01-03: Plot Styles

1 - Purpose

- Create a scatterplot with continuous values
- Changing styles on points, lines, and text
- Adding a regression line
- Adding titles and labels

2 - Getting data from a CSV file

For this lesson, we will be using weather data obtained from NOAA/NCDC for Lansing, Michigan in 2016. You can download the csv file here. Save the file, ***LansingNOAA2016.csv***, to the ***data*** folder in your Root Folder.

2.1 - Start new script in the project and include the data

Open the RStudio Project you have been using the last couple of lessons.

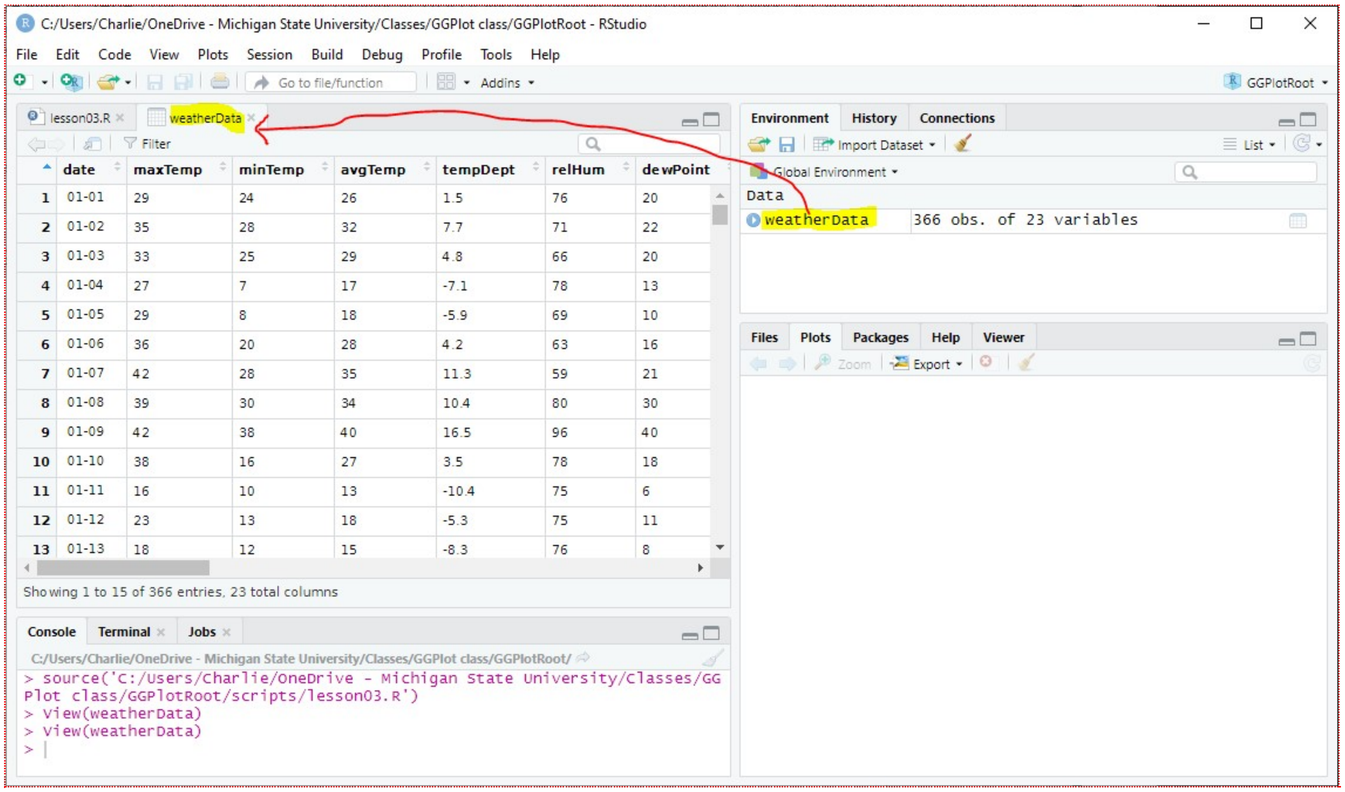
Add a new script file to your project:
1. click ***File*** -> ***New File*** -> ***R Script***
2. save the new blank script file as ***lesson03.r*** inside the ***scripts*** folder

The first two lines in the script file includes:
1. reading in the ***reference.r*** script (like last lesson)
2. saving the data from ***LansingNOAA2016.csv*** to a dataframe variable called ***weatherData***

```
source(file="scripts/reference.R");   # include the reference.r file
weatherData = read.csv(file="data/LansingNOAA2016.csv",
                       stringsAsFactors = FALSE);
```

In ***read.csv()***, the default value for ***stringsAsFactors*** is **TRUE**. This means that R treats columns in a data frame that contain any non-numeric values as categorical (i.e., a factor). This instructor believes that this behavior is too aggressive and makes it harder in RStudio to view the values in a columns. So, I almost always set ***stringsAsFactors*** to ***FALSE***. If needed, we can later convert a column's values to factors, which we do in later lessons.

When we execute the script with the two lines, the data from the CSV file is saved to the data frame, ***weatherData. weatherData*** appears in the Environment Window and double-clicking on it opens it up in the Main Window (*fig 1*).


C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
GGPlotRoot
lesson03.R
weatherData
Filter
Environment History Connections
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
Zoom
Export
Console Terminal Jobs

	date	maxTemp	minTemp	avgTemp	tempDept	relHum	dewPoint
1	01-01	29	24	26	1.5	76	20
2	01-02	35	28	32	7.7	71	22
3	01-03	33	25	29	4.8	66	20
4	01-04	27	7	17	-7.1	78	13
5	01-05	29	8	18	-5.9	69	10
6	01-06	36	20	28	4.2	63	16
7	01-07	42	28	35	11.3	59	21
8	01-08	39	30	34	10.4	80	30
9	01-09	42	38	40	16.5	96	40
10	01-10	38	16	27	3.5	78	18
11	01-11	16	10	13	-10.4	75	6
12	01-12	23	13	18	-5.3	75	11
13	01-13	18	12	15	-8.3	76	8

Showing 1 to 15 of 366 entries, 23 total columns

```
Data
weatherData        366 obs. of 23 variables
```

```
C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot/
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GG
Plot class/GGPlotRoot/scripts/lesson03.R')
> View(weatherData)
> View(weatherData)
> |
```

Fig 1: Viewing the data frame in RStudio

3 - Plot Temperature vs Humidity (continuous x-axis)

Like last lesson we will

- Make a call to initialize a GGPlot canvas, using ***ggplot()*** with the ***weatherData*** data frame:

```
ggplot(data=weatherData)
```

- Add a scatterplot component, ***geom_point(),*** that uses the ***mapping*** subcomponent to describe how the data gets represented in the plot: ***relHum*** *(y-axis)* vs ***avgTemp*** (x-axis):

```
geom_point( mapping=aes(x=avgTemp, y=relHum) )
```

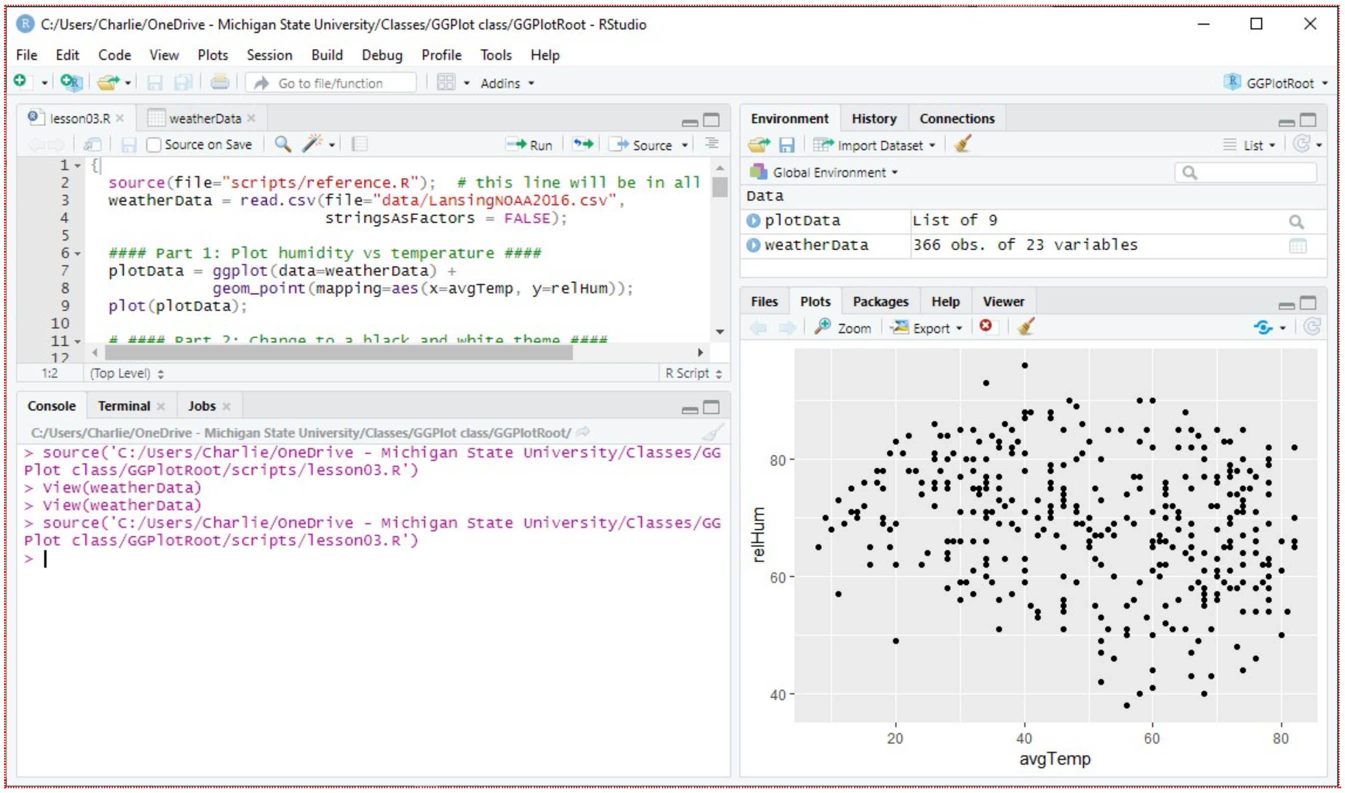
- Save the ggplot canvas information to a variable, ***plotData***, and use ***plot()*** to display the canvas:

```
plotData = ggplot( data=weatherData ) +
           geom_point( mapping=aes(x=avgTemp, y=relHum) );
plot(plotData);
```

Putting all the code together:

```
source(file="scripts/reference.R");   # include the reference.r file
weatherData = read.csv(file="data/LansingNOAA2016.csv",
                        stringsAsFactors = FALSE);

#### Part 1: Plot humidity vs temperature ####
plotData = ggplot( data=weatherData ) +
            geom_point( mapping=aes(x=avgTemp, y=relHum) );
plot(plotData);
```



C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot - RStudio

```
{
  source(file="scripts/reference.R");  # this line will be in all
  weatherData = read.csv(file="data/LansingNOAA2016.csv",
                         stringsAsFactors = FALSE);

  #### Part 1: Plot humidity vs temperature ####
  plotData = ggplot(data=weatherData) +
             geom_point(mapping=aes(x=avgTemp, y=relHum));
  plot(plotData);

  # #### Part 2: Change to a black and white theme ####
```

```
Data
plotData      List of 9
weatherData   366 obs. of 23 variables
```

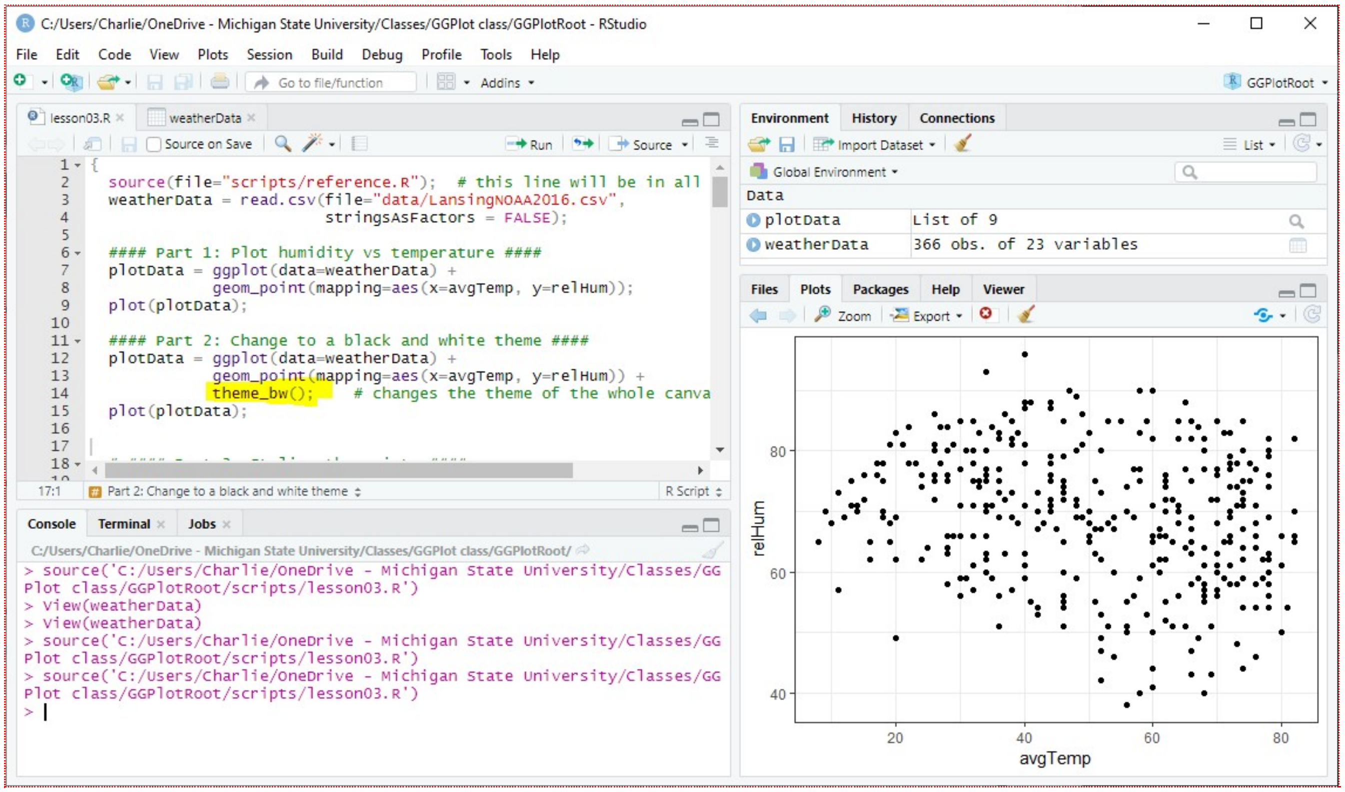
```
C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot/
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GG
Plot class/GGPlotRoot/scripts/lesson03.R')
> View(weatherData)
> View(weatherData)
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GG
Plot class/GGPlotRoot/scripts/lesson03.R')
> 
```

Fig 2: Scatterplot of humidity vs temperature

3.1 - Resetting the theme

I am not a big fan of GGPlot's default grey canvas theme, but the theme can be easily changed. In this case, I will change to a more standard black and white theme (*fig 3*) using the GGPlot component ***theme_bw()***, which is one of many themes that can be applied to the plot.

```
#### Part 2: Change to a black and white theme ####
plotData = ggplot( data=weatherData ) +
            geom_point( mapping=aes(x=avgTemp, y=relHum) ) +
            theme_bw();     # changes the theme of the whole canvas
plot(plotData);
```


C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
lesson03.R
weatherData
Source on Save
Run
Source
Environment History Connections
Import Dataset
Global Environment
Files Plots Packages Help Viewer
Zoom
Export
Console Terminal Jobs
17:1
Part 2: Change to a black and white theme
R Script
relHum
avgTemp

```
{
  source(file="scripts/reference.R");  # this line will be in all
  weatherData = read.csv(file="data/LansingNOAA2016.csv",
                         stringsAsFactors = FALSE);

  #### Part 1: Plot humidity vs temperature ####
  plotData = ggplot(data=weatherData) +
             geom_point(mapping=aes(x=avgTemp, y=relHum));
  plot(plotData);

  #### Part 2: Change to a black and white theme ####
  plotData = ggplot(data=weatherData) +
             geom_point(mapping=aes(x=avgTemp, y=relHum)) +
             theme_bw();    # changes the theme of the whole canva
  plot(plotData);
```

```
Data
plotData        List of 9
weatherData     366 obs. of 23 variables
```

```
C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot/
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GG
Plot class/GGPlotRoot/scripts/lesson03.R')
> View(weatherData)
> View(weatherData)
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GG
Plot class/GGPlotRoot/scripts/lesson03.R')
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GG
Plot class/GGPlotRoot/scripts/lesson03.R')
> |
```

Fig 3: Using the black-white theme in ggplot

4 - Styling the points

Pretty much everything in a GGPlot canvas can be styled. We are going to start by styling the points in the ***geom_point()*** component using the subcomponents: ***color***, ***size***, and ***shape***.

The possible values for these subcomponents are:

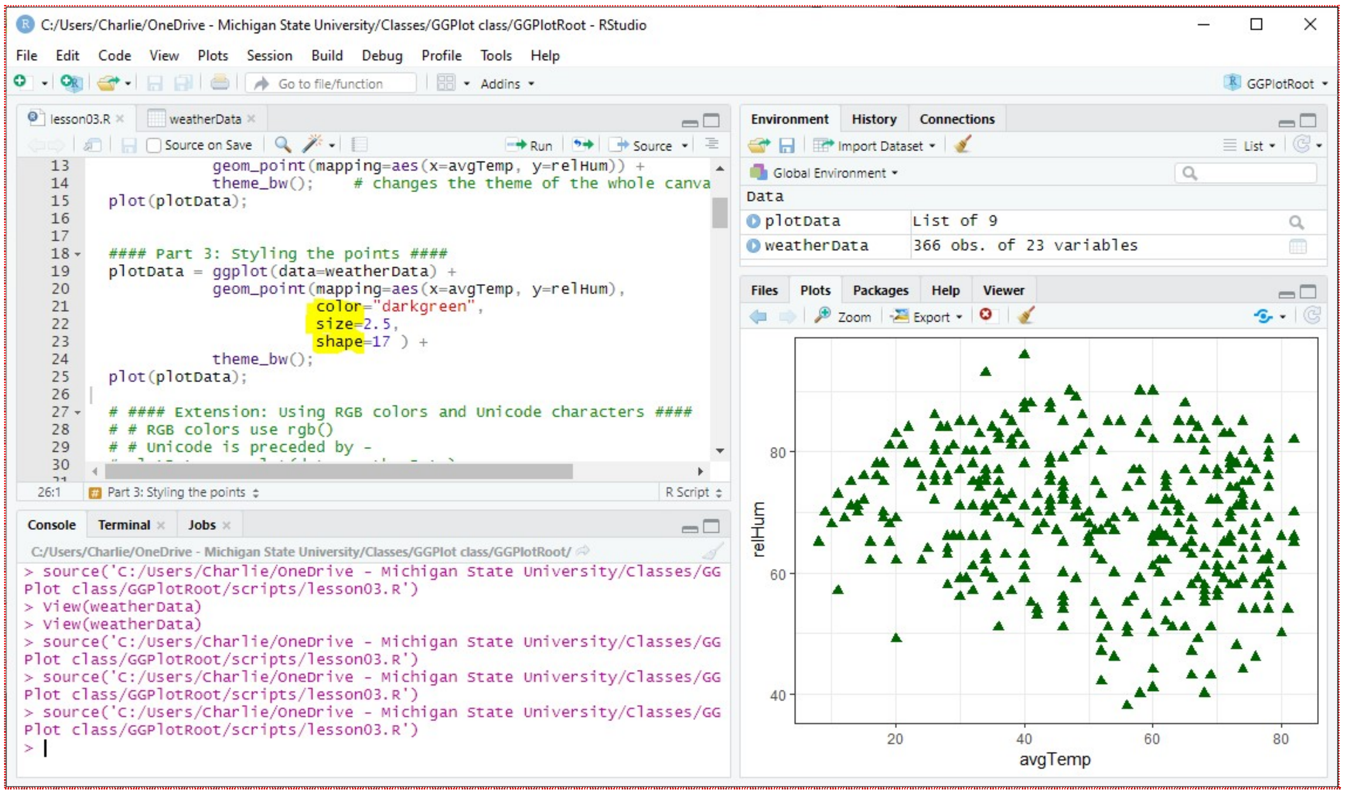
- ***color***: pretty much any color name you can think of can be used -- here is a complete list of all the color names.
- ***size***: a number that acts as a multiplier -- so ***size=3*** is 3 times the original size, ***size=0.5*** is half the original size
- ***shape***: there are two options for shape
  - the standard R shapes (numbered 0 through 25)
  - any character on the keyboard in quotes (so, ***shape = "~"*** would use the tilde as a point character)

```
#### Part 3: Styling the points ####
plotData = ggplot( data=weatherData ) +
           geom_point( mapping=aes(x=avgTemp, y=relHum),
                       color="darkgreen",
                       size=2.5,
                       shape=17 ) +
           theme_bw();
plot(plotData);
```

In this example, we set four subcomponents in ***geom_point()***:

- ***mapping***: maps the column data ***relHum*** vs ***avgTemp*** to the plot

- ***color***: changed the point color to **darkgreen**
- ***size***: multiplied the point size by **2.5**
- ***shape***: changed the point shape to filled triangles (**17**)


C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
GGPlotRoot
lesson03.R
weatherData
Source on Save
Run
Source
Environment History Connections
Import Dataset
List
Global Environment
Files Plots Packages Help Viewer
Zoom
Export
relHum
avgTemp
26:1
Part 3: Styling the points
R Script
Console Terminal Jobs

```
          geom_point(mapping=aes(x=avgTemp, y=relHum)) +
          theme_bw();    # changes the theme of the whole canva
plot(plotData);


#### Part 3: Styling the points ####
plotData = ggplot(data=weatherData) +
          geom_point(mapping=aes(x=avgTemp, y=relHum),
                     color="darkgreen",
                     size=2.5,
                     shape=17 ) +
          theme_bw();
plot(plotData);

# #### Extension: Using RGB colors and Unicode characters ####
# # RGB colors use rgb()
# # Unicode is preceded by -
```

```
Data
plotData      List of 9
weatherData   366 obs. of 23 variables
```

```
C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot/
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GG
Plot class/GGPlotRoot/scripts/lesson03.R')
> View(weatherData)
> View(weatherData)
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GG
Plot class/GGPlotRoot/scripts/lesson03.R')
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GG
Plot class/GGPlotRoot/scripts/lesson03.R')
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GG
Plot class/GGPlotRoot/scripts/lesson03.R')
> 
```

Fig 4: Styling the points on a plot

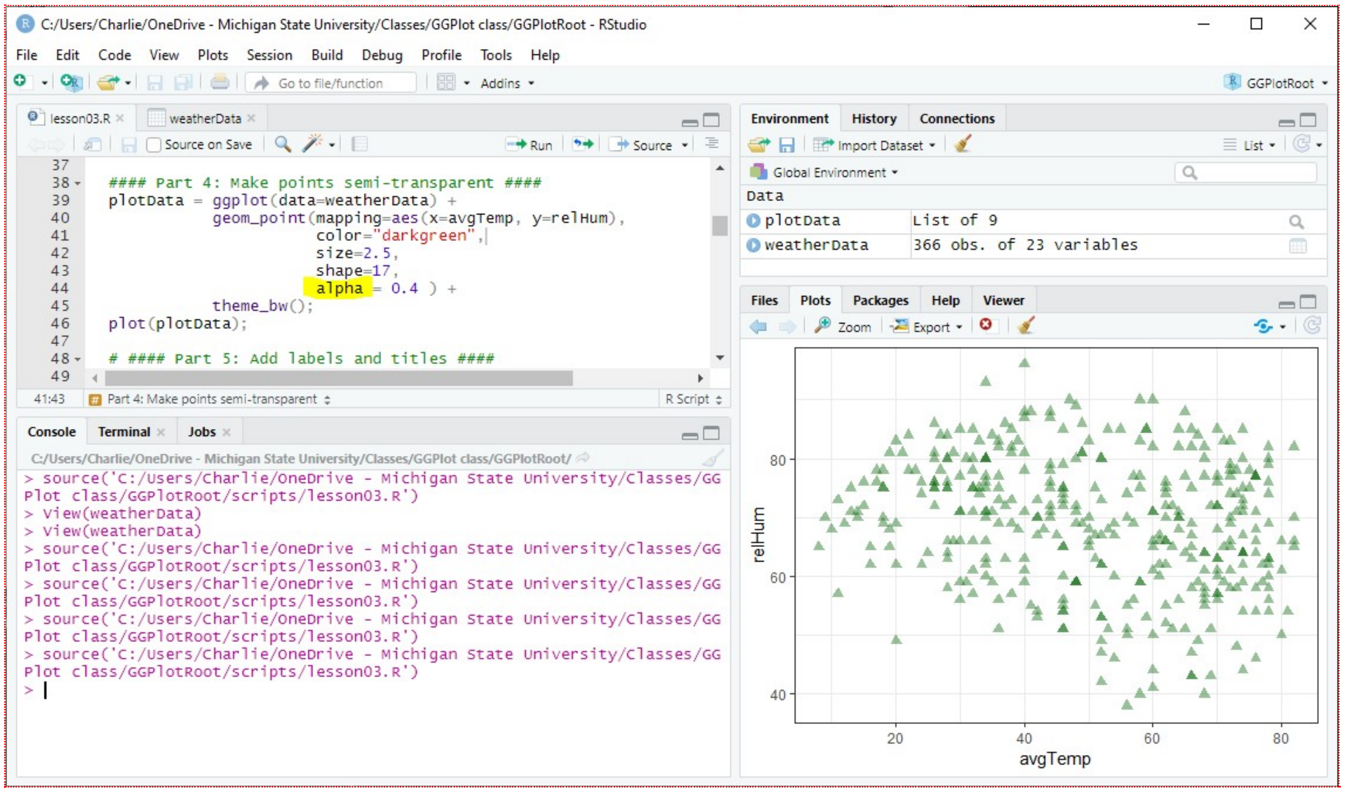
If you want even more options for ***color*** and ***shape***, then look at *Extension: rgb colors and Unicode characters*

4.1 - Adding transparency (opacity) to the point

When you have a lot of points overlapping each-other, it is often helpful to make the points partially transparent so you can better see the clustering of points. To do this we change the ***alpha*** subcomponent -- ***alpha*** is a value between **0** (completely transparent) and **1** (completely opaque).

alpha is also a subcomponent in the ***geom_point()*** component. For this example, we use an ***alpha*** of **0.4** (i.e., 40% opaque):

```
#### Part 4: Make points semi-transparent ####
plotData = ggplot( data=weatherData ) +
           geom_point( mapping=aes(x=avgTemp, y=relHum),
                       color="darkgreen",
                       size=2.5,
                       shape=17,
                       alpha = 0.4 ) +
           theme_bw();
plot(plotData);
```



```
#### Part 4: Make points semi-transparent ####
plotData = ggplot(data=weatherData) +
           geom_point(mapping=aes(x=avgTemp, y=relHum),
                      color="darkgreen",
                      size=2.5,
                      shape=17,
                      alpha = 0.4 ) +
           theme_bw();
plot(plotData);

# #### Part 5: Add labels and titles ####
```

```
C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot/
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GG
Plot class/GGPlotRoot/scripts/lesson03.R')
> View(weatherData)
> View(weatherData)
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GG
Plot class/GGPlotRoot/scripts/lesson03.R')
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GG
Plot class/GGPlotRoot/scripts/lesson03.R')
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GG
Plot class/GGPlotRoot/scripts/lesson03.R')
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GG
Plot class/GGPlotRoot/scripts/lesson03.R')
> 
```

```
Data
plotData       List of 9
weatherData    366 obs. of 23 variables
```

Fig 5: Adding an alpha value (transparency) to the points to better see clustering

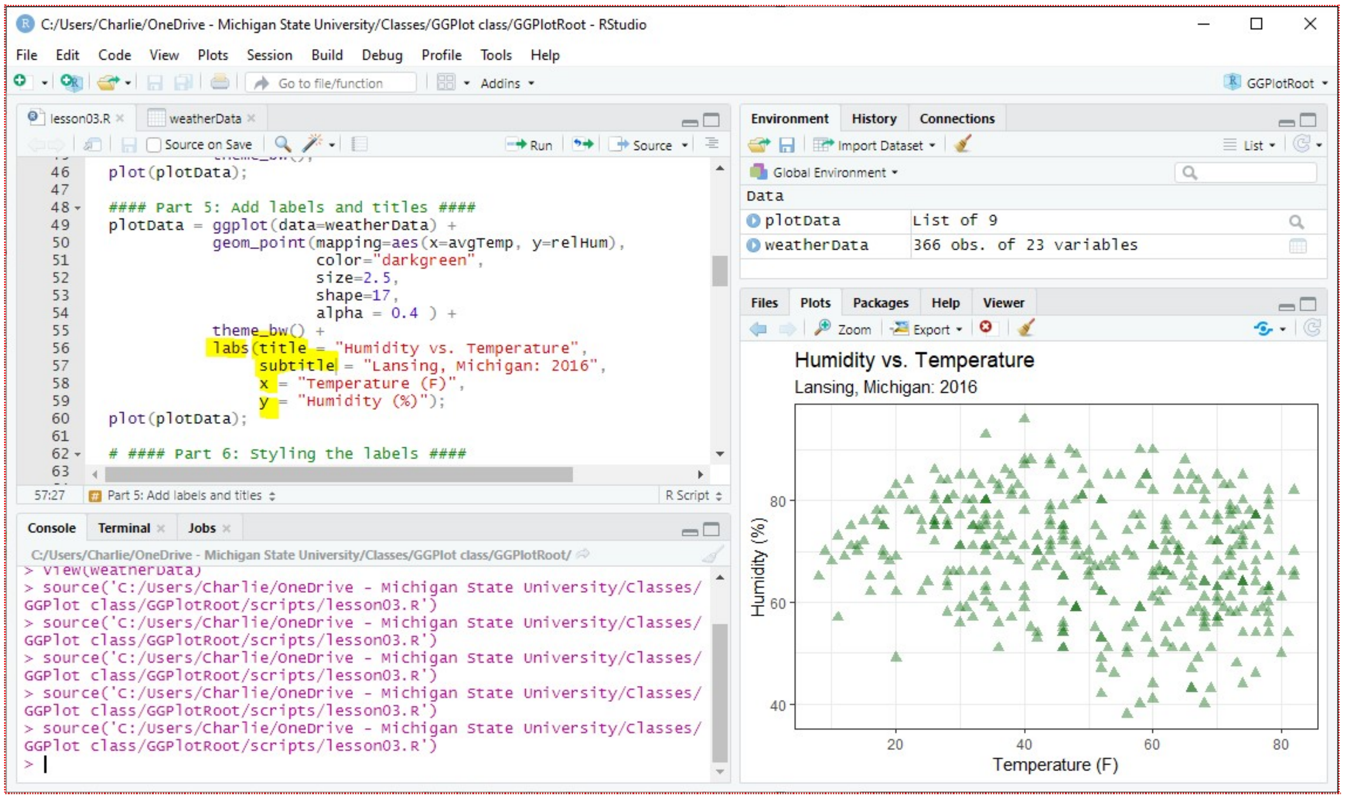
5 - Labels and Titles

We can add labels to the axes and a title to the plot using the ***labs()*** component.

The subcomponents in the ***labs()*** component that we will modify are:

- ***title***: title of plot
- ***subtitle***: subtitle of plot
- ***x***: x-axis label
- ***y***: y-axis label

```
#### Part 5: Add labels and titles ####
plotData = ggplot( data=weatherData ) +
           geom_point( mapping=aes(x=avgTemp, y=relHum),
                       color="darkgreen",
                       size=2.5,
                       shape=17,
                       alpha = 0.4 ) +
           theme_bw() +
           labs(title = "Humidity vs. Temperature",
                subtitle = "Lansing, Michigan: 2016",
                x = "Temperature (F)",
                y = "Humidity (%)");
plot(plotData);
```


C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
lesson03.R
weatherData
Source on Save
Run
Source
Environment History Connections
Import Dataset
Global Environment
Files Plots Packages Help Viewer
Zoom
Export
Console Terminal Jobs
Humidity vs. Temperature
Lansing, Michigan: 2016
Humidity (%)
Temperature (F)
80
60
40
20
40
60
80

```
plot(plotData);

#### Part 5: Add labels and titles ####
plotData = ggplot(data=weatherData) +
           geom_point(mapping=aes(x=avgTemp, y=relHum),
                      color="darkgreen",
                      size=2.5,
                      shape=17,
                      alpha = 0.4 ) +
           theme_bw() +
           labs(title = "Humidity vs. Temperature",
                subtitle = "Lansing, Michigan: 2016",
                x = "Temperature (F)",
                y = "Humidity (%)");
plot(plotData);

# #### Part 6: Styling the labels ####
```

plotData	List of 9
weatherData	366 obs. of 23 variables

```
> View(weatherData)
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot/scripts/lesson03.R')
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot/scripts/lesson03.R')
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot/scripts/lesson03.R')
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot/scripts/lesson03.R')
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot/scripts/lesson03.R')
> 
```

Fig 6: Adding labels and titles to a plot

*Note: In a previous lesson, we used the component **ggtitle()** to change the title on the plot -- in this lesson we use **labs()**. In GGPlot, there are often multiple ways to perform the same task.*

5.1 - Styling text labels

We can change the default font styles using the ***theme()*** component. ***theme()*** is probably the most commonly used component and can be used to modify pretty much every element in the ggplot canvas -- and there are many elements than can be modified.

We are going to look at many of the subcomponents in ***theme()*** in later lessons, for now we are just going to look at these four:

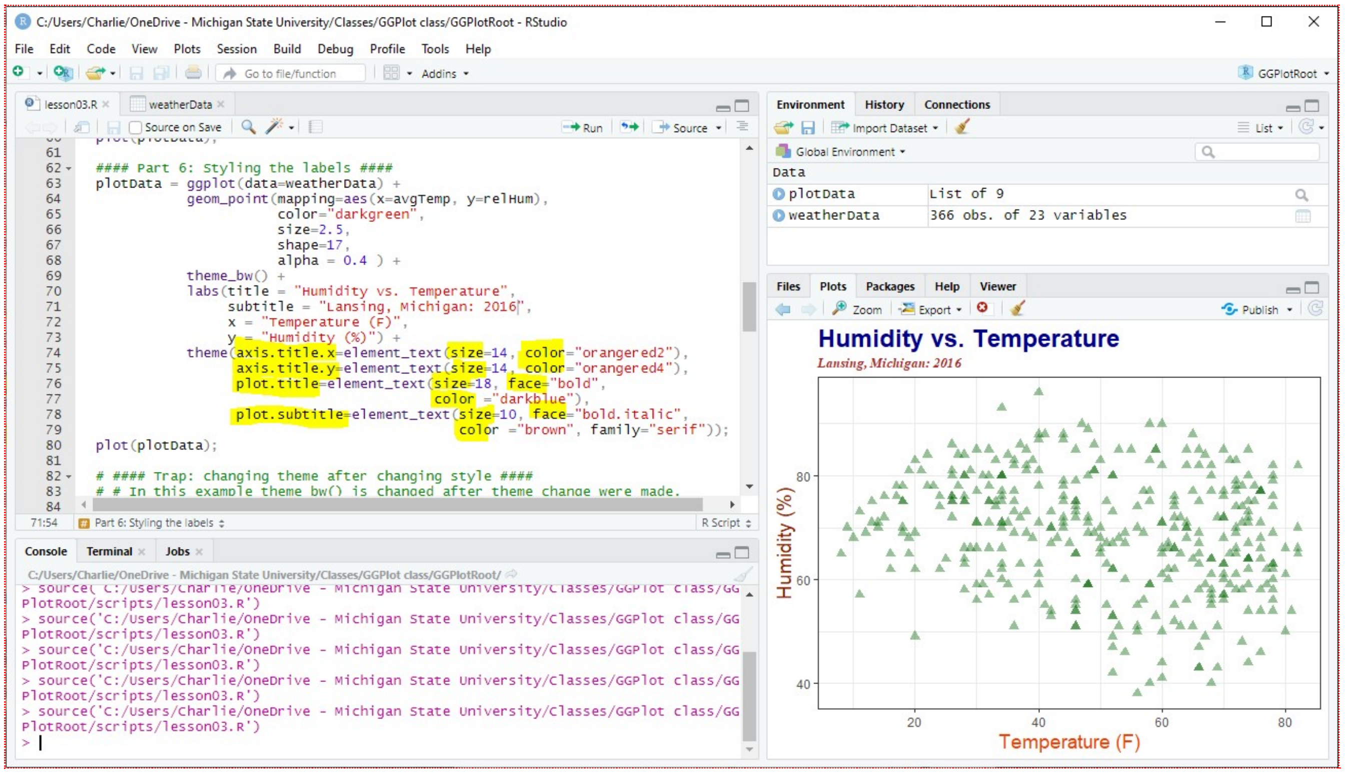
- ***axis.title.x***: x-axis label style
- ***axis.title.y***: y-axis label style
- ***plot.title***: plot title style
- ***plot.subtitle***: plot subtitle style

And for the above elements, we will modify the:

- ***size***: font size in pixels
- ***color***: font color
- ***face***: font modification (bold, italic, bold.italic)
- ***family***: font type (this should be used with caution!)

```
#### Part 6: Styling the labels ####
plotData = ggplot( data=weatherData ) +
           geom_point( mapping=aes(x=avgTemp, y=relHum),
                        color="darkgreen",
                        size=2.5,
                        shape=17,
                        alpha = 0.4 ) +
            theme_bw() +
            labs(title = "Humidity vs. Temperature",
                 subtitle = "Lansing, Michigan: 2016",
                 x = "Temperature (F)",
                 y = "Humidity (%)") +
            theme(axis.title.x=element_text(size=14, color="orangered2"),
                  axis.title.y=element_text(size=14, color="orangered4"),
                  plot.title=element_text(size=18, face="bold",
                                          color ="darkblue"),
                  plot.subtitle=element_text(size=10, face="bold.italic",
                                             color ="brown", family="serif"));
plot(plotData);
```

Trap: Changing theme after using a theme component



```
#### Part 6: Styling the labels ####
plotData = ggplot(data=weatherData) +
           geom_point(mapping=aes(x=avgTemp, y=relHum),
                      color="darkgreen",
                      size=2.5,
                      shape=17,
                      alpha = 0.4 ) +
           theme_bw() +
           labs(title = "Humidity vs. Temperature",
                subtitle = "Lansing, Michigan: 2016",
                x = "Temperature (F)",
                y = "Humidity (%)") +
           theme(axis.title.x=element_text(size=14, color="orangered2"),
                 axis.title.y=element_text(size=14, color="orangered4"),
                 plot.title=element_text(size=18, face="bold",
                                         color ="darkblue"),
                 plot.subtitle=element_text(size=10, face="bold.italic",
                                            color ="brown", family="serif"));
plot(plotData);

# #### Trap: changing theme after changing style ####
# # In this example theme_bw() is changed after theme change were made.
```

```
Data
plotData      List of 9
weatherData   366 obs. of 23 variables
```

```
C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot/
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GG
PlotRoot/scripts/lesson03.R')
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GG
PlotRoot/scripts/lesson03.R')
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GG
PlotRoot/scripts/lesson03.R')
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GG
PlotRoot/scripts/lesson03.R')
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GG
PlotRoot/scripts/lesson03.R')
> 
```

Fig 7: Styling labels and titles

5.2 - Elements of a plot

Let's look a little deeper at how subcomponents within ***theme()*** are set...

In the following code, the ***size*** of ***axis.title.x***, is set to **14** and the ***color*** *of* ***axis.title.x*** is set to ***orangered2***:

```
  axis.title.x = element_text(size=14, color="orangered2")
```

This is probably not an intuitive way to set styles. In R, ***axis.title.x*** is a text element (or ***element_text***). Instead of directly setting the ***size*** and ***color*** of ***axis.title.x***, you set ***axis.title.x*** to a text element (***element_text***) that has the ***size*** and ***color*** you want -- in this case, ***14*** and ***orangered2***.

In *figure 7*, we are modifying four subcomponents that are text element, so they are all set to an **element_text** that has the styles we want. In future lessons we will look at other types of elements.

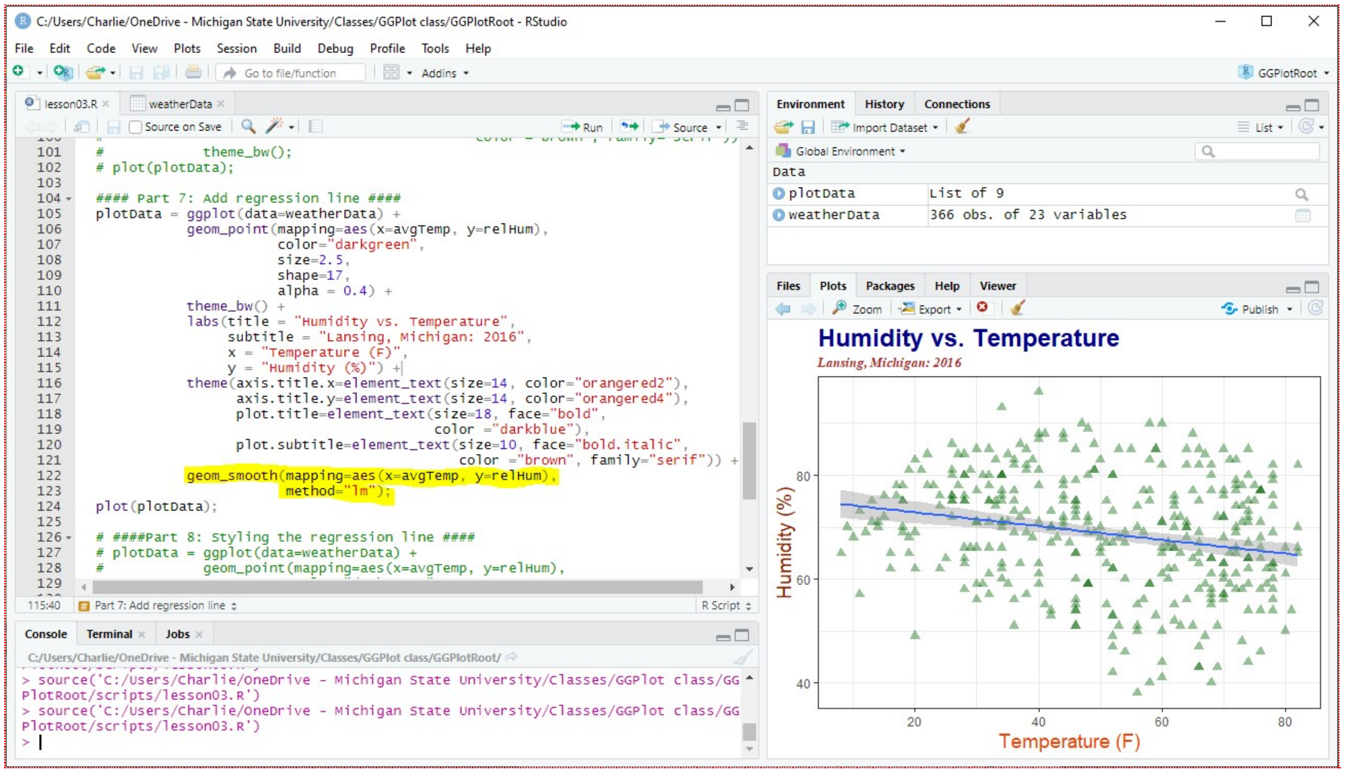
6 - Add Regression line

Next, we will add a regression line to the canvas. This is done using the plotting component ***geom_smooth()***.

Since ***geom_smooth()*** adds data to the plot area, we need to use the ***mapping*** subcomponent to tell ***GGPlot*** what is being added to the plot area and how. In this case we are mapping ***relHum*** vs. ***avgTemp***.

We also add the ***method*** subcomponent to ***geom_smooth()*** to set the smoothing method we will use on the data. In this case, linear model (***lm***).

```
#### Part 7: Add regression line ####
plotData = ggplot( data=weatherData ) +
           geom_point( mapping=aes(x=avgTemp, y=relHum),
                       color="darkgreen",
                       size=2.5,
                       shape=17,
                       alpha = 0.4 ) +
           theme_bw() +
           labs(title = "Humidity vs. Temperature",
                subtitle = "Lansing, Michigan: 2016",
                x = "Temperature (F)",
                y = "Humidity (%)") +
           theme(axis.title.x=element_text(size=14, color="orangered2"),
                 axis.title.y=element_text(size=14, color="orangered4"),
                 plot.title=element_text(size=18, face="bold",
                                         color ="darkblue"),
                 plot.subtitle=element_text(size=10, face="bold.italic",
                                            color ="brown", family="serif")) +
           geom_smooth( mapping=aes(x=avgTemp, y=relHum),
                        method="lm" );
plot(plotData);
```



```
#               theme_bw();
# plot(plotData);

#### Part 7: Add regression line ####
plotData = ggplot(data=weatherData) +
           geom_point(mapping=aes(x=avgTemp, y=relHum),
                      color="darkgreen",
                      size=2.5,
                      shape=17,
                      alpha = 0.4) +
           theme_bw() +
           labs(title = "Humidity vs. Temperature",
                subtitle = "Lansing, Michigan: 2016",
                x = "Temperature (F)",
                y = "Humidity (%)") +
           theme(axis.title.x=element_text(size=14, color="orangered2"),
                 axis.title.y=element_text(size=14, color="orangered4"),
                 plot.title=element_text(size=18, face="bold",
                                         color ="darkblue"),
                 plot.subtitle=element_text(size=10, face="bold.italic",
                                            color ="brown", family="serif")) +
           geom_smooth(mapping=aes(x=avgTemp, y=relHum),
                       method="lm");
plot(plotData);

# ####Part 8: Styling the regression line ####
# plotData = ggplot(data=weatherData) +
#               geom_point(mapping=aes(x=avgTemp, y=relHum),
```

```
Data
plotData      List of 9
weatherData   366 obs. of 23 variables
```

```
C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot/
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GG
PlotRoot/scripts/lesson03.R')
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GG
PlotRoot/scripts/lesson03.R')
> 
```

Fig 8: Adding a regression line

7 - Styling the line

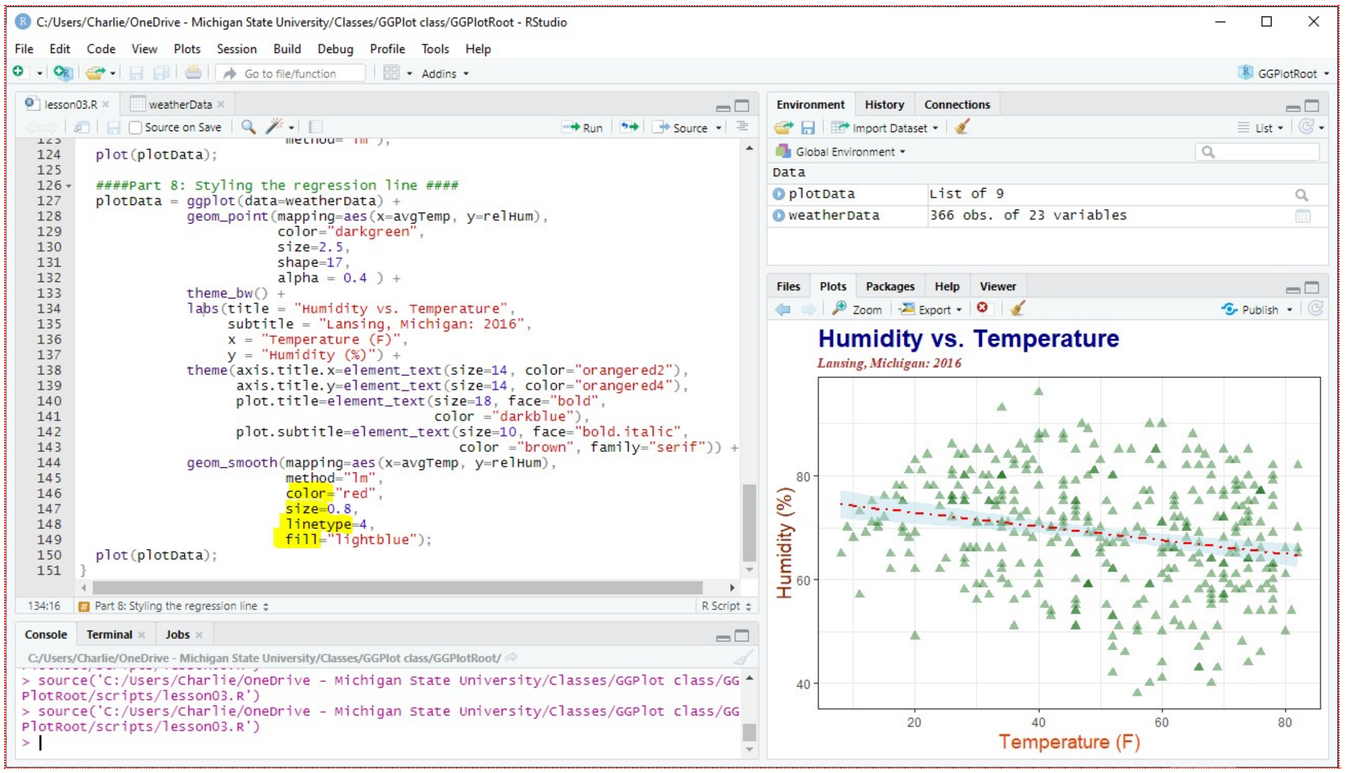
Lastly, we can also apply styles to the regression line using subcomponents in ***geom_smooth()***.

The four subcomponents of the regression line we will change are:

- ***color***: the color of the line
- ***linetype***: a number from 1 to 6 that gives the type of line -- the line types are shown here
- ***size***: multiplier for the size of the line
- ***fill***: color of the area representing the error-range of the regression line

```
source(file="scripts/reference.R");   # include the reference.r file
weatherData = read.csv(file="data/LansingNOAA2016.csv",
                        stringsAsFactors = FALSE);


plotData = ggplot( data=weatherData ) +
           geom_point( mapping=aes(x=avgTemp, y=relHum),
                       color="darkgreen", size=2.5, shape=17,
                       alpha = 0.4 ) +
           theme_bw() +
           labs(title = "Humidity vs. Temperature",
                subtitle = "Lansing, Michigan: 2016",
                x = "Temperature (F)",
                y = "Humidity (%)") +
           theme(axis.title.x=element_text(size=14, color="orangered2"),
                 axis.title.y=element_text(size=14, color="orangered4"),
                 plot.title=element_text(size=18, face="bold",
                                         color ="darkblue"),
                 plot.subtitle=element_text(size=10, face="bold.italic",
                                            color ="brown", family="serif")) +
           geom_smooth(mapping=aes(x=avgTemp, y=relHum),
                       method="lm",
                       color="red",
                       size=0.8,
                       linetype=4,
                       fill="lightblue");
plot(plotData);
```



```
  plot(plotData);

  ####Part 8: Styling the regression line ####
  plotData = ggplot(data=weatherData) +
             geom_point(mapping=aes(x=avgTemp, y=relHum),
                        color="darkgreen",
                        size=2.5,
                        shape=17,
                        alpha = 0.4 ) +
             theme_bw() +
             labs(title = "Humidity vs. Temperature",
                  subtitle = "Lansing, Michigan: 2016",
                  x = "Temperature (F)",
                  y = "Humidity (%)") +
             theme(axis.title.x=element_text(size=14, color="orangered2"),
                   axis.title.y=element_text(size=14, color="orangered4"),
                   plot.title=element_text(size=18, face="bold",
                                           color ="darkblue"),
                   plot.subtitle=element_text(size=10, face="bold.italic",
                                              color ="brown", family="serif")) +
             geom_smooth(mapping=aes(x=avgTemp, y=relHum),
                         method="lm",
                         color="red",
                         size=0.8,
                         linetype=4,
                         fill="lightblue");
  plot(plotData);
}
```

```
Data
plotData       List of 9
weatherData    366 obs. of 23 variables
```

```
C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GGPlotRoot/
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GG
PlotRoot/scripts/lesson03.R')
> source('C:/Users/Charlie/OneDrive - Michigan State University/Classes/GGPlot class/GG
PlotRoot/scripts/lesson03.R')
> 
```

Fig 9: Adding styles to the regression line

8 - Application

Create a script file in your Project's script ***folder*** called ***app03.r*** and, using the data from **LansingNOAA2016.csv:**

1. Create a scatterplot of Average Wind Speed (***windSpeed***) vs. Daily Temperature Departure (***tempDept***)
2. Make the following modifications to the points:
   A. set the **alpha** of the points to 70% opaque
   B. set the **size** of the point to twice the normal size
   C. change the **color** of the points to **rgb(red=0, green=.6, blue=.6)**
   D. use filled-diamonds as the point **shape**

3. Add a regression line (***windSpeed*** vs ***tempDept***) to the plot
4. Change the line-type, color, and fill of the regression line.
5. Change the angle the values on the y-axis are drawn at
6. Add a **title** and **subtitle** and change the x and y-axis labels
7. Change the subtitle's **color** and **size**
8. Change the x-axis labels font **face** and **color**
9. Put a comment in the script file about the what the scatterplot shows
10. ***tempDept*** can be positive or negative but we might only care about the absolute change -- not the direction of change. So make all ***tempDept*** values positive by using the absolute function: ***abs(tempDept)*** in both the scatterplot and the regression line.

9 - Extension: RGB Colors and Unicode Characters

The are many more colors than names of colors and there are many more character than keys on a keyboard. This section will give a brief introduction on how one accesses the full spectrum of colors and characters.

9.1 - RGB colors

The are over 16 million colors we can choose from and to access all these colors, we need to use their ***rgb*** color codes.

rgb stands for red, green, blue and represents the three *primary light colors*. This is not the same as the primary pigment colors that you are probably familiar with (red, yellow, blue -- or, more accurately: magenta, yellow, cyan). The different is that *primary light colors are additive* (they combine to make white), whereas the *primary pigment colors are subtractive* (they combine to make black -- or a really dark brown).

In R, you can set color to an ***rgb*** value using ***rgb()***. ***rgb()*** has three parameter (**red**, **green**, **blue**), and each color is set to a value from **0** (completely off) to **1** (completely on).

So:
black is: rgb(red=0, green=0, blue=0)
white is: rgb(red=1, green=1, blue=1)
bright blue is: rgb(red=0, green=0, blue=1)
dark blue is: rgb(red=0, green=0, blue=0.3)
yellow is: rgb(red=1, green=1, blue=0)
dark grey is: rgb(red=.3, green=.3, blue=.3)

It takes some time to get used to mixing light colors, but it is worth it in the end to learn how to do it.

9.2 - Unicode Characters

Unicode characters represent the most complete digital set of characters used in the world and, like hexadecimal colors, are represented by a number. You use Unicode to get characters that you cannot find on your keyboard. Wikipedia says that there are currently over 136,000 characters, and this number is growing.

In R, Unicode character are preceded by a dash (**-**) and follow by the number representing the character code.

9.3 - Script using RGB colors and Unicode characters

In the following script, I used **RGB** colors to set the color of the points, and **Unicode** to represent the shape of the points.

- The color ***rgb(red=0.8, green=0.4, blue=0)*** is most red light, a little green light, and no blue light. It is orange-red (remember these are light color, not pigment colors)
- The shape is Unicode character **6235**, which is the Mongolian letter *toda nia*: ᡛ

```
source("scripts/reference.R");     # include the script file reference.r

weatherData = read.csv(file="data/LansingNOAA2016.csv",
                       stringsAsFactors = FALSE);

plotData = ggplot(data=weatherData) +
           geom_point(mapping=aes(x=avgTemp, y=relHum),
                      color=rgb(red=0.8, green=0.4, blue=0),
                      size=2.5,
                      shape=-6235 ) +
           theme_bw();
plot(plotData);
```

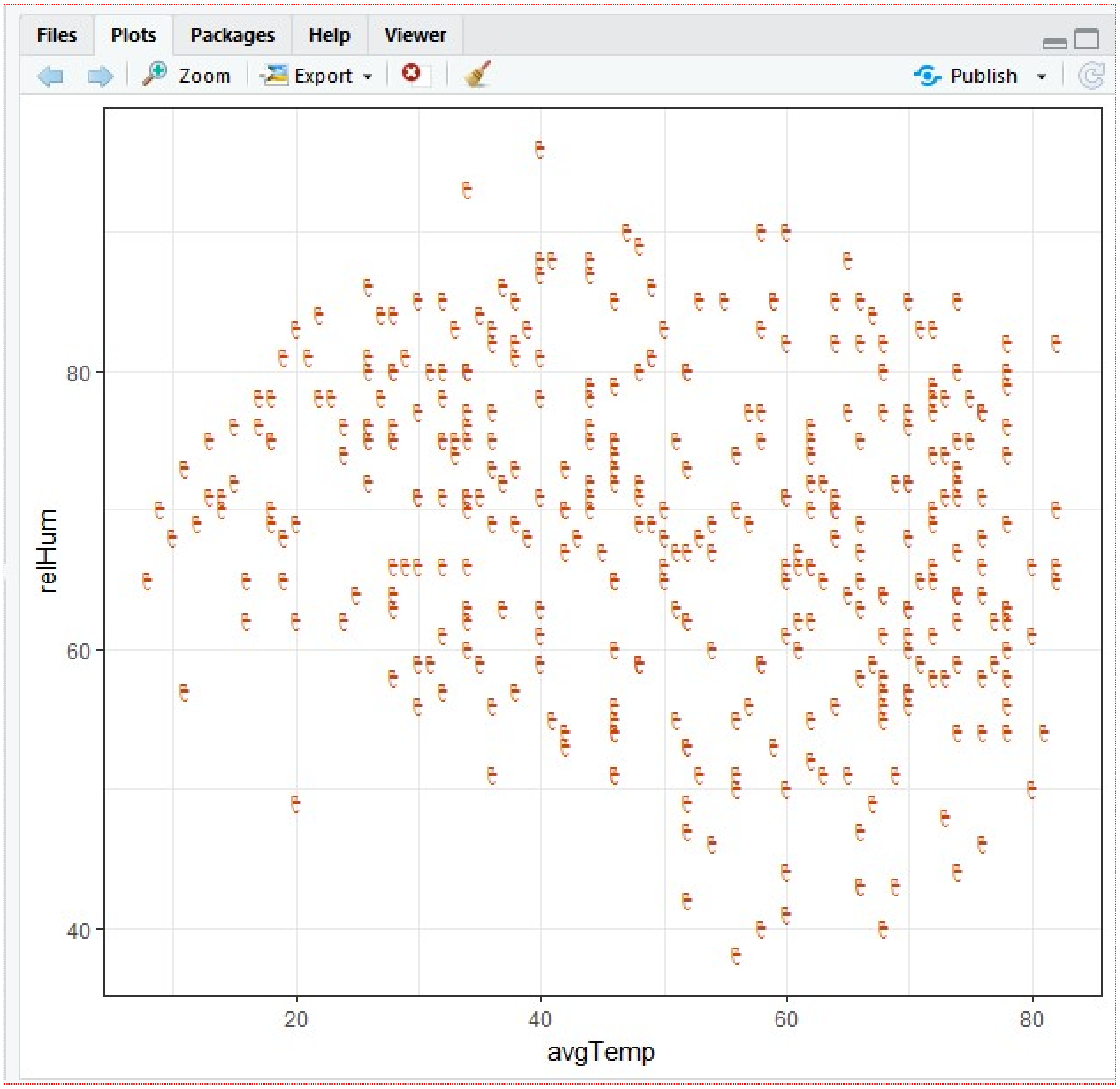


Fig 10: Using hexadecimal color codes and Unicode

10 - Trap: Changing a theme after modifying

Is this example, ***theme()*** is used to modify various label and title components, but then ***theme_bw()*** is called. ***theme_bw()*** reset all components of the theme basically undoing the changes we made beforehand. Make sure ***theme_bw()*** is called before other theme changes are made.

```
source("scripts/reference.R");   # include the script file reference.r

weatherData = read.csv(file="data/LansingNOAA2016.csv",
                        stringsAsFactors = FALSE);
plotData = ggplot(data=weatherData) +
           geom_point(mapping=aes(x=avgTemp, y=relHum),
                       color="darkgreen", size=2.5, shape=17,
                       alpha = 0.4 ) +
           labs(title = "Humidity vs. Temperature",
                subtitle = "Lansing, Michigan: 2016",
                x = "Temperature (F)",
                y = "Humidity (%)") +
           theme(axis.title.x=element_text(size=14, color="orangered2"),
                 axis.title.y=element_text(size=14, color="orangered4"),
                 plot.title=element_text(size=18, face="bold",
                                         color ="darkblue"),
                 plot.subtitle=element_text(size=10, face="bold.italic",
                                            color ="brown", family="serif")) +
           theme_bw();
plot(plotData);
```

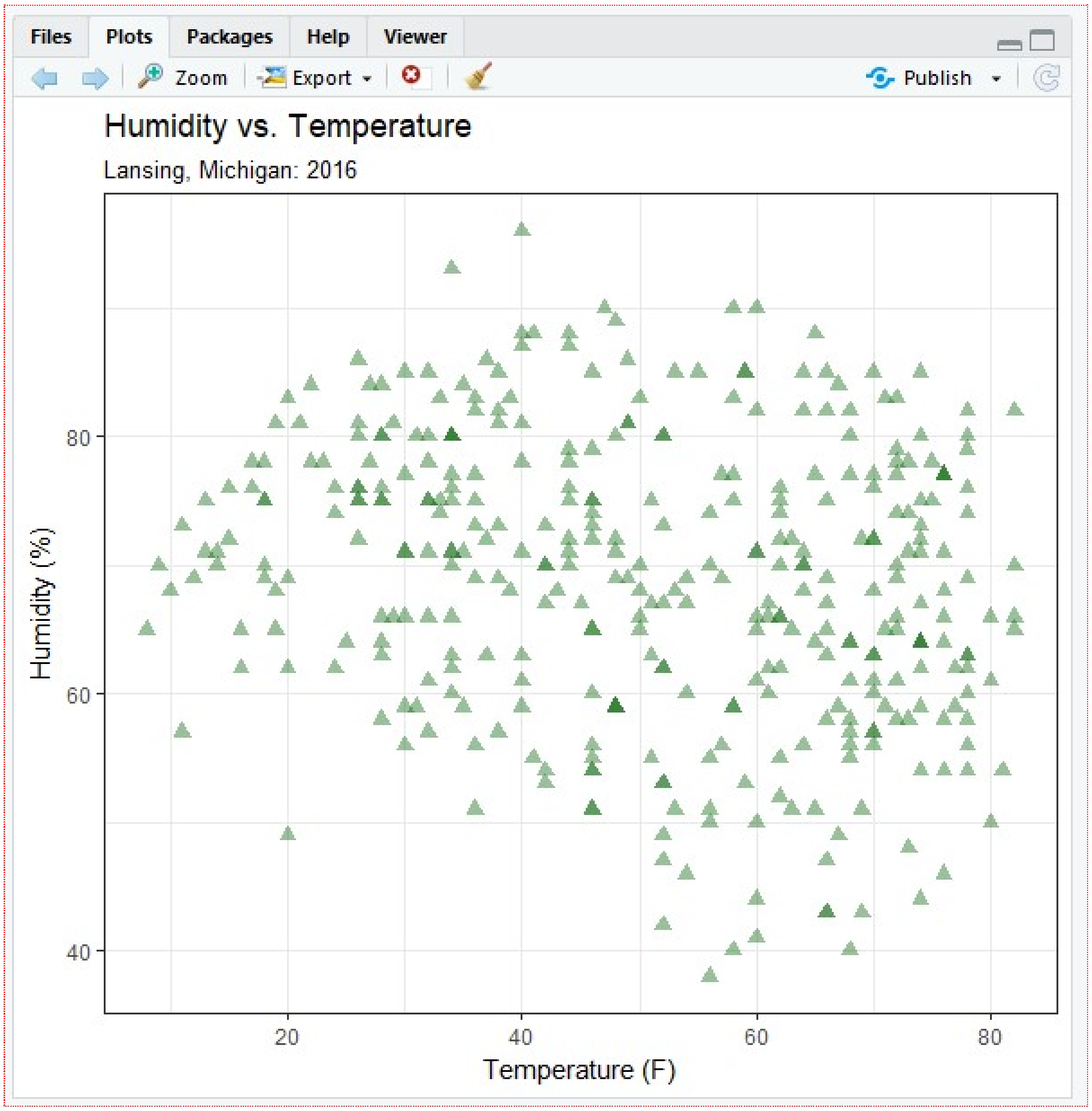


*Fig 11: **theme_bw()** resets theme undoing the changes made before using **theme()***