01-07: Boxplots 2

1 - Purpose

- Adding facets and groupings to a boxplot
- Styling individual boxes in a boxplot
- Changing titles and labels on a legend

2 - Concepts

3 - Get data

For this lesson, we will work from the data file we created last lesson, LansingNOAA2016-3.csv.

```
source( file="scripts/reference.R" );
weatherData = read.csv( file="data/LansingNOAA2016-3.csv",
                        stringsAsFactors = FALSE );
```

4 - Another way to group discrete variables

We are going to start with the same boxplot as the last lesson: ***changeMaxTemp*** vs ***WindDir.*** In the last lesson we set the order of the wind direction levels using ***factor()*** for the x-axis of the ***geom_boxplot()***. This time we will set the order using the ***scale_x_discrete()*** component -- the subcomponent ***limits*** is set to a vector that contains the order the categories will appear on the x-axis (**North**, **East**, **South**, **West**).

```
#### Part 1: A different way to arrange x-axis values
thePlot = ggplot(data=weatherData) +
          geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp),
                       na.rm=TRUE) +
          scale_x_discrete(limits=c("North", "East", "South", "West")) +
          theme_bw() +
          labs(title = "Change in Temperature vs. Wind Direction",
               subtitle = "Lansing, Michigan: 2016",
               x = "Wind Direction",
               y = "Degrees (Fahrenheit)");
plot(thePlot);
```

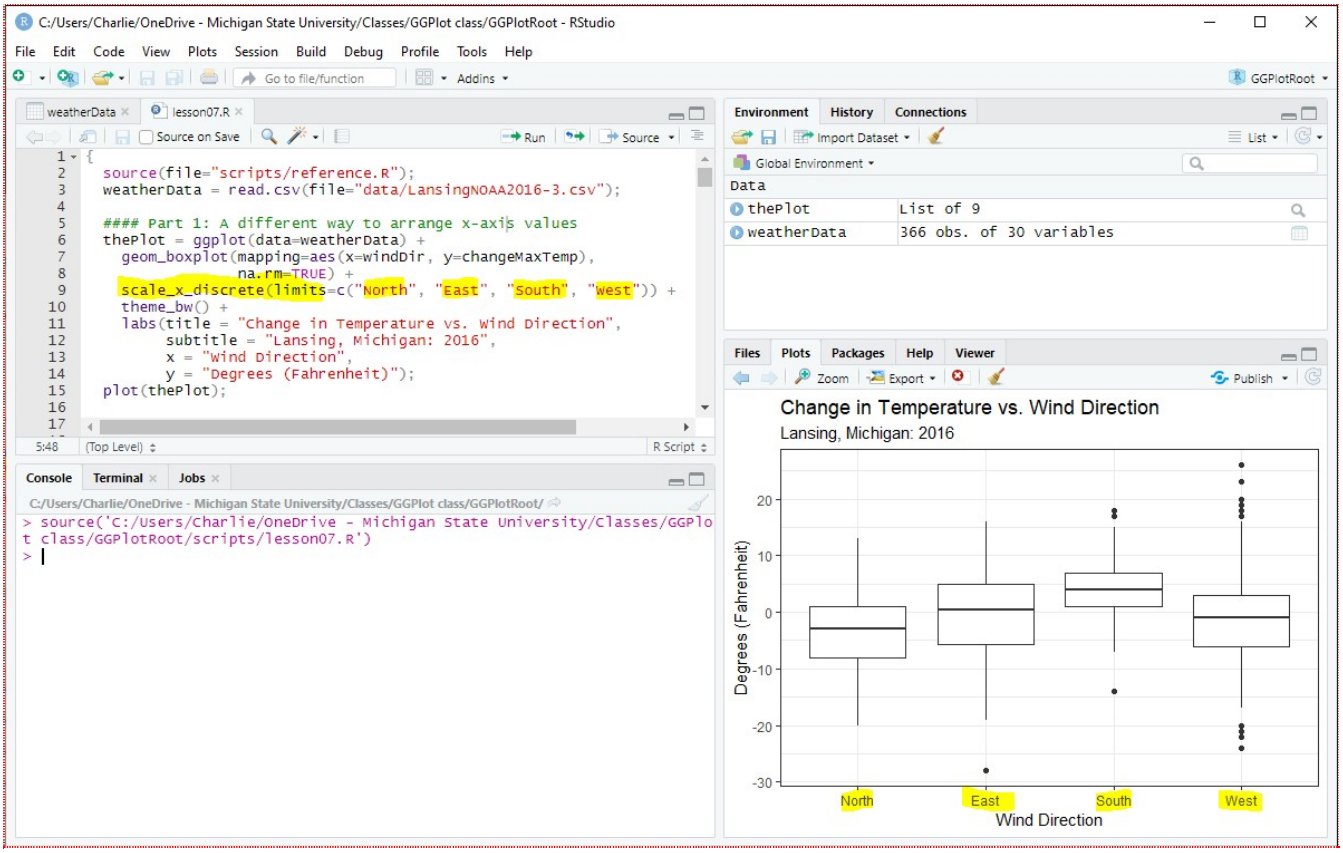

*Fig 1: Using **scale_x_discrete** to set the x-axis values*

Extension: Changing quantiles

5 - Grouping boxplots by levels

We can see in the previous boxplot (*fig 1*) that northerly winds seem to coincide with lower temperatures and southerly winds seem to coincide with higher temperatures. This makes sense because, in Michigan, northerly winds tend to come from the Arctic and southerly winds come from the Gulf of Mexico.

Let's try to tease out more information by applying ***windSpeedLevel*** to the plot. We are going to use ***windSpeedLevel*** as a grouping variable so there will be three boxes representing different levels of wind (**Low**, **Medium**, **High**) for each of the four cardinal directions (**N**, **E**, **S**, **W**).

In the ***mapping*** for the boxplot, we use ***fill*** to add ***windSpeedLevel*** as a grouping variable.

*Note: GGPlot automatically generates a legend when the **fill** parameter is used.*

```
### Part 2: Group boxplots by wind speed levels
thePlot = ggplot(data=weatherData) +
          geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp,
                                   fill=windSpeedLevel),
                       na.rm=TRUE) +
              scale_x_discrete(limits=c("North", "East", "South", "West")) +
              theme_bw() +
              labs(title = "Change in Temperature vs. Wind Direction",
```

```
                    subtitle = "Lansing, Michigan: 2016",
                    x = "Wind Direction",
                    y = "Degrees (Fahrenheit)");
plot(thePlot);
```

We now have 12 boxes, representing all combinations of three levels (Low, Medium, High) and four directions (North, East, South, West).

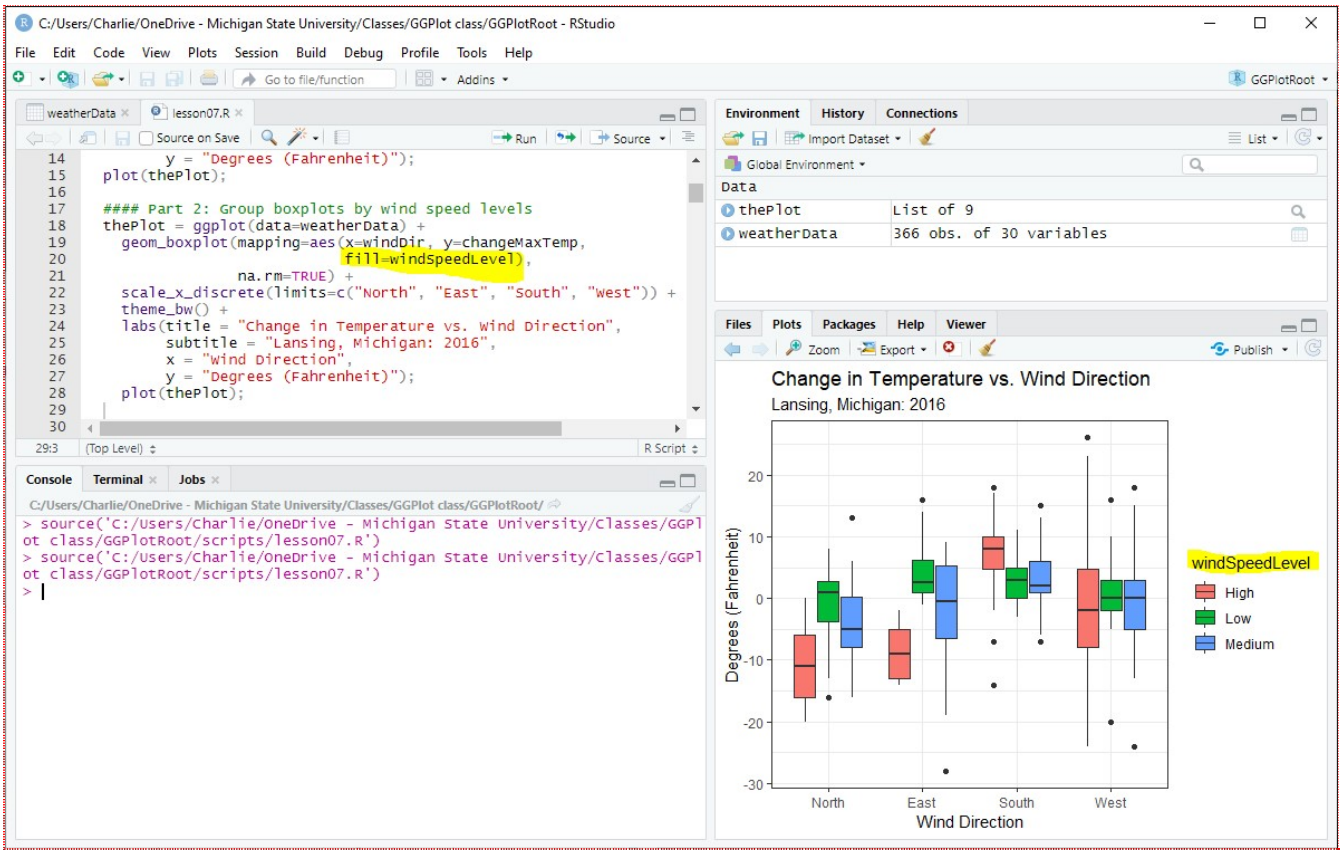

*Fig 2: Grouping **maxChangeTemp** vs **windDir** by the strength of the wind (**fill=windSpeedLevel**)*

5.1 - Reorder groups

```
  fill = windSpeedLevel
```

...puts the ***windSpeedLevel*** categories into the legend alphabetical order **(High, Low, Medium)**. But, we want the the levels to be in order of strength (**Low**, **Medium**, **High**).

We can use ***factor()*** to cast ***windSpeedLevel*** into a factor vector with the ***levels*** defined:

```
  factor(windSpeedLevel, levels=c("Low", "Medium", "High"))
```

And use the factored ***windSpeedLevel*** as the ***fill*** in the ***mapping*** for the ***geom_boxplot()*** component:

```
### Part 3: Re-order group as factors
thePlot = ggplot(data=weatherData) +
          geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp,
                                   fill=factor(windSpeedLevel,
```

```
                                        levels=c("Low", "Medium", "High"))),
                        na.rm=TRUE) +
          theme_bw() +
          scale_x_discrete(limits = c("North", "East", "South", "West")) +
          labs(title = "Change in Temperature vs. Wind Direction",
               subtitle = "Lansing, Michigan: 2016",
               x = "Wind Direction",
               y = "Degrees (Fahrenheit)");
plot(thePlot);
```

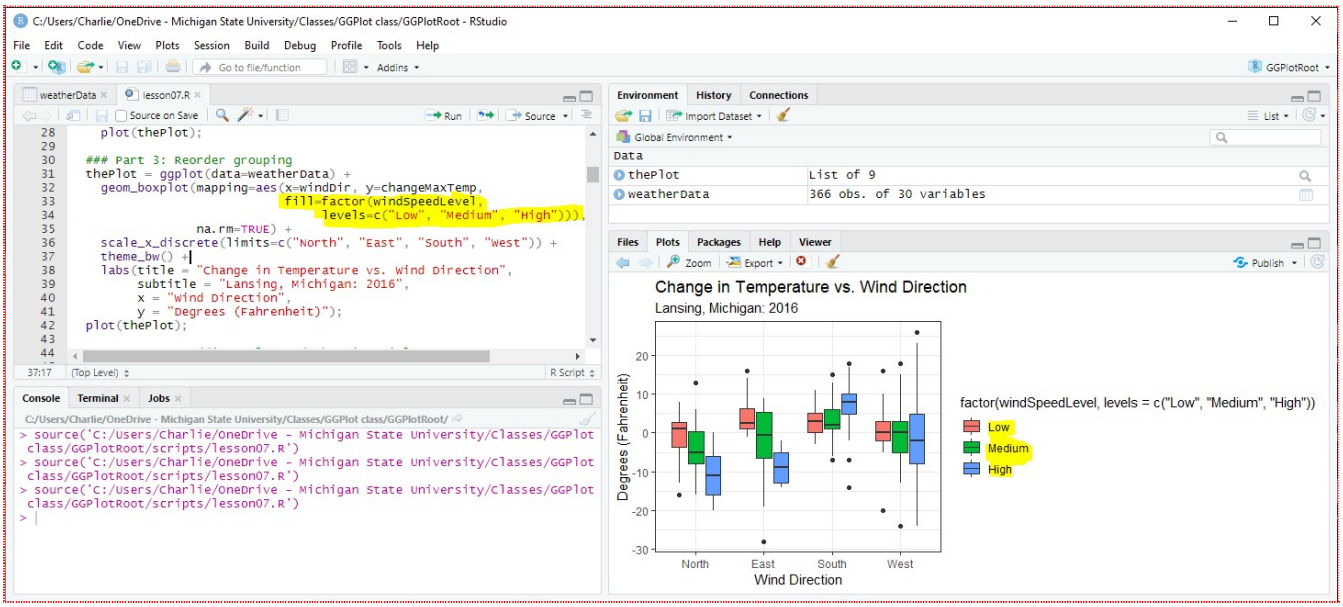

Fig 3: Reordering the fill grouping.

5.2 - Changing the legend title

We now have the grouping in the right order but, by default, the title for the legend is the text value of the ***fill*** in the ***mapping***, which is a little awkward (*fig 3*). We can change the legend title use the ***fill*** subcomponent in the ***labs()*** component.

```
### Part 3: Re-order group as factors
thePlot = ggplot(data=weatherData) +
          geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp,
                                   fill=factor(windSpeedLevel,
                                               levels=c("Low", "Medium", "High"))),
                       na.rm=TRUE) +
          theme_bw() +
          scale_x_discrete(limits = c("North", "East", "South", "West")) +
          labs(title = "Change in Temperature vs. Wind Direction",
               subtitle = "Lansing, Michigan: 2016",
               x = "Wind Direction",
               y = "Degrees (Fahrenheit)",
```

```
                fill = "Wind Speeds");
plot(thePlot);
```

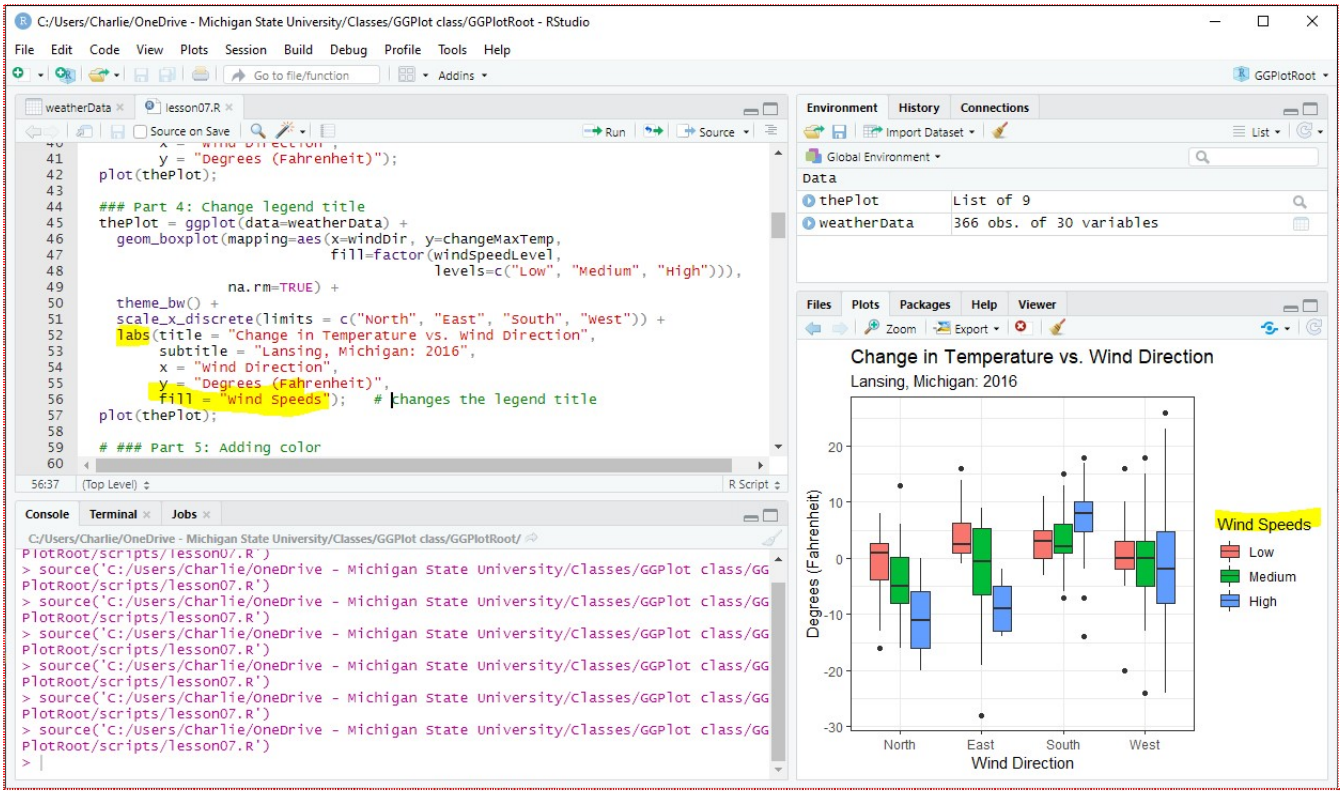

Fig 4: Changing the legend title using ***labs()***

5.3 - Style and title changes to the legend and fill

We can change the fill colors using the ***scale_fill_manual()*** component. The parameter to set is ***values*** and we need to supply a vector with three color values, representing the three levels (**low**, **medium**, **high**). Instead of color names, we will use red-green-blue colors, or RGB.

```
scale_fill_manual(values = c(rgb(red=1, green=1, blue=0),      # low
                             rgb(red=1, green=0.2, blue=0),    # medium
                             rgb(red=0.5, green=0, blue=0.8))  # high
```

RGB (red-green-blue) colors have values from **0** (no light) to **1** (light completely on) so:

- ***rgb(red=1, green=1, blue=0)***: all red, all green, and no blue (yellow -- *remember these are light colors, not paint pigments*)
- ***rgb(red=1, green=0.2, blue=0)***: all red, a little green, and no blue (red-orange)
- ***rgb(red=0.5, green=0, blue=0.8)***: some red, no green, and a lot of blue (purple)

Extension: More about RGB Colors

The code for the new plot is now:

```
### Part 5: Adding color using rgb()
thePlot = ggplot(data=weatherData) +
          geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp,
```

```
                                    fill=factor(windSpeedLevel,
                                                levels=c("Low", "Medium", "High"))),
                            na.rm=TRUE) +
          theme_bw() +
          scale_x_discrete(limits = c("North", "East", "South", "West")) +
          scale_fill_manual(values = c(rgb(red=1, green=1, blue=0),         # low
                                       rgb(red=1, green=0.2, blue=0),       # medium
                                       rgb(red=0.5, green=0, blue=0.8))) + # high
          labs(title = "Change in Temperature vs. Wind Direction",
               subtitle = "Lansing, Michigan: 2016",
               x = "Wind Direction",
               y = "Degrees (Fahrenheit)",
               fill = "Wind Speeds");   # changes the legend (fill) title
plot(thePlot);
```

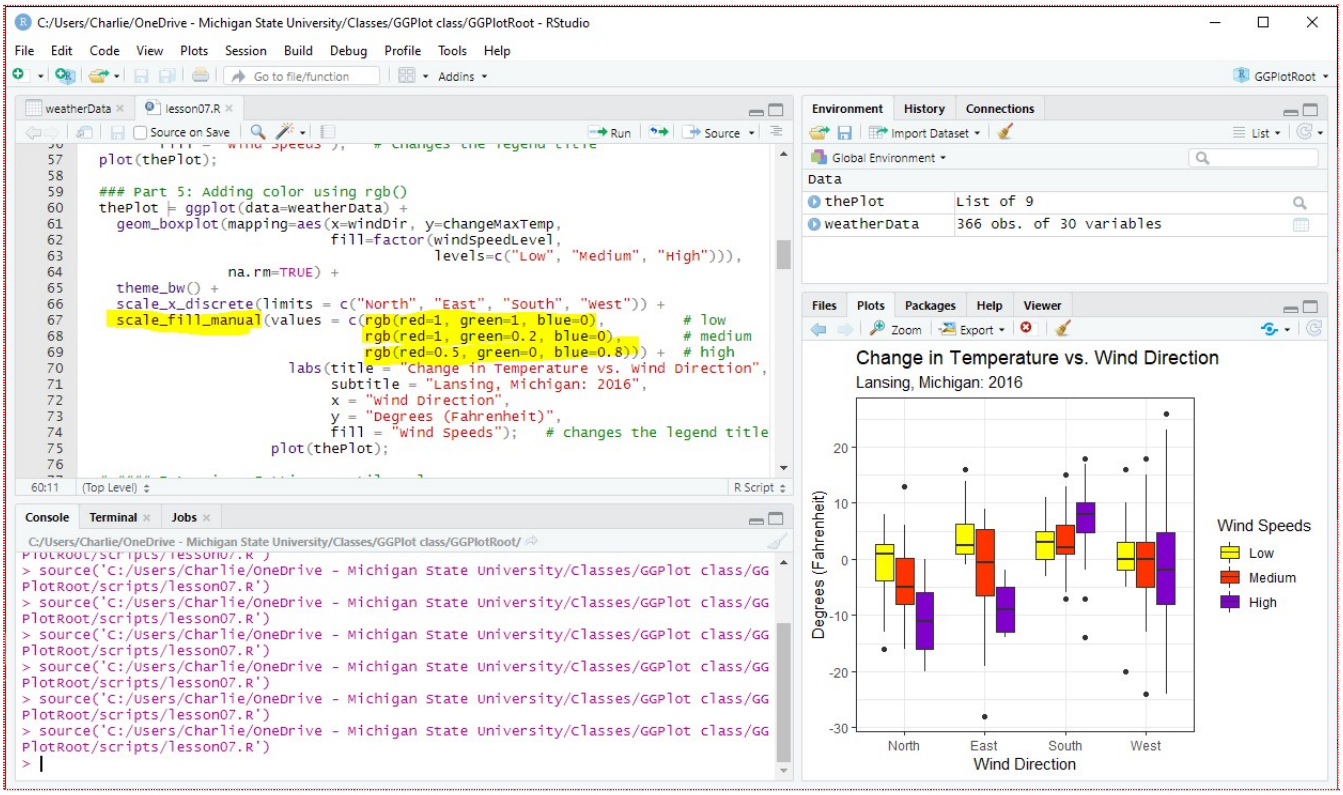

Fig 5: Changing the legend title and the boxplot fill colors

6 - Using facets in a boxplot

We can also group the ***windSpeedLevel*** category using facets. Where ***fill*** grouped by a *variable within the plot*, facets uses the grouping variable (in this case: ***windSpeedLevel***) to *create multiple plots*.

The component is called ***facet_grid()*** and the subcomponent we are setting is ***facets***. The subcomponent facets requires both a y-axis and x-axis variable in the form *y-axis~x-axis*. Most of the time, you will only use facets along one axis, so you use a dot (**.**) to represent an axis that has no facet.

```
  facet_grid(facets=windSpeedLevel ~ .)  # facet in vertical direction
```

In this example, ***windSpeedLevel*** is set as the facet for the y-axis and the x-axis has no facet.

```
### Part 6: Using facets along the y-axis
thePlot = ggplot(data=weatherData) +
          geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp),
                       na.rm=TRUE) +
          theme_bw() +
          scale_x_discrete(limits = c("North", "East", "South", "West")) +
          facet_grid(facets=windSpeedLevel ~ .) + # facet in vertical direction
          labs(title = "Change in Temperature vs. Wind Direction",
               subtitle = "Lansing, Michigan: 2016",
               x = "Wind Direction",
               y = "Degrees (Fahrenheit)");
plot(thePlot);
```

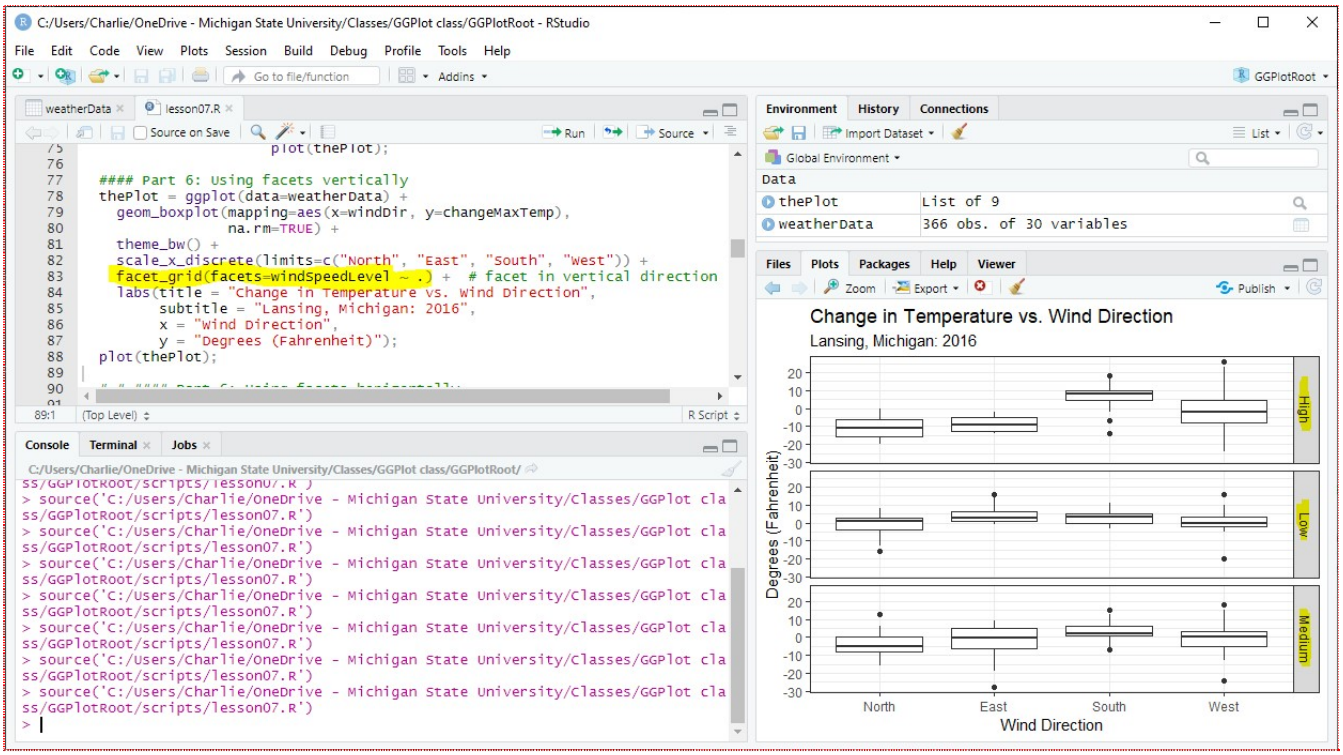

Fig 6: Adding facet in the vertical direction

6.1 - Adding a x-axis facet

In this example the plots would probably look better if ***windSpeedLevel*** was arranged along the x-axis. To do this, we switch the axis order for the ***facets*** parameter **from:**

```
  (facets= windSpeedLevel ~ .)
```

to:

```
  (facets= . ~ windSpeedLevel)    # facet in horizontal direction
```

```
### Part 7: Using facets along the x-axis
```

```
thePlot = ggplot(data=weatherData) +
          geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp),
                       na.rm=TRUE) +
          theme_bw() +
          scale_x_discrete(limits = c("North", "East", "South", "West")) +
          facet_grid(facets=. ~ windSpeedLevel) + # facet in horizontal direction
          labs(title = "Change in Temperature vs. Wind Direction",
               subtitle = "Lansing, Michigan: 2016",
               x = "Wind Direction",
               y = "Degrees (Fahrenheit)");
plot(thePlot);
```

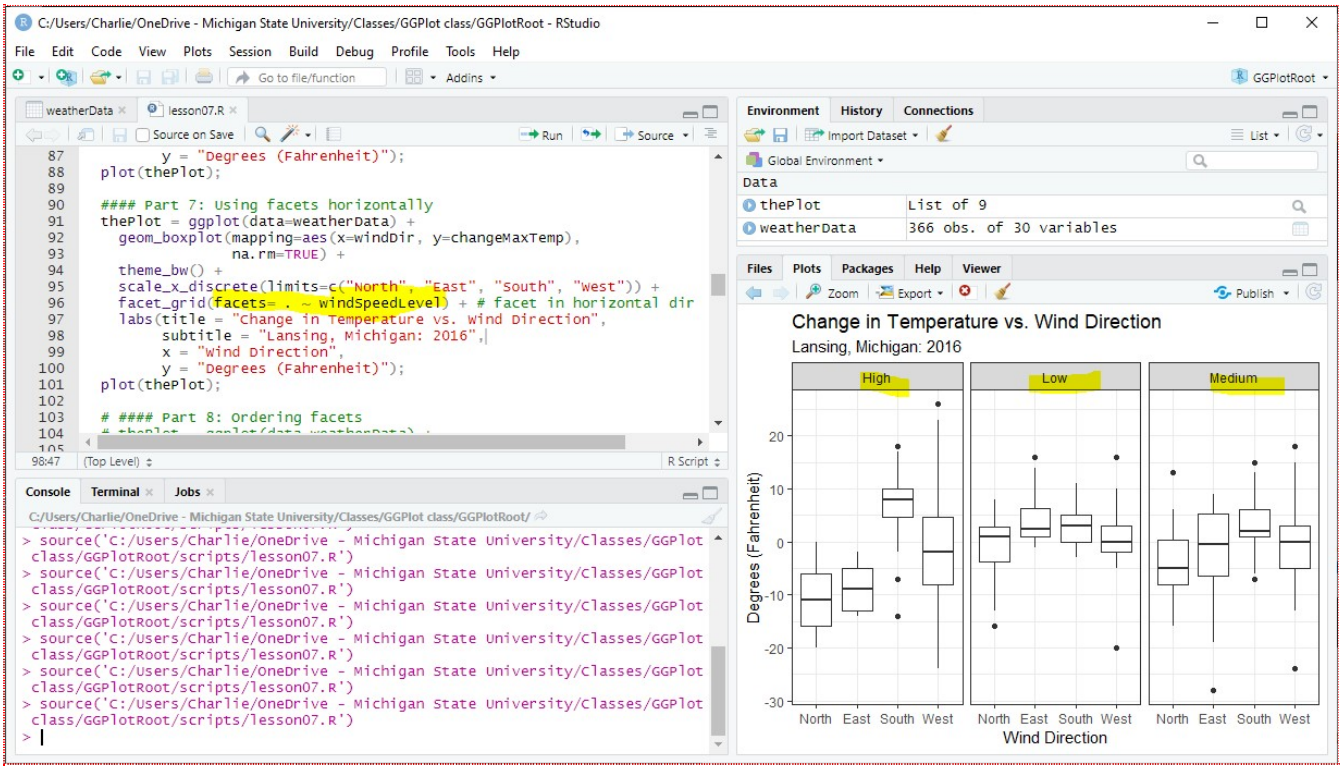

Fig 7: Adding a facet in the horizontal direction

Note: you can add both an horizontal and vertical facet to the plot -- the plot will get a bit busy if you do that, though.

7 - Reordering the facet values

Once again, we have the issue that GGPlot orders categorical values alphabetically but we want ***windSpeedLevel*** in order of strength. Just like with fill before, we can force ***windSpeedLevel*** into a ***factor()*** and then set the order of the factor values.

```
### Part 8: Ordering facets
thePlot = ggplot(data=weatherData) +
          geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp), na.rm=TRUE) +
```

```
          theme_bw() +
          scale_x_discrete(limits = c("North", "East", "South", "West")) +
          facet_grid(facets= . ~ factor(windSpeedLevel,
                                        levels=c("Low", "Medium", "High"))) +
          labs(title = "Change in Temperature vs. Wind Direction",
               subtitle = "Lansing, Michigan: 2016",
               x = "Wind Direction",
               y = "Degrees (Fahrenheit)");
plot(thePlot);
```

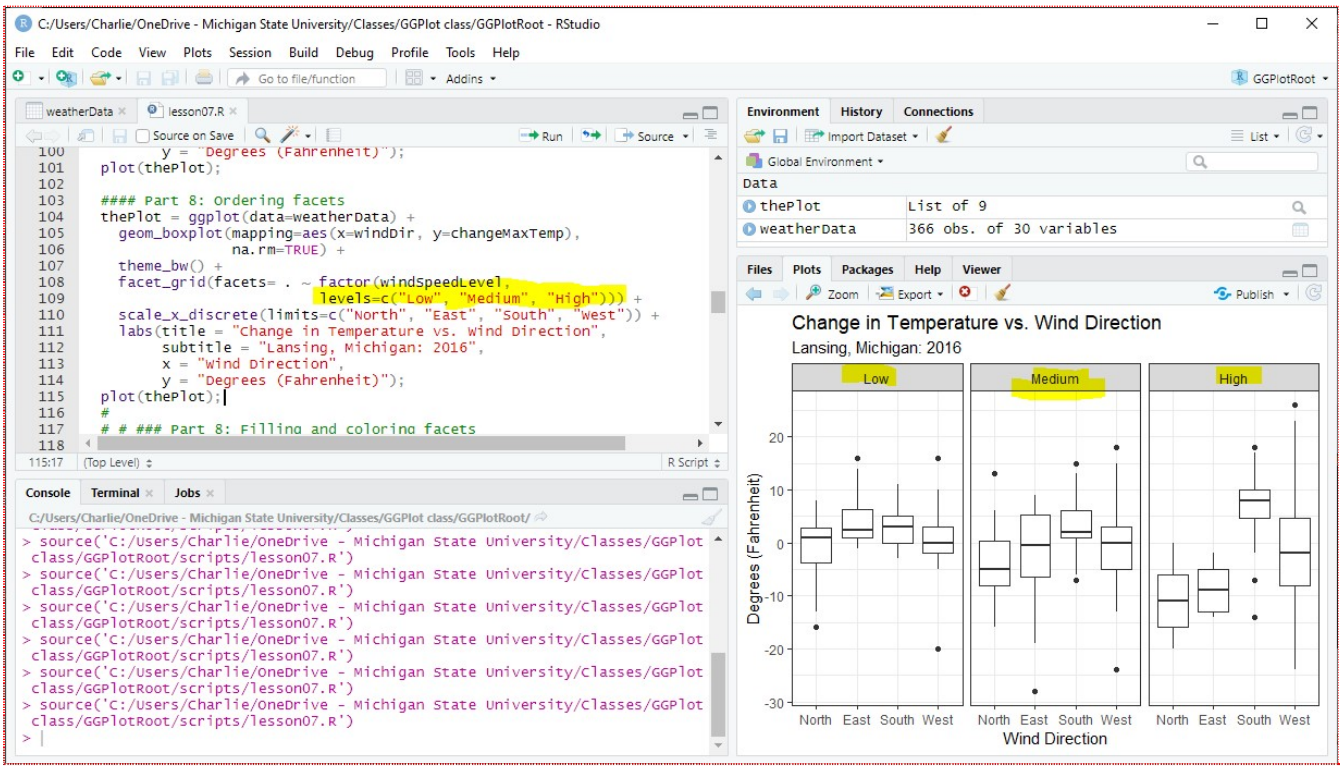

Fig 8: Reordering the facet values

8 - Adding color to the boxplots

We are going to modify the fill color and outline color of the boxplots. This is done using the ***color*** (outline color of box) and ***fill*** (fill color of box) subcomponents in ***geom_boxplot()***.

8.1 - One Color

If you want to set one color for all plots, you add ***color*** and ***fill*** subcomponents to the ***mapping()*** component:

```
### Part 9: Filling and coloring facets
thePlot = ggplot(data=weatherData) +
          geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp),
                       na.rm=TRUE,
                       color="blue",   # outline color
```

```
                        fill="red")     # fill color
          theme_bw() +
          scale_x_discrete(limits = c("North", "East", "South", "West")) +
          facet_grid(facets= . ~ factor(windSpeedLevel,
                                levels=c("Low", "Medium", "High"))) +
          labs(title = "Change in Temperature vs. Wind Direction",
               subtitle = "Lansing, Michigan: 2016",
               x = "Wind Direction",
               y = "Degrees (Fahrenheit)");
plot(thePlot);
```

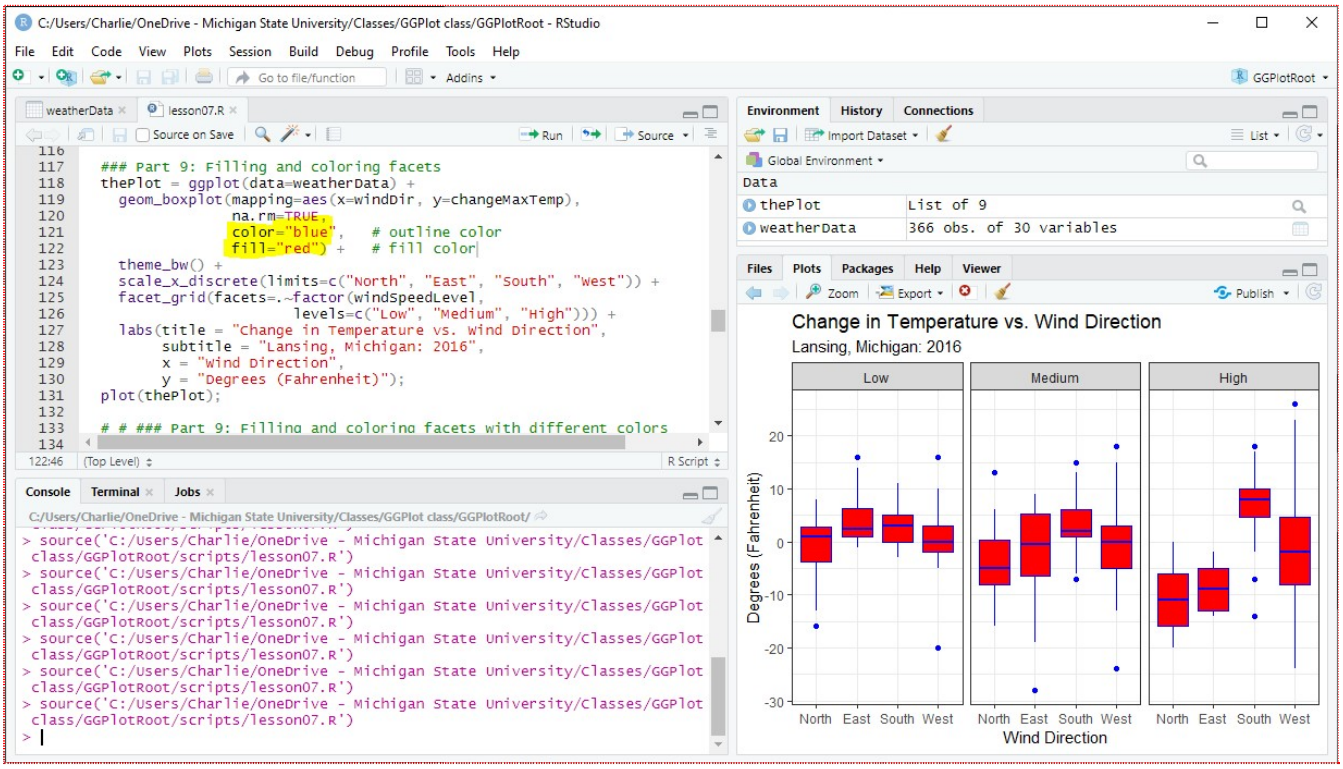

Fig 9: Filling and outlining the boxplots with color

8.2 - All colors

If we want the plots to be different color then we need to set ***color*** and ***fill*** to a vector that contains the same number of values as there are plots (**12**).

To set the ***color*** value for all **12** plots, we need to use a vector with **12** color values like this:

```
color = c("blue", "black", "black", "black",
          "green", "black", "black", "black",
          "orange", "black", "black", "black"),
```

This will set the color of the first plot to **blue**, fifth plot to **green**, ninth plot to **orange**, and every other plot to black. However, we have repeating **black** values, so we can make use of ***rep()*** to avoid writing out the same value multiple times:

```
  color = c("blue", rep("black", 3),     # blue, black, black, black, blue...
            "green", rep("black", 3),    # green, black, black, black, green...
            "orange", rep("black", 3)),  # orange, black, black, black, orange...
```

We can also set ***fill*** to a vector with **12** color values. In this case, we will use ***NA*** for plots that we do not want to have a fill color.

```
  fill = c(NA, NA, NA, NA, NA, NA, NA, NA, "red", "red", "red", NA)
```

or

```
  fill = c(rep(NA, 8), rep("red", 3), NA)  # 8 NA, 3 red, 1 NA
```

Putting the full code together:

```
### Part 10: Filling and coloring facets
thePlot = ggplot(data=weatherData) +
          geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp),
                       na.rm=TRUE,
                       color=c("blue", rep("black", 3),
                               "green", rep("black", 3),
                               "orange", rep("black", 3)),
                       fill=c(rep(NA, 8), rep("red", 3), NA)) +
          theme_bw() +
          scale_x_discrete(limits = c("North", "East", "South", "West")) +
          facet_grid(facets=.~factor(windSpeedLevel,
                                     levels=c("Low", "Medium", "High"))) +
          labs(title = "Change in Temperature vs. Wind Direction",
               subtitle = "Lansing, Michigan: 2016",
               x = "Wind Direction",
               y = "Degrees (Fahrenheit)");
plot(thePlot);
```

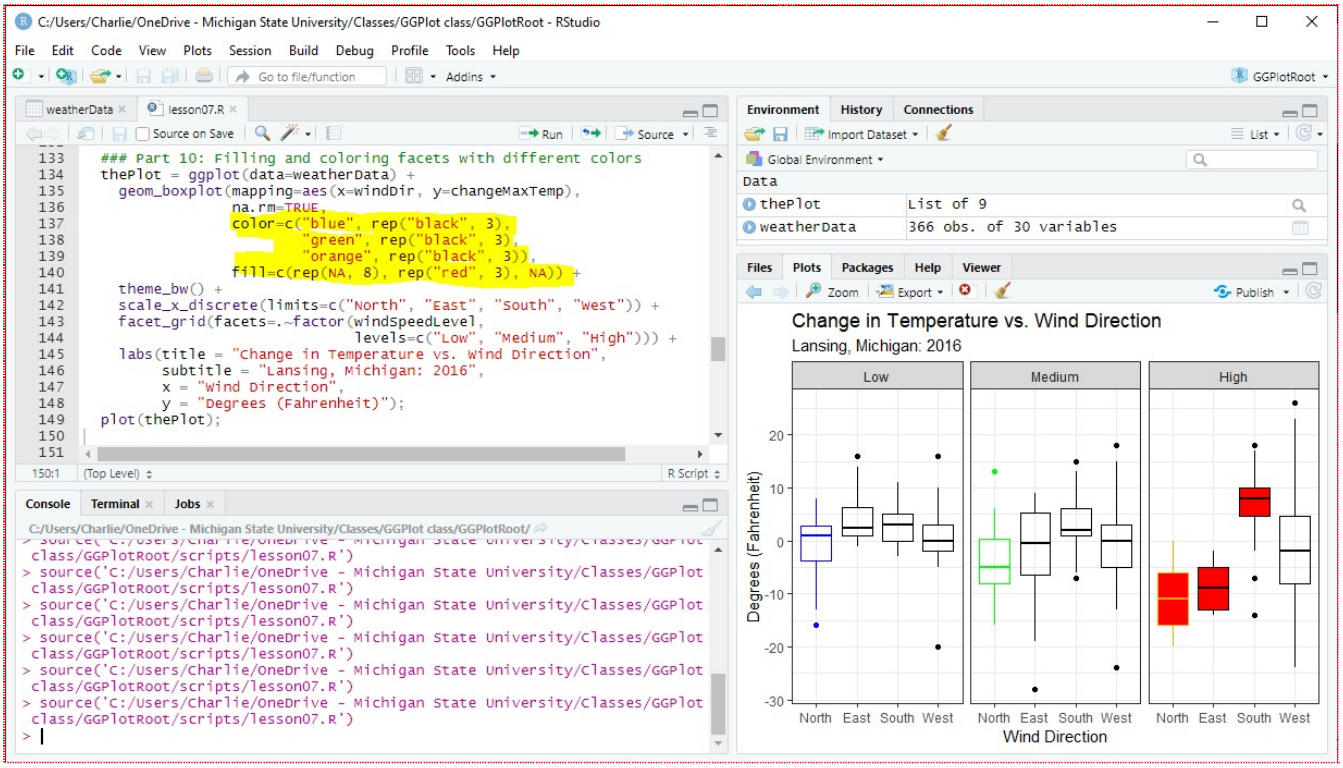

Fig 10: All fill and outline colors to the boxplots

8.3 - Wrong number of values

There are 12 boxplots in the plot area, *so we need to make sure that we have either 1 or 12 color values* -- otherwise, GGPlot will give the mismatch error: *Error: Aesthetics must be either length 1 or the same as the data (12)*

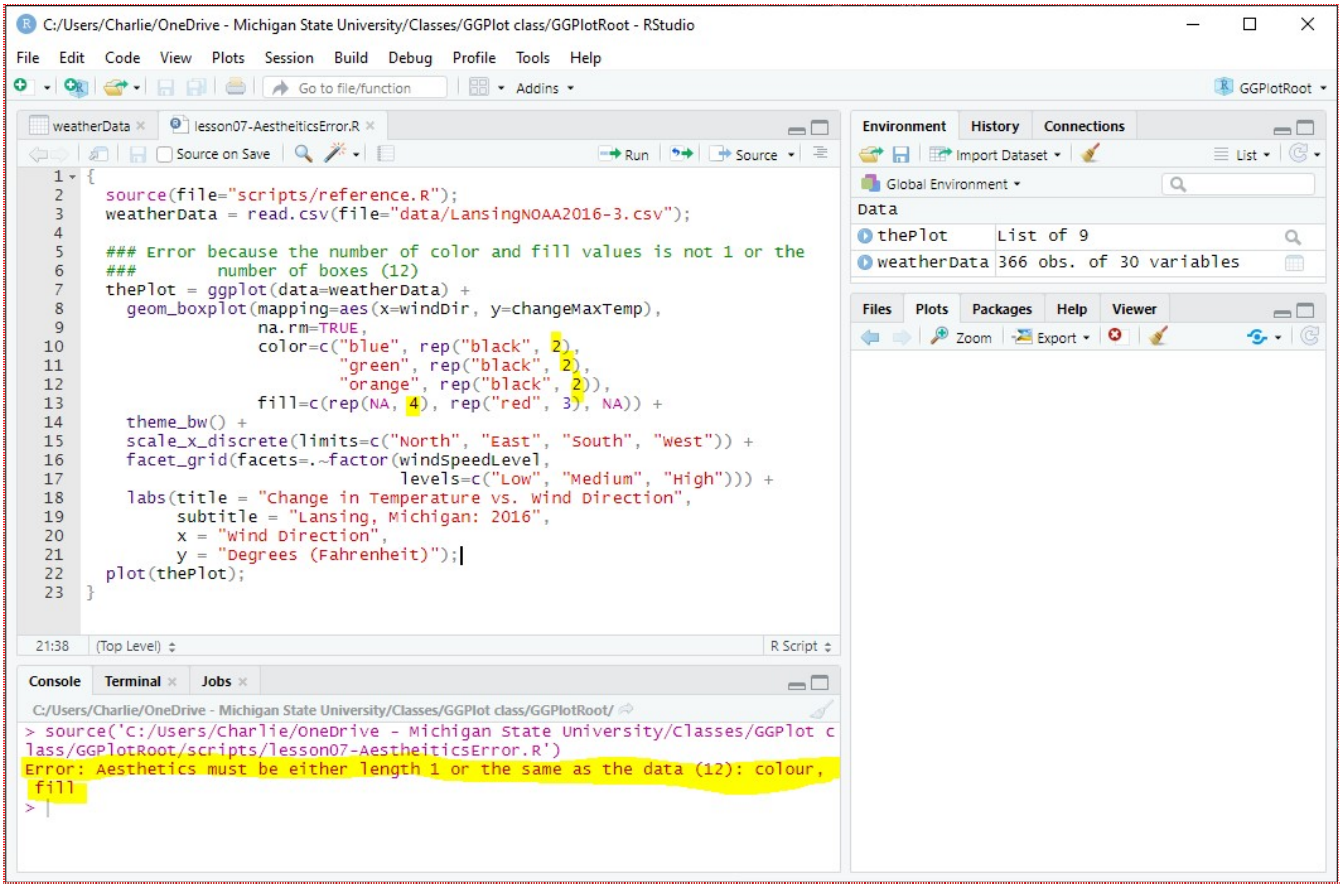

*Fig 11: Putting the wrong number of values in the **fill** and **color** vectors*

9 - Changing facet labels

Lastly, the labels on the facet are the values in the ***windSpeedLevel*** column: **Low**, **Medium**, and **High**. Many times, we want to change the labels on the graph to be more descriptive.

Adding customized labels is a two-step process:

1. Create a vector with the customized labels
2. Use this vector for the ***labeller*** subcomponent in the ***facet_grid()*** component

9.1 - Components of facet labelling

We need to create a vector that maps the wind speed values (**Low**, **Medium**, **High**) with the label we want for each value:

```
windLabels = c(Low = "Light Winds",
               Medium = "Medium Winds",
               High = "Strong Winds");
```

and then use the parameter ***labeller*** in the component ***facet_grid()*** to add these labels to the legend.

```
labeller=as_labeller(windLabels)
```

*Note: the function **as_labeller()** explicitly says that the vector is a vector of labels*

9.2 - Putting the code together

```
### Part 11: Changing facet labels
windLabels = c(Low = "Light Winds",
               Medium = "Medium Winds",
               High = "Strong Winds");

thePlot = ggplot(data=weatherData) +
          geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp),
                       na.rm=TRUE,
                       color=c("blue", rep("black", 3),
                               "green", rep("black", 3),
                               "orange", rep("black", 3)),
                       fill=c(rep(NA, 8), rep("red", 3), NA)) +
          theme_bw() +
          scale_x_discrete(limits = c("North", "East", "South", "West")) +
          facet_grid(facets=.~factor(windSpeedLevel,
                                     levels=c("Low", "Medium", "High")),
                     labeller=as_labeller(windLabels)) +
          labs(title = "Change in Temperature vs. Wind Direction",
               subtitle = "Lansing, Michigan: 2016",
               x = "Wind Direction",
               y = "Degrees (Fahrenheit)");
plot(thePlot);
```

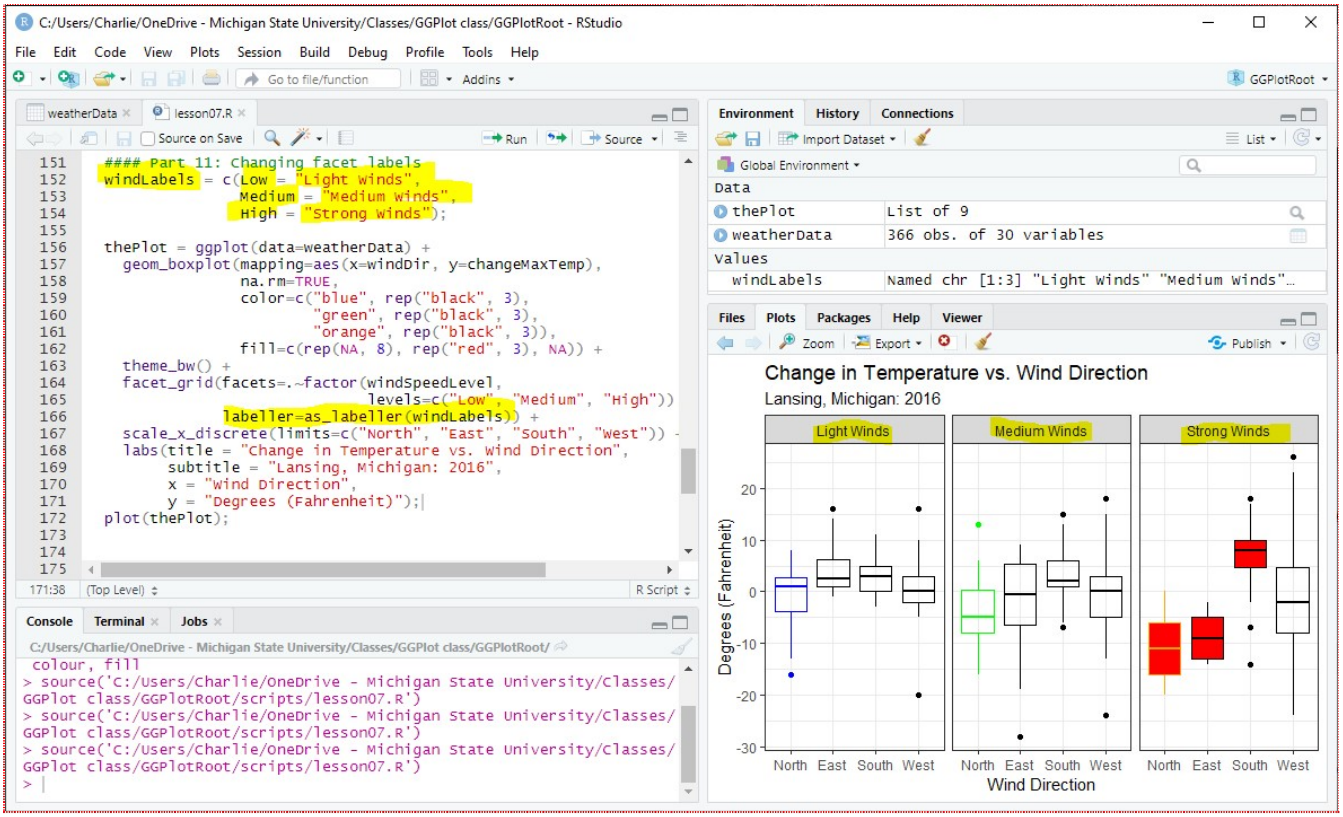

Fig 12: Changing labels on a facet

10 - Application

1. Create a script file in your GGPlot Class Project called ***app07.r***.
2. Create a new column called ***humidityLevel*** that creates three levels for ***relHumidity***
   - Levels are low: 0-30%, medium: 30-70%, high: 70-100%
3. Create a boxplot of ***stnPressure*** vs ***humidityLevel***
   - Put ***humidityLevel*** in order of increasing humidity
4. Add ***windDir*** as a ***fill*** to the boxplot
5. Add a y-axis facet using ***windSpeedLevel***

11 - Extension: RGB Colors

RGB colors represent the three *primary light colors:* red, green, and blue.

Each color has a values from **0** to **1,** with **0** meaning the light is completely off, **1** meaning the light is fully on, and values in between mean the light is partially on.

So:

- **rgb(red=1, green=0, blue=0)** means the red light is fully on, green and blue are off
- **rgb(red=1, green=1, blue=1)** is all three lights fully on (white)
- **rgb(red=1, green=1, blue=1)** is all three lights off (black)
- **rgb(red=0.5, green=0.5, blue=0.5)** is all three light half-on (grey)

12 - Extension: Changing Quantiles

The percentiles GGPlot uses to calculate the quantiles in a boxplot are:
ymin: 5%
lower: 25%
middle: 50%
upper: 75%
ymax: 95%

GGPlot allows the user to set values for the quantiles -- this is shown in the **Aesthetics** section of the geom_boxplot() help page. However, the parameters **lower**, **upper**, **middle**, **ymin**, **ymax** can only be used to set quantile values -- not percentiles. And, if you have multiple boxplots then the parameters need to be a vector with the same number of values as the number of boxplots.

There is no easy way to do this. The way we will use is:
1. Create a function that calculate quantile values and return a vector
2. Call function and get the vector
3. Apply vector to the quantile parameters

12.1 - Quantile function

In the reference.r script file, we will add a new function called ***findQuants()***.

findQuants() will take four parameters:
- vector for the dependent variable
- vector for the independent variable
- a list of factors
- the percentile level

findQuants() will return a vector that gives the quantile value for each factor. So, the return vector will have the same number of values as the factor list.

```
1  findQuants = function(yVar, xVar, factors, percentile)
2  {
3    quants = c();
4    for(i in factors)
5    {
6      quantIndex = which(xVar == i);
7      quants[i] = quantile(yVar[quantIndex], percentile, na.rm=TRUE);
8    }
9    return(quants)
10 }
```

12.2 - Calling findQuants()

For the following example, we are going to change the lower quantile to 35% and the upper quantile to 65% for all boxplots.

To do this we need to call ***findQuants()*** twice: once for ***lower*** and once for ***upper***. The return values from

findQuants() are saved to the vectors *lowVal* and ***highVal***.

```
#### Extension: Setting quantile values
lowVal = findQuants(yVar = weatherData$changeMaxTemp,
                    xVar = weatherData$windDir,
                    factors = c("North", "East", "South", "West"),
                    percentile = 0.35);

highVal = findQuants(yVar = weatherData$changeMaxTemp,
                     xVar = weatherData$windDir,
                     factors = c("North", "East", "South", "West"),
                     percentile = 0.65);
```

12.3 - Applying quant values to the plot

We are going to set the parameters ***lower*** and ***upper*** to the vector ***lowVal*** and ***highVal***.

```
#### Extension: Applying quantile values
thePlot = ggplot(data=weatherData) +
          geom_boxplot(mapping=aes(x=windDir, y=changeMaxTemp),
                       na.rm=TRUE,
                       lower=lowVal,
                       upper=highVal) +
          scale_x_discrete(limits=c("North", "East", "South", "West")) +
          theme_bw() +
          labs(title = "Change in Temperature vs. Wind Direction",
               subtitle = "Lansing, Michigan: 2016",
               x = "Wind Direction",
               y = "Degrees (Fahrenheit)");
plot(thePlot);
```

Executing the script creates boxplots with a smaller box that in the original (*fig 1*):

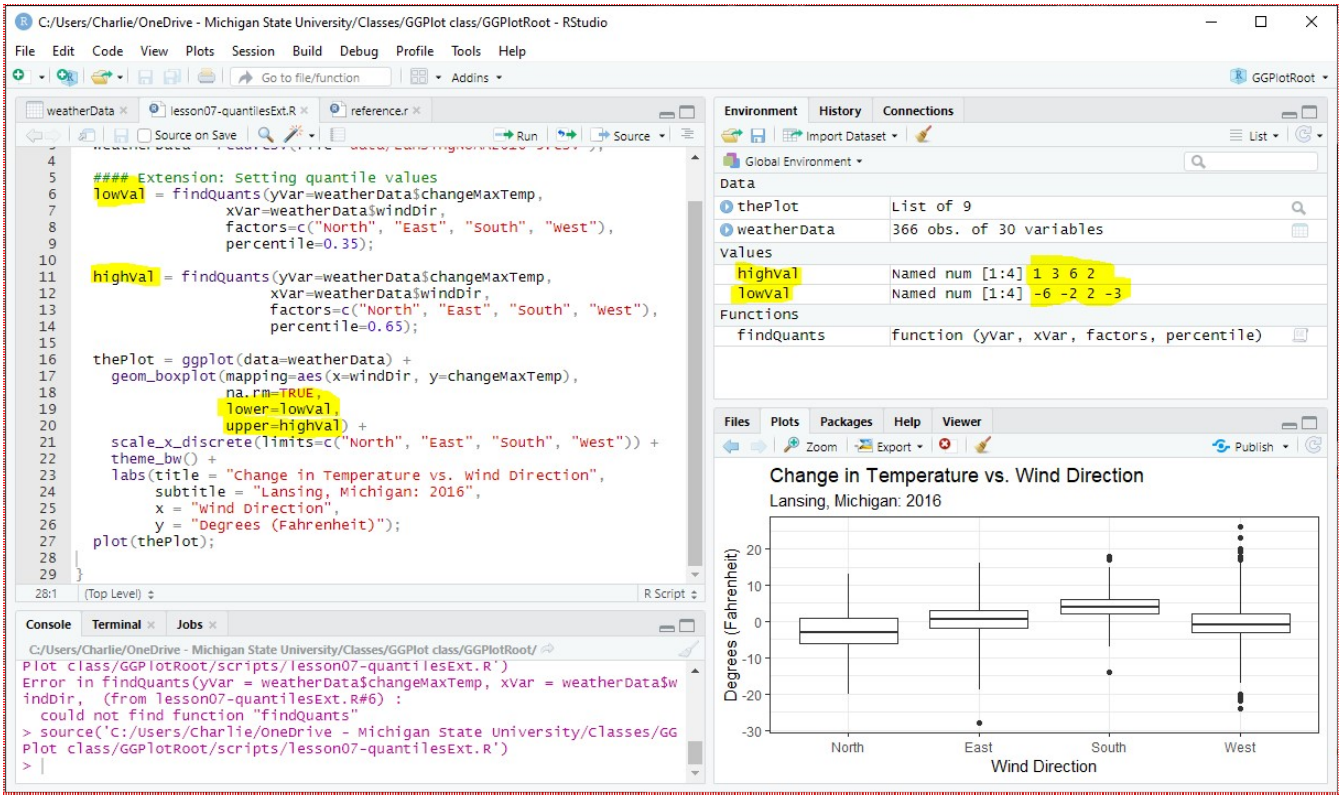

Fig 13: Applying new quantile values to the boxplots