02-09: Plots 2

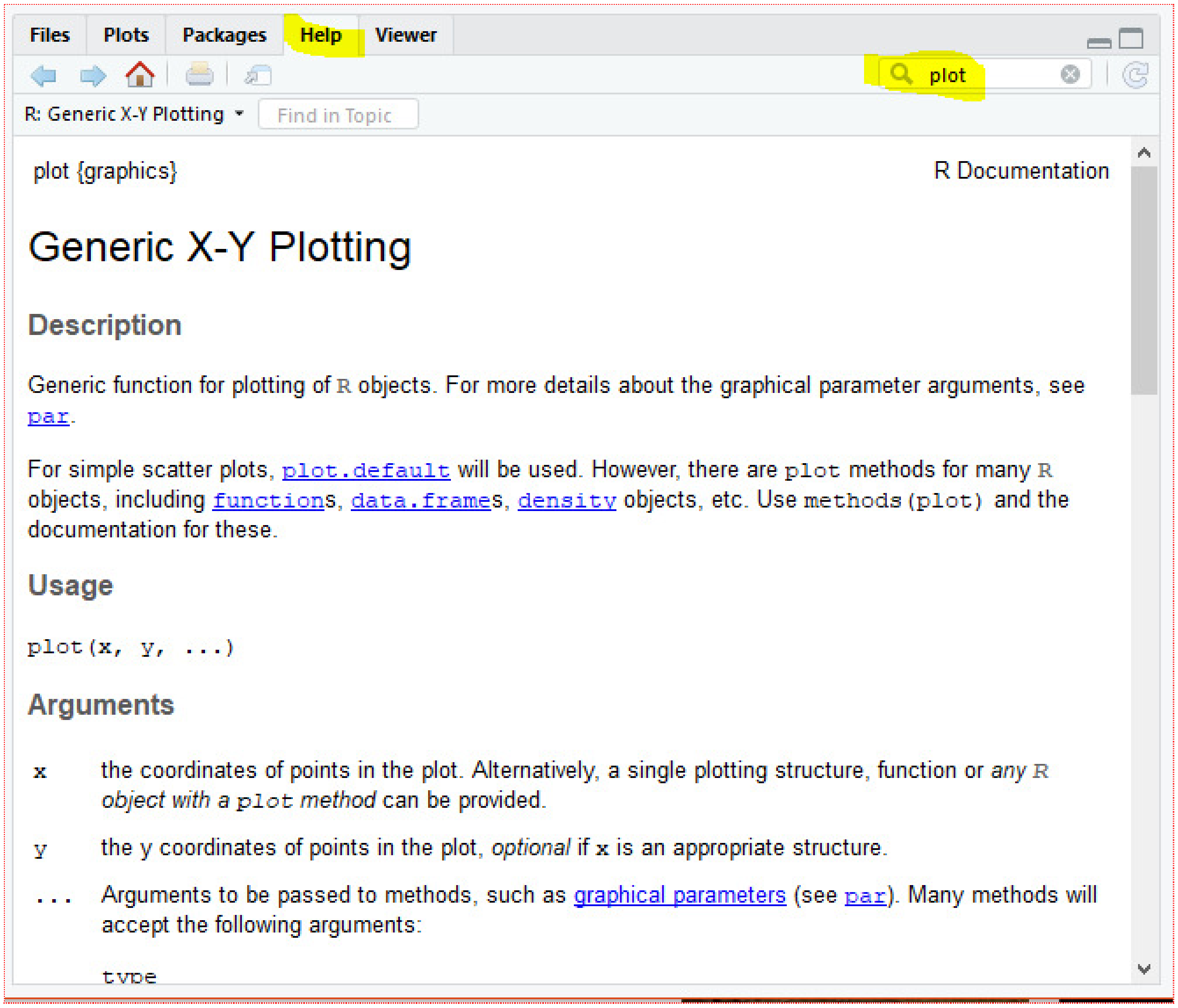
1 - Purpose

- Create a histogram, barplot, and a boxplot
- Understanding and using the parameters in the plot functions
- Style the plots and change axis ranges

2 - Concepts

3 - plot() is a function

plot() is a function somewhere within R. This means that there is some function named ***plot()*** that takes in a bunch of parameters, executes a codeblock, and outputs a plot. Information about the function ***plot()*** can be found here. You can also type "plot" in the search bar of the Help Window (*Fig 1*). The information does not show you the code inside the ***plot()***, just the parameters and an explanation of the purpose of the function and the parameters.



plot {graphics} R Documentation

Generic X-Y Plotting

Description

Generic function for plotting of `R` objects. For more details about the graphical parameter arguments, see `par`.

For simple scatter plots, `plot.default` will be used. However, there are `plot` methods for many `R` objects, including `functions`, `data.frames`, `density` objects, etc. Use `methods(plot)` and the documentation for these.

Usage

```
plot(x, y, ...)
```

Arguments

`x`	the coordinates of points in the plot. Alternatively, a single plotting structure, function or *any* `R` *object with a* `plot` *method* can be provided.
`y`	the y coordinates of points in the plot, *optional* if `x` is an appropriate structure.
`...`	Arguments to be passed to methods, such as graphical parameters (see `par`). Many methods will accept the following arguments:

`type`

Fig 1: Help Window information on ***plot()****.*

Unlike the ***pythagoras()*** function we created in the previous lesson, you do not have to assign a value to every parameters in ***plot()***. Many of the parameters in ***plot()*** have default values -- similar to how Microsoft Word has default settings for font type (Times New Roman) and font size (12) -- but you can change these values.

In this lesson we are going to create three different types of plots: histograms, bar plots, and box plots. We are going to modify some of the parameters in each plot and learn how to add additional lines and points to the plots.

4 - The histogram

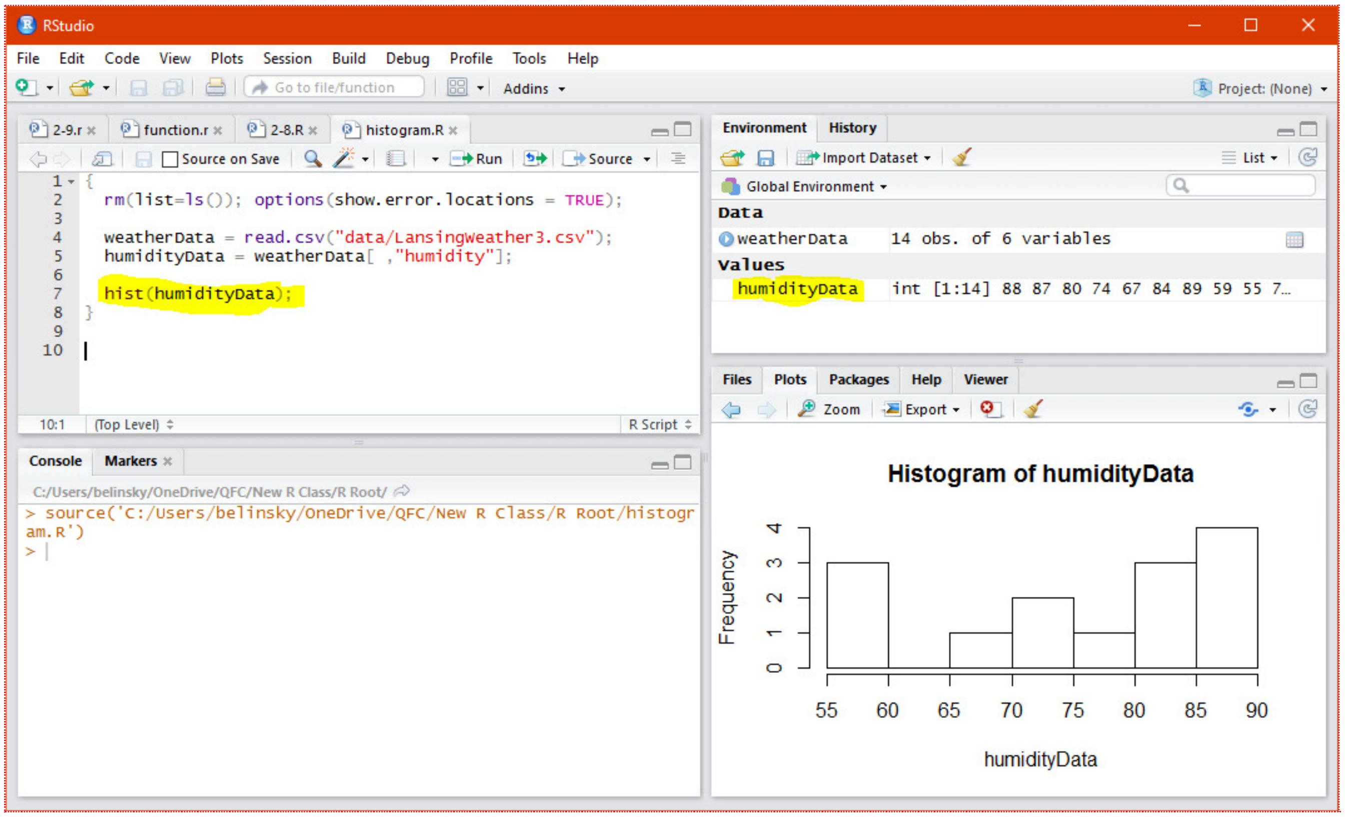
We will start by making a histogram of the humidity values from the 14 days in ***LansingWeather3.csv***. To do this we call the predefined R function ***hist()*** and pass in the humidity vector.

```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather3.csv");
  humidityData = weatherData[ ,"humidity"];

  hist(humidityData);
}
```

A histogram is a plot that shows the distribution of a numeric variable. The values are distributed into equally spaced bins (in the case, of length 5) and the frequency of counts in each bin is plotted. If you add up all of the frequencies, you will see that it adds up to 14 -- for the 14 humidity values.



```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather3.csv");
  humidityData = weatherData[ ,"humidity"];

  hist(humidityData);
}

```

```
10:1   (Top Level)   R Script
```

```
Data
weatherData    14 obs. of 6 variables
Values
humidityData   int [1:14] 88 87 80 74 67 84 89 59 55 7…
```

```
Console  Markers
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/histogr
am.R')
> 
```

Fig 2: Plotting as histogram of the 14 humidity values

4.1 - Modifying the histogram with function parameters

So we have a pretty basic histogram telling us that humidity ranged from 55-90%, and there was high humidity (> 80%) on 7 of the days, which explains why we had so much rain during that time.

As in the previous plotting lesson, there are numerous modifications we can make to the histogram. The function hist(), along with the parameters, is described at this page. Or, you can type "hist" in the Help Window.

The ***hist()*** function looks like this (copied from the link above):

```
hist(x, breaks = "Sturges",
     freq = NULL, probability = !freq,
     include.lowest = TRUE, right = TRUE,
         density = NULL, angle = 45, col = NULL,
         border = NULL,  main = paste("Histogram of" , xname),
         xlim = range(breaks), ylim = NULL,
         xlab = xname, ylab,
         axes = TRUE, plot = TRUE, labels = FALSE,
         nclass = NULL, warn.unused = TRUE, ...)
```

As you can see, there are many parameters that can be set and the (***...***) indicates there are many more that are not given here. In the link above there is a description of what the different parameters do. The majority of the parameters already have assigned (default) values -- all except **x,** which is assigned the data.

So:

```
hist(x=humidityData);
```

assigns the values of ***humidityData*** to ***x***

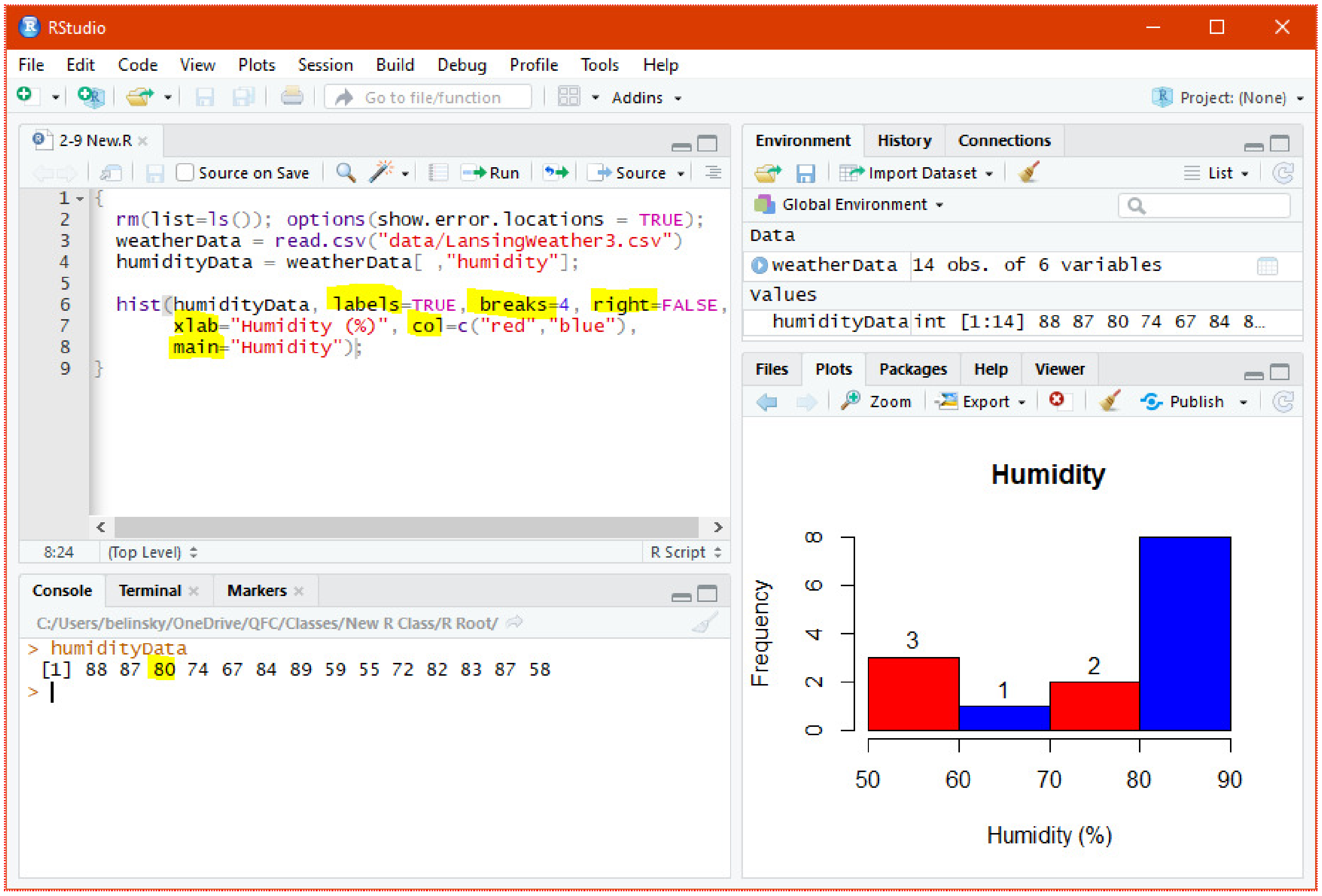
Extension: The x value in hist().

4.2 - Assigning values to the parameters that have default values

Now we will assign value to a few of the other parameters -- the parameters highlighted above.

```
{
  rm(list=ls()); options(show.error.locations = TRUE);
  weatherData = read.csv("data/LansingWeather3.csv")
  humidityData = weatherData[ ,"humidity"];

  hist(humidityData, labels=TRUE, breaks=4, right=FALSE,
       xlab="Humidity (%)", col=c("red","blue"),
       main="Humidity");
}
```



```
{
  rm(list=ls()); options(show.error.locations = TRUE);
  weatherData = read.csv("data/LansingWeather3.csv")
  humidityData = weatherData[ ,"humidity"];

  hist(humidityData, labels=TRUE, breaks=4, right=FALSE,
       xlab="Humidity (%)", col=c("red","blue"),
       main="Humidity");
}
```

```
Data
weatherData    14 obs. of 6 variables
Values
humidityData   int [1:14] 88 87 80 74 67 84 8…
```

```
C:/Users/belinsky/OneDrive/QFC/Classes/New R Class/R Root/
> humidityData
 [1] 88 87 80 74 67 84 89 59 55 72 82 83 87 58
> 
```

Fig 3: Changing some of the parameters in the ***hist()*** *function*

Let's look at what these parameters did and I encourage you to play with these values.

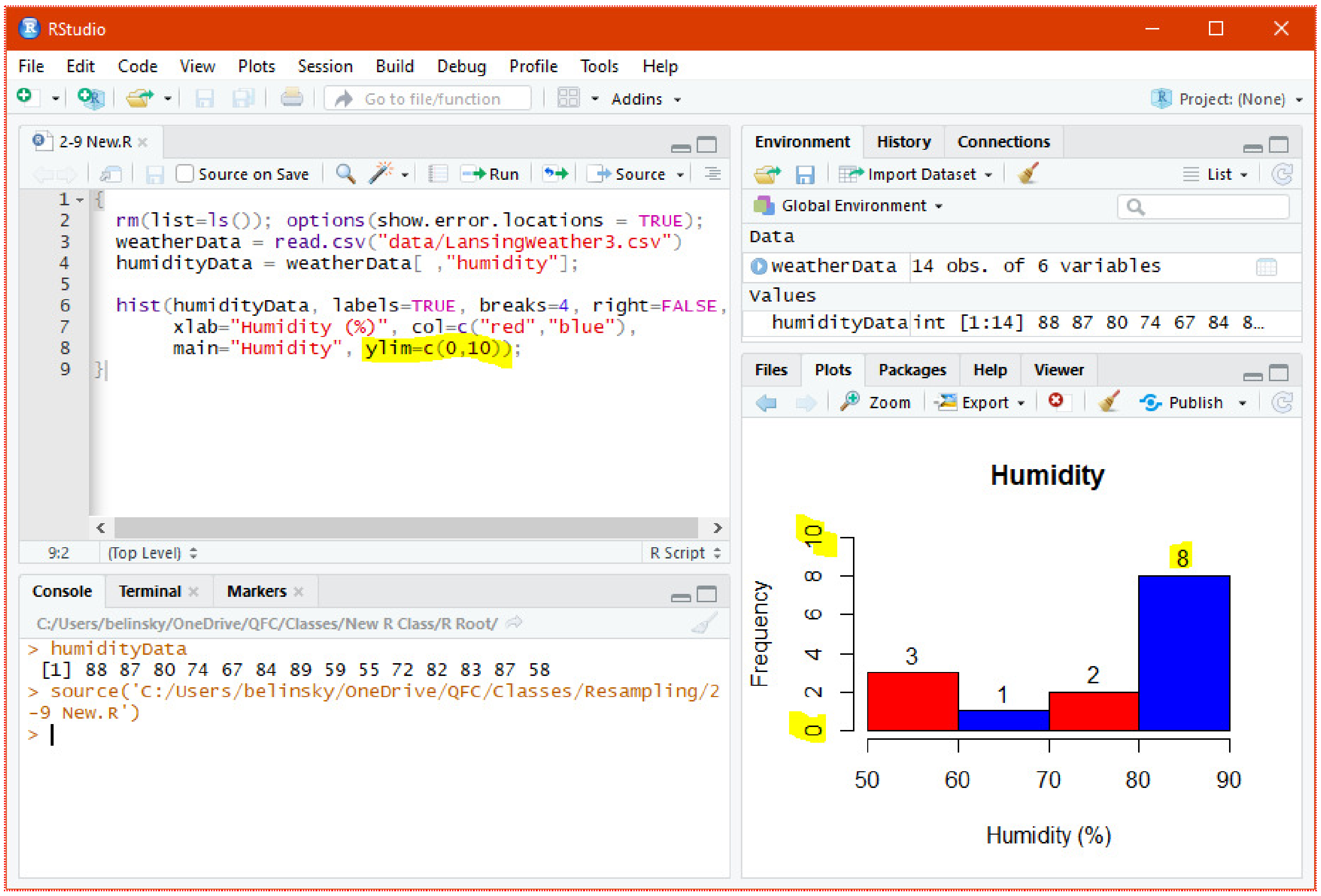
- ***labels = TRUE*** put the frequency numbers on top of the bars
- ***breaks = 4*** sets the number of bins (bars) in the plot. Note: the number assigned to ***breaks*** is more of a suggestion and is often not the exact number that R will use -- try out different numbers.
- ***right*** tell ***hist()*** how to handle values right on the border of two bins. There is a humidity value of 80, which is on the border of the last 2 bins
  - ***right = TRUE*** (default) assigns the border value, 80, to the right bin
  - ***right = FALSE*** assigns the border value, 80, to the left bin.
- ***xlab*** *sets* the label on the x-axis
- ***main*** sets the title of the plot
- ***col*** sets the color of the bars. You can put in multiple colors and R will recycle the colors-- so, in this case, the bin colors will go **red-blue-red-blue-red-blue**...

4.3 - Fixing the y-axis range

The last plot has a small issue with it. The last frequency bar has no label on it. Actually, it has a label but it is cut off. We need to increase the range of the y-axis to include the label. The property to change is ***ylim***. ***ylim*** take a vector with two values: the low and high value of the axis.

```
{
  rm(list=ls()); options(show.error.locations = TRUE);
  weatherData = read.csv("data/LansingWeather3.csv")
  humidityData = weatherData[ ,"humidity"];
hist(humidityData, labels=TRUE, breaks=4, right=FALSE,
     xlab="Humidity (%)", col=c("red","blue"),
     main="Humidity", ylim =c(0,10) );
}
```

So we have increased the high end of the y-axis from **8** to **10** and the last frequency label now appears.



```
{
  rm(list=ls()); options(show.error.locations = TRUE);
  weatherData = read.csv("data/Lansingweather3.csv")
  humidityData = weatherData[ ,"humidity"];

  hist(humidityData, labels=TRUE, breaks=4, right=FALSE,
       xlab="Humidity (%)", col=c("red","blue"),
       main="Humidity", ylim=c(0,10));
}
```

```
C:/Users/belinsky/OneDrive/QFC/Classes/New R Class/R Root/
> humidityData
 [1] 88 87 80 74 67 84 89 59 55 72 82 83 87 58
> source('C:/Users/belinsky/OneDrive/QFC/Classes/Resampling/2
-9 New.R')
> 
```

```
Data
weatherData    14 obs. of 6 variables
Values
  humidityData int [1:14] 88 87 80 74 67 84 8…
```

Fig 4: Changing the limits of the y-axis so the frequency label does not get cut off.

4.4 - Adding a line to the histogram

Next, we will add a vertical line to represent the mean humidity. First we need to find the mean of the humidity values using the ***mean()*** function. Then we call the function ***abline()*** to add a line at the mean value to the histogram. Information about abline() can be found here.

abline() is a function that adds lines to a plot. We will assign values to three parameters:

- ***v***: the vertical line variables -- we set it to ***meanHumidity*** (approximately 76)
- ***col***: the color of the line (darkgreen)
- ***lty***: the line type which can accept value form 1-6

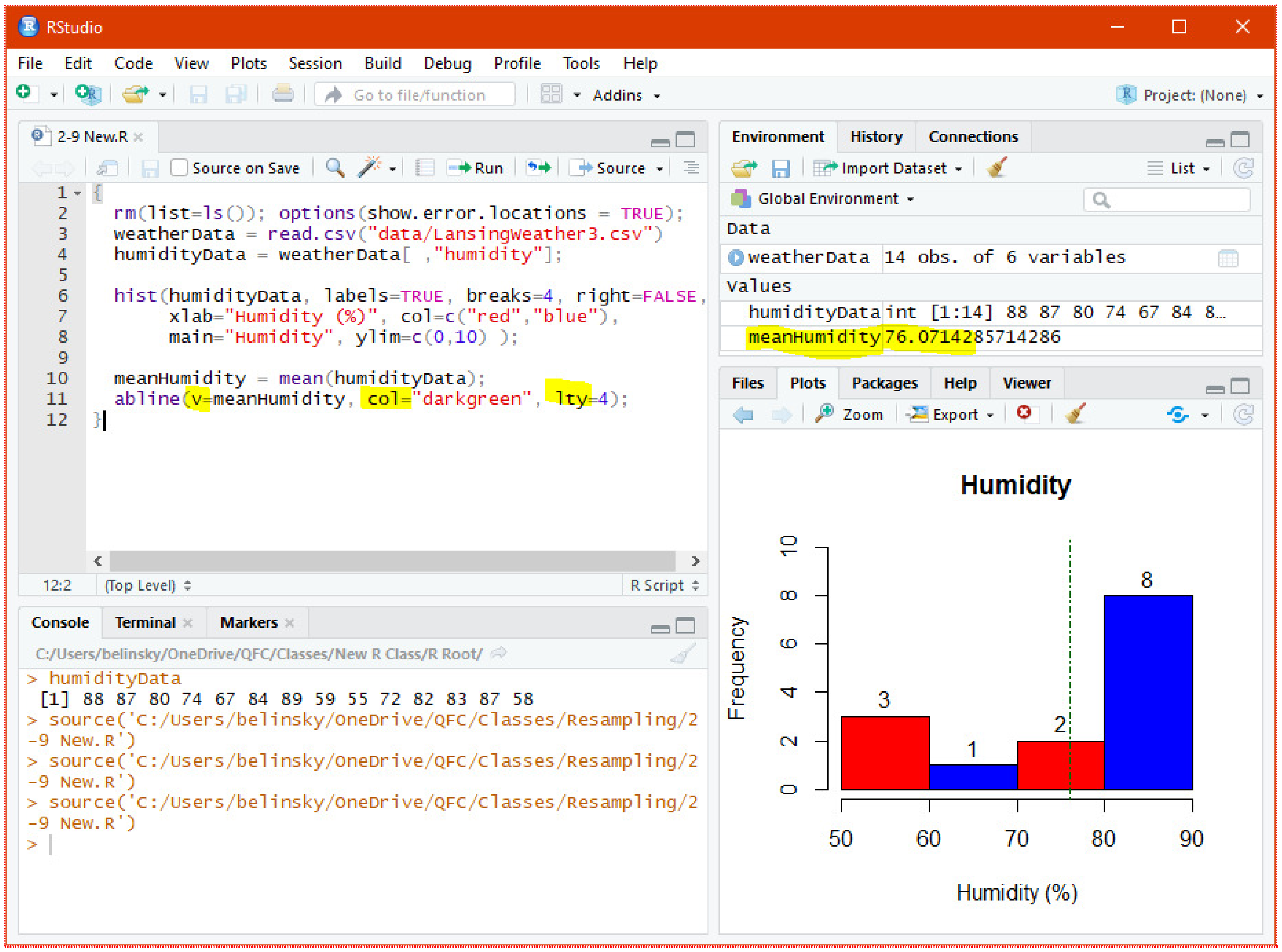
Extension: The (...) parameters for abline()

```
{
  rm(list=ls()); options(show.error.locations = TRUE);
  weatherData = read.csv("data/LansingWeather3.csv");
  humidityData = weatherData[ ,"humidity"];

  hist(humidityData, labels=TRUE, breaks=4, right=FALSE,
  xlab="Humidity (%)", col=c("red","blue"),
  main="Humidity", ylim=c(0,10) );

  meanHumidity = mean(humidityData);
  abline(v=meanHumidity, col="darkgreen", lty=4);
}
```

The mean humidity was calculate to be about 76 and then ***abline()*** put a vertical line at the mean humidity.



```
{
  rm(list=ls()); options(show.error.locations = TRUE);
  weatherData = read.csv("data/LansingWeather3.csv")
  humidityData = weatherData[ ,"humidity"];

  hist(humidityData, labels=TRUE, breaks=4, right=FALSE,
       xlab="Humidity (%)", col=c("red","blue"),
       main="Humidity", ylim=c(0,10) );

  meanHumidity = mean(humidityData);
  abline(v=meanHumidity, col="darkgreen", lty=4);
}
```

```
Data
weatherData     14 obs. of 6 variables
Values
humidityData    int [1:14] 88 87 80 74 67 84 8…
meanHumidity    76.0714285714286
```

```
C:/Users/belinsky/OneDrive/QFC/Classes/New R Class/R Root/
> humidityData
 [1] 88 87 80 74 67 84 89 59 55 72 82 83 87 58
> source('C:/Users/belinsky/OneDrive/QFC/Classes/Resampling/2
-9 New.R')
> source('C:/Users/belinsky/OneDrive/QFC/Classes/Resampling/2
-9 New.R')
> source('C:/Users/belinsky/OneDrive/QFC/Classes/Resampling/2
-9 New.R')
> |
```

Fig 5: Adding a vertical line to represent the mean humidity.

5 - The barplot

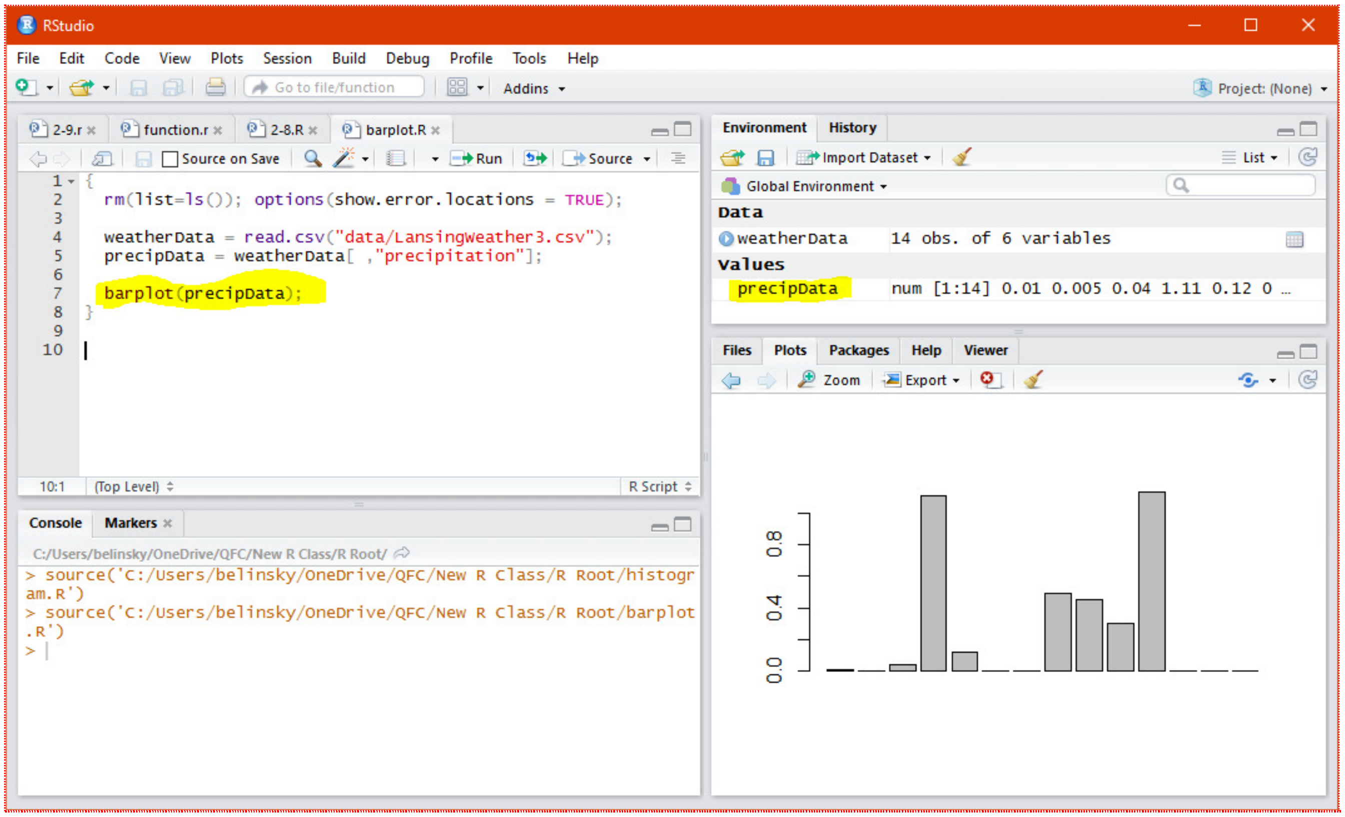
R has a function that creates barplots called ***barplot()*** and it works in a similar manner to ***hist()***

We can simply pass a vector (in this case, precipitation) into the ***barlpot()***. The values in this vector represent the height of each bar.

```
{
  rm(list=ls()); options(show.error.locations = TRUE);
  weatherData = read.csv("data/LansingWeather3.csv");
  precipData = weatherData[ ,"precipitation"];

  barplot(precipData);
}
```

And this produces a very plain barplot that shows there were several days with rain -- and twice the rain was over one inch.



```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather3.csv");
  precipData = weatherData[ ,"precipitation"];

  barplot(precipData);
}
```

```
Data
weatherData     14 obs. of 6 variables
Values
precipData      num [1:14] 0.01 0.005 0.04 1.11 0.12 0 ...
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/histogr
am.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/barplot
.R')
>
```

Fig 6: A barplot of the precipitation from the 14 days of data

5.1 - Setting parameters in barplots

The function ***barplot()*** looks very similar to hist()

```
barplot(height, width = 1, space = NULL,
     names.arg = NULL, legend.text = NULL, beside = FALSE,
     horiz = FALSE, density = NULL, angle = 45,
     col = NULL, border = par("fg"),
     main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
     xlim = NULL, ylim = NULL, xpd = TRUE, log = "",
     axes = TRUE, axisnames = TRUE,
     cex.axis = par("cex.axis"), cex.names = par("cex.axis"),
        inside = TRUE, plot = TRUE, axis.lty = 0, offset = 0,
        add = FALSE, args.legend = NULL, ...)
```

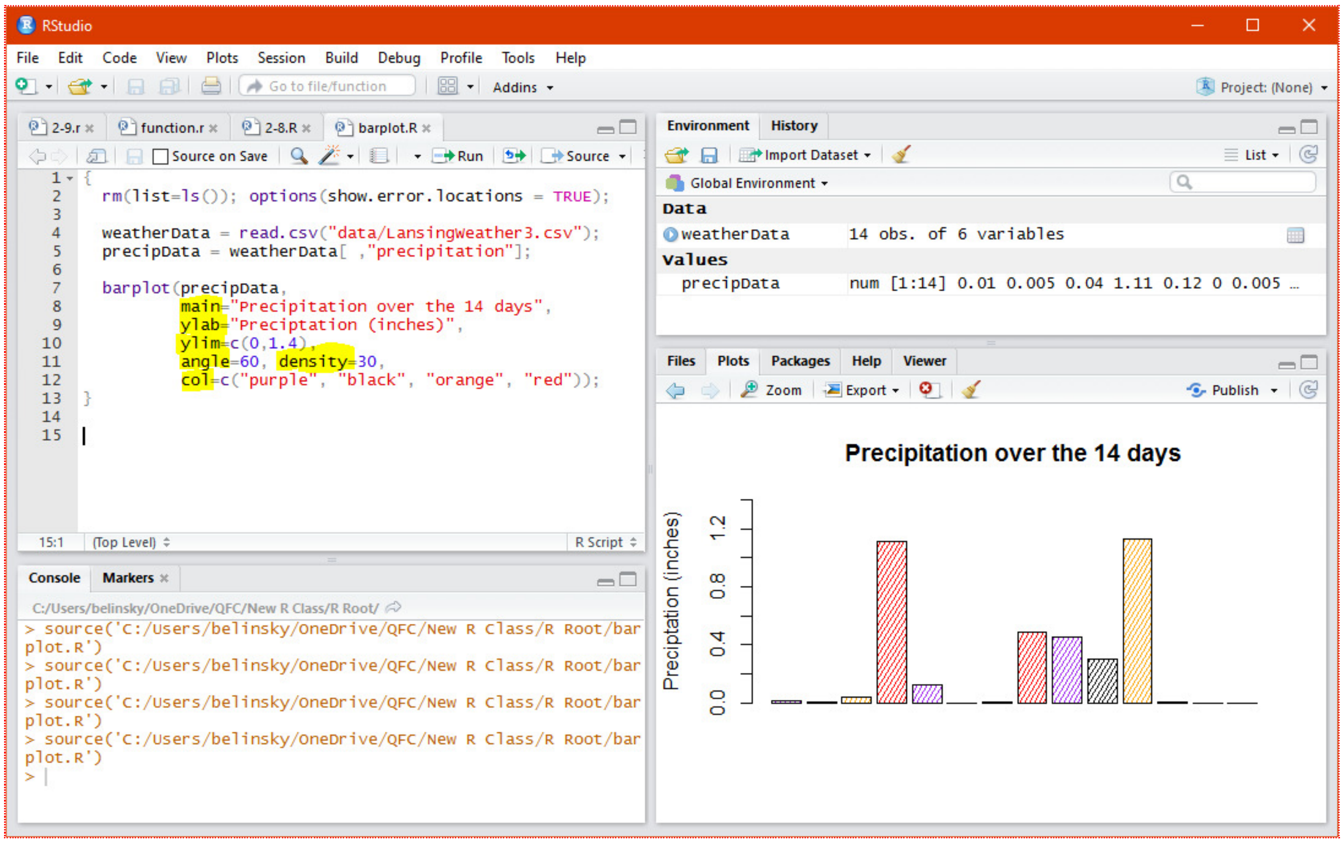
Let's now assign values to some of the parameters that have default values (the ones highlighted above)

```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather3.csv");
  precipData = weatherData[ ,"precipitation"];

  barplot(precipData,
          main="Precipitation over the 14 days",
          ylab="Preciptation (inches)",
          ylim=c(0,1.4),
          angle=60, density=30,
          col=c("purple", "black", "orange", "red"));
}
```

Now we have a barplot that is more colorful and the y-axis limits have been increased to match our precipitation values.



```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather3.csv");
  precipData = weatherData[ ,"precipitation"];

  barplot(precipData,
          main="Precipitation over the 14 days",
          ylab="Preciptation (inches)",
          ylim=c(0,1.4),
          angle=60, density=30,
          col=c("purple", "black", "orange", "red"));
}

```

```
Data
weatherData     14 obs. of 6 variables
Values
precipData      num [1:14] 0.01 0.005 0.04 1.11 0.12 0 0.005 ...
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/bar
plot.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/bar
plot.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/bar
plot.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/bar
plot.R')
> |
```

Fig 7: Barplot with a bunch of parameters changed

Let's look at the parameters we changed:

- ***main***: adds a title

- ***ylab***: adds a label to the y axis
- ***ylim***: changes the limits of the y-axis (it now goes from 0 to 1.4)
- ***angle***: sets the angle of the hatch lines inside the bars to 60 degrees
- ***density***: sets the numbers of hatch lines in the bars to 30 per inch
- ***col***: sets the color by bar. In this case the order goes purple-black-orange-red-purple-black...

5.2 - Adding values to the x-axis

The x-axis is pretty barren on this plot. It would be nice to put useful information on the x-axis, like the date. We have the dates in the ***weatherData*** data frame, which we will save to the vector ***dateData***:

```
  dateData = weatherData[ ,"date"];
```

We can use the ***dateData*** vector as x-axis labels using the ***names.arg*** parameter in the ***barplot()*** function

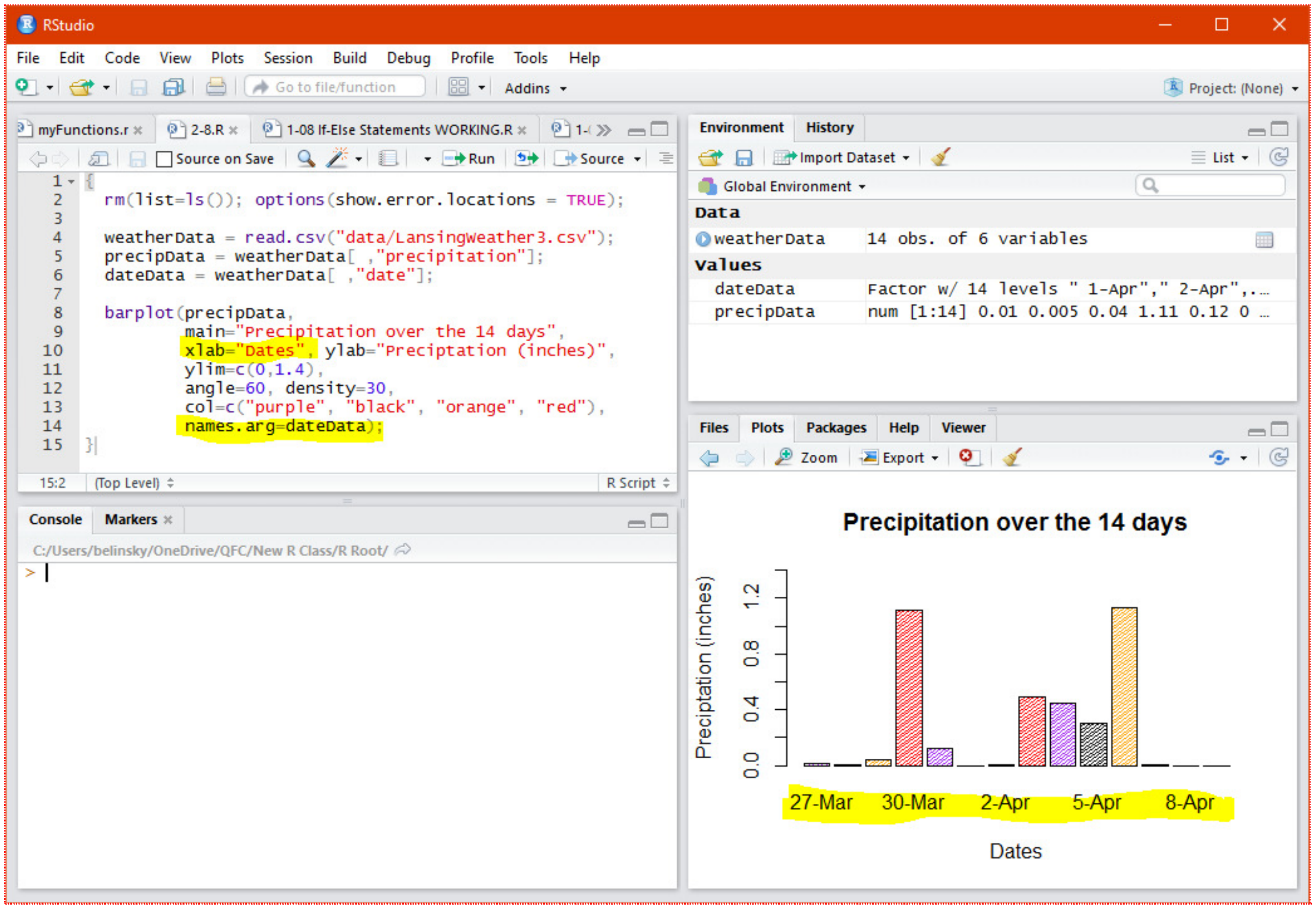
```
  names.arg=dateData
```

And we also assign the value "Dates" to the pararmeter ***xlab*** and the full script becomes...

```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather3.csv");
  precipData = weatherData[ ,"precipitation"];
  dateData = weatherData[ ,"date"];

  barplot(precipData,
          main="Precipitation over the 14 days",
          xlab="Dates", ylab="Preciptation (inches)",
          ylim=c(0,1.4),
          angle=60, density=30,
          col=c("purple", "black", "orange", "red"),
          names.arg=dateData);
}
```



```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather3.csv");
  precipData = weatherData[ ,"precipitation"];
  dateData = weatherData[ ,"date"];

  barplot(precipData,
          main="Precipitation over the 14 days",
          xlab="Dates", ylab="Preciptation (inches)",
          ylim=c(0,1.4),
          angle=60, density=30,
          col=c("purple", "black", "orange", "red"),
          names.arg=dateData);
}
```

```
Data
weatherData     14 obs. of 6 variables
Values
dateData        Factor w/ 14 levels " 1-Apr"," 2-Apr",...
precipData      num [1:14] 0.01 0.005 0.04 1.11 0.12 0 ...
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
>
```

Fig 8: Adding dates to the x-axis

5.3 - Adding a line to represent the mean

Just like with the histogram of humidity, we can add a mean line for the precipitation. The difference here is that the mean line needs to be placed horizontally.

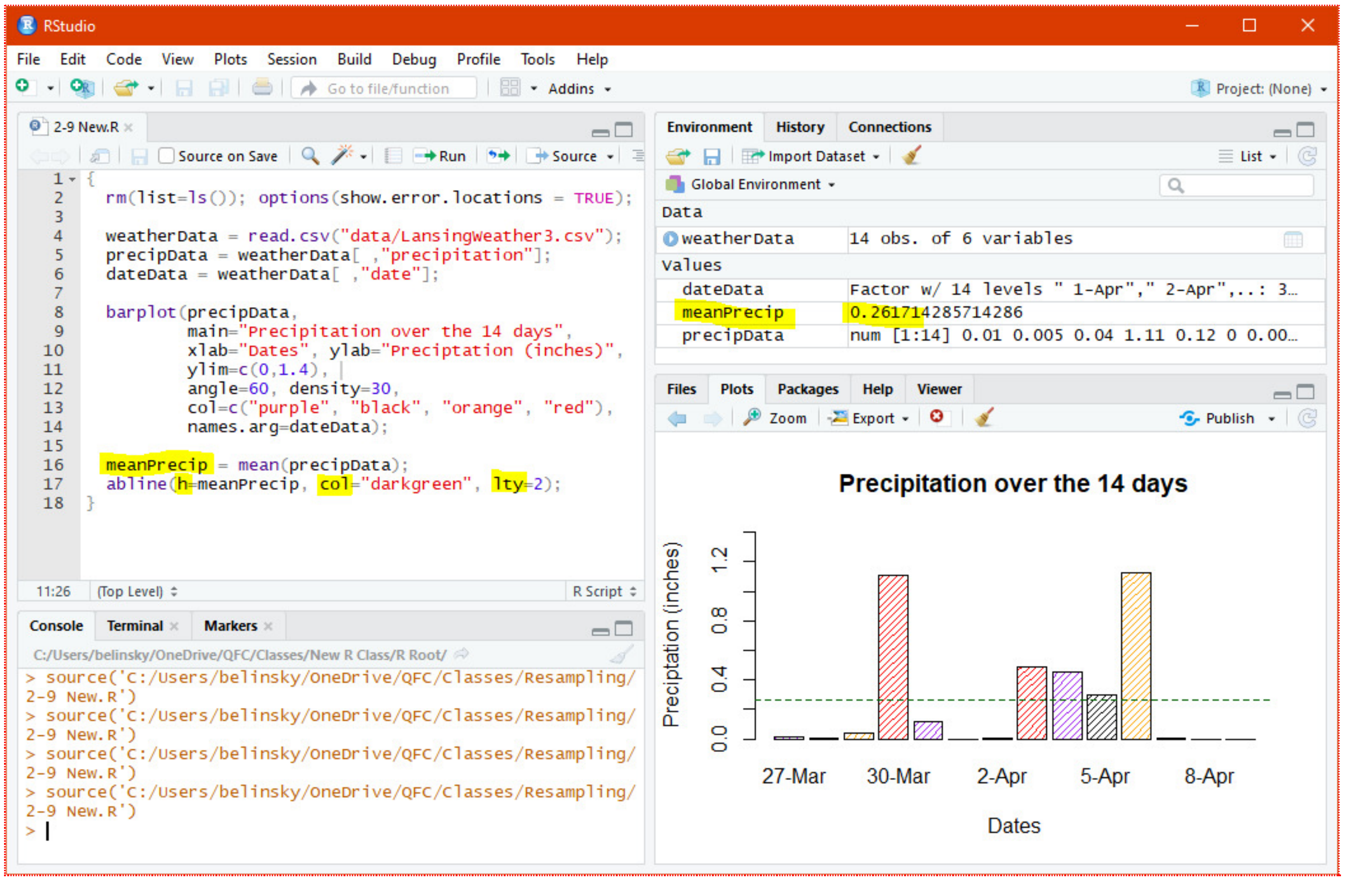
```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather3.csv");
  precipData = weatherData[ ,"precipitation"];
  dateData = weatherData[ ,"date"];

  barplot(precipData,
          main="Precipitation over the 14 days",
          xlab="Dates", ylab="Preciptation (inches)",
          ylim=c(0,1.4),
          angle=60, density=30,
          col=c("purple", "black", "orange", "red"),
          names.arg=dateData);

  meanPrecip = mean(precipData);
  abline(h=meanPrecip, col="darkgreen", lty=2);
}
```

Just like with the histogram, we found ***meanPrecip*** using ***mean()*** and added a line using ***abline()***. The difference is that we added a horizontal line so we assigned the vector ***meanPrecip*** to the parameter ***h*** instead of the parameter ***v***.



```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather3.csv");
  precipData = weatherData[ ,"precipitation"];
  dateData = weatherData[ ,"date"];

  barplot(precipData,
          main="Precipitation over the 14 days",
          xlab="Dates", ylab="Preciptation (inches)",
          ylim=c(0,1.4),
          angle=60, density=30,
          col=c("purple", "black", "orange", "red"),
          names.arg=dateData);

  meanPrecip = mean(precipData);
  abline(h=meanPrecip, col="darkgreen", lty=2);
}
```

```
Data
weatherData    14 obs. of 6 variables
Values
dateData       Factor w/ 14 levels " 1-Apr"," 2-Apr",..: 3…
meanPrecip     0.261714285714286
precipData     num [1:14] 0.01 0.005 0.04 1.11 0.12 0 0.00…
```

```
C:/Users/belinsky/OneDrive/QFC/Classes/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/Classes/Resampling/
2-9 New.R')
> source('C:/Users/belinsky/OneDrive/QFC/Classes/Resampling/
2-9 New.R')
> source('C:/Users/belinsky/OneDrive/QFC/Classes/Resampling/
2-9 New.R')
> source('C:/Users/belinsky/OneDrive/QFC/Classes/Resampling/
2-9 New.R')
> 
```

Fig 9: Adding the mean line to the precipitation barplot

6 - The boxplot

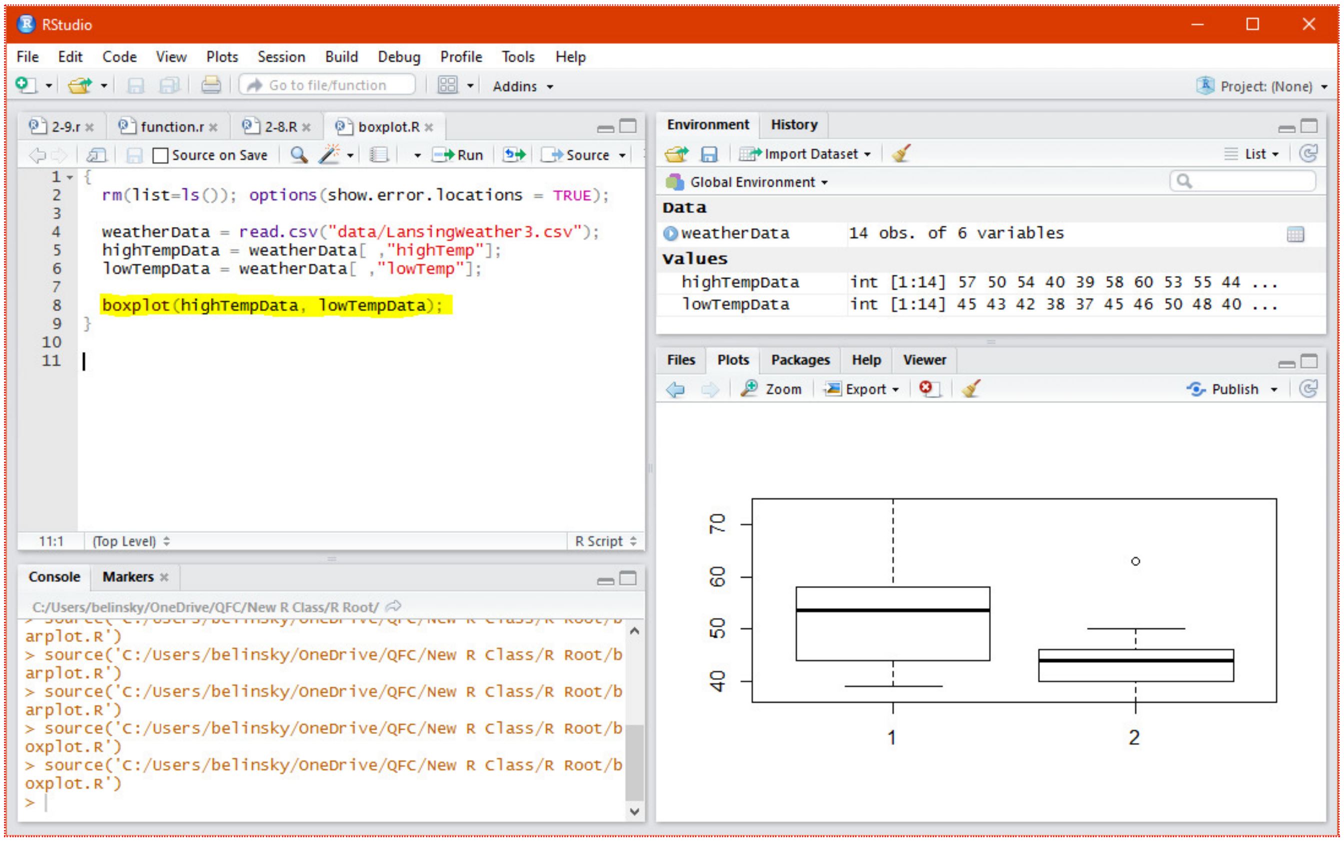
Our last plot is going to be a boxplot, which is another way to plot the distribution of data similar to ***hist()***. We are going to put two boxplots on one graph-- one for ***highTempData*** and one for the ***lowTempData.***

```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather3.csv");
  highTempData = weatherData[ ,"highTemp"];
  lowTempData = weatherData[ ,"lowTemp"];

  boxplot(highTempData, lowTempData);
}
```

R places the two boxplots on the graph but a part of each boxplot is broken off because the y-axis values are too restrictive. <strange behavior -- need to redo>



```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather3.csv");
  highTempData = weatherData[ ,"highTemp"];
  lowTempData = weatherData[ ,"lowTemp"];

  boxplot(highTempData, lowTempData);
}
```

Data
weatherData 14 obs. of 6 variables
Values
highTempData int [1:14] 57 50 54 40 39 58 60 53 55 44 ...
lowTempData int [1:14] 45 43 42 38 37 45 46 50 48 40 ...

```
arplot.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/b
arplot.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/b
arplot.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/b
oxplot.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/b
oxplot.R')
>
```

Fig 10: A boxplot of the high and low temperatures

Extension: unknown number of parameters.

6.1 - Changing some of the boxplot parameters

We are going to assign values to two parameters in ***boxplot()***:

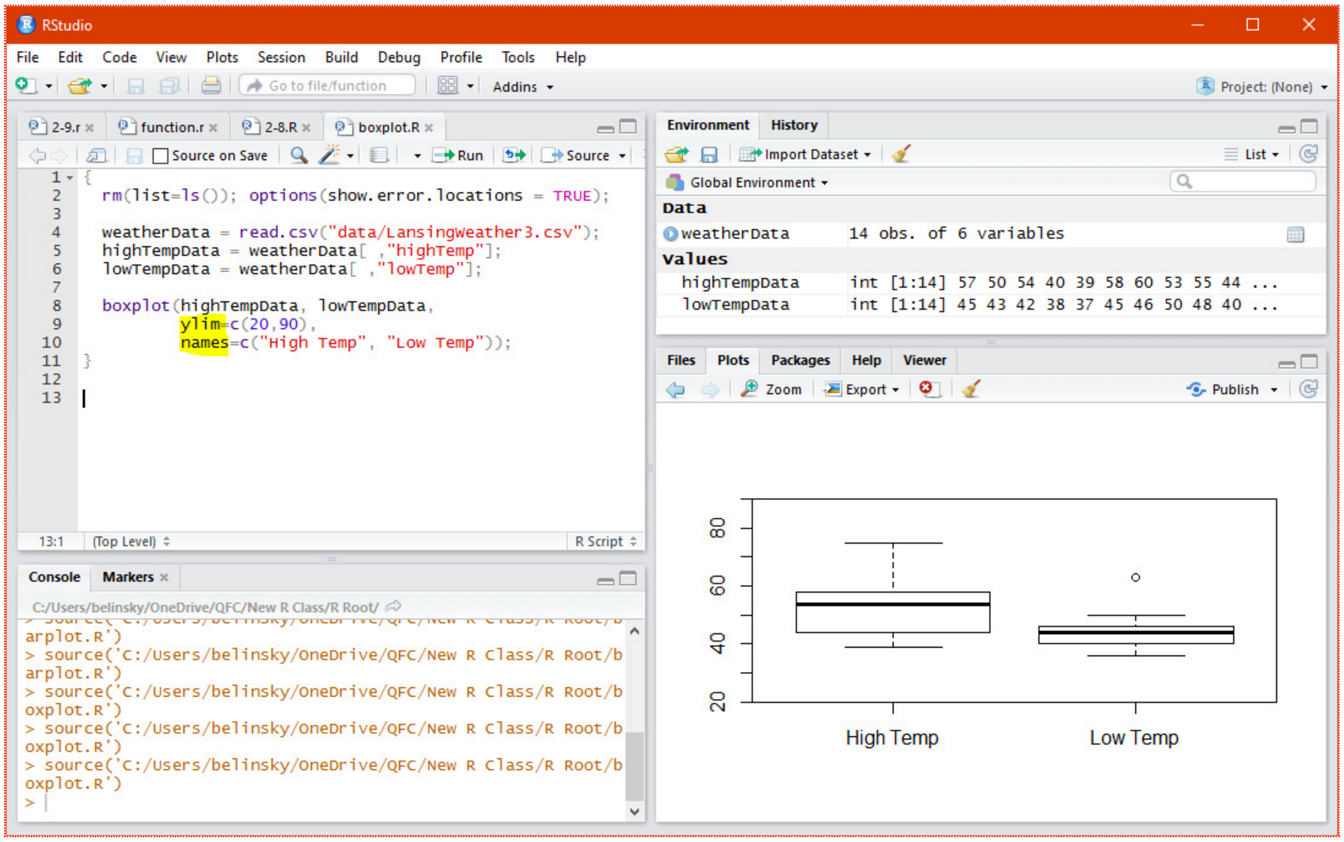
1) ***names***: replace the sequence label on the x-axis with names for the boxplots
2) ***ylim***: change the limits on the y-axis to fit all boxplots' data

```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather3.csv");
  highTempData = weatherData[ ,"highTemp"];
  lowTempData = weatherData[ ,"lowTemp"];

  boxplot(highTempData, lowTempData,
          ylim=c(20,90),
          names=c("High Temp", "Low Temp"));
}
```

Now we have boxplots that can be easily viewed.



```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather3.csv");
  highTempData = weatherData[ ,"highTemp"];
  lowTempData = weatherData[ ,"lowTemp"];

  boxplot(highTempData, lowTempData,
          ylim=c(20,90),
          names=c("High Temp", "Low Temp"));
}
```

```
weatherData     14 obs. of 6 variables
Values
  highTempData  int [1:14] 57 50 54 40 39 58 60 53 55 44 ...
  lowTempData   int [1:14] 45 43 42 38 37 45 46 50 48 40 ...
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
arplot.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/b
arplot.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/b
oxplot.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/b
oxplot.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/b
oxplot.R')
>
```

Fig 11: Changing the x and y axis values on the boxplot

6.2 - Adding points to the boxplot

We can add points to the boxplot using the ***points()*** function described on this page.

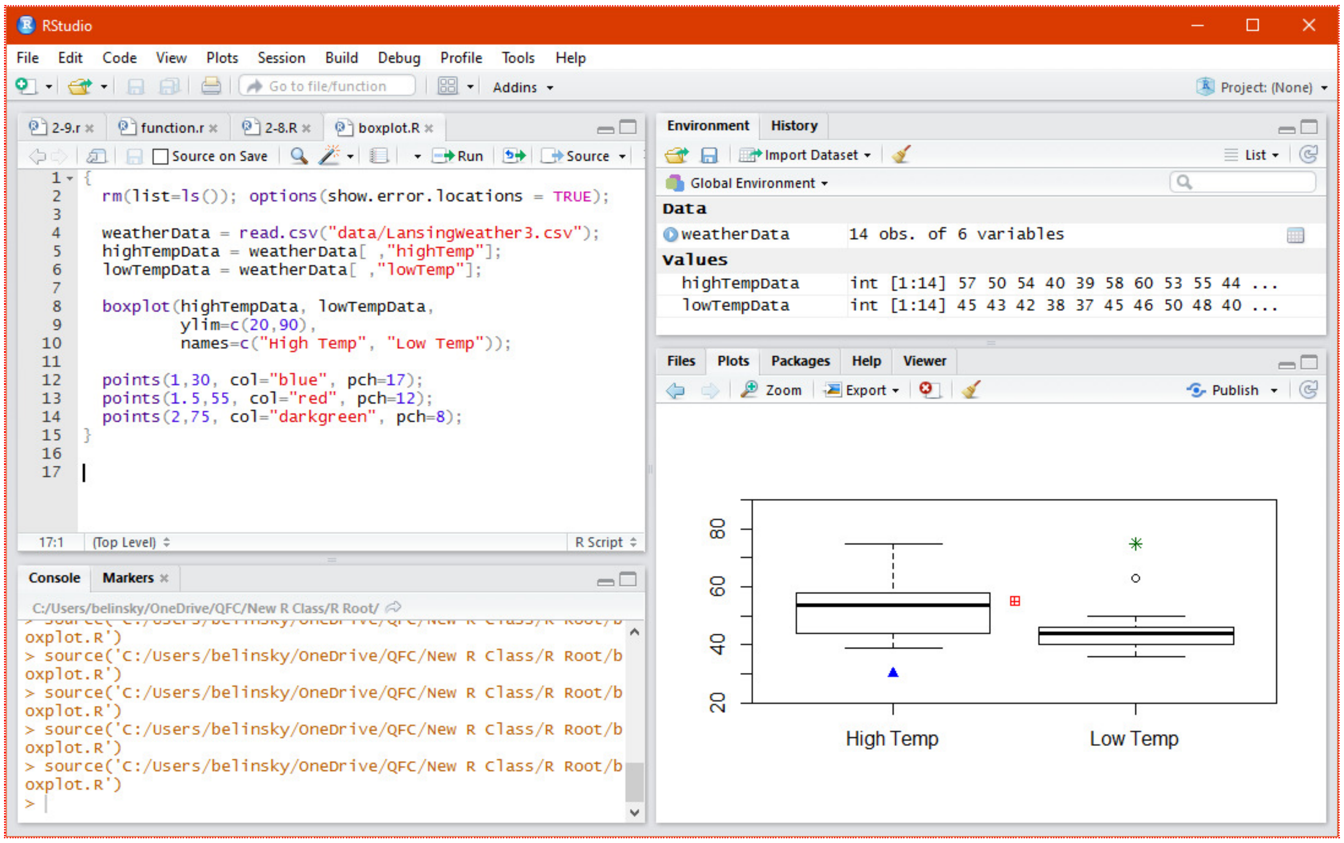
```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather3.csv");
  highTempData = weatherData[ ,"highTemp"];
  lowTempData = weatherData[ ,"lowTemp"];

  boxplot(highTempData, lowTempData,
          ylim=c(20,90),
          names=c("High Temp", "Low Temp"));

  points(1,30, col="blue", pch=17);
  points(1.5,55, col="red", pch=12);
  points(2,75, col="darkgreen", pch=8);
}
```

We added three different points, each with a different color (***col="blue", "green", and "red"***) and shape (***pch= 17,12, and 8***). Here is a list of the different types of shapes that can be used as point characters (***pch***).



```
{
  rm(list=ls()); options(show.error.locations = TRUE);

  weatherData = read.csv("data/LansingWeather3.csv");
  highTempData = weatherData[ ,"highTemp"];
  lowTempData = weatherData[ ,"lowTemp"];

  boxplot(highTempData, lowTempData,
          ylim=c(20,90),
          names=c("High Temp", "Low Temp"));

  points(1,30, col="blue", pch=17);
  points(1.5,55, col="red", pch=12);
  points(2,75, col="darkgreen", pch=8);
}
```

```
Data
weatherData      14 obs. of 6 variables
Values
highTempData     int [1:14] 57 50 54 40 39 58 60 53 55 44 ...
lowTempData      int [1:14] 45 43 42 38 37 45 46 50 48 40 ...
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
oxplot.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/b
oxplot.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/b
oxplot.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/b
oxplot.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/b
oxplot.R')
> 
```

Fig 12: Adding points to the boxplot

Let's look at the parameters of ***points()***

- ***#,#***: the x and y coordinates of the point. The "values" on the x-axis are the sequence values, with the number 1 represents the first boxplot on the x-axis, the number 2 represents the second number. Decimal number will give you positions in between objects.
- ***col***: the color of the point
- ***pch***: the shape of the point. The different point shapes you can use are described here.

7 - Application

1) Create a histogram of the difference between the high and low temperature for the 14 days. Add labels to both axes and a title. Make the bins in the histogram 2 unit wide. Add an average change in temperature line to the plot.

2) Create a barplot of the the high and low temperatures (so, 28 values). Label both axes, add a title, and use three colors for the bins. Add an average temperature line to the barplot.
Hint: create a vector that contains all the high temp and low temp values

3) Create a boxplot of humidity and precipitation on the same plot. Label the axis and give it a title. Hint, divide the humidity values by 100 so that they are decimal values. Add points at...

8 - Extension: The x value in hist()

The function for ***hist()*** includes the parameter ***x.*** ***x*** is the data to be plotted. Because ***x*** is the first parameter, the name is often skipped as the first value in the function is assumed by R to be the value of the first parameter.

So

```
hist(humidityData);
```

is functionally the same as

```
hist(x=humidityData);
```

Of course, you need to name any other parameter that you assign values to unless *you keep the parameters in the exact same order* as the function declaration. No one does this and it would be a real pain given how many parameters are in functions like ***hist()***:

```
hist(x, breaks = "Sturges",
     freq = NULL, probability = !freq,
      include.lowest = TRUE, right = TRUE,
      density = NULL, angle = 45, col = NULL, border = NULL,
      main = paste("Histogram of" , xname),
      xlim = range(breaks), ylim = NULL,
      xlab = xname, ylab,
      axes = TRUE, plot = TRUE, labels = FALSE,
      nclass = NULL, warn.unused = TRUE, ...)
```

This is true of many other functions that involves data and a bunch of other parameters. Some, like ***plot()***, have two parameters for the data (***x*** and ***y***).

So, you will often see

```
plot(humidityData, precipData);
```

which is equivalent to

```
plot(x=humidityData, y=precipData);
```

which is *also* equivalent to

```
plot(y=precipData, x=humidityData);
```

9 - Extension: The (...) parameters of abline()

The function ***abline()*** is given at this help page as:

```
abline(a = NULL, b = NULL, h = NULL, v = NULL, reg = NULL,
       coef = NULL, untf = FALSE, ...)
```

Notice that ***col*** and ***lty*** are not variables in the ***abline()*** function in the link above. They are actually part of the (***...***) -- as described further down in the help page. ***col*** and ***lty*** are part of the graphical parameters in the function ***par()***, which is a very large set of parameters that you can assign values to.

You can type "par" in the search bar of the Help Window to see the various parameters that can be set.

10 - Extension: Unknown number of parameters

boxplot() has an unusual function declaration:

```
boxplot(x, ..., range = 1.5, width = NULL, varwidth = FALSE,
        notch = FALSE, outline = TRUE, names, plot = TRUE,
        border = par("fg"), col = NULL, log = "",
        pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),
        horizontal = FALSE, add = FALSE, at = NULL)
```

The (...) represent the fact that ***boxplot()*** can handle an unknown amount of data, each set of data getting a separate boxplot. In this case, there is no specific parameter value for the data so the data is passed in to the ***boxplot()*** *without a parameters name*.

So:

```
boxplot(highTempData, lowTempData);
```

outputs a boxplot for both ***highTempData*** and ***lowTempData***

and you could produce more boxplots on the same plot:

```
boxplot(highTempData, lowTempData, humidityData, precipData);
```

But, there is no parameter name for these types of values.