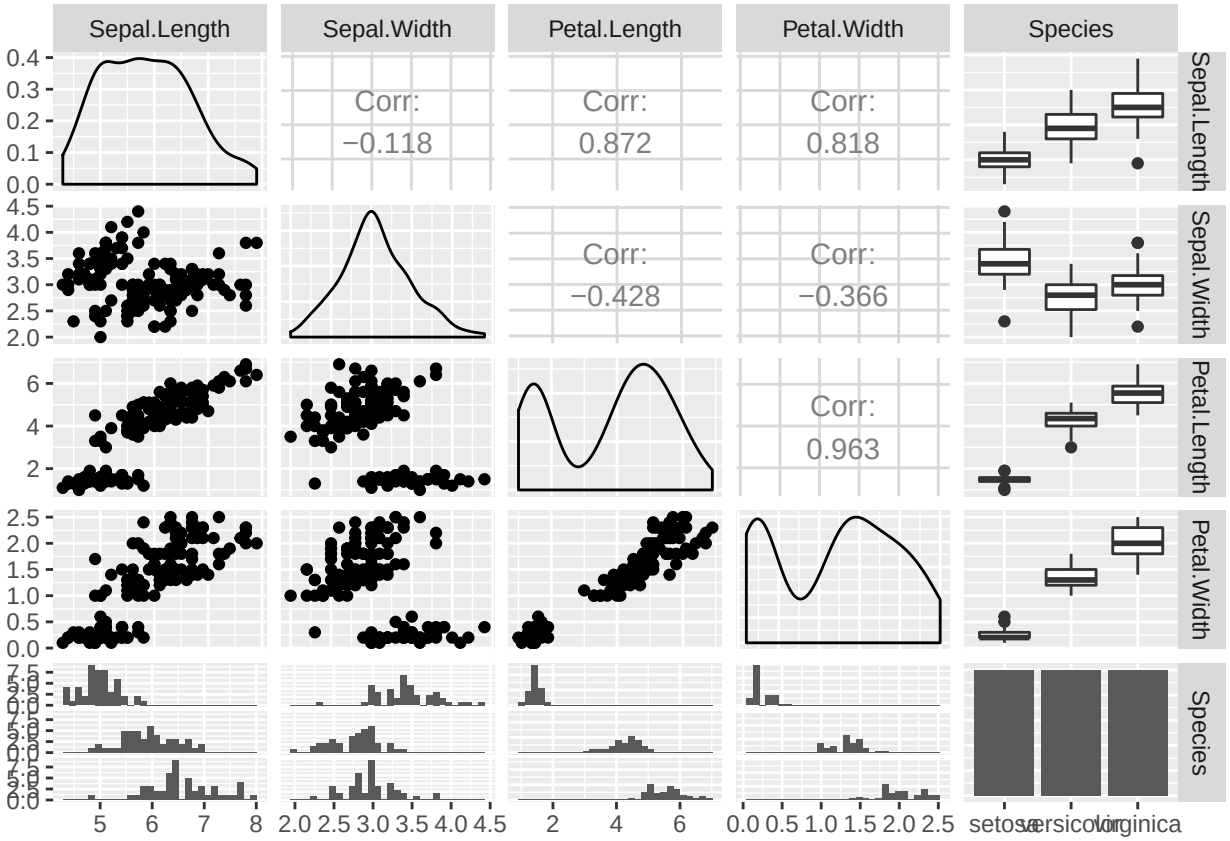
Exercise 19 It may be difficult to explore all pairs of variables in your data. R includes some helper functions to quickly visualize relationships between variables. In base R, you can use the *pairs* function:

```
pairs(iris)
```



In the *GGally* package you can use the *ggpairs* function, which is MUCH more informative. (If you have not installed this package yet, you may want to give it a try!)

```
library(GGally)
ggpairs(iris,progress=FALSE)
```

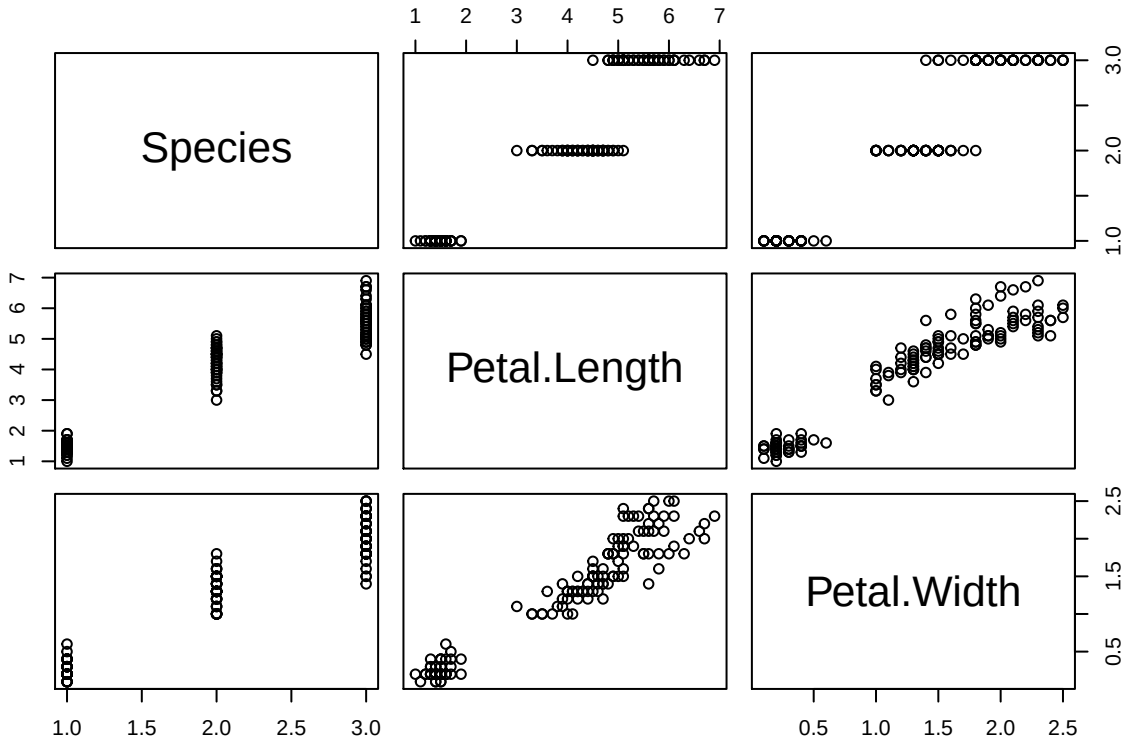


It can be helpful to select only a few variables for comparison. For example,

```
petal <- iris[,c("Species","Petal.Length","Petal.Width")]
```

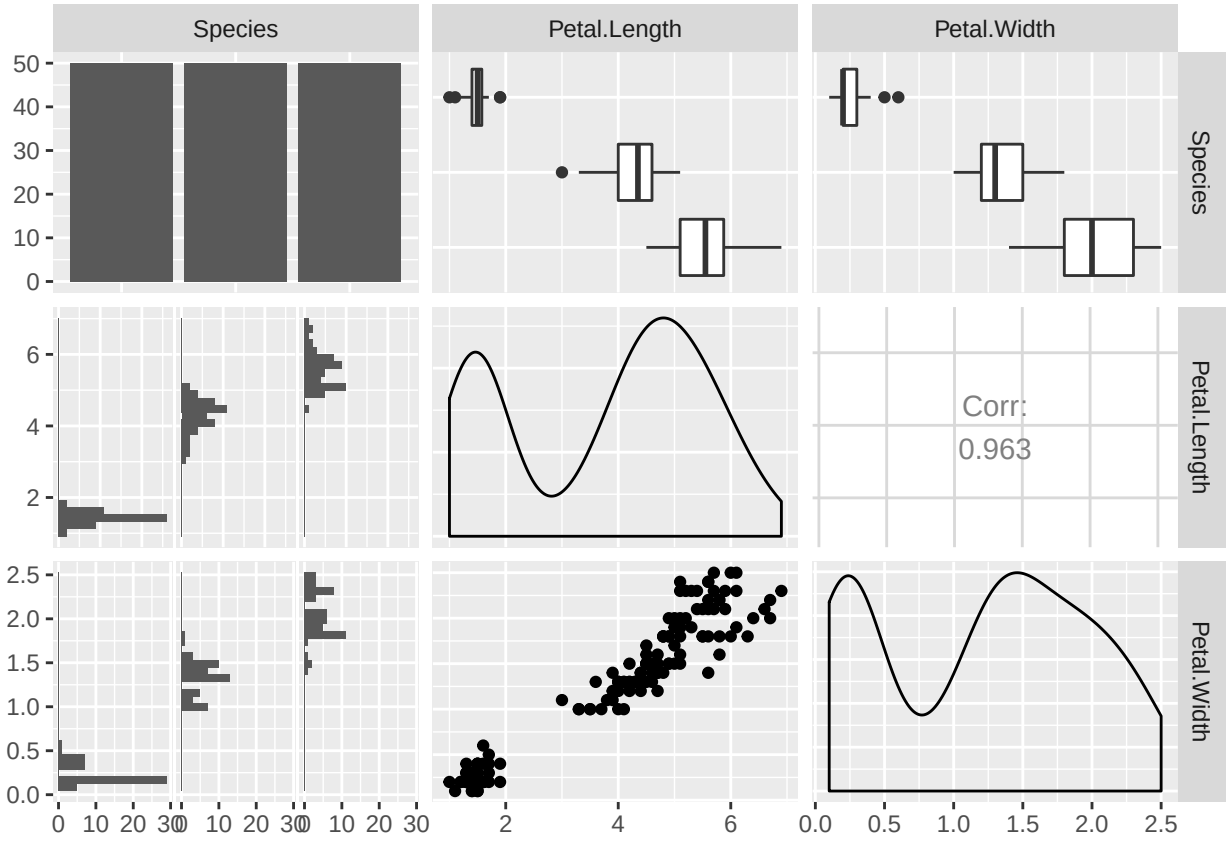
We then have in base R

```
pairs(petal)
```



and with *ggpairs*

```
library(GGally)
ggpairs(petal,progress=FALSE)
```



Try using both *pairs* and *ggpairs* on the sepal length and sepal width variables (include the species variable as well).

Applications

An exploratory analysis of a data set should have the following components:

- An explanation of, or link to an explanation of, the meanings of all the variables in the dataset.
- Some tabular and graphical summaries of the variables and their relationships in the dataset.
- Some tentative conclusions.
- Some follow-up questions. These questions will be addressed in further experiments, either physical or statistical.

Exercise 20 Use the concepts in exercises 1 through 19 to produce an exploratory analysis of the *ToothGrowth* dataset.

Exercise 21 Use the concepts in exercises 1 through 19 to produce an exploratory analysis of the *mtcars* dataset.