**Biostatistics**

**Lab 5**

*Tasks*

Visualization, ggplot2 (Boxplots, Histograms), Colors

*Introduction*

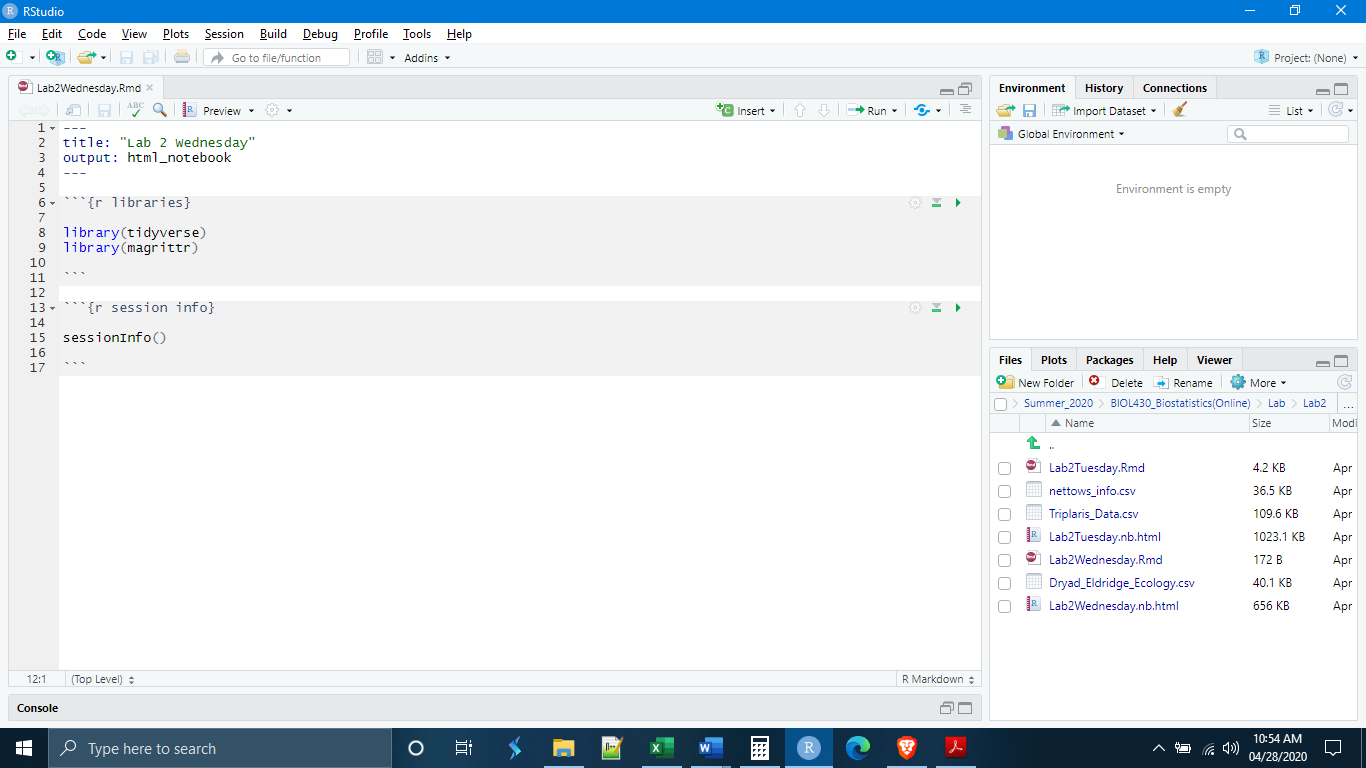
Thus far we have focused on the fundamentals of data importation and data manipulation in R. Data manipulation skills are essential for proficiency in any statistical computing software. At the same time, logical expressions and matrix manipulation is difficult intellectual labor and can be at times a bit draining. As a result, I want to try and lighten the load today by shifting to the end product of a statistical analysis, the visualization.

The visualization package associated with the tidyverse is ggplot2. I would encourage you to have a look at the ggplot cheat sheet, as it will be a useful resource throughout this process. ggplot2 is one of the gems of the tidyverse and is used by professional academics and statisticians to create informative, intuitive, and good-looking data visualizations.

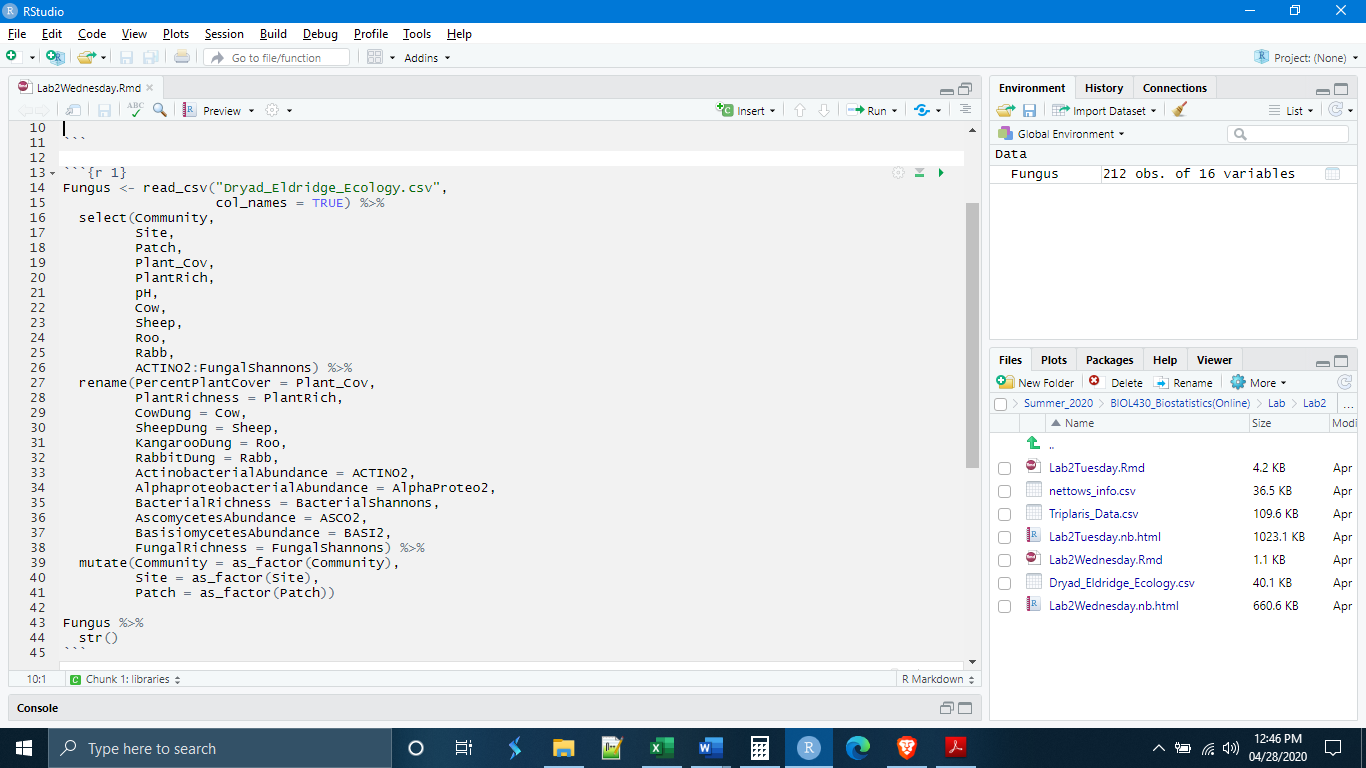
*Data Reading and Cleaning in dplyr*

If you have not done so already, download all of the data files necessary for the lab this week and save them in the Lab2 folder you made yesterday. The file that will be most important for today is the Dryad\_Eldridge\_Ecology.csv data.

Start R Studio and open a new R Notebook file. Save this file in the Lab2 folder where the data files are. Delete the text that comes in the notebook template, and use the first chunk to load libraries. It is good practice to also present your session info, make a last chunk to do this.



The first thing that we will need to do is read in our data and clean it. We will not spend a lot of time on this because these are the dplyr tools that we learned last week and yesterday. The data for today’s exercises also comes from a study conducted in Australia. Here, the researchers were looking at changes in soil microbial community composition as a result of animal grazing.

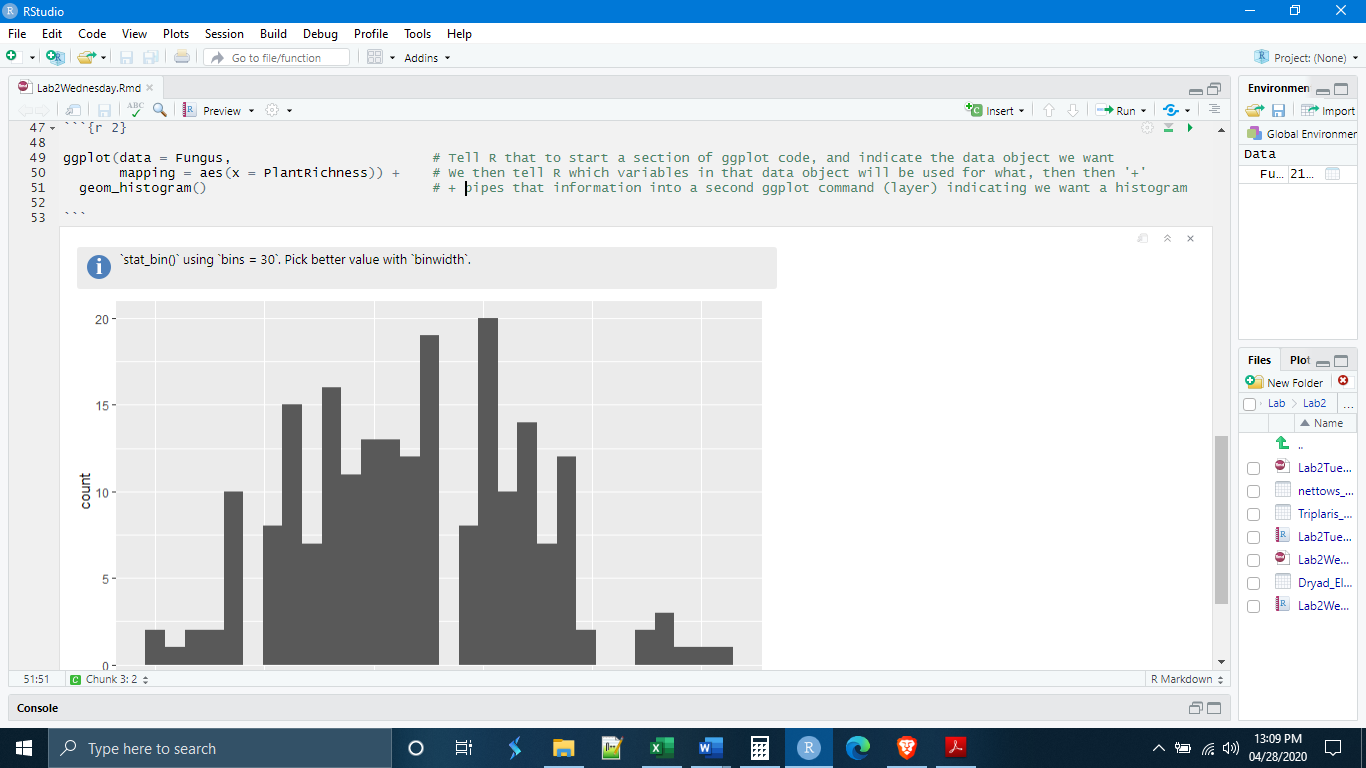


*ggplot Basics*

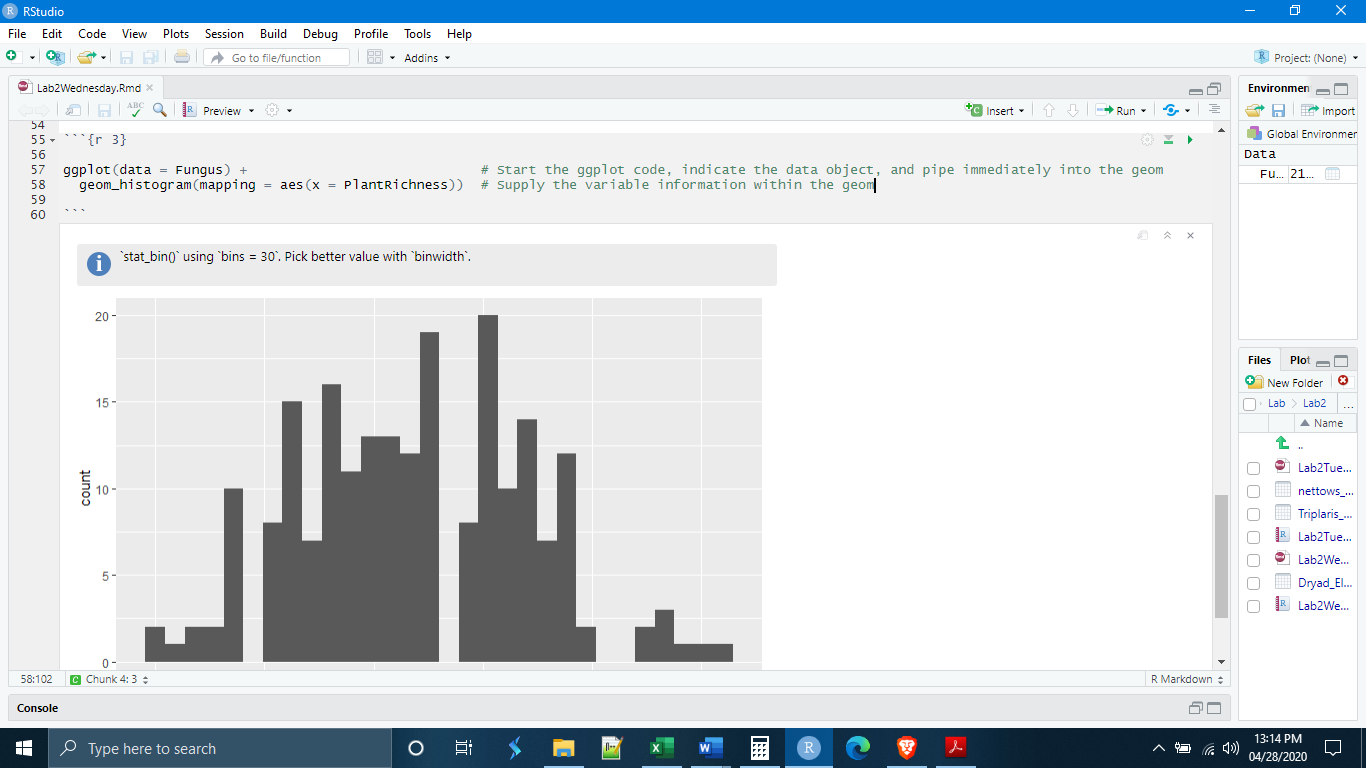
ggplot2 was developed before the forward pipe operator was developed. As a result, ggplot2 uses a slightly different syntax than dplyr. In place of the %>% that is familiar to us, ggplot uses +. ggplot3 is in development and may use the pipe operator from dplyr, but for now we have to get used to the fact that within ggplot code, we have to use + instead of %>%.

Second, ggplot functions through layers. Each time you add a new term to your ggplot code, it places a new layer on top of your visualization. This will make more sense in a few minutes when we start to code, but I mention it because sometimes the order of layers matters.

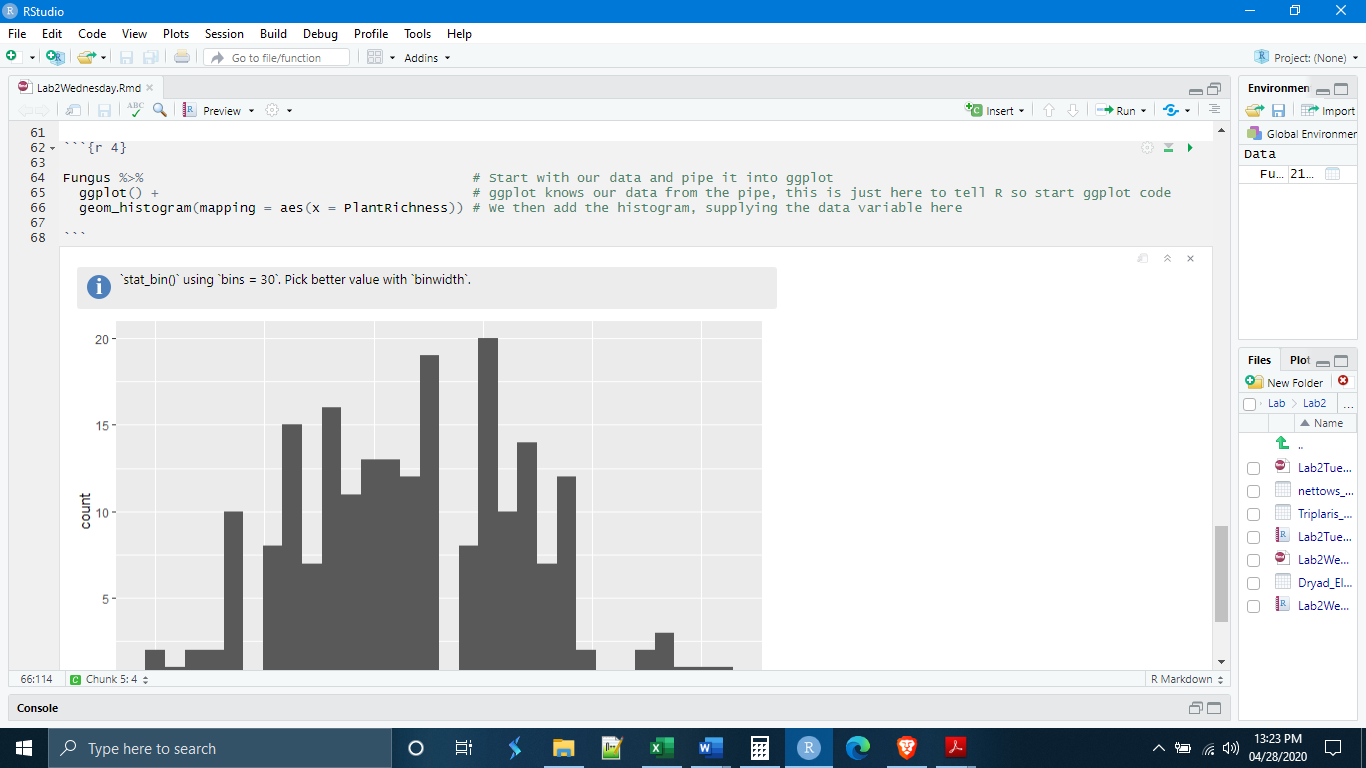
Let’s say we want to make a histogram of the species richness values. If, for a moment, we forget what we have learned from our dplyr labs, we could write the code like this



It is also possible to supply the information about variables later, within the geom\_histogram() command.

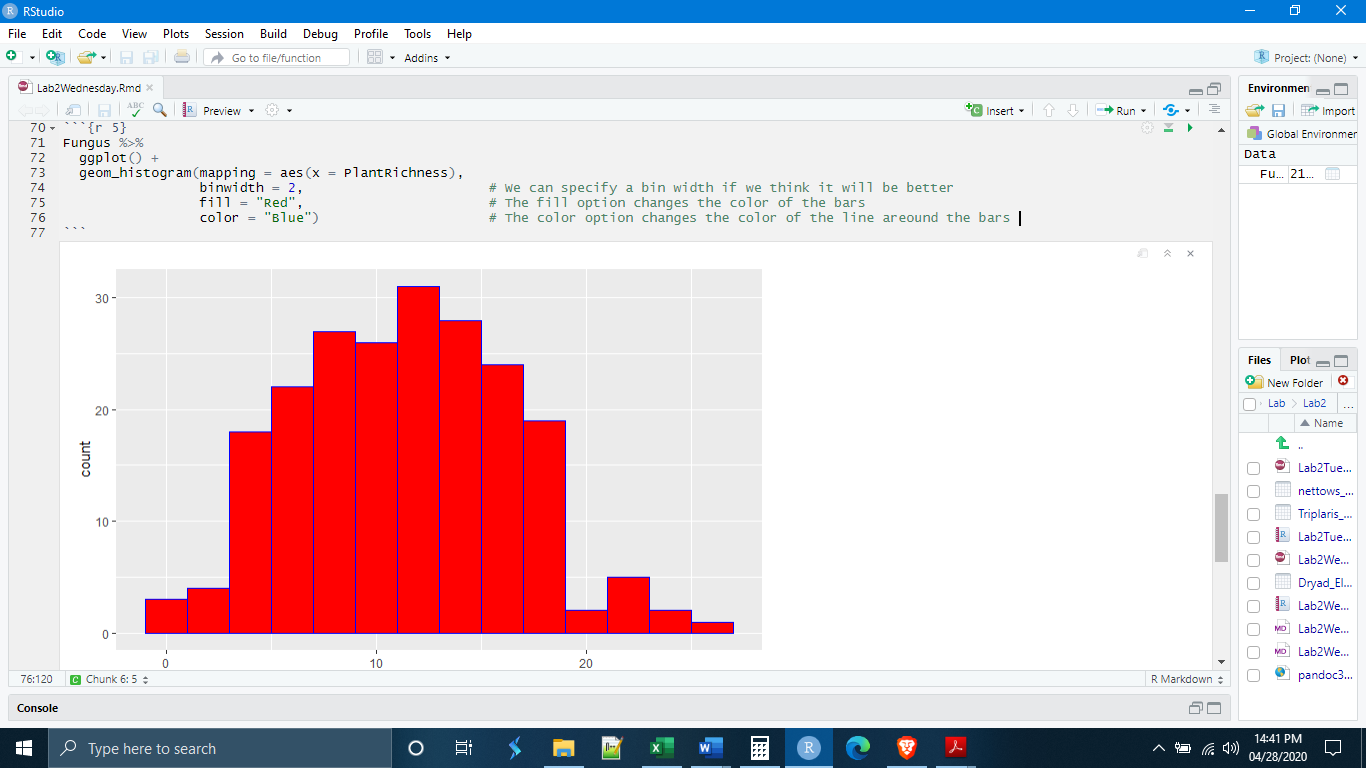


The final way is to use the dplyr pipe to pipe the data into ggplot. This is the preferred method because it allows you to make modifications to the data necessary for plotting, without needing to create a separate data object. Instead, you run a short dplyr pipe from the main data object before beginning ggplot, and pipe the modified data directly into ggplot.

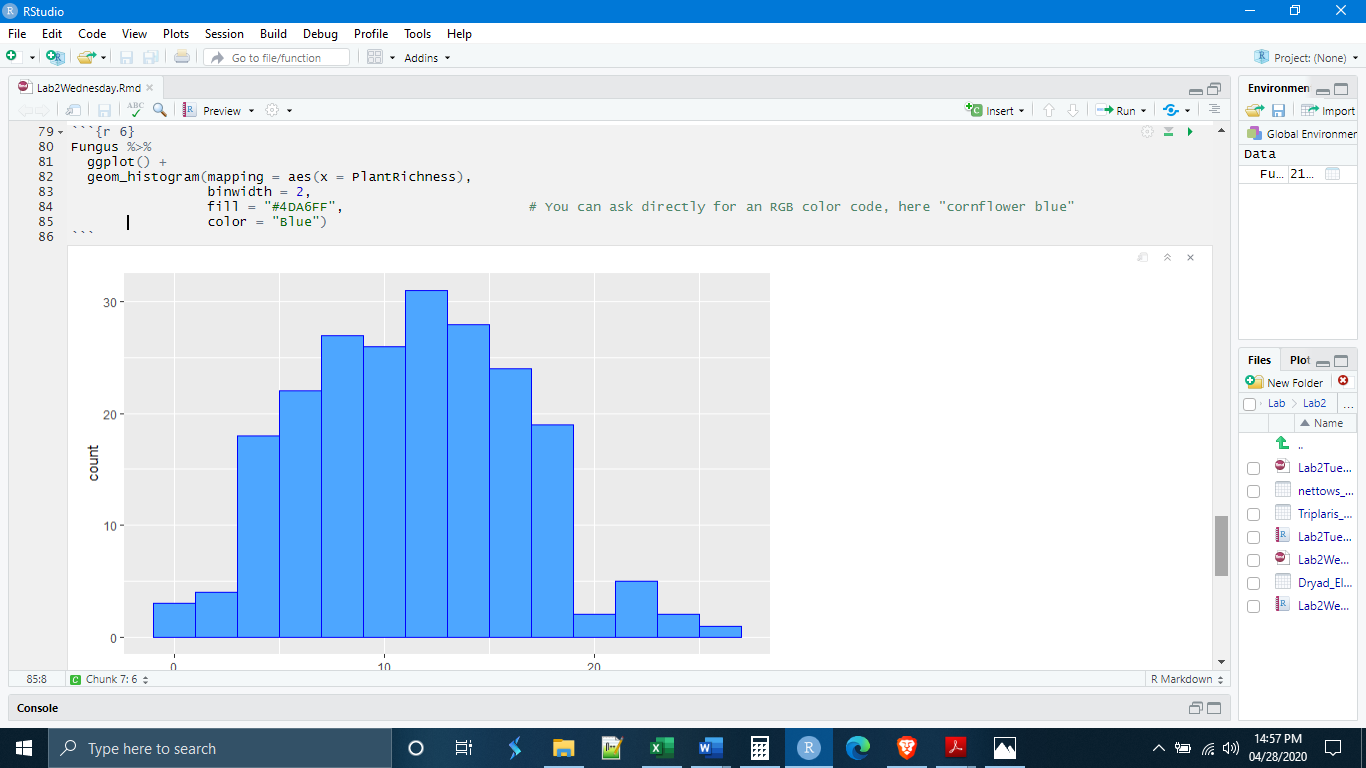


*Histograms*

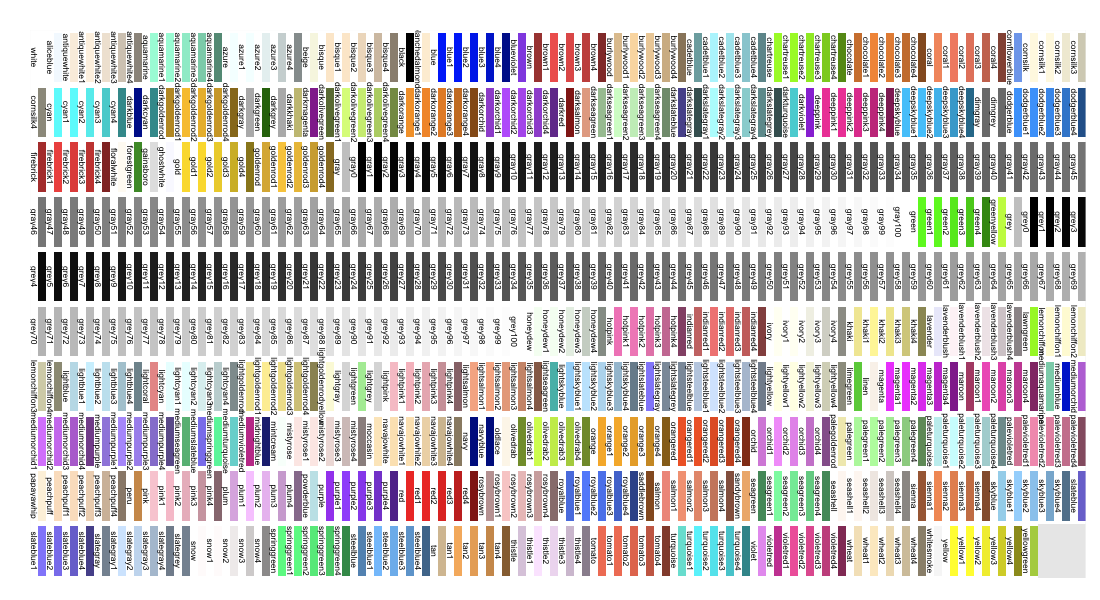
The histogram takes a single variable and plots the observation values on the x axis and the frequency of observation values on the y axis. The histogram helps to visualize the data and determine its distribution. The histogram ‘bins’ observation values, particularly if they do not fall into natural categories. As a default, geom\_histogram() will automatically choose the bin size. We can manually select the bin size, as well as the colors, with the code below.



ggplot has a good number of pre-loaded colors in its palette. It will accept any RGB color code (ex. cornflower blue = #4DA6FF).



It is also possible to call certain colors by name so that you don’t need to look up the code number. We have already done this with basic colors like “red” or “blue”, but there are quite a few options. Under no circumstances will you be expected to know these colors off the top of your head. This information is only for your reference.



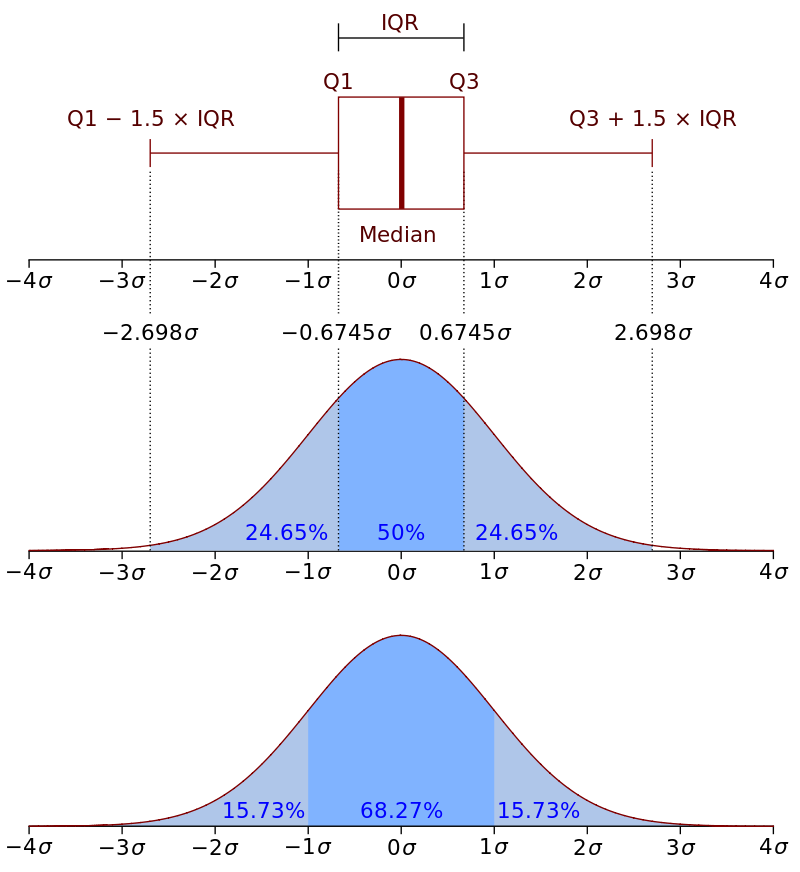
(http://sape.inf.usi.ch/quick-reference/ggplot2/colour)

*The geoms*

The ggplot command for a histogram is geom\_histogram(). The geom\_...() is a standard vocabulary for the different kinds of coordinate grid objects and plots. There are many of them. The ggplot cheat sheet will be your friend. It lists a large number of the geoms and their uses. There are such things as geom\_histogram(), geom\_bar(), geom\_point(), geom\_boxplot(), as well as things like line objects such as geom\_abline() and geographic map functions such as geom\_map(). Again, it would be a good idea to familiarize yourself with the ggplot cheat sheet.

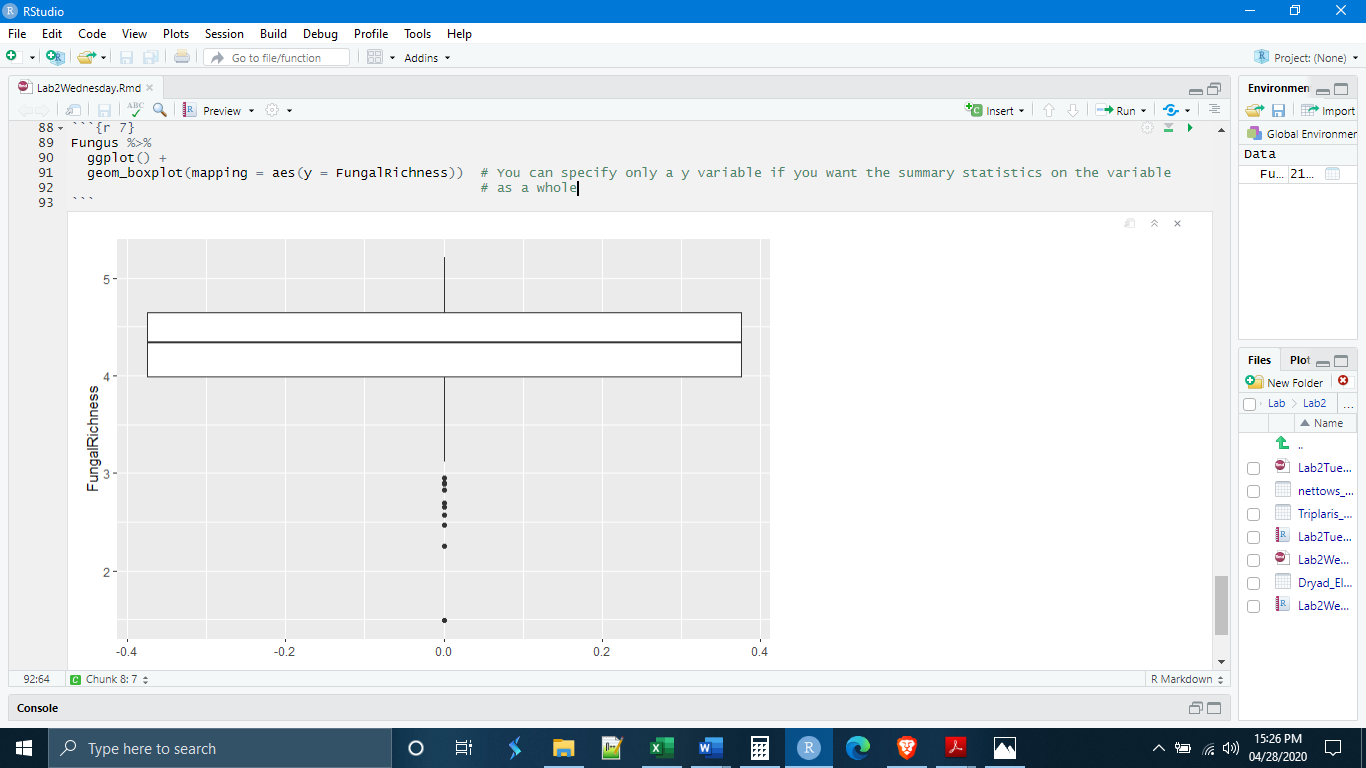
*Boxplots*

A boxplot (also sometimes called a box and whisker plot) is another plot designed to investigate the distribution of a data set. The boxplot provides quantitative information about the center and spread of the data that the histogram does not. The boxplot shows the median, the middle quartiles, and “whiskers” that represent the major part of the tails. If the distribution of observations is normal, fifty percent of the observations will fall inside of the “box”, most of the remaining fifty percent of the observations in the “whiskers”, and only observations in percentiles less than 1 and greater than 99 will fall outside of this range.

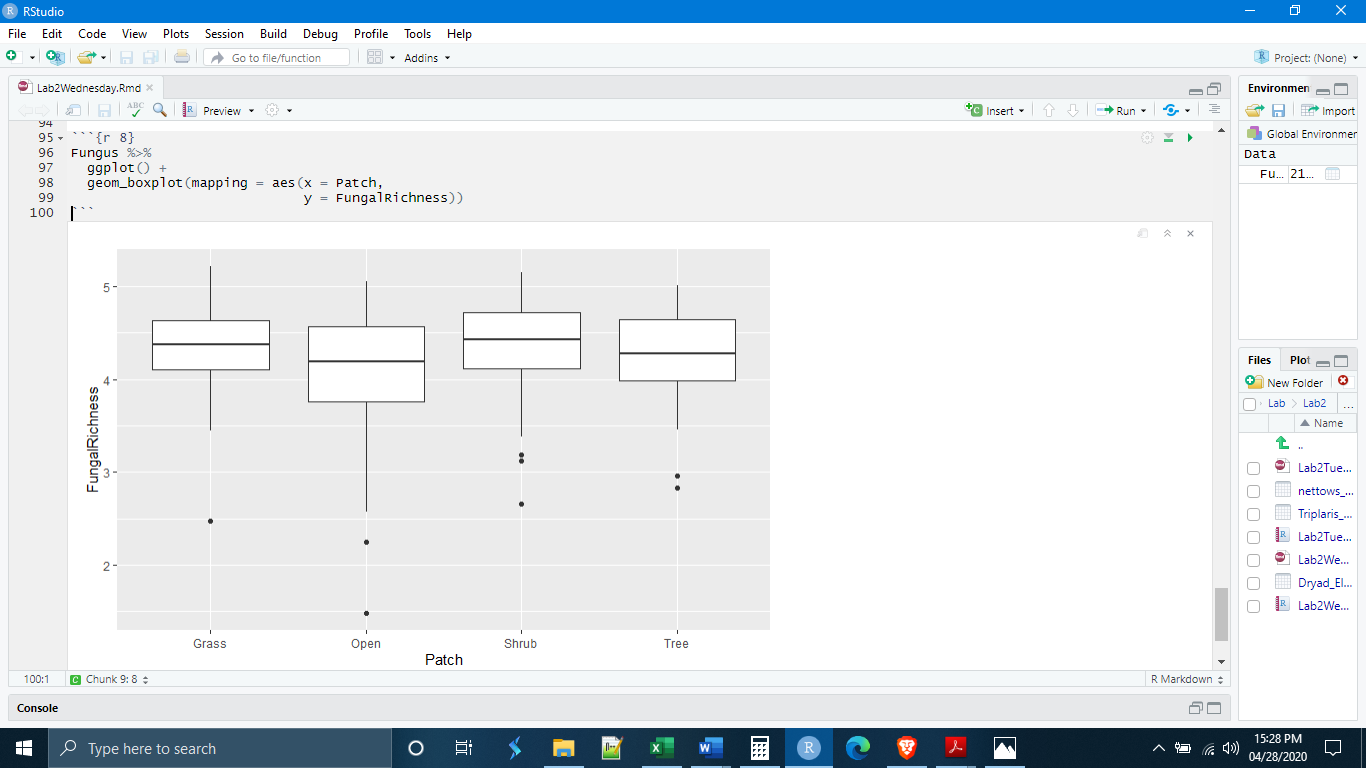


(https://upload.wikimedia.org/wikipedia/commons/1/1a/Boxplot\_vs\_PDF.svg)

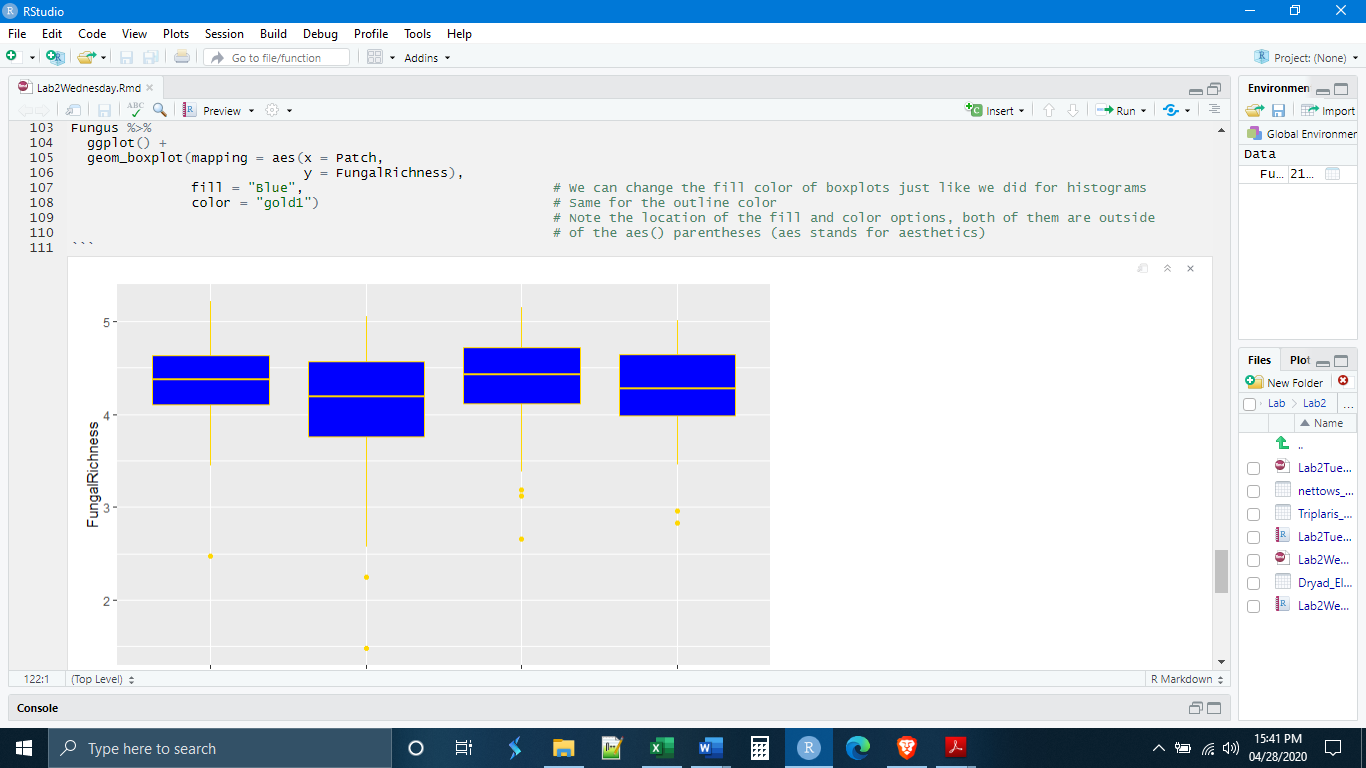
For a basic boxplot, the code in ggplot will look very similar to that of a histogram. A boxplot can be made for a single variable to investigate the center and spread of that variable or group of observations. This can be accomplished in ggplot by specifying only a y variable. ggplot will fill in the x axis with cartesian coordinates.



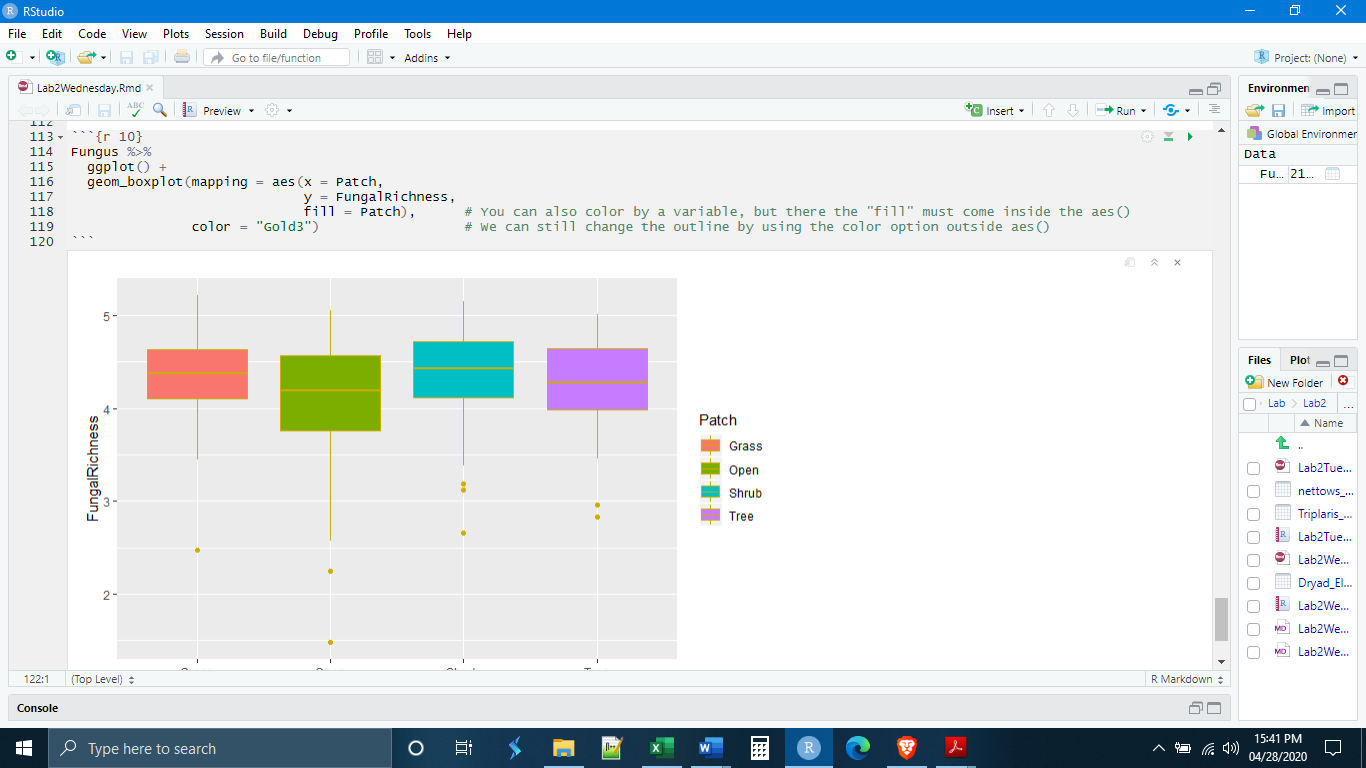
It is also useful, however, to compare the center and spread of multiple groups of data. For example, in today’s data, one of the variables was the patch types where fungal diversities were measured. We can use the patch type as the x variable to get a boxplot for each patch type.



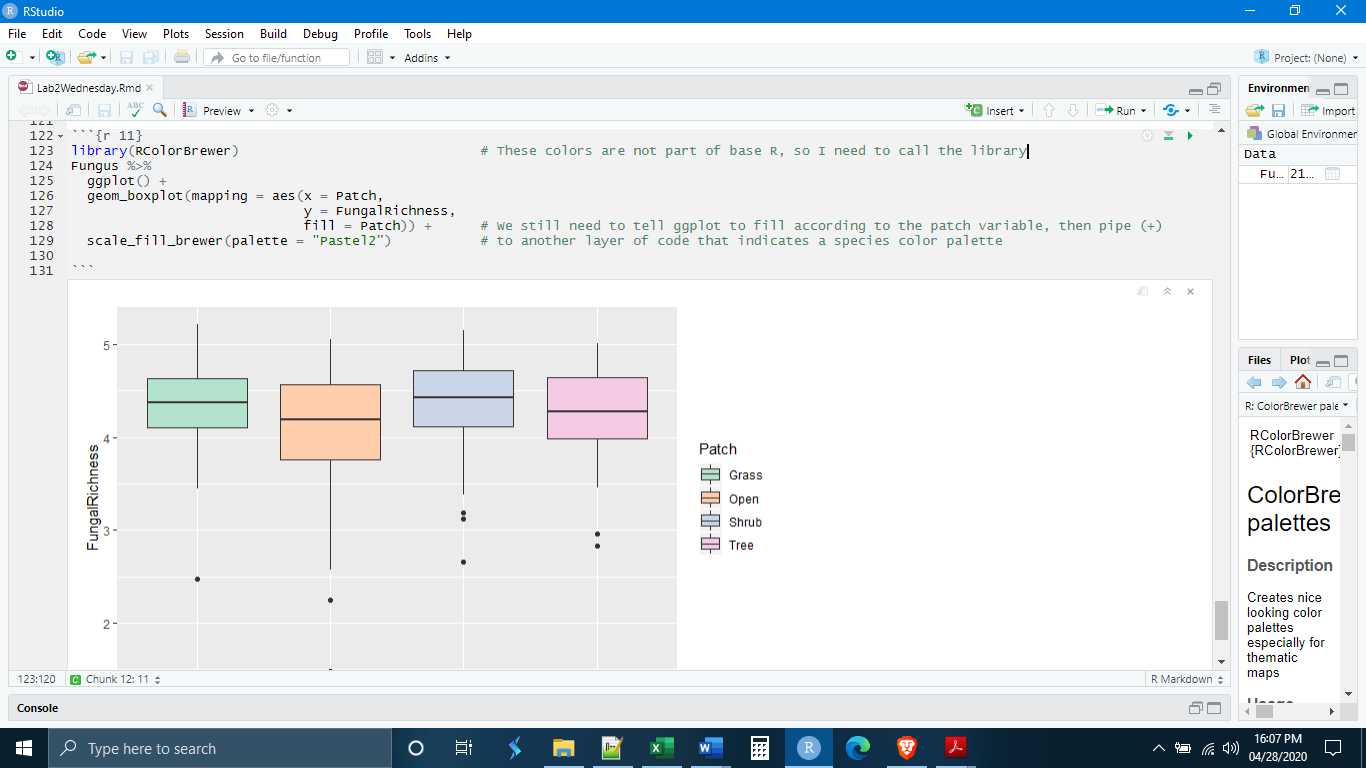
Changing general colors in boxplots can be accomplished in the same way as with the histogram. One thing to notice is where we are placing the fill and color options; they are outside the aes() parentheses. Also, the colors are placed inside quotation marks. If you leave out the quotation marks, R will think you are referencing a particular data object.



It is also possible to color the boxplots according to the observations in a particular variable. For example, if we wanted to highlight the patch types, we could stipulate a fill according to the Patch variable itself. In this case, ggplot will give each of the patch types its own color. Note, however, that this fill option goes inside the aes() parentheses.



Finally, when ggplot colors according to a variable, it selects from one of its default color palettes (collections of colors intended to look good together) depending on how many colors it needs (in our case, it would need four colors). We can manually change this color palette by adding another layer to our ggplot code specifying a special color palette. Here I am selecting from the RColorBrewer, which is a collection of color palettes that can be downloaded. Notice that I had to call a new library before the code to remind R that is had RColorBrewer installed.



*Individual Exercises*

**1.** Make a new chunk called “chunk 5” and, using the Fungus data we made today (you will need to copy and rerun the cleaning/import code from the beginning of today), write code to make a histogram of the pH variable. Set the bin width to 0.1. Describe the distribution. Is the distribution skewed? Record the answer as a comment in the chunk.

**2.** Make a new chunk called “chunk 6” and, using the Fungus data, write code to make boxplots of the ActinobacterialAbundance variable for each of the communities (similar to what we did on pages 17-19). Color the boxplots based on the Community variable.

**3.** Make a new chunk called “chunk 7” and, using the Fugus data, write code that first filters out any PercentPlantCover above 25, and then pipes this data directly into ggplot to make boxplots of the PercentPlantCover variable in each of the patch types. Color the boxplots based on the patch types.

**4.** Make a new chunk called “chunk 8” and paste in the code below (next page). You have seen that in these labs I have sometimes put in comments to indicate what each of the lines of code is doing. I want you to add comments for each line of this code indicating what that line is doing. Feel free to run the code first to see what happens.

Fungus %>%

group\_by(Community,

Site) %>%

summarize(SiteKangDung = mean(KangarooDung)) %>%

ggplot() +

geom\_boxplot(mapping = aes(x = Community,

y = SiteKangDung,

fill = Community))