02-07: Plots 1

1 - Purpose

- Create a scatter plot and a line plot
- Adding lines to a plot
- Style the plot and change axis ranges
- Include a legend

2 - Questions about the material...

If you have any questions about the material in this lesson or your Class Project *feel free to email them to the instructor here*.

3 - A New Data Frame

For this lesson we will be using an expanded weather data set that includes ***dayOfYear*** and ***humidity***. ***dayOfYear*** assigns a numerical value for each day of the year from 1-365 (or 366 on a leap year). Copy the data below and save it as ***LansingWeather3.csv*** in your ***data*** folder.

```
date,dayOfYear,highTemp, lowTemp, precipitation,humidity
27-Mar,86,57,45,0.01,88
28-Mar,87,50,43,0.005,87
29-Mar,88,54,42,0.04,80
30-Mar,89,40,38,1.11,74
31-Mar,90,39,37,0.12,67
1-Apr,91,58,45,0,84
2-Apr,92,60,46,0.005,89
3-Apr,93,53,50,0.49,59
4-Apr,94,55,48,0.45,55
5-Apr,95,44,40,0.3,72
6-Apr,96,39,36,1.13,82
7-Apr,97,53,43,0.004,83
8-Apr,98,61,45,0,87
9-Apr,99,75,63,0,58
```

4 - Your first plot

We will first plot a time series of the high temperatures from the data above.

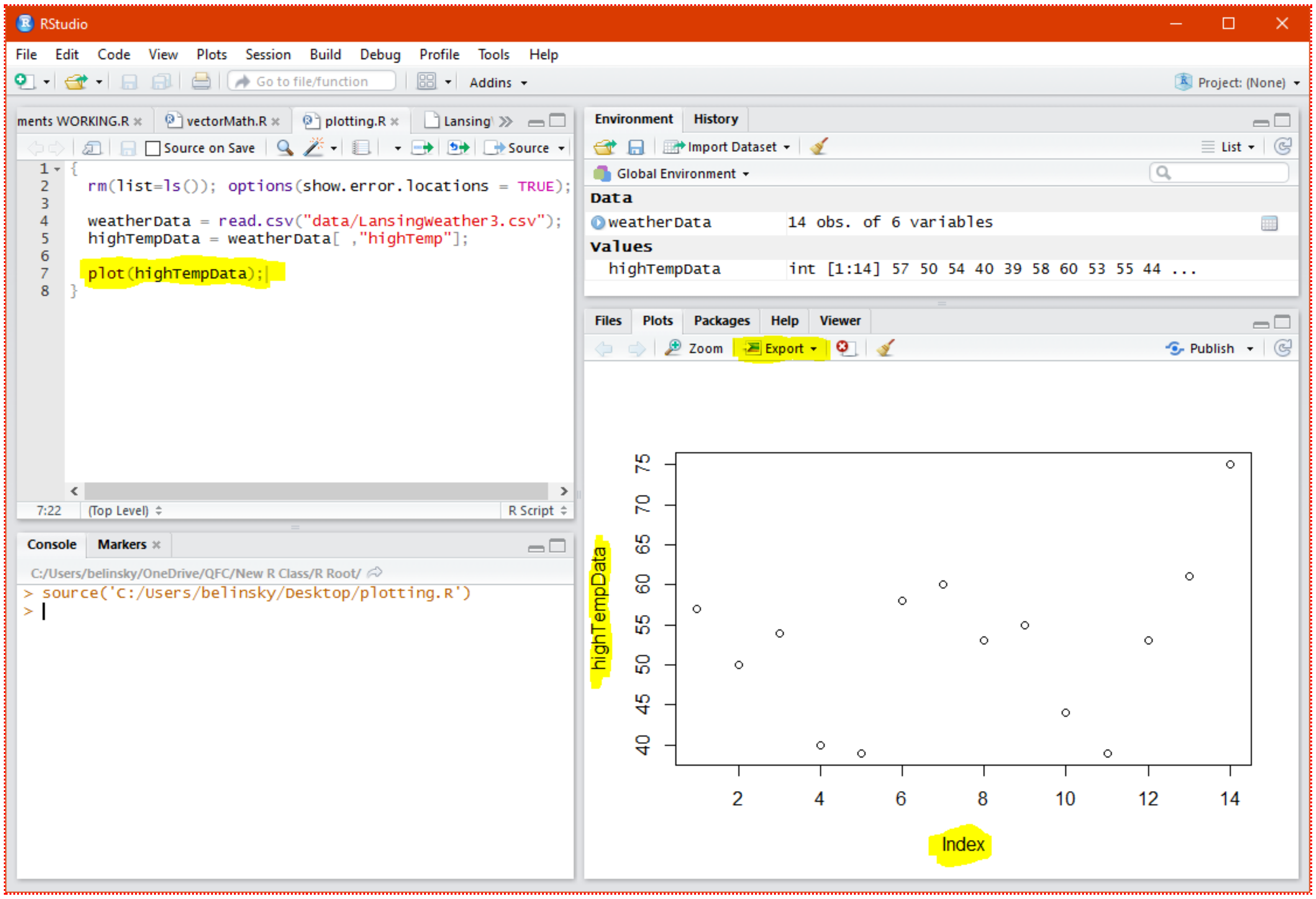
The R function we will use is ***plot()*** and there are many parameters you can include that control the plot's properties and appearance. There are numerous ways in which you can modify the plot and we will learn some of these in this and future lessons.

For now we will simply add the vector ***highTempData*** as a parameter to ***plot()***.

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

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

  plot(highTempData);
}
```



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

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

  plot(highTempData);
}
```

```
Data
weatherData        14 obs. of 6 variables
Values
highTempData       int [1:14] 57 50 54 40 39 58 60 53 55 44 ...
```

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

*Fig 1: Scatterplot of the high temperature from **weatherData***

In the Plot Window, we see a scatterplot of the high temperature values plotted against the high temperature's vector index.

The plot can be resized by dragging the gray area surrounding the Plot Window. *Extension: Full-sized window for plots.*

You can also use the ***Export*** button to save the plot as an image, a PDF, or to the clipboard (to be pasted to a document).

Trap: A very common, and very unintuitive, plot error

4.1 - Plotting X and Y variables

The plot above just plots the high temperature vs. the index number of the values. Instead of plotting against the index number, we can plot the high temperature against the ***dayOfYear***. The ***dayOfYear*** value for March 27 (the first day in the data) is 86. Note: we use ***dayOfYear*** instead of ***date*** because ***date*** (e.g., **2-Apr**) contains non-numeric values.

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

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

  plot(formula=highTempData~dateData);  # plot(y~x)
}
```

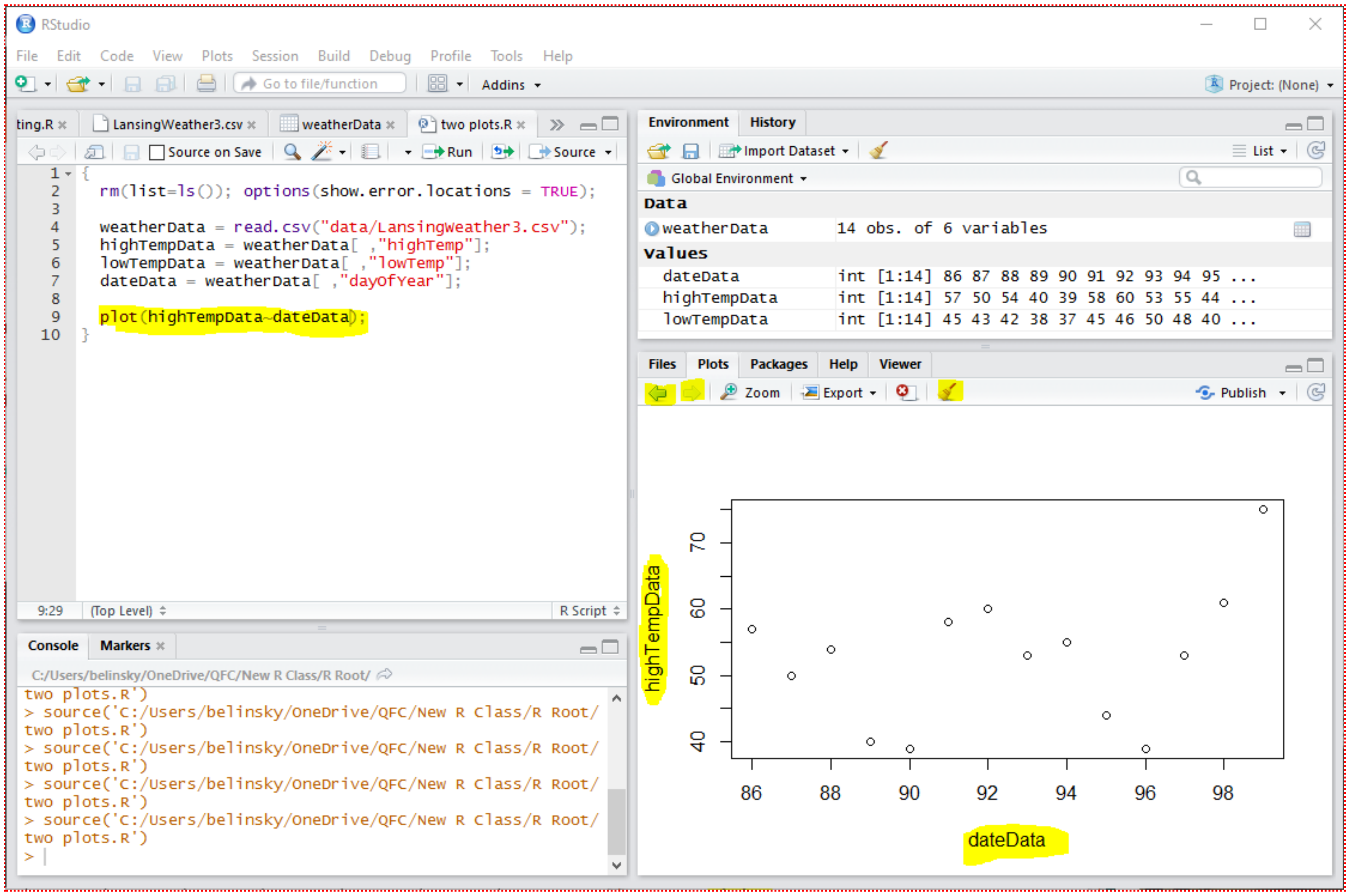
In the last line of the script:

```
plot(formula=highTempData~dateData);
```

highTempData are the dependent values (y) and ***dateData*** are the independent values (x)

The line can also be written in this form -- ***plot(x, y)***

```
plot(x=dateData, y=highTempData);
```



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

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

  plot(highTempData~dateData);
}
```

```
Data
weatherData       14 obs. of 6 variables
Values
  dateData        int [1:14] 86 87 88 89 90 91 92 93 94 95 ...
  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/
two plots.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
two plots.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
two plots.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
two plots.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
two plots.R')
> |
```

Fig 2: Plotting high temperature vs. the day-of-year

Note that the Plot Window stores all the previous plots and the plots can be cycled through by clicking the arrow buttons. You can use the brush button to remove all of the plots.

4.2 - Plotting lines

If we want to connect the points -- in other words, we want a line graph, we need to add an parameter to ***plot()*** called ***type***. Note: the points are connected according to their order in the vector.

the parameter ***type*** can have the following values:

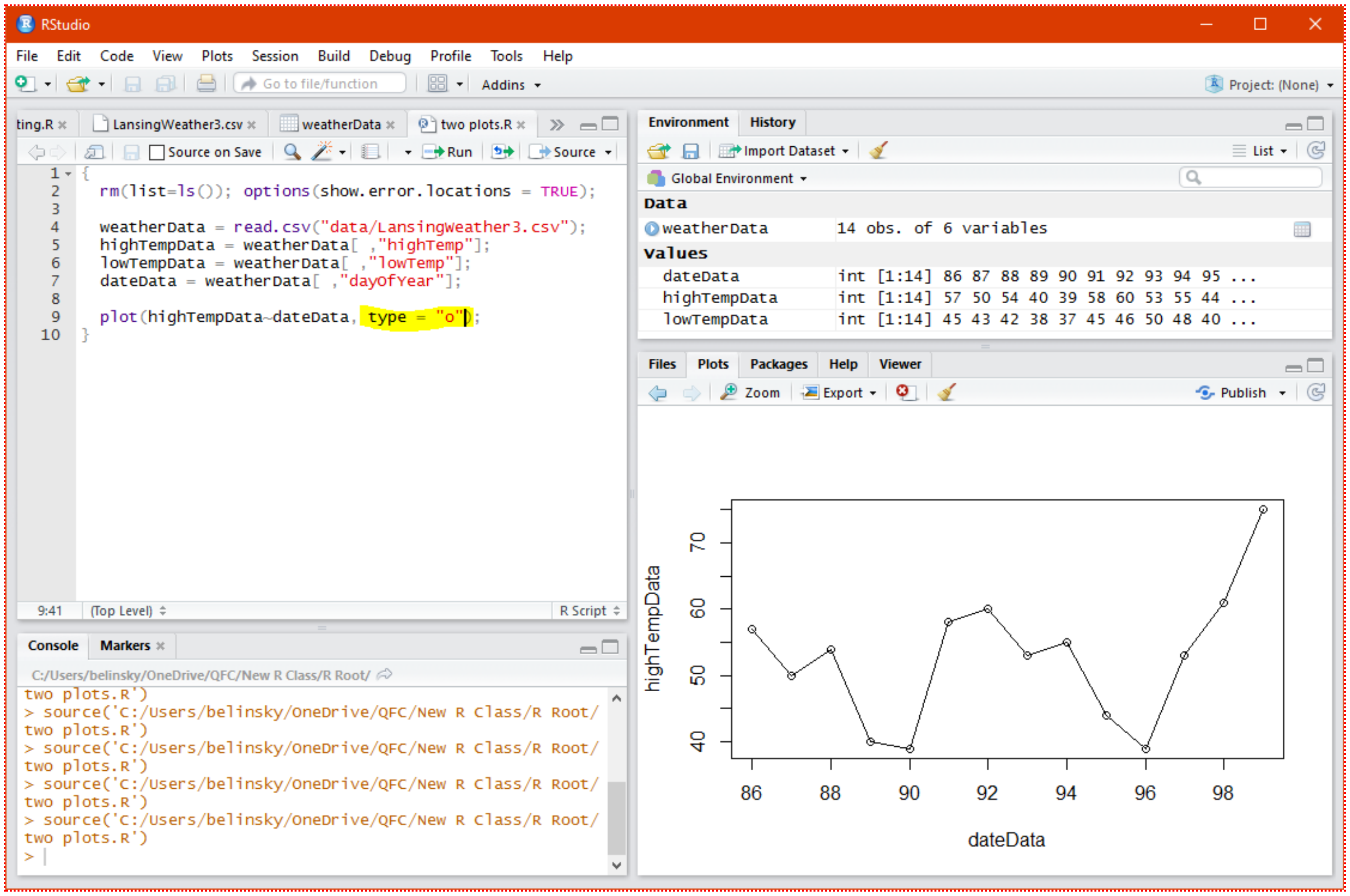
- **"p"**: points only
- **"l"**: line only
- **"o"**: points and lines
- **"n"**: neither points nor lines (empty plot)

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

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

  plot(highTempData~dateData, type = "o");
}
```

*Don't forget to put a comma between the parameters in **plot()**.*



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

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

  plot(highTempData~dateData, type = "o");
}
```

```
Data
weatherData      14 obs. of 6 variables
Values
dateData         int [1:14] 86 87 88 89 90 91 92 93 94 95 ...
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/
two plots.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
two plots.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
two plots.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
two plots.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
two plots.R')
> |
```

Fig 3: Connecting the points on a scatterplot.

4.3 - Graph labels and title

The default labels for the axes are the names of the variables used in the plot (e.g., ***dateData*** and ***highTempData***) and the default for the title is blank. We can edit the labels using the parameters ***xlab*** and ***ylab*** and add a title using the parameter ***main***.

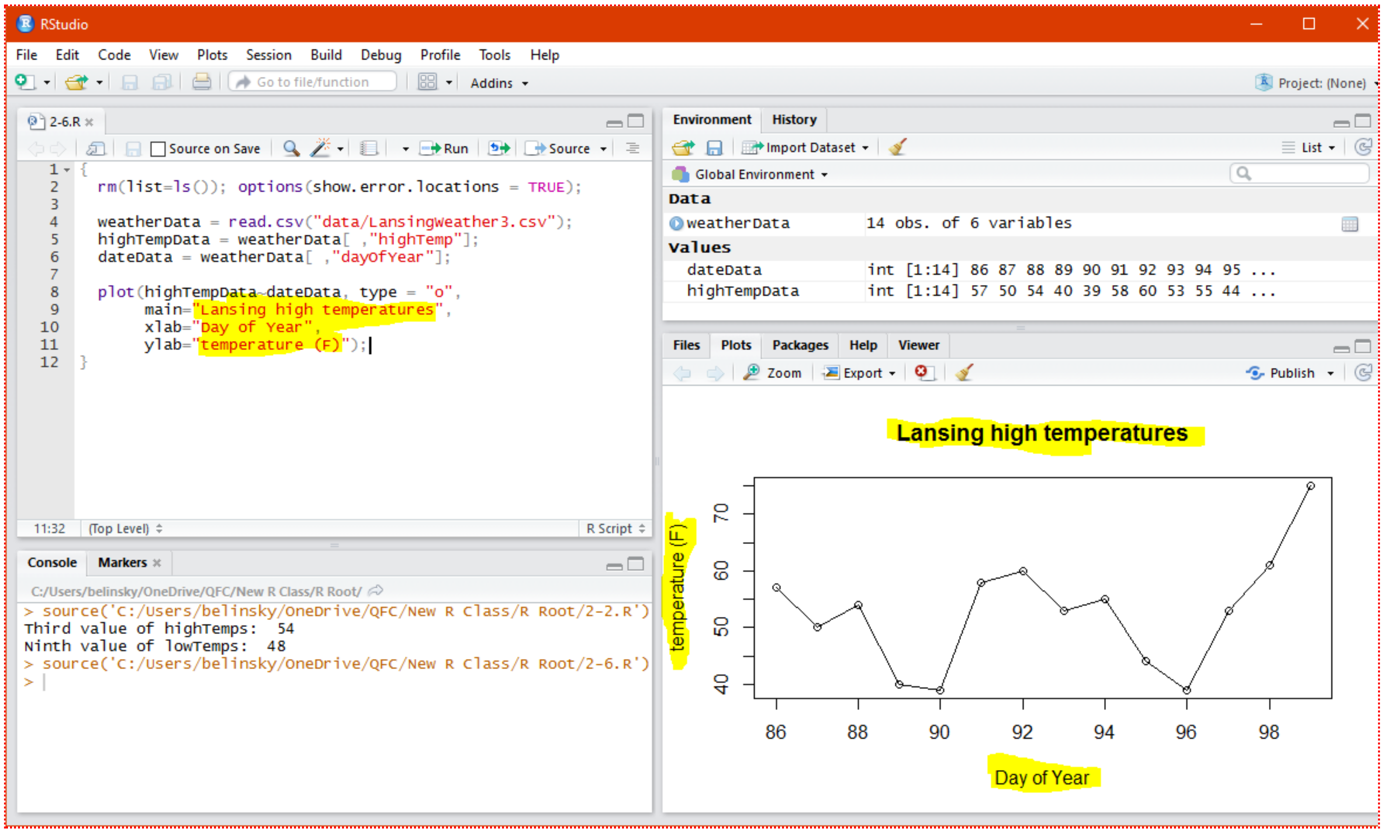
```
{
  rm(list=ls()); options(show.error.locations = TRUE);

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

  dateData = weatherData[ ,"dayOfYear"];

  plot(highTempData~dateData, type = "o",
       main="Lansing high temperatures",
       xlab="Day of Year",
       ylab="temperature (F)");
}
```

Note, I split the function ***plot()*** into multiple lines to make it easier to see each of the parameters.



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

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

  plot(highTempData~dateData, type = "o",
       main="Lansing high temperatures",
       xlab="Day of Year",
       ylab="temperature (F)");
}
```

```
Data
weatherData      14 obs. of 6 variables
Values
dateData         int [1:14] 86 87 88 89 90 91 92 93 94 95 ...
highTempData     int [1:14] 57 50 54 40 39 58 60 53 55 44 ...
```

```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/2-2.R')
Third value of highTemps:  54
Ninth value of lowTemps:  48
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/2-6.R')
> 
```

Fig 4: Adding labels to the axes and a title to the plot.

5 - Adding a second plot

The next step is to add another set of values to the plot -- specifically the low temperature data. We cannot use ***plot()*** to add the low temperature data because ***plot()*** creates a new plot so ***plot()** cannot be used to append data to an existing plot*.

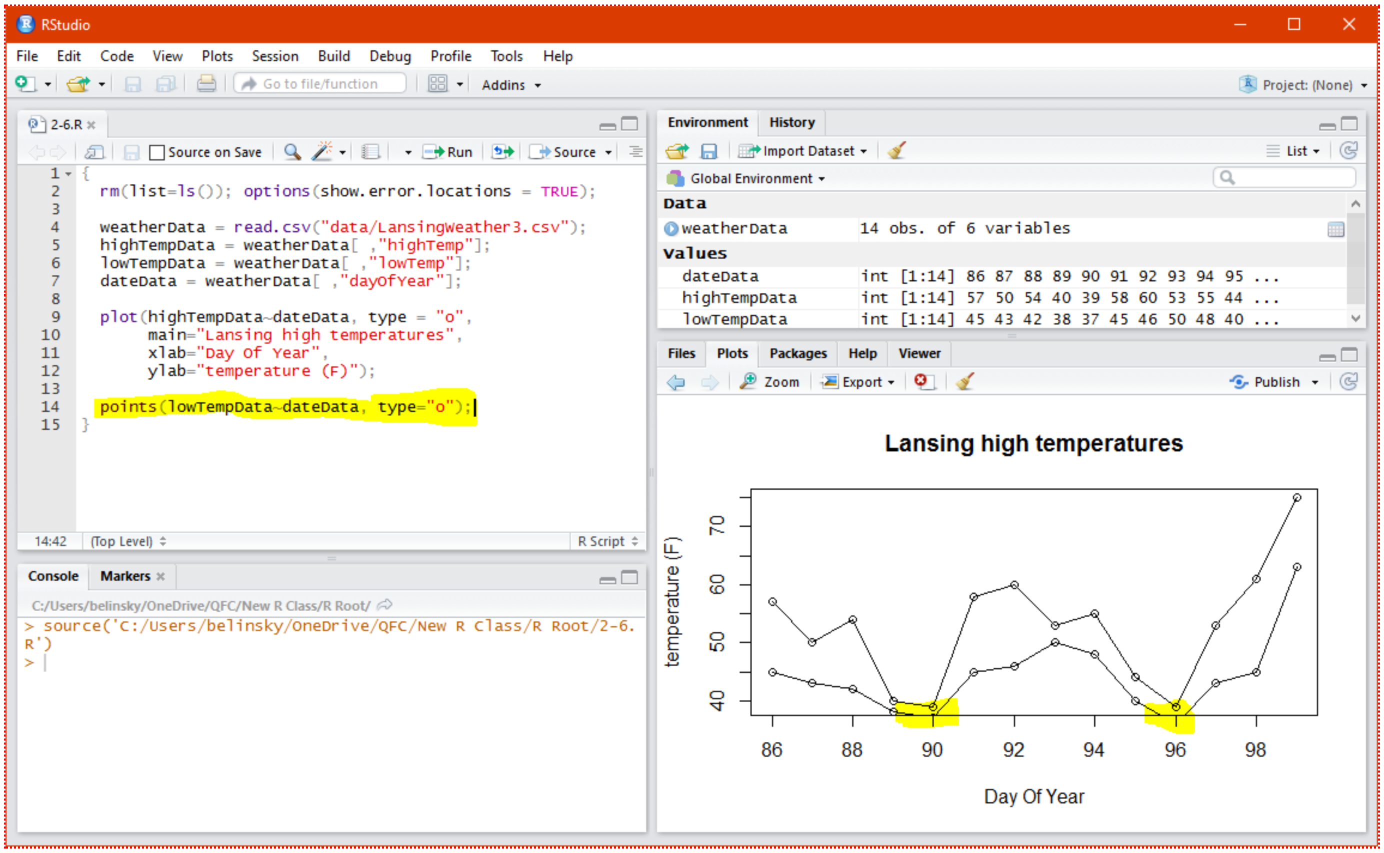
Instead we use the function ***points()***. ***points()*** appends data to the current plot and the parameters for ***points()*** are almost the same as ***plot()*** -- including ***type***="o" to connect the points.

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

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

  plot(highTempData~dateData, type = "o",
        main="Lansing high temperatures",
        xlab="Day of Year",
        ylab="temperature (F)");
        ylab="temperature (F)");

  points(lowTempData~dateData, type="o");
}
```



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

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

  plot(highTempData~dateData, type = "o",
       main="Lansing high temperatures",
       xlab="Day Of Year",
       ylab="temperature (F)");

  points(lowTempData~dateData, type="o");
}
```

```
Data
weatherData        14 obs. of 6 variables
Values
  dateData         int [1:14] 86 87 88 89 90 91 92 93 94 95 ...
  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/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/2-6.
R')
> 
```

Fig 5: Plotting out both the high and low temperatures on one plot.

5.1 - Changing the limits of the axes

The low temperatures were added to the plot but some of the points are off the plot. This is because the y-axis was not adjusted to accommodate the new values; the original y-axis limits were based on the minimum and maximum from the ***highTempData*** (approximately 40 to 75), but the limits do not adjust when new points are added, even if the points go beyond the y-axis limits.

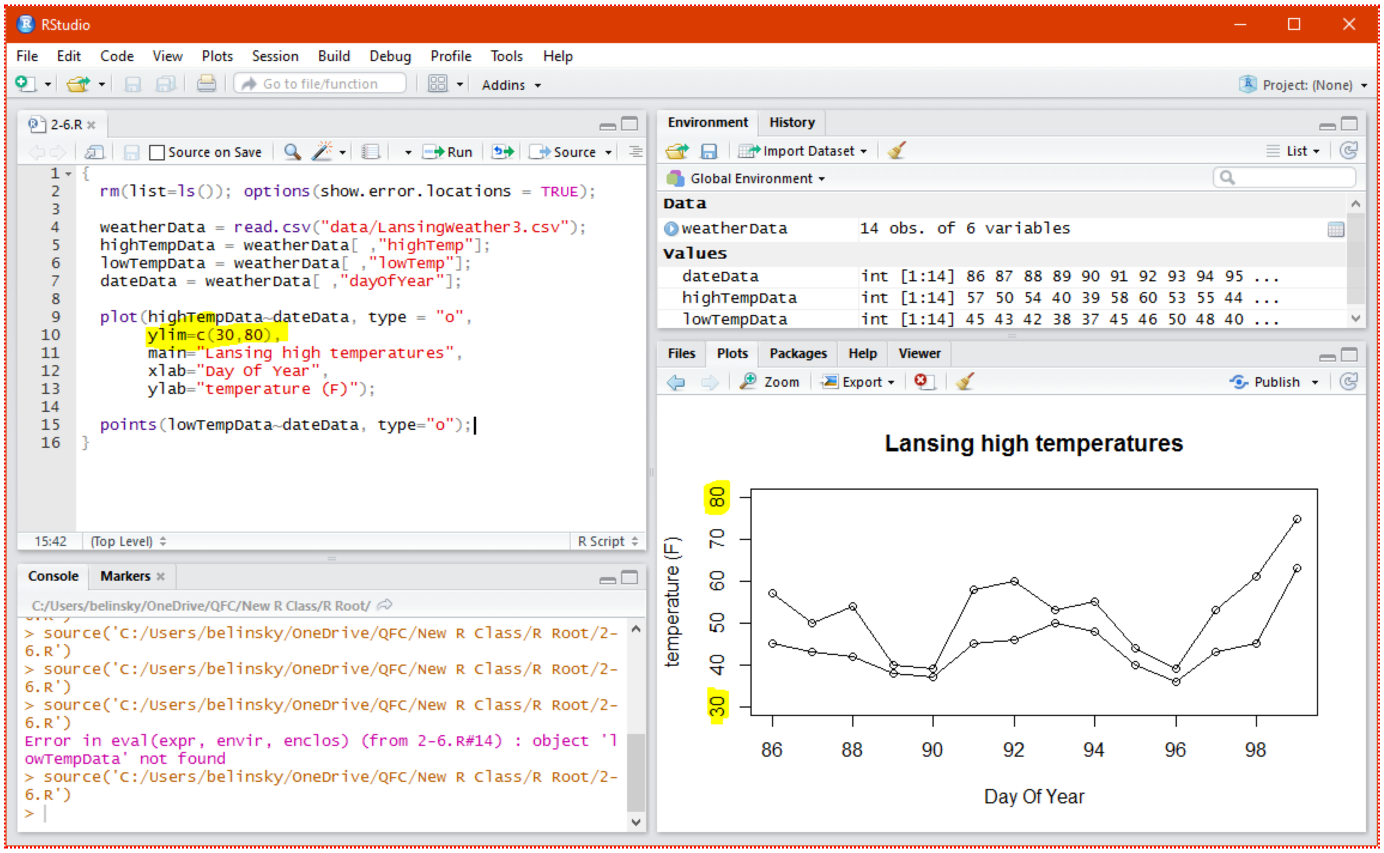
We need to change the y-axis limits in the original ***plot()*** to accommodate the new temperature values. The parameters for changing the axis limits are ***xlim*** and ***ylim***. In this case, we only need to change the ***ylim*** parameter. To change to ***ylim*** parameter, we set ***ylim*** to a vector of two values: ***ylim*** = ***c(min, max)***. In this case we will use 30 and 80.

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

  weatherData = read.csv("data/LansingWeather3.csv");
  highTempData = weatherData[ ,"highTemp"];
  lowTempData = weatherData[ ,"lowTemp"];
  dateData = weatherData[ ,"dayOfYear"];
  plot(highTempData~dateData, type = "o",
       ylim=c(30,80),
       main="Lansing high temperatures",
       xlab="Day Of Year",
       ylab="temperature (F)");

  points(lowTempData~dateData, type="o");
}
```

Note that the y-axis now goes from 30 to 80 due to assigning values to the parameter ***ylim.***



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

  weatherData = read.csv("data/Lansingweather3.csv");
  highTempData = weatherData[ ,"highTemp"];
  lowTempData = weatherData[ ,"lowTemp"];
  dateData = weatherData[ ,"dayOfYear"];

  plot(highTempData~dateData, type = "o",
       ylim=c(30,80),
       main="Lansing high temperatures",
       xlab="Day Of Year",
       ylab="temperature (F)");

  points(lowTempData~dateData, type="o");
}
```

```
Data
weatherData     14 obs. of 6 variables
Values
dateData        int [1:14] 86 87 88 89 90 91 92 93 94 95 ...
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/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/2-
6.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/2-
6.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/2-
6.R')
Error in eval(expr, envir, enclos) (from 2-6.R#14) : object 'l
owTempData' not found
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/2-
6.R')
> |
```

Fig 6: Changing the y-axis limits on a plot to accommodate new values

5.2 - Changing the color of the plots

We now have two lines and it would be nice to distinguish them. We are going to make the first plot blue and the second plot red using the ***col*** parameter.

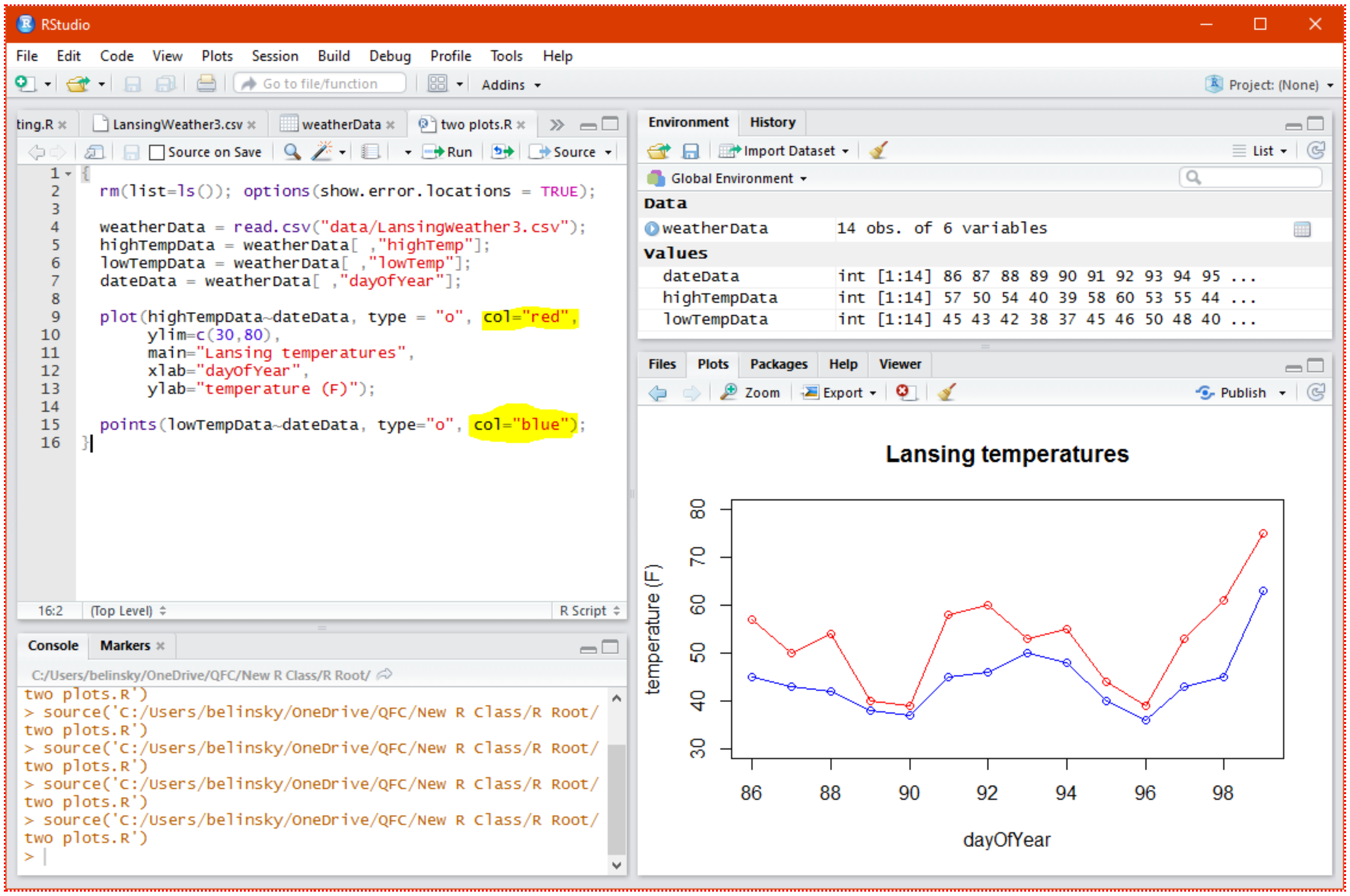
```
{

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

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

  plot(highTempData~dateData, type = "o", col="red",
          ylim=c(30,80),
          main="Lansing high temperatures",
          xlab="Day Of Year",
          ylab="temperature (F)");

  points(lowTempData~dateData, type="o", col="blue");
}
```



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

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

  plot(highTempData~dateData, type = "o", col="red",
       ylim=c(30,80),
       main="Lansing temperatures",
       xlab="dayOfYear",
       ylab="temperature (F)");

  points(lowTempData~dateData, type="o", col="blue");
}
```

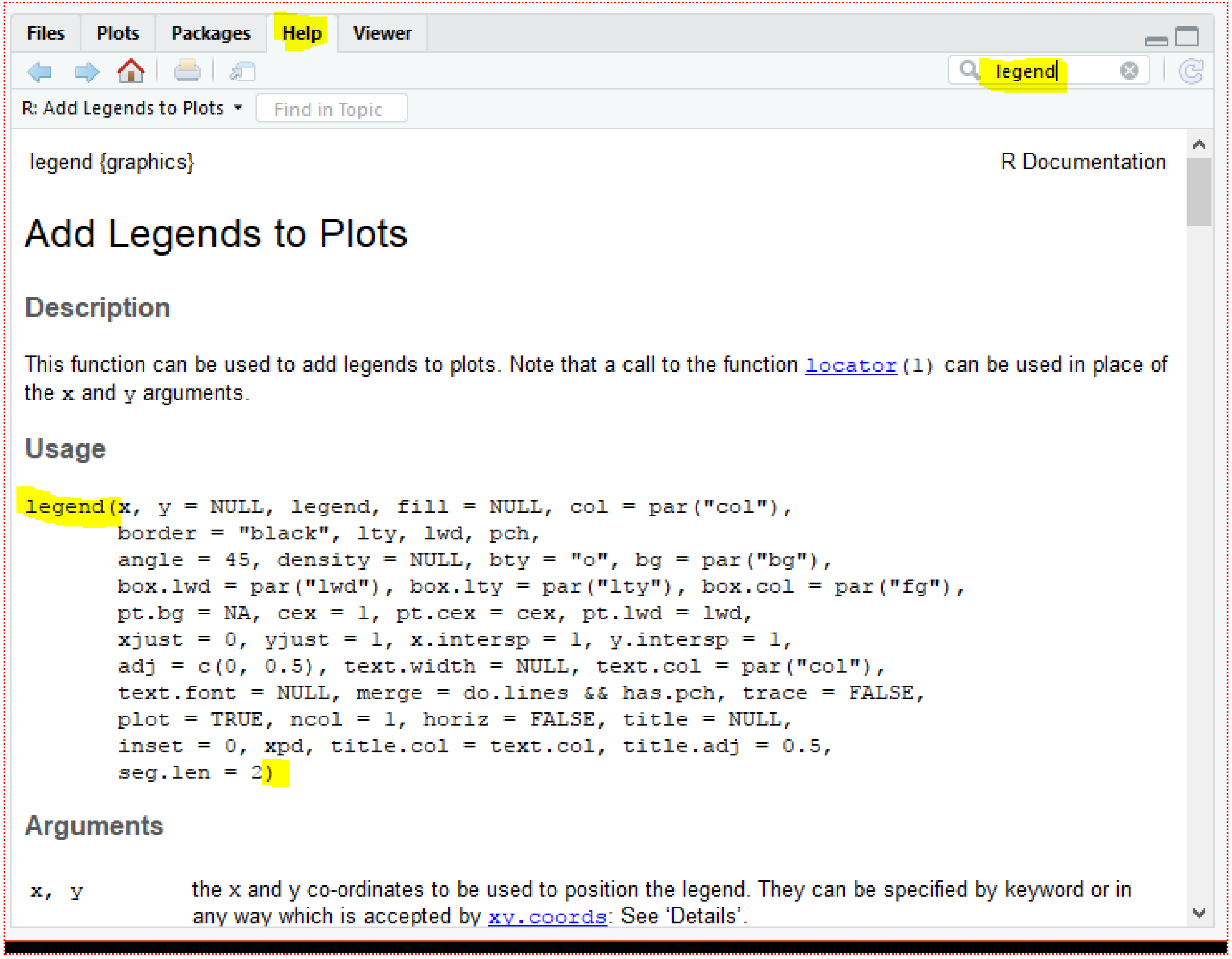
```
Data
weatherData      14 obs. of 6 variables
Values
  dateData       int [1:14] 86 87 88 89 90 91 92 93 94 95 ...
  highTempData   int [1:14] 57 50 54 40 39 58 60 53 55 44 ...
  lowTempData    int [1:14] 45 43 42 38 37 45 46 50 48 40 ...
```

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

Fig 7: Changing the color of the lines in a plot

6 - Adding a legend

Lastly, we would like a way to annotate the different lines and the best way is to use a legend. In R the function to add a legend to a plot is ***legend()***. ***legend()***, like ***plot()*** and ***points()*** has numerous properties that can be modified. We can find the help page for ***legend*** by switching to the Help tab in the Plot Window and typing "legend" in the search bar (*Fig.8*).



legend {graphics} R Documentation

Add Legends to Plots

Description

This function can be used to add legends to plots. Note that a call to the function `locator(1)` can be used in place of the `x` and `y` arguments.

Usage

```
legend(x, y = NULL, legend, fill = NULL, col = par("col"),
       border = "black", lty, lwd, pch,
       angle = 45, density = NULL, bty = "o", bg = par("bg"),
       box.lwd = par("lwd"), box.lty = par("lty"), box.col = par("fg"),
       pt.bg = NA, cex = 1, pt.cex = cex, pt.lwd = lwd,
       xjust = 0, yjust = 1, x.intersp = 1, y.intersp = 1,
       adj = c(0, 0.5), text.width = NULL, text.col = par("col"),
       text.font = NULL, merge = do.lines && has.pch, trace = FALSE,
       plot = TRUE, ncol = 1, horiz = FALSE, title = NULL,
       inset = 0, xpd, title.col = text.col, title.adj = 0.5,
       seg.len = 2)
```

Arguments

`x, y` the x and y co-ordinates to be used to position the legend. They can be specified by keyword or in any way which is accepted by `xy.coords`: See 'Details'.

Fig 8: Searching for "legend" in the Help Window.

For this example we are only going to use the parameters that are needed to get the legend up and functional. They are:

- ***x***: set the position of the legend
- ***legend***: labels in the legend
- ***lty***: line type (needed to put lines in the legend)
- ***pch***: point character (needed to put points in the legend)
- ***col***: color

Note: R does not provide the most intuitive parameter names!

We will start by positioning the legend and putting the labels in:

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

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

  dateData = weatherData[ ,"dayOfYear"];

  plot(highTempData~dateData, type = "o", col="red",
  ylim=c(30,80),
  main="Lansing temperatures",
  xlab="Day of Year",
  ylab="temperature (F)");

  points(lowTempData~dateData, type="o", col="blue");

  legend(x="topleft",                 # position
         legend=c("High Temp", "Low Temp")); # labels
}
```

Here we position the legend in the top-left corner:

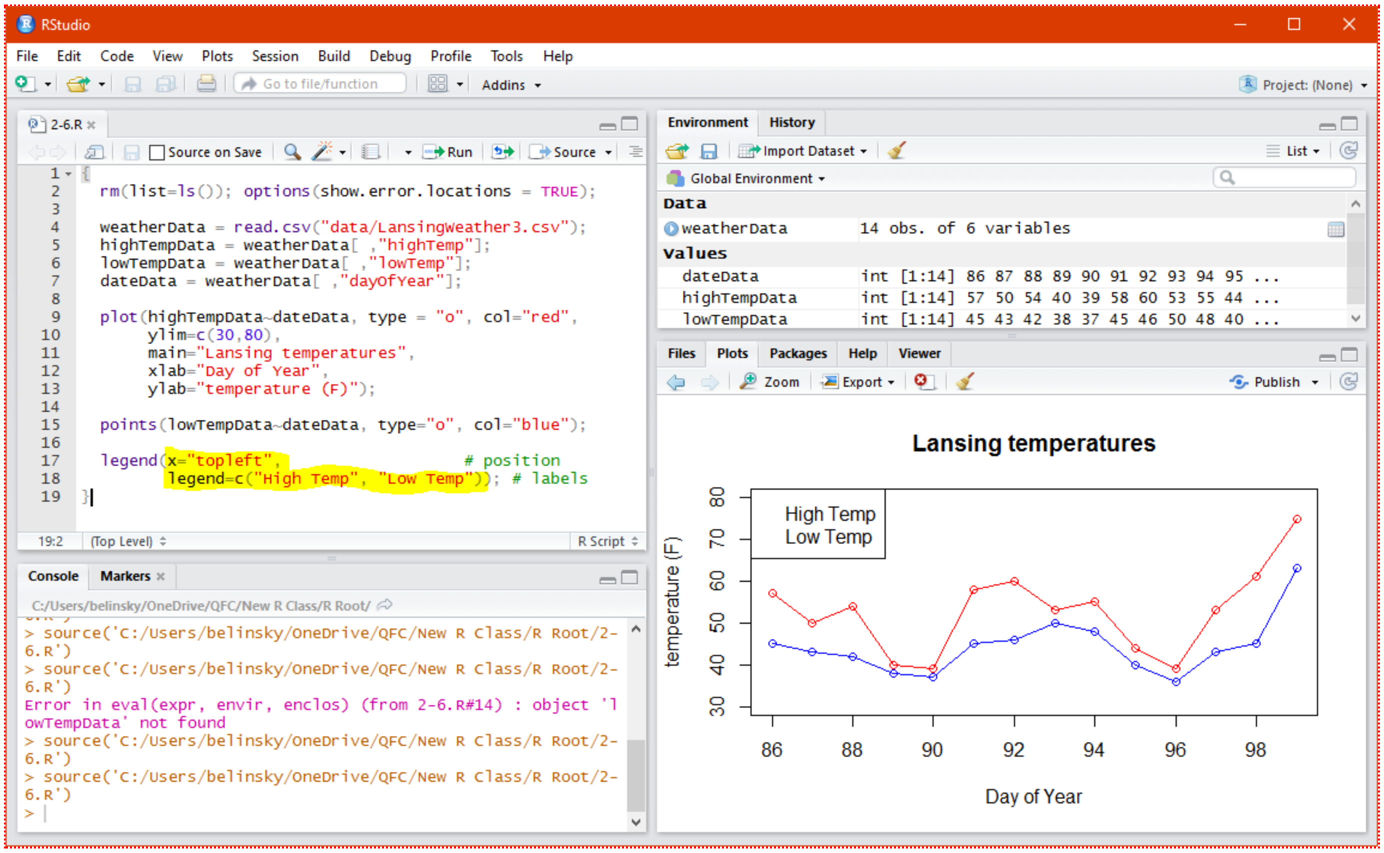
```
x="topleft"
```

There are nine values you can use for position: "topleft", "topright", "top, "center", "right", "left", "bottomleft", and "bottomright".

The labels

```
legend=c("High Temp", "Low Temp")
```

are put in a vector and R recognizes each value in the vector ("High Temp", "Low Temp) as a separate label.



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

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

  plot(highTempData~dateData, type = "o", col="red",
       ylim=c(30,80),
       main="Lansing temperatures",
       xlab="Day of Year",
       ylab="temperature (F)");

  points(lowTempData~dateData, type="o", col="blue");

  legend(x="topleft",                 # position
         legend=c("High Temp", "Low Temp")); # labels
}
```

```
Data
weatherData      14 obs. of 6 variables
Values
dateData         int [1:14] 86 87 88 89 90 91 92 93 94 95 ...
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/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/2-6.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/2-6.R')
Error in eval(expr, envir, enclos) (from 2-6.R#14) : object 'lowTempData' not found
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/2-6.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/2-6.R')
>
```

Fig 9: Legend placed in the plot

Now we want to put lines and points in the legend that look like the lines and points in the plot.

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

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

  plot(highTempData~dateData, type = "o", col="red",
  ylim=c(30,80),
  main="Lansing temperatures",
  xlab="Day Of Year",
  ylab="temperature (F)");

  points(lowTempData~dateData, type="o", col="blue");

  legend(x="topleft",                   # position
         legend=c("High Temp", "Low Temp"), # labels
         lty=c(1,1),                    # line type
         pch=c(1,1),                    # point type
         col=c("red","blue"));          # colors
}
```

lty and ***pch*** need to have values if you want a line and/or a point next to the label.

```
lty = c(1,1),
pch = c(1,1)
```

means we will use the first line type (solid) and the first point type (open circle) for both the **HighTemp** and **LowTemp** labels.

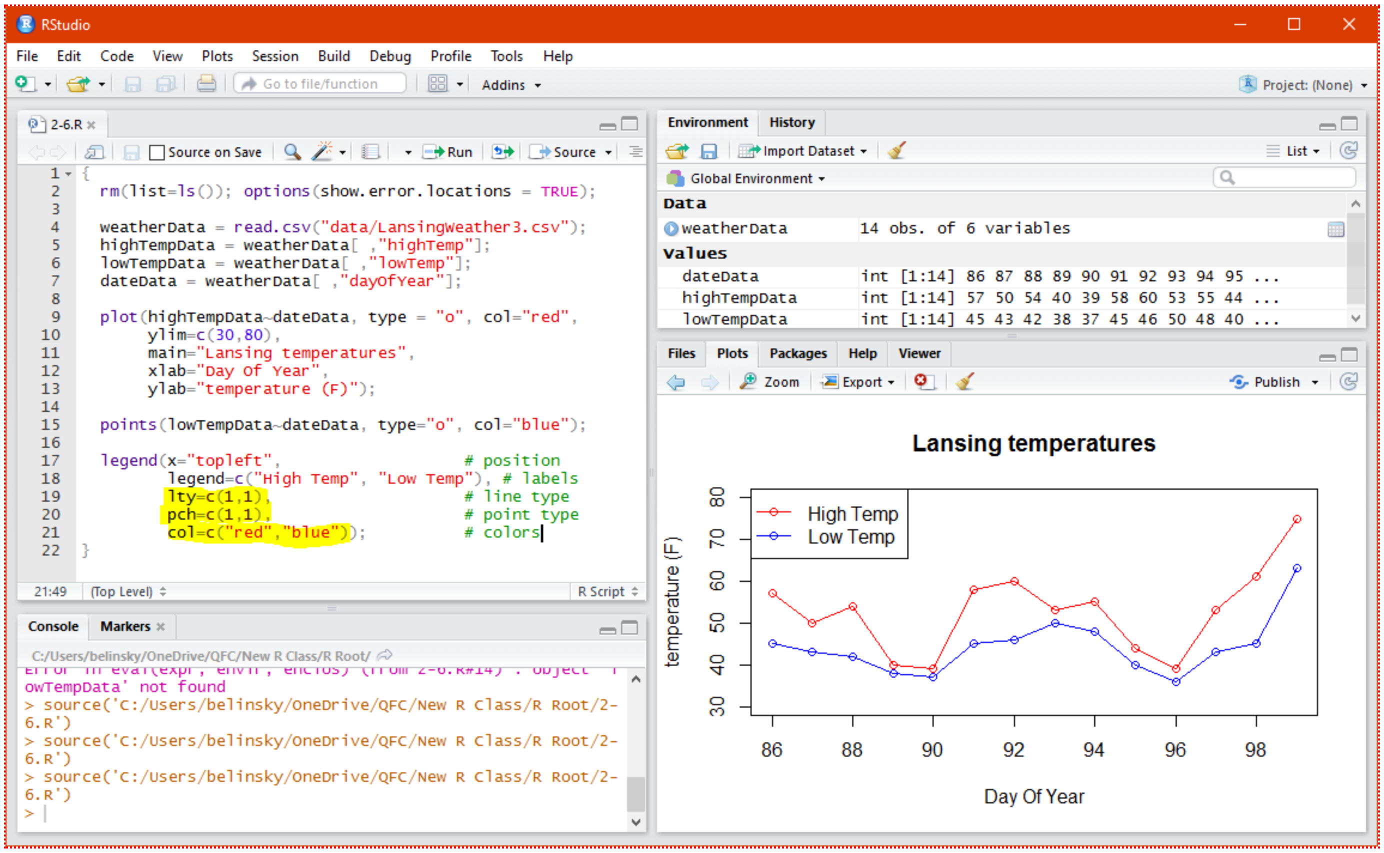
If we wanted to use the 3rd line type for High Temp and the 5th line type for Low Temp we would use:

```
lty = c(3, 5)
```

You can change the numbers and see how the lines and points change. A complete list of line and point types can be found at http://www.statmethods.net/advgraphs/parameters.html. Scroll down to the Plotting Symbols and Lines section of the webpage.

```
col = c("red", "blue")
```

mean the ***highTemp*** lines and points are red and the ***lowTemp*** lines and points are blue



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

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

  plot(highTempData~dateData, type = "o", col="red",
       ylim=c(30,80),
       main="Lansing temperatures",
       xlab="Day Of Year",
       ylab="temperature (F)");

  points(lowTempData~dateData, type="o", col="blue");

  legend(x="topleft",                    # position
         legend=c("High Temp", "Low Temp"), # labels
         lty=c(1,1),                      # line type
         pch=c(1,1),                      # point type
         col=c("red","blue"));            # colors
}
```

```
Data
weatherData          14 obs. of 6 variables
Values
  dateData           int [1:14] 86 87 88 89 90 91 92 93 94 95 ...
  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/
Error in eval(expr, envir, enclos) (from 2-6.R#14) : object 'l
owTempData' not found
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/2-
6.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/2-
6.R')
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/2-
6.R')
>
```

Fig 10: Legend added with lines and points

7 - Application

If you have any questions regarding this application or your Class Project, feel free to email them to the instructor here. You can attach the whole Root Folder as a zipped file.

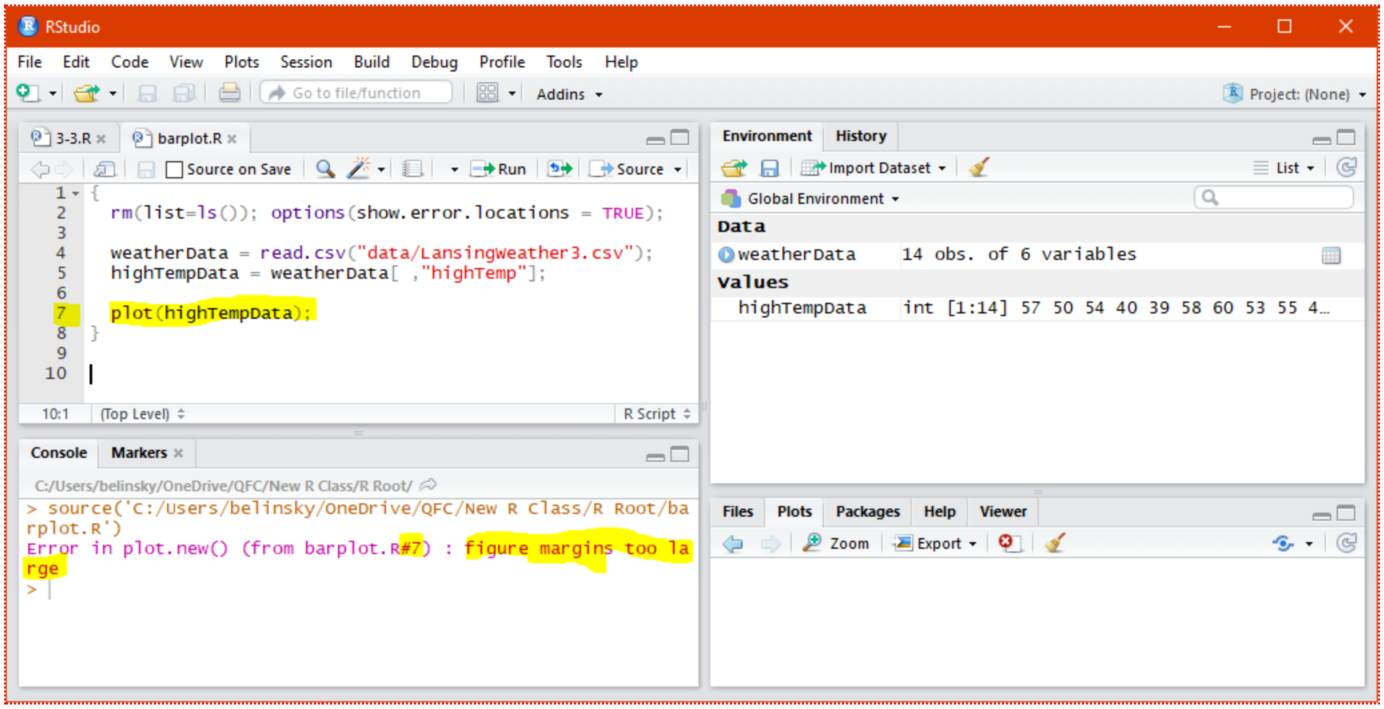
A) Produce a plot of ***precipitation*** vs ***humidity*** using the data from **LansingWeather3.csv**
- label the axes: humidity (%) and precipitation (inches)
- title the plot: Precipitation vs. Humidity
- make the plot points-only (no lines)

B) Add a ***changeInTemp*** vector to the ***highTemp*** and ***lowTemp*** plot created earlier in the lesson (*Fig.10*)
- ***changeInTemp*** is ***highTemp*** minus ***lowTemp***
- adjust the axis so all points from all three plots can be seen
- add ***changeInTemp*** to the legend
- make the color of the ***changeInTemp*** line purple in the plot and the legend
Note: The legend will probably cover part of the plot -- we will deal with this is the next plotting lesson.

*Save you script file as **app2-7.r** in the **scripts** folder of your RStudio Project for the class.*

8 - Trap: A very common plot error

If you have plots in your script then a common error in RStudio is:
Error in plot.new() (from <some file>.R##) : figure margins too large



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

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

  plot(highTempData);
}

```

```
weatherData    14 obs. of 6 variables
highTempData   int [1:14] 57 50 54 40 39 58 60 53 55 4…
```

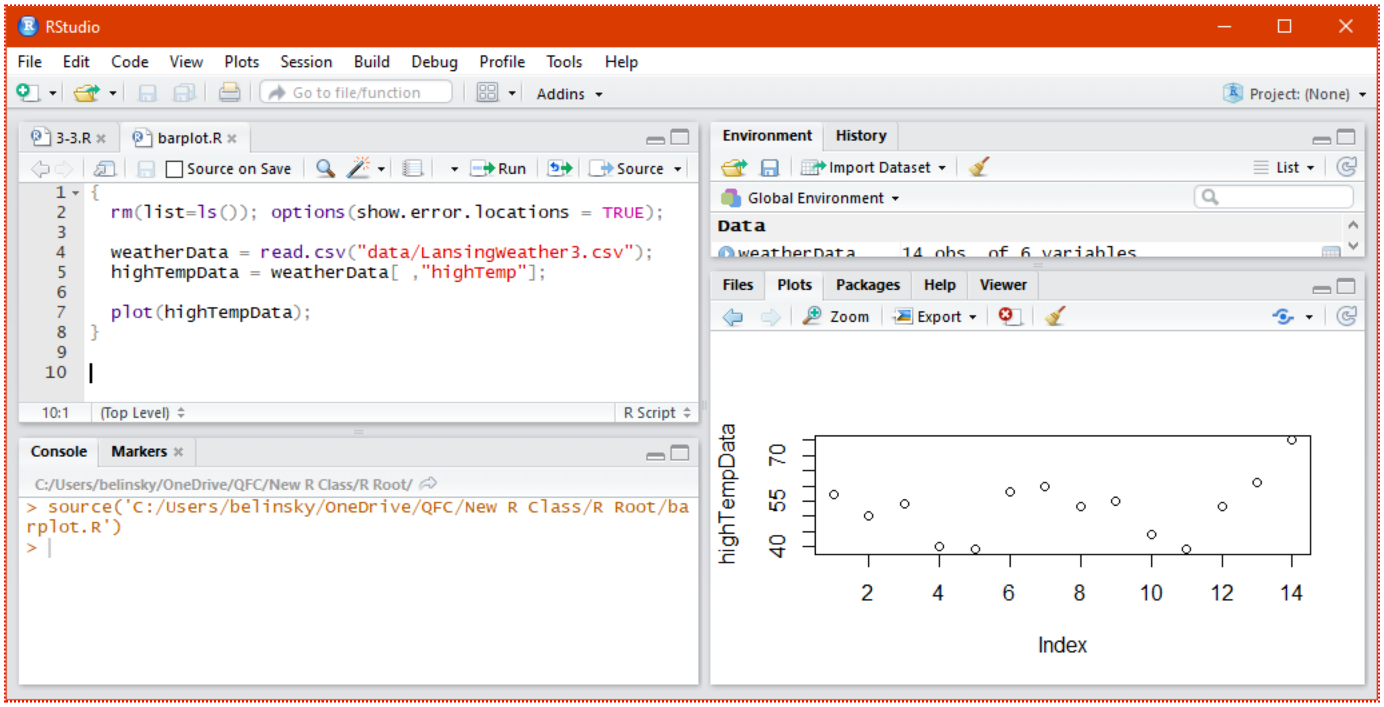
```
C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/
> source('C:/Users/belinsky/OneDrive/QFC/New R Class/R Root/barplot.R')
Error in plot.new() (from barplot.R#7) : figure margins too large
>
```

Fig 11: Figure margin too large error.

This is not an error in your script!

The error occurs because the RStudio Plot Window is too small to hold the plot. Technically speaking, the error is saying that the figure margins on your plot are larger than the Plot Window.

To fix the problem, just make the Plot Window larger.



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

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

  plot(highTempData);
}

```

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

Fig 12: Increased Plot Window size and the error goes away.

Note: Just like with any error, all code after the error will not be executed. So, if your script is 100 lines long and you are plotting something on line 10, then lines 11-100 will not be executed.

One possible workaround for this error is described below in *Extension: Full-sized window for plots.*

9 - Extension: Full-sized window for plots

The Plot Window in RStudio can often be too small or cumbersome for our needs. Sometimes it is easier to open up a separate window the holds any plots created in the script.

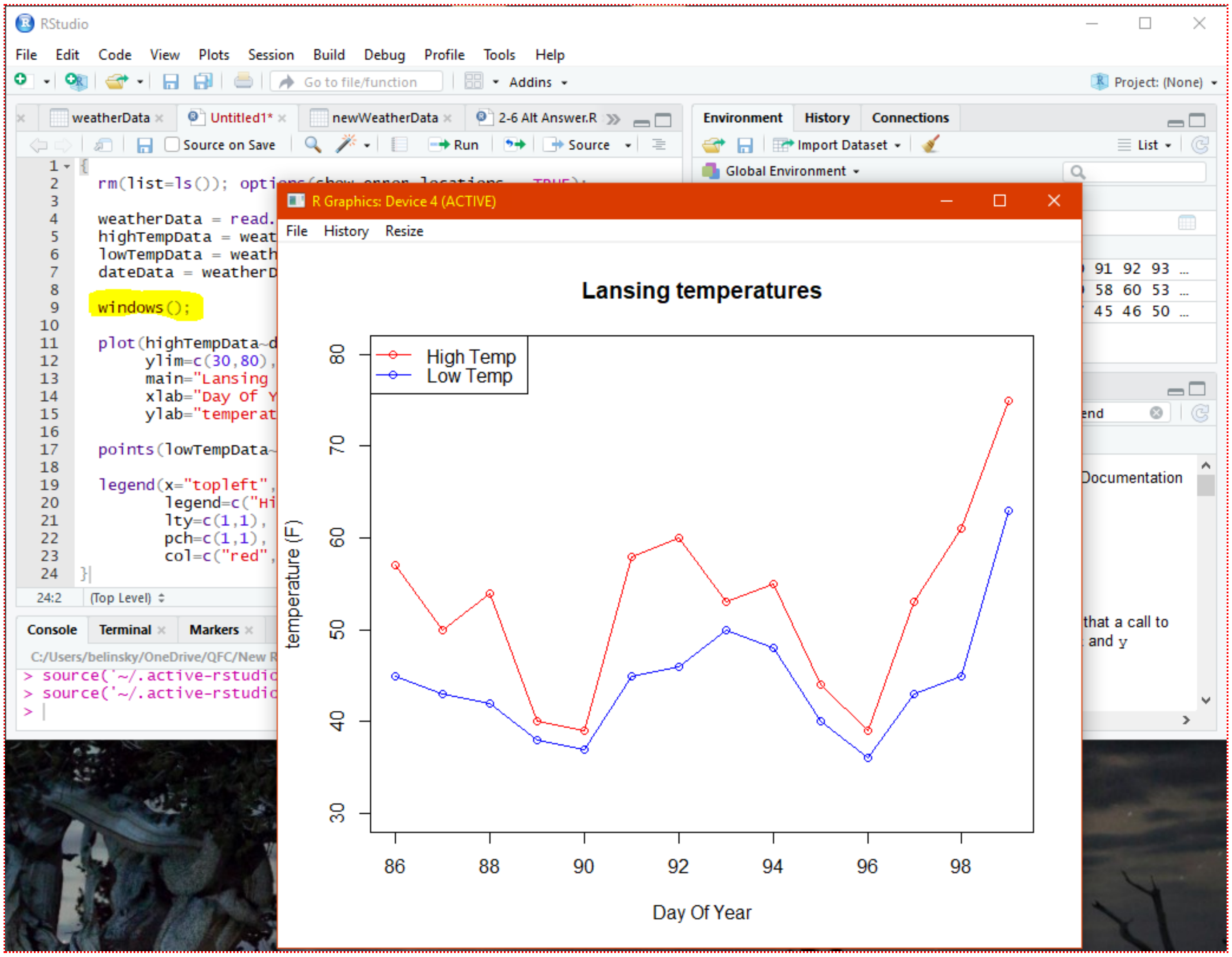
We can do that by adding to the script:

```r
windows();  # if this is a windows machine
```

or

```r
quartz();   # if this is a MAC machine
```

Note: These functions must be executed before the plots in your script.



```r
{
  rm(list=ls()); opti

  weatherData = read.
  highTempData = weat
  lowTempData = weath
  dateData = weatherD

  windows();

  plot(highTempData~d
       ylim=c(30,80),
       main="Lansing
       xlab="Day of Y
       ylab="temperat

  points(lowTempData~

  legend(x="topleft",
         legend=c("Hi
         lty=c(1,1),
         pch=c(1,1),
         col=c("red",
}
```

```
91 92 93 ...
58 60 53 ...
45 46 50 ...
```

```
C:/Users/belinsky/OneDrive/QFC/New R
> source('~/.active-rstudic
> source('~/.active-rstudic
> 
```

*Fig 13: Opening plots in a separate window using **windows()**.*