Introduction to Data

```
library(fastR2)
```

All of the plots in this notebook make use of packages contained in the fastR2 library.

Introduction

Data are the raw material of statistics and machine learning. This notebook provides an introduction to working with data in R. Most often, we will organize data into a 2-dimensional array (a generalization of a matrix), which we can think of as rows and columns in a spreadsheet. We will call this **rectanglular data.** Furthermore, we organize the data so that each row of our data table corresponds to an observation and each column corresponds to a variable. In R, rectangular data is represented as a dataframe. Consider for example the iris dataset in R, the first few rows of which are shown with the head command:

```
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
```

Notice that each row is an observation for a sinlge flower where four different measurements and the flower species are recorded. Each column corresponds to a different variable.

The dim command will tell us the total number of rows and columns of the data:

```
dim(iris)
```

```
## [1] 150   5
```

Thus there are 150 rows (observations) and 5 columns (variables).

The names of the variables are stored as the names of the columns of the dataframe which can be accessed as follows:

```
names(iris)
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

You can access individual columns of the dataframe with the dollar sign operator, for example, to acces the Species column one does:

```
iris$Species
```

```
##   [1] setosa     setosa     setosa     setosa     setosa     setosa
##   [7] setosa     setosa     setosa     setosa     setosa     setosa
##  [13] setosa     setosa     setosa     setosa     setosa     setosa
##  [19] setosa     setosa     setosa     setosa     setosa     setosa
##  [25] setosa     setosa     setosa     setosa     setosa     setosa
##  [31] setosa     setosa     setosa     setosa     setosa     setosa
```

```
##  [37] setosa     setosa     setosa     setosa     setosa     setosa
##  [43] setosa     setosa     setosa     setosa     setosa     setosa
##  [49] setosa     setosa     versicolor versicolor versicolor versicolor
##  [55] versicolor versicolor versicolor versicolor versicolor versicolor
##  [61] versicolor versicolor versicolor versicolor versicolor versicolor
##  [67] versicolor versicolor versicolor versicolor versicolor versicolor
##  [73] versicolor versicolor versicolor versicolor versicolor versicolor
##  [79] versicolor versicolor versicolor versicolor versicolor versicolor
##  [85] versicolor versicolor versicolor versicolor versicolor versicolor
##  [91] versicolor versicolor versicolor versicolor versicolor versicolor
##  [97] versicolor versicolor versicolor versicolor virginica  virginica
## [103] virginica  virginica  virginica  virginica  virginica  virginica
## [109] virginica  virginica  virginica  virginica  virginica  virginica
## [115] virginica  virginica  virginica  virginica  virginica  virginica
## [121] virginica  virginica  virginica  virginica  virginica  virginica
## [127] virginica  virginica  virginica  virginica  virginica  virginica
## [133] virginica  virginica  virginica  virginica  virginica  virginica
## [139] virginica  virginica  virginica  virginica  virginica  virginica
## [145] virginica  virginica  virginica  virginica  virginica  virginica
## Levels: setosa versicolor virginica
```

The summary function we return summary statistics for all of the variables in a dataframe:

```
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
##
##
##
```

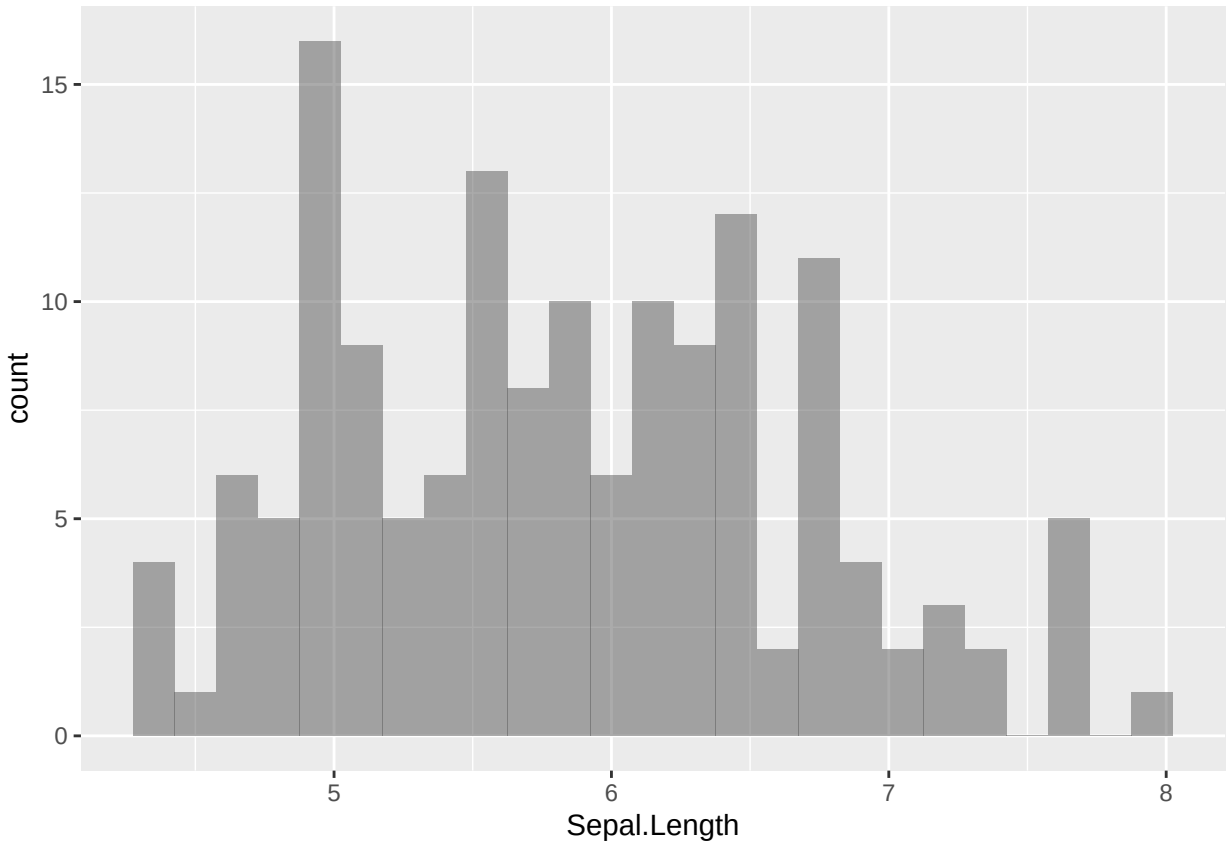
Notice that the measurement variables Sepal.Length, Sepal.Width, Petal.Length, Petal.Width are all numeric while the variable is non-numeric. We call numeric variables like the measurement variables quantitative and non-numeric variables like the Species variable **categorical**. Observe that summary returns counts for a categorical variable like Species but numerical statistics for quantitative varviables like Sepal.Length, Sepal.Width, Petal.Length, Petal.Width.

Graphical Summaries

Numerical summaries such as those returned by summary are useful. However, it is often far better to have a graphical summary of data. We will now cover some of the most common ways to summarise data graphically.

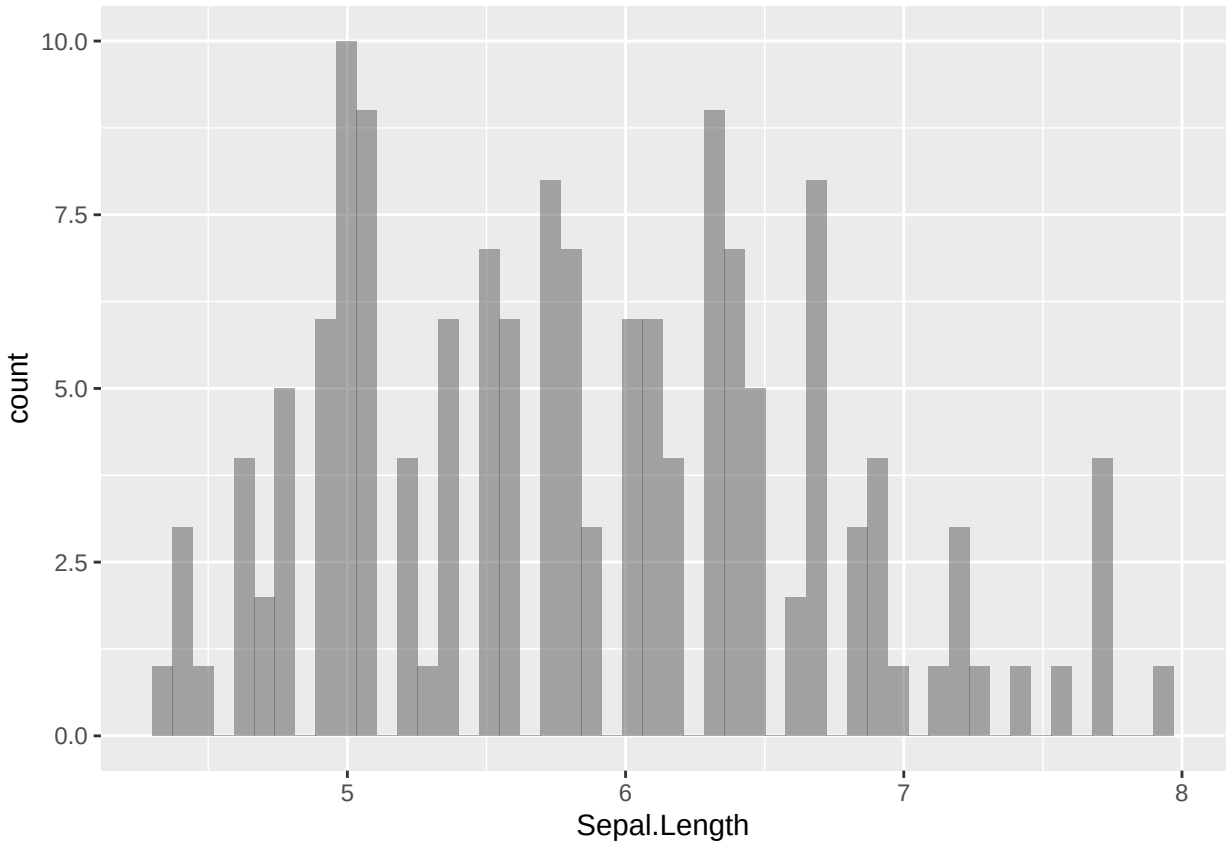
For a numeric variable such as with values from a continuous range of real numbers two common graphical summaries are a histogram and a boxplot. We can obtain a histogram as follows:

```
iris %>% gf_histogram(~Sepal.Length)
```



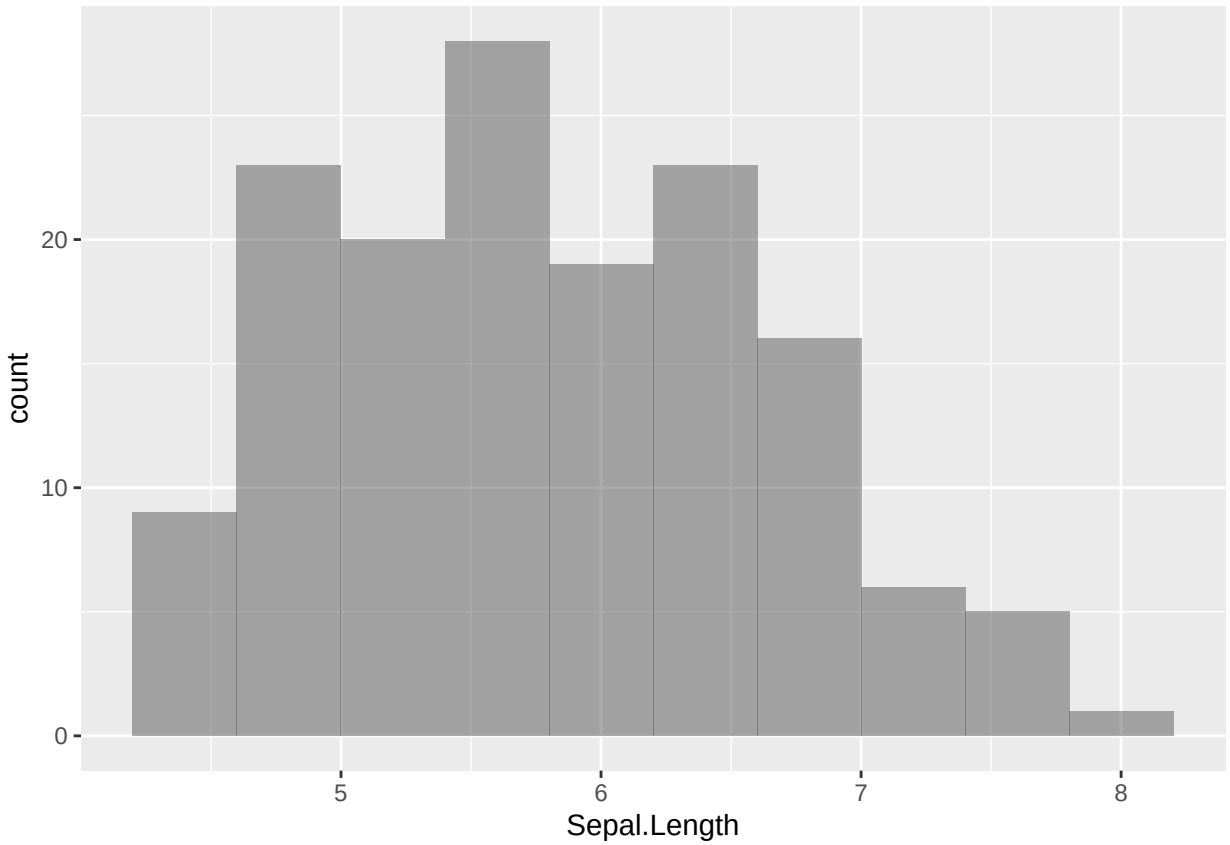
A histogram creates "bins" which are a sequence of subintervals of the range of the data. Then, a histogram displays the number of observations from the data that fall within each bin. Histograms provide some idea about the distribution of quantitative data. The size or number of bins is a choice (the deafult in R is about 30). For example, if we increase the number of bins we could get someting like this:

```
iris %>% gf_histogram(~Sepal.Length,bins = 50)
```



whereas if we decrease the number of bins we could get something like this:

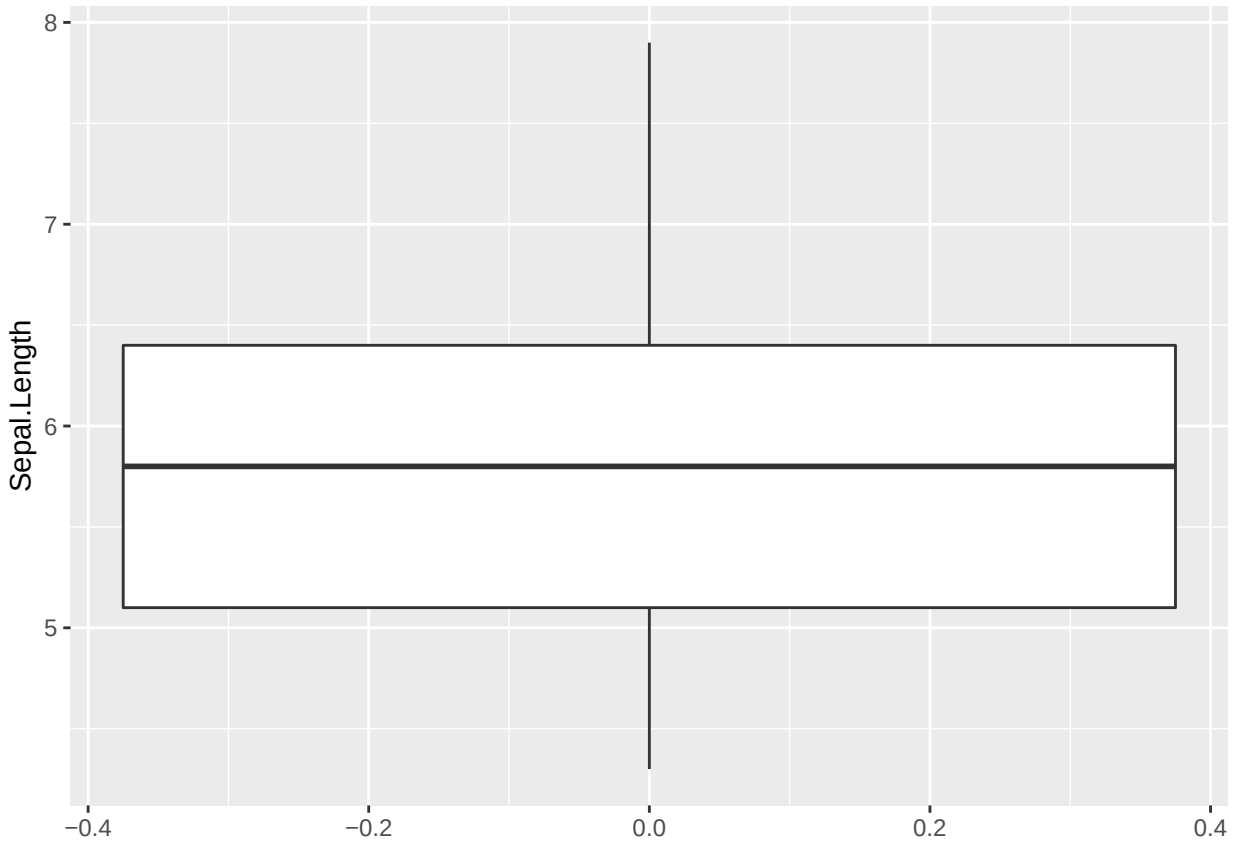
```
iris %>% gf_histogram(~Sepal.Length,bins=10)
```



When you plot a histogram, it is a good idea to try out several different values for the number of bins to get an idea of how changing the number of bins affects the appearance of the histogram.

Another way to get a sense of the distribution of our data is with a boxplot, for example:

```
iris %>% gf_boxplot(~Sepal.Length)
```



This plot shows the minimum value, the maximum value, the range of the middle 50% of the data (that is the box), and the median value (solid line in the middle of the box). In other words, a boxplot displays the so-called five number summary of quantitative data. These numerical values can be obtained as follows:

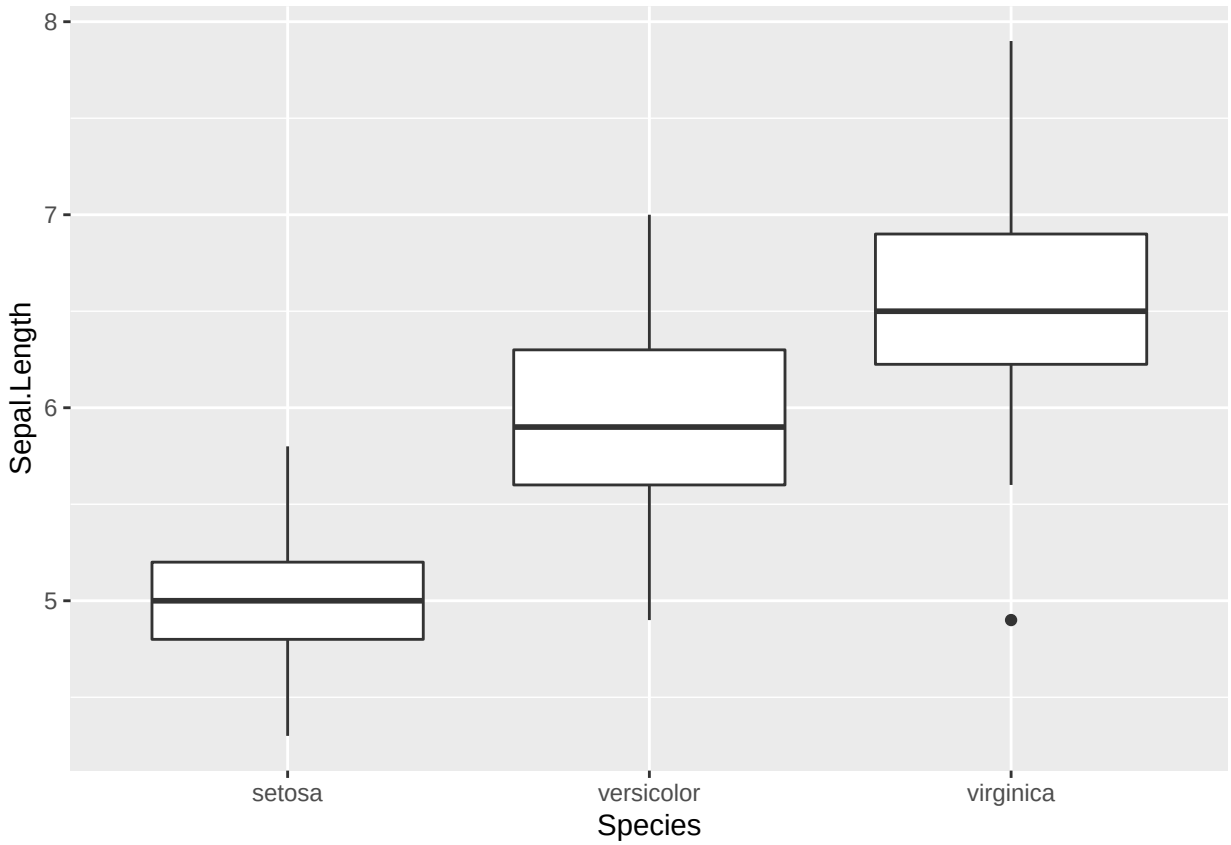
```
fivenum(iris$Sepal.Length)
```

```
## [1] 4.3 5.1 5.8 6.4 7.9
```

A boxplot will also indicate any outliers although there are none in this example.

Since the measurements of our data are taken for flowers from different species, it might be useful to see how the distribution of a measurement compares across species. This can be done as follows:

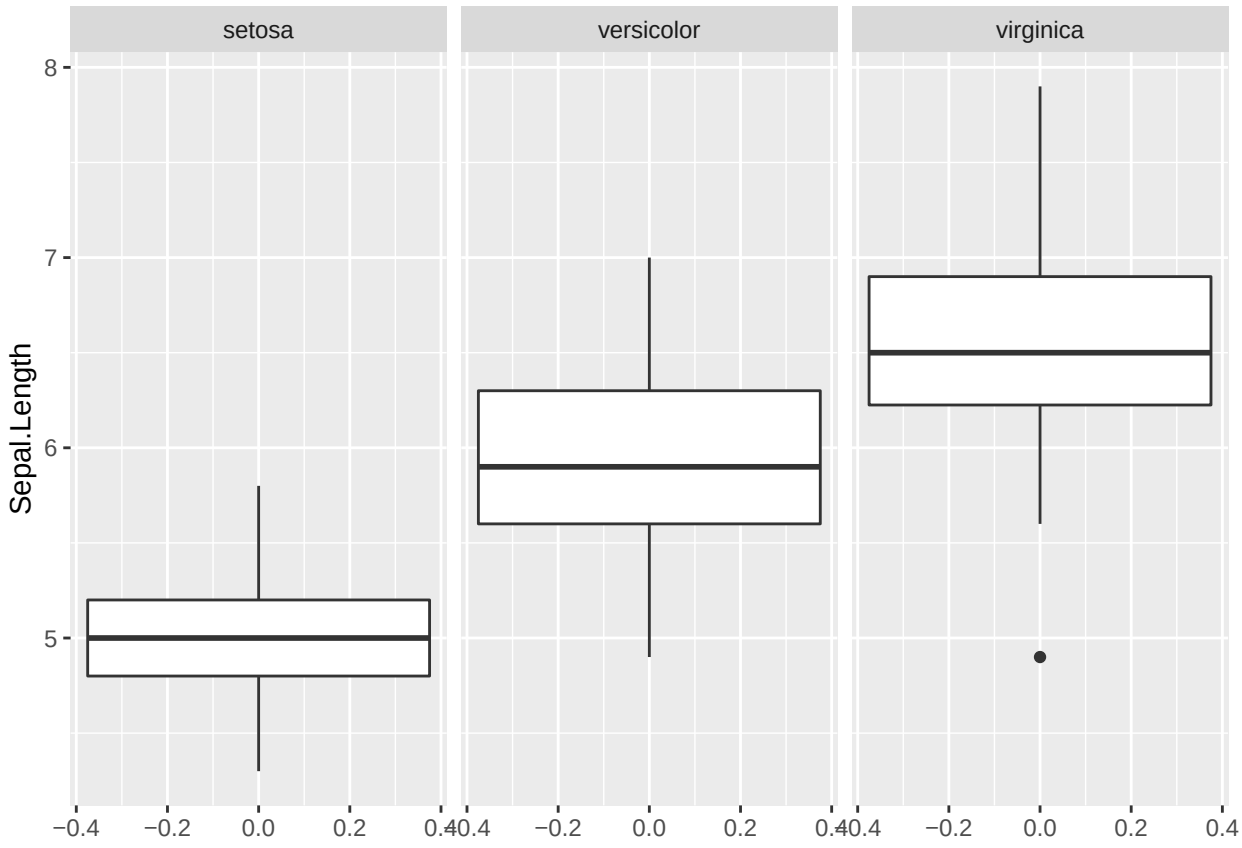
```
iris %>% gf_boxplot(Sepal.Length ~ Species)
```



This plot suggests that there is a different distribution of sepal length values for flowers of species setosa than for flowers of species versicolor or virginica.

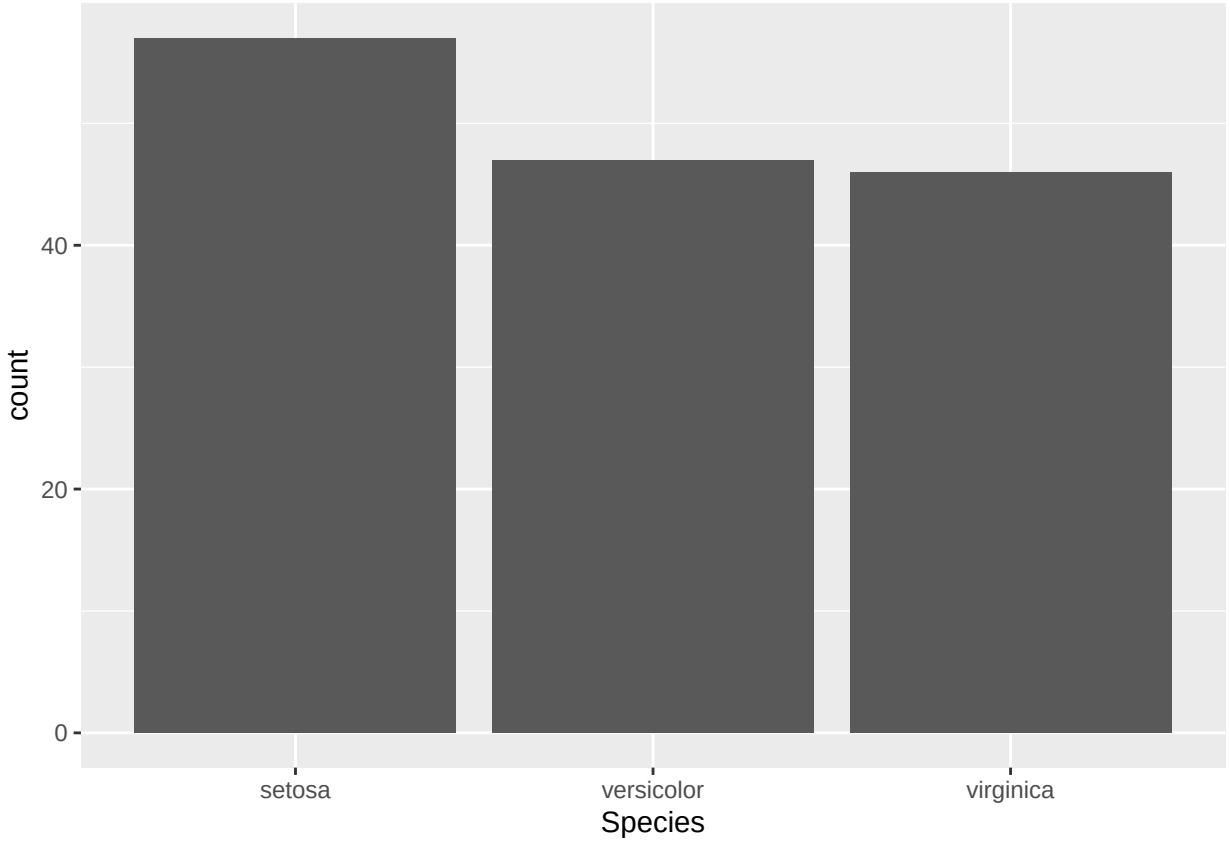
Another way to obtain a similar plot is as follows:

```
iris %>% gf_boxplot(~Sepal.Length) %>% gf_facet_wrap(~Species)
```



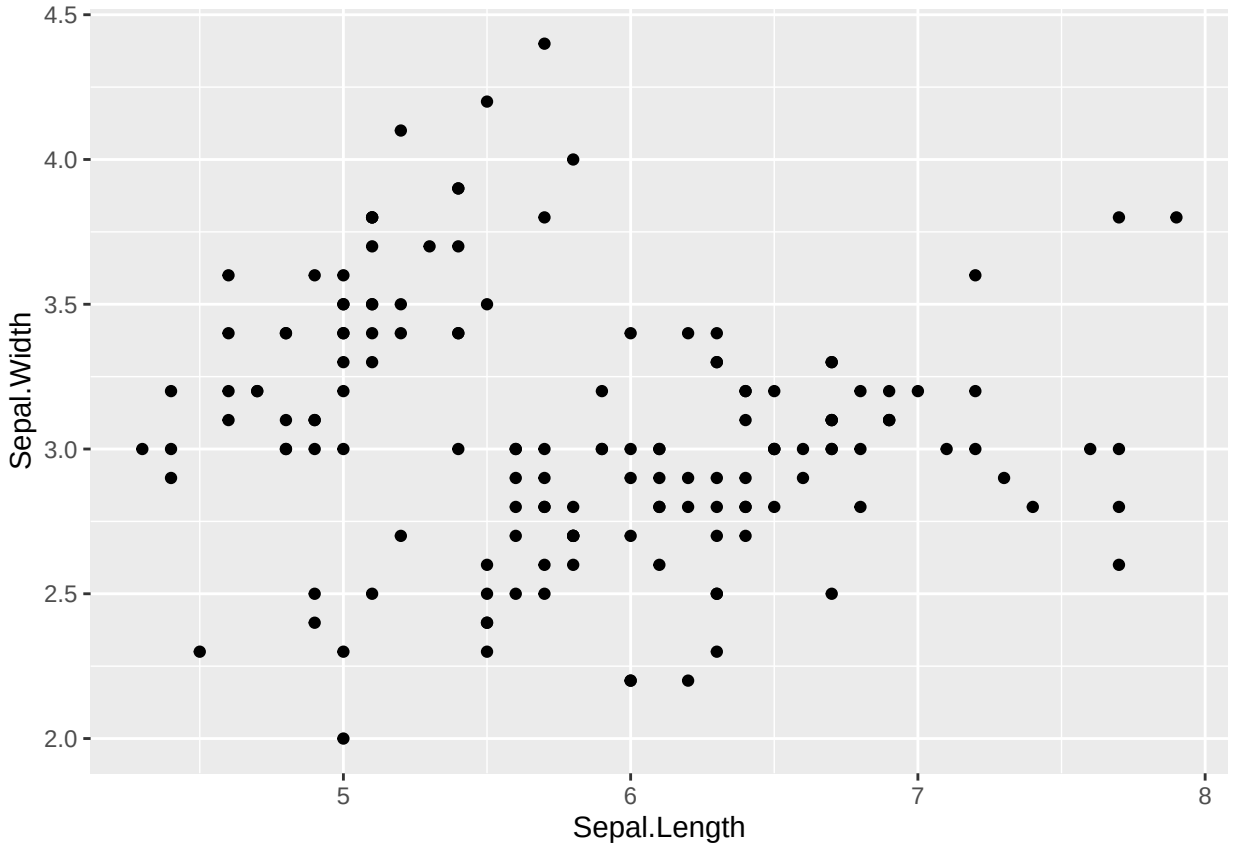
How do we obtain a graphical summary for a categorical variable like species? A common approach is the use a bar plot. We have already seen that in the iris dataset there are 50 observations from each of the three species. Let's take a random sample of out data, this might lead to different numbers of each species. A bar plot will allow us to detect if this is the case or not.

```
sample_n(iris,150,replace = TRUE) %>% gf_bar(~Species)
```



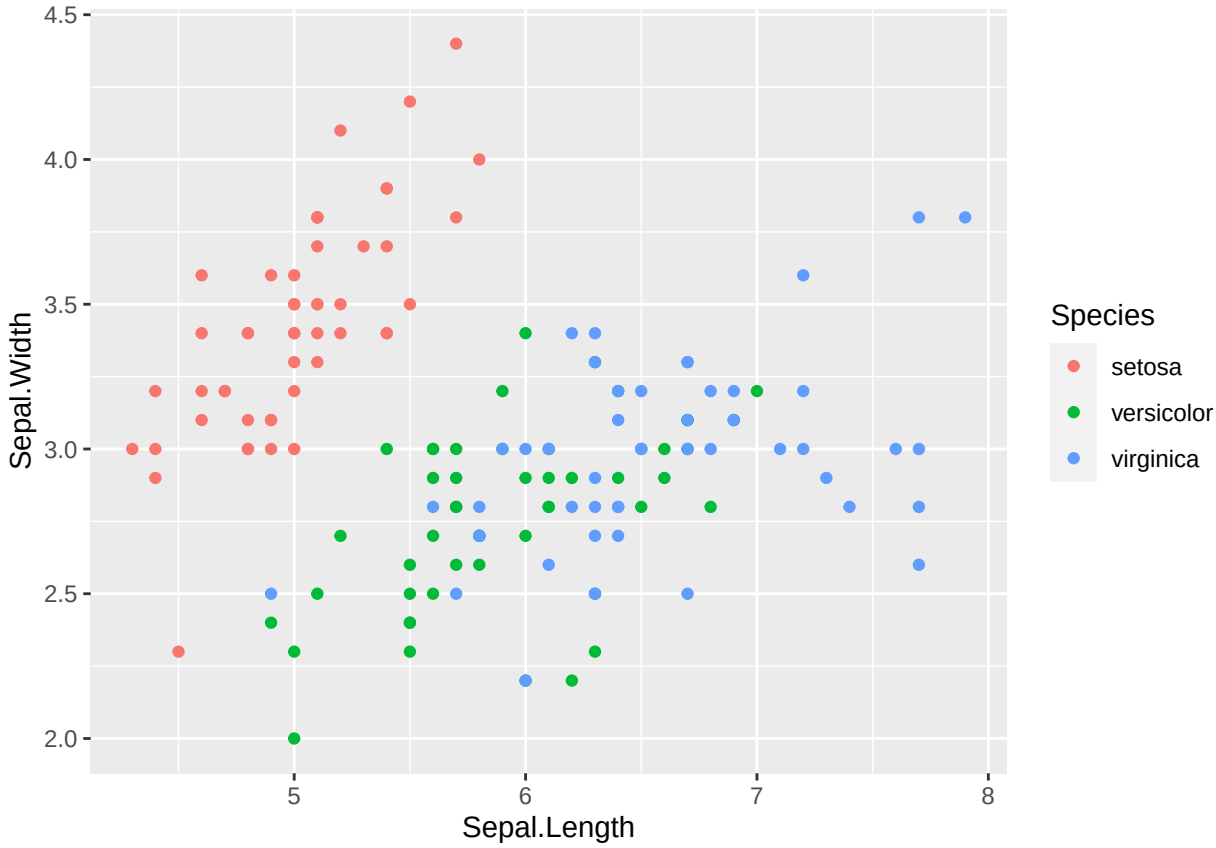
Finally, we might be interested to see how two quantitative variables are potentially related. A scatter plot is a common method used to explore this. For example,

```
iris %>% gf_point(Sepal.Width~Sepal.Length)
```



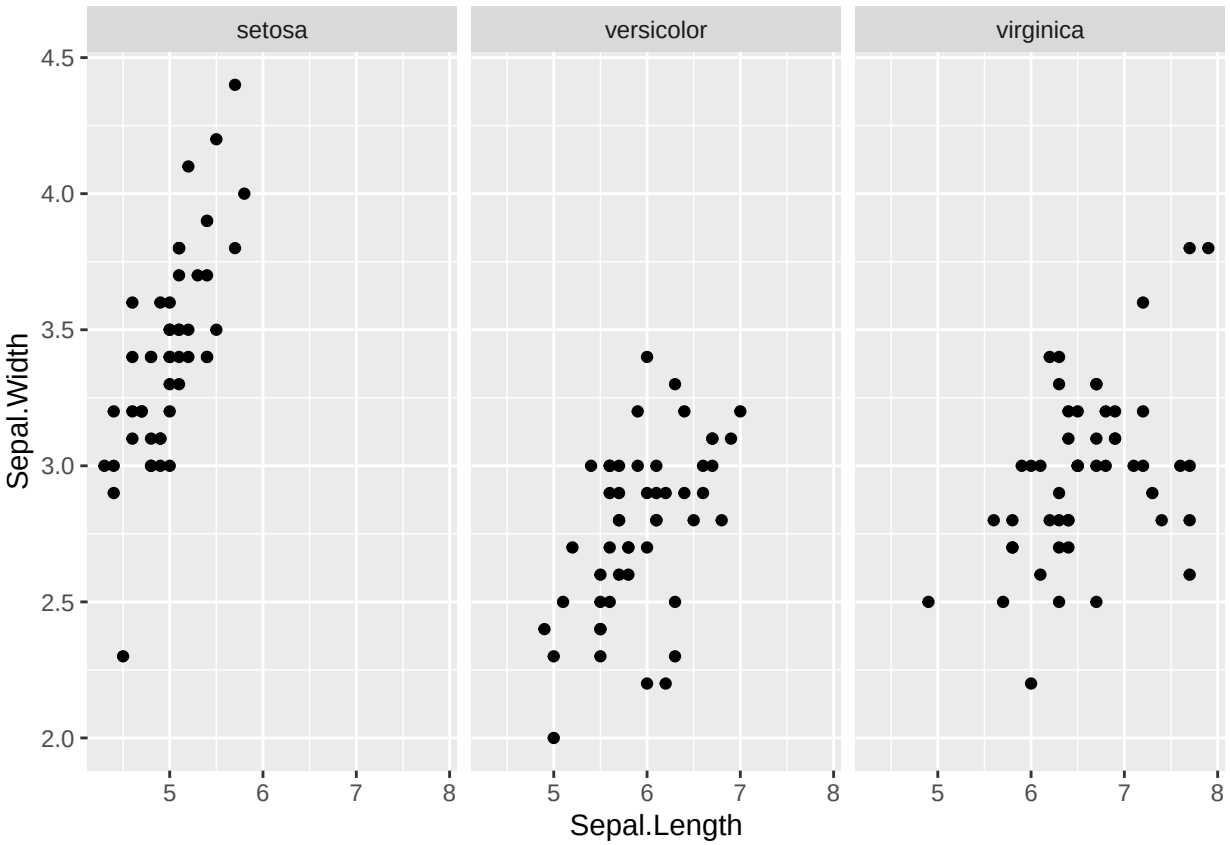
It might be interesting to see how the values shown vary by species. We can do this by adding color with a different color associated to each species:

```
iris %>% gf_point(Sepal.Width~Sepal.Length,color = ~Species)
```



An alternative way to explore the same variation is as follows:

```
iris %>% gf_point(Sepal.Width~Sepal.Length|Species)
```



As we move into our study of statistics, we will make repeated use of plots like the ones described in this notebook.