**Biostatistics**

**Lab 6**

*Tasks*

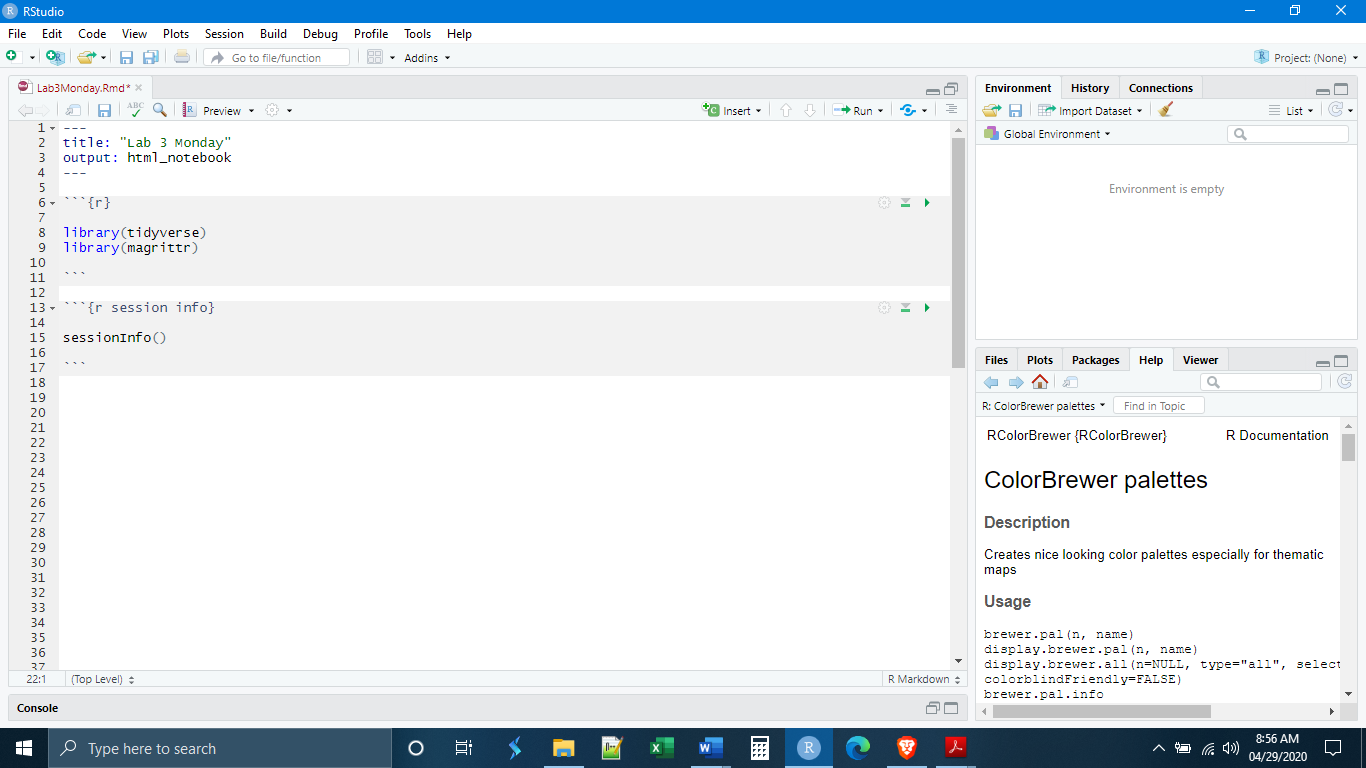
More ggplot2 (Scatter Plots, Facets, Coord Flips, Chart Labels)

*Introduction*

We had a basic introduction to ggplot last week. One of the advantages of working within the tidyverse is that once you have a basic understanding of “the rules” so to speak the rest is just variations on the same theme. The goal for today is to reinforce what we know about dplyr and ggplot, as well as learn some new visualization techniques in ggplot.

As usual, begin by creating a new folder, this time “Lab3” which we will use for all our files for this week. Second, download from Blackboard all of the data files necessary for the lab this week and save them in this same folder. The files that will be most important for today are Dryad\_Eldridge\_Ecology.csv and Sebastes\_raw\_morphology.csv.

Start R Studio and open a new R Notebook file. Save this file in the “Lab3” 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.



Next, we will want to read in, clean, and save our two data sets. The dplyr code for cleaning the fungus data is the same as last week and for the sake of time I have included it here. Simply copy-paste it into a new chunk.

Fungus <- read\_csv("Dryad\_Eldridge\_Ecology.csv",

col\_names = TRUE) %>%

select(Community,

Site,

Patch,

Plant\_Cov,

PlantRich,

pH,

Cow,

Sheep,

Roo,

Rabb,

ACTINO2:FungalShannons) %>%

rename(PercentPlantCover = Plant\_Cov,

PlantRichness = PlantRich,

CowDung = Cow,

SheepDung = Sheep,

KangarooDung = Roo,

RabbitDung = Rabb,

ActinobacterialAbundance = ACTINO2,

AlphaproteobacterialAbundance = AlphaProteo2,

BacterialRichness = BacterialShannons,

AscomycetesAbundance = ASCO2,

BasisiomycetesAbundance = BASI2,

FungalRichness = FungalShannons) %>%

mutate(Community = as\_factor(Community),

Site = as\_factor(Site),

Patch = as\_factor(Patch))

Fungus %>%

str()

Similarly, for the sake of time we will not work through the cleaning of the fish data. If you have any question about the code, feel free to ask. These data are from a taxonomic study or rockfishes in the north Pacific Ocean. Most of the variables are morphological measurements.

FishData <- read\_csv("Sebastes\_raw\_morphology.csv",

col\_names = TRUE) %>%

rename(BodyLength = "TL",

BodyWidth = "INTOPERC",

EyeWidth = "EYEWID",

UpperJawLength = "UPPJAW",

LowerJawLength = "LOWJAW",

GillRakers = "RAKENUM",

RakerLength = "RAKELEN",

PectoralFinLength = "PECTLEN",

PectoralFinWidth = "PECTHT") %>%

select(sampleID,

Species,

CommonName,

BodyLength,

BodyWidth,

EyeWidth,

UpperJawLength,

LowerJawLength,

GillRakers,

RakerLength,

PectoralFinLength,

PectoralFinWidth) %>%

mutate(Species = as\_factor(Species),

CommonName = as\_factor(CommonName),

NumObs = 1)

FishData %>%

group\_by(CommonName) %>%

summarize(Obs = sum(NumObs)) %>%

filter(Obs > 10) %>%

arrange(desc(Obs))

FishData <- FishData %>%

mutate(Desired = str\_replace\_all(CommonName, c("COPPER" = "1",

"BLACK" = "1",

"BLUE" = "1",

"CANARY" = "1",

"VERMILLION" = "1",

"BANK" = "1",

"BROWN" = "1",

"YELLOWTAIL" = "1",

"GREENSTRIPE" = "1",

"PINKROSE" = "1")),

Desired = as.numeric(Desired)) %>%

filter(Desired == 1) %>%

mutate(Species = as\_factor(Species),

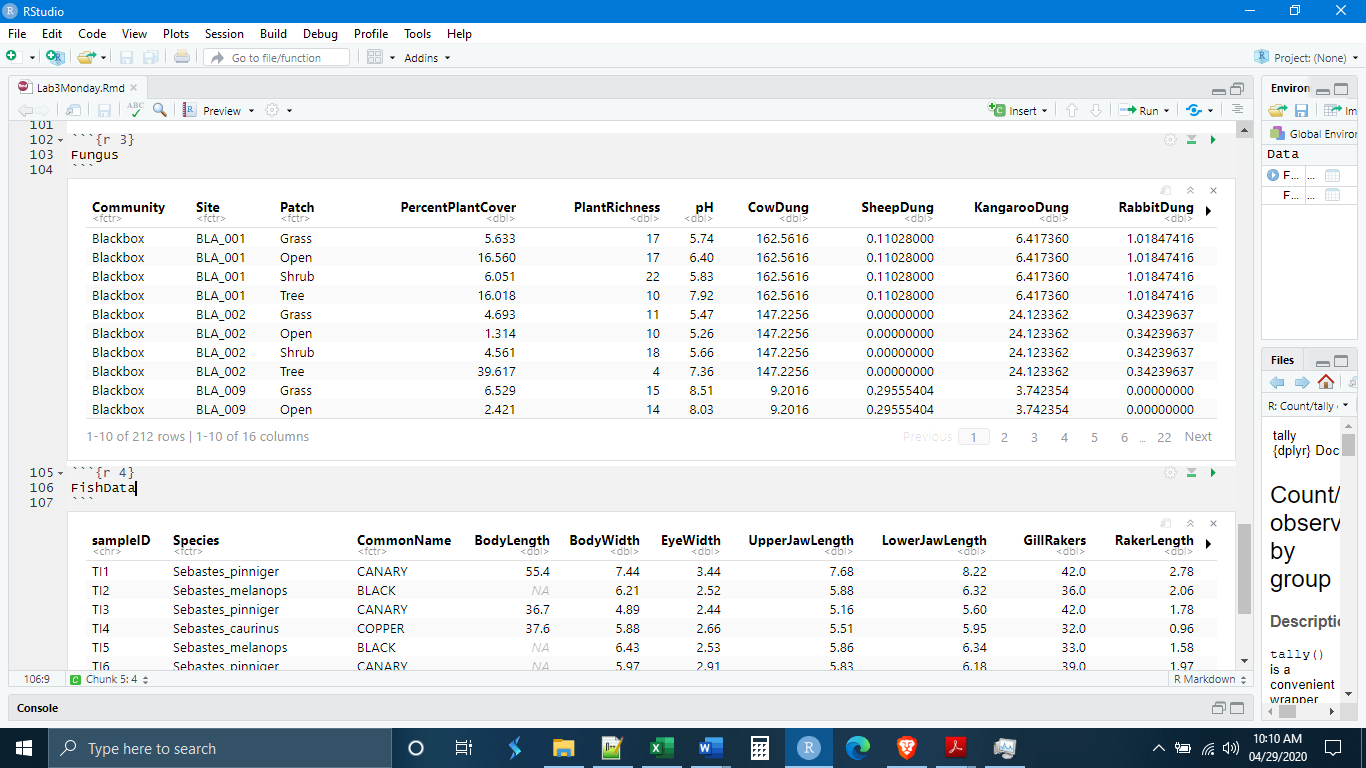
CommonName = as\_factor(CommonName)) %>%

select(-NumObs,

-Desired)

FishData %>%

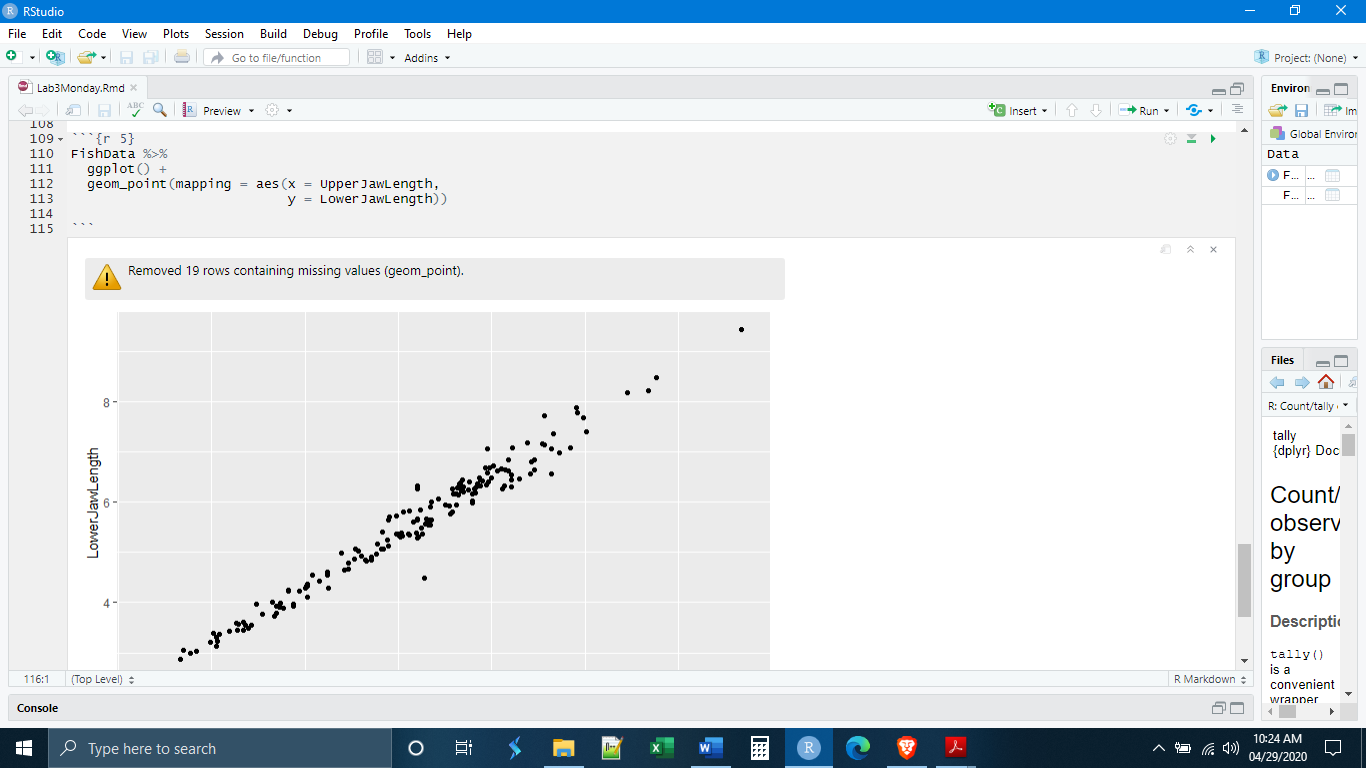
summary()



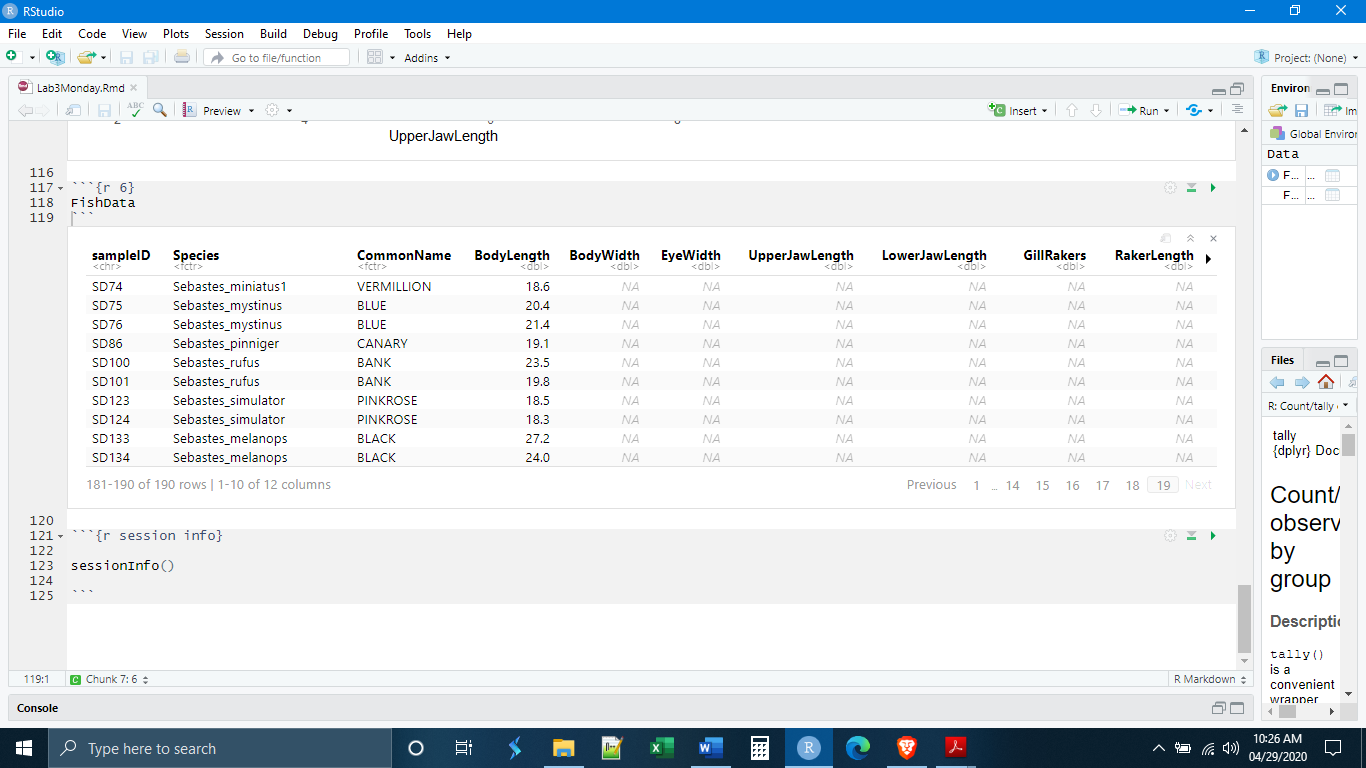
*Scatter Plots*

Now that we have our data, we can move on to plotting it. Last week we learned how to make a histogram, which is a single variable plot, and a boxplot, which can be a single or two-variable plot where one of the variables is discrete and the other continuous. Today we will learn how to construct another two-variable plot, the scatter plot. Scatter plots are useful for visualizing the relationship between two continuous, numeric variables. Scatter plots are particularly useful visualization tools when assessing regression and correlation.

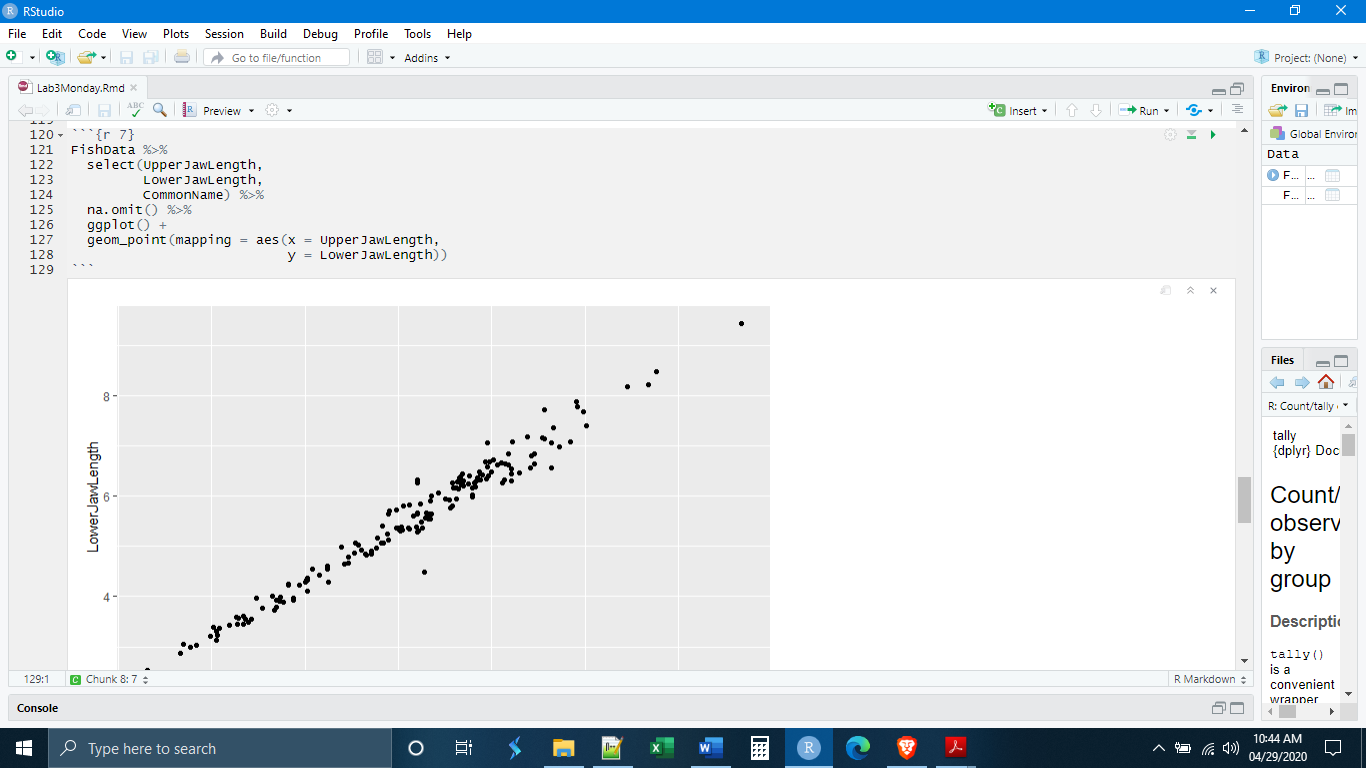
The ggplot function for a scatter plot is geom\_point(). Like the boxplot, it will take an x and a y variable. Let’s begin by looking at the fish data and plotting the lengths of the upper and lower jaw.



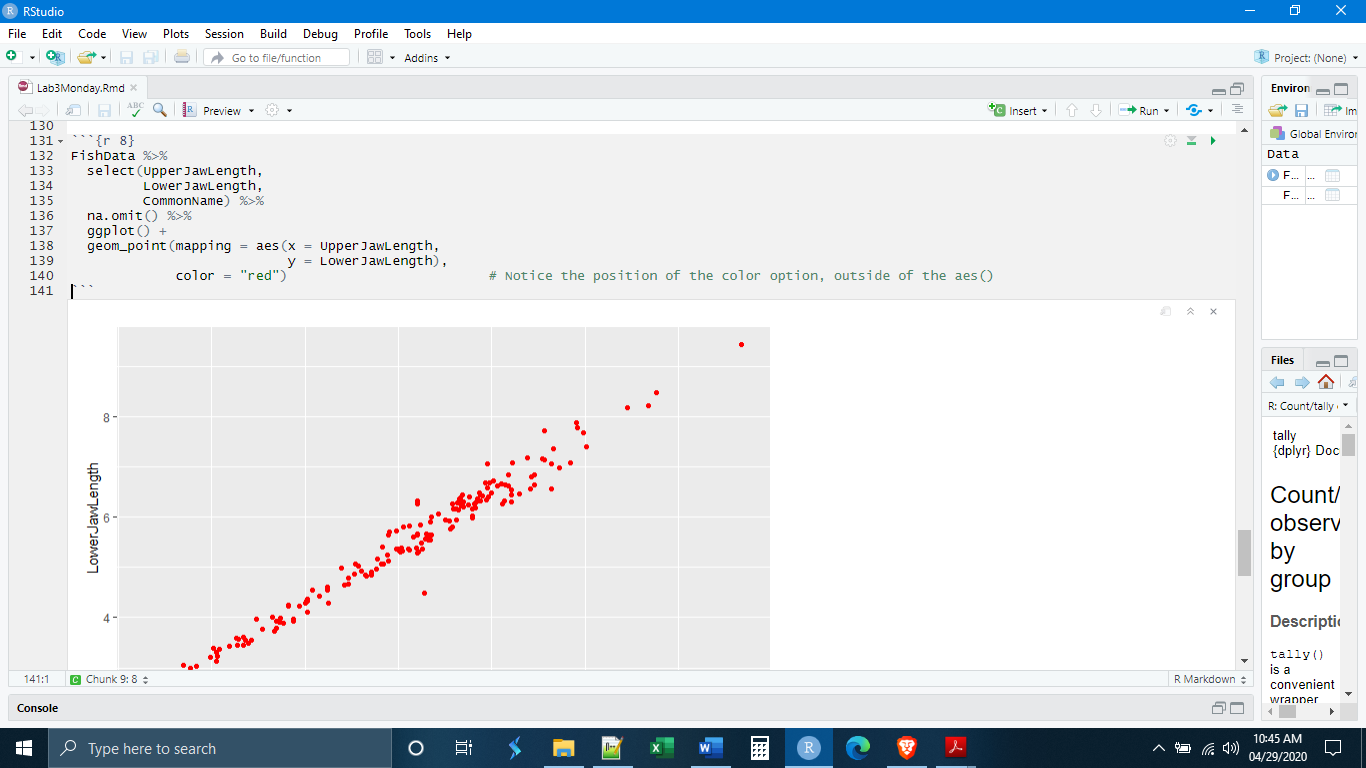
Notice that R has given us a warning message saying that it has omitted a number of observations because the data were missing. If we look at our data table and scroll all the way to the bottom, we see where these missing data are.



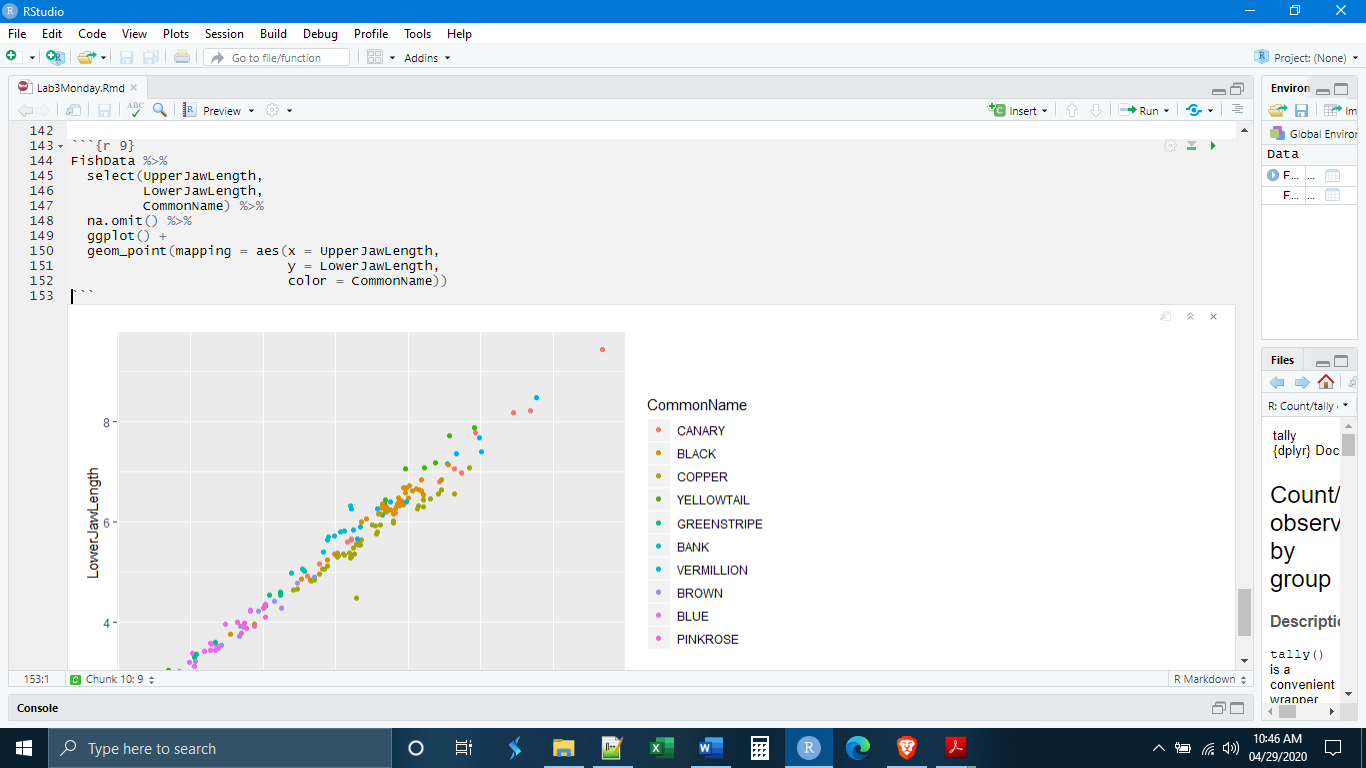
We can remove these data before they get into ggplot with the na.omit() command. Note that the na.omit() command will remove any observation (row) that has any missing data, so it is best to limit the variables that you apply na.omit() to, as I have done below. But you will see that we no longer get the warning message.



For scatterplots, if you want to change the color of all of the points together, you must use the color option. The fill option will not do anything. Also, as was the case for the histogram and boxplot, to change colors throughout you need to use the color option outside of aes().

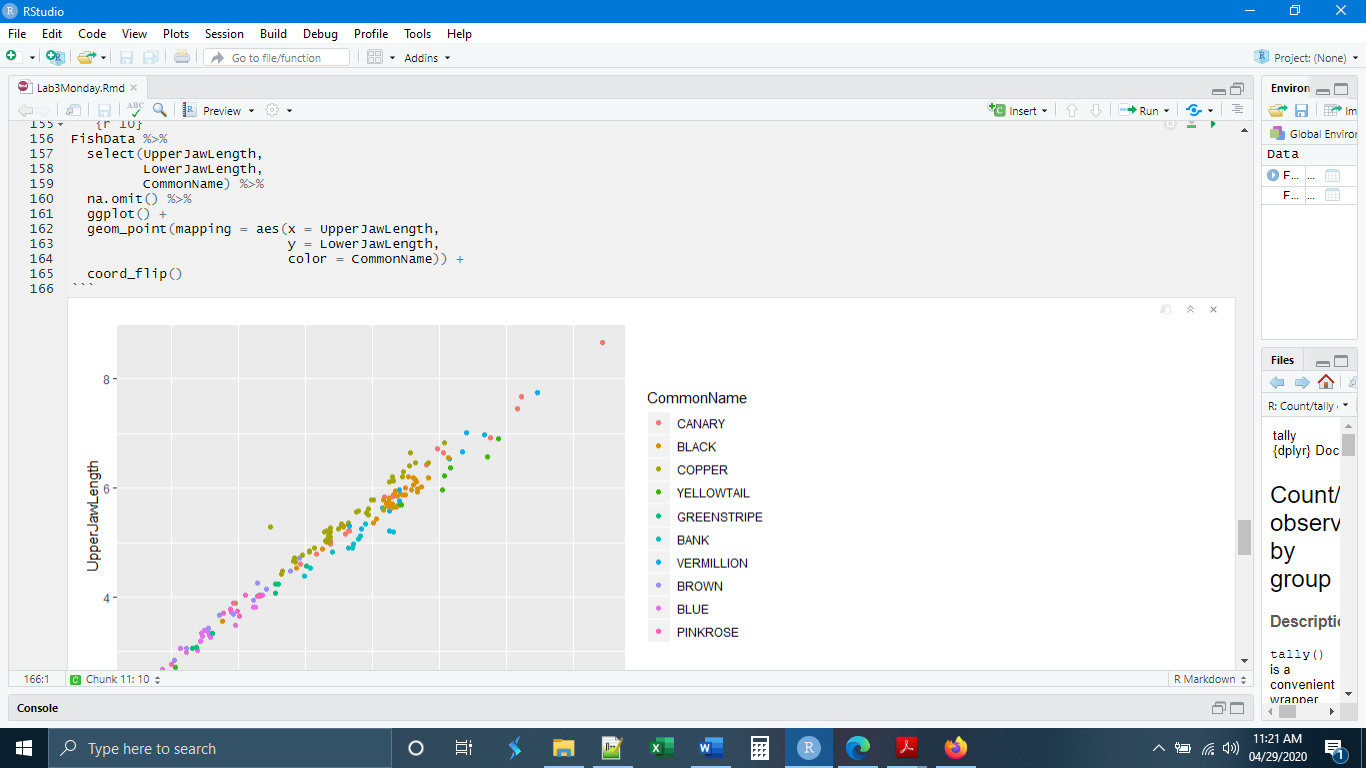


Like with the boxplots, we can also color by a variable, here let’s try coloring by the common name of the fish. Again, note that when coloring by a variable, you must use the color option inside of aes() and with no quotation marks.

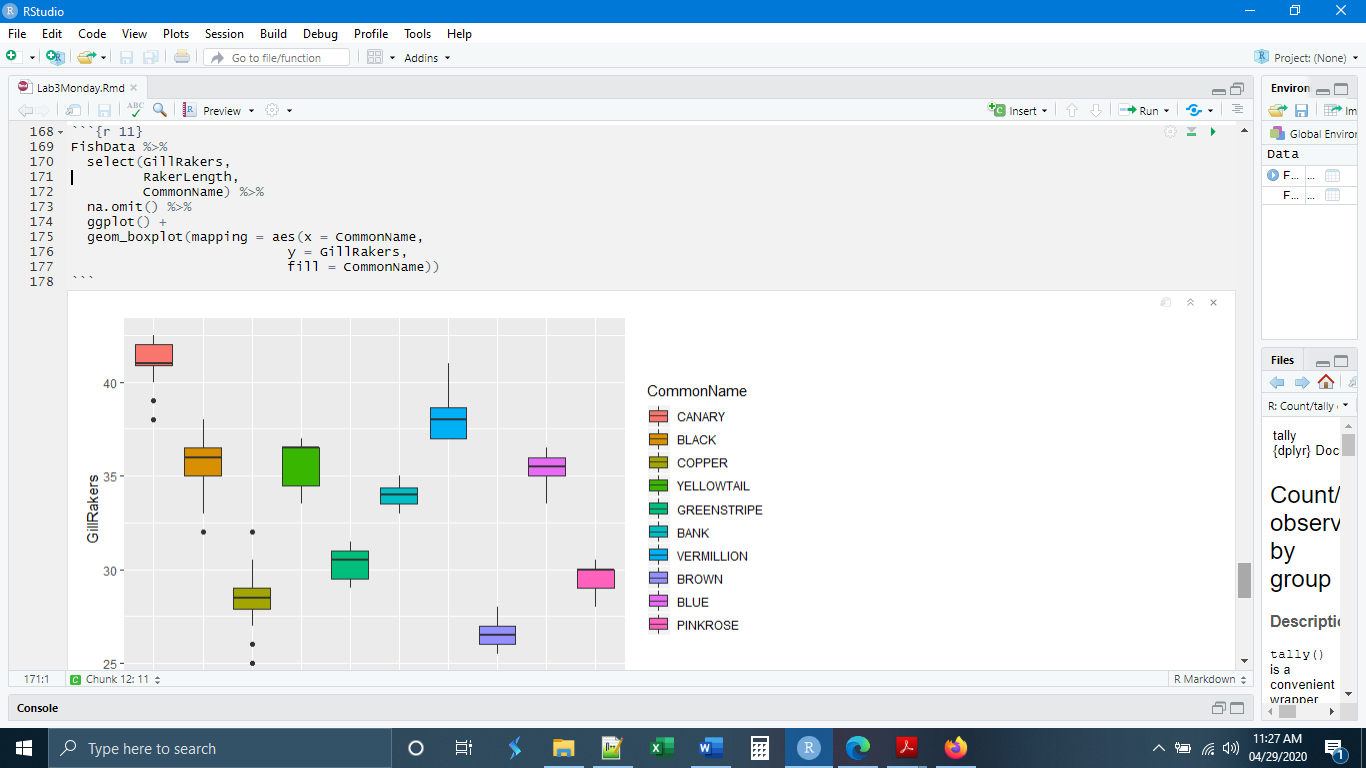


*Coord Flips*

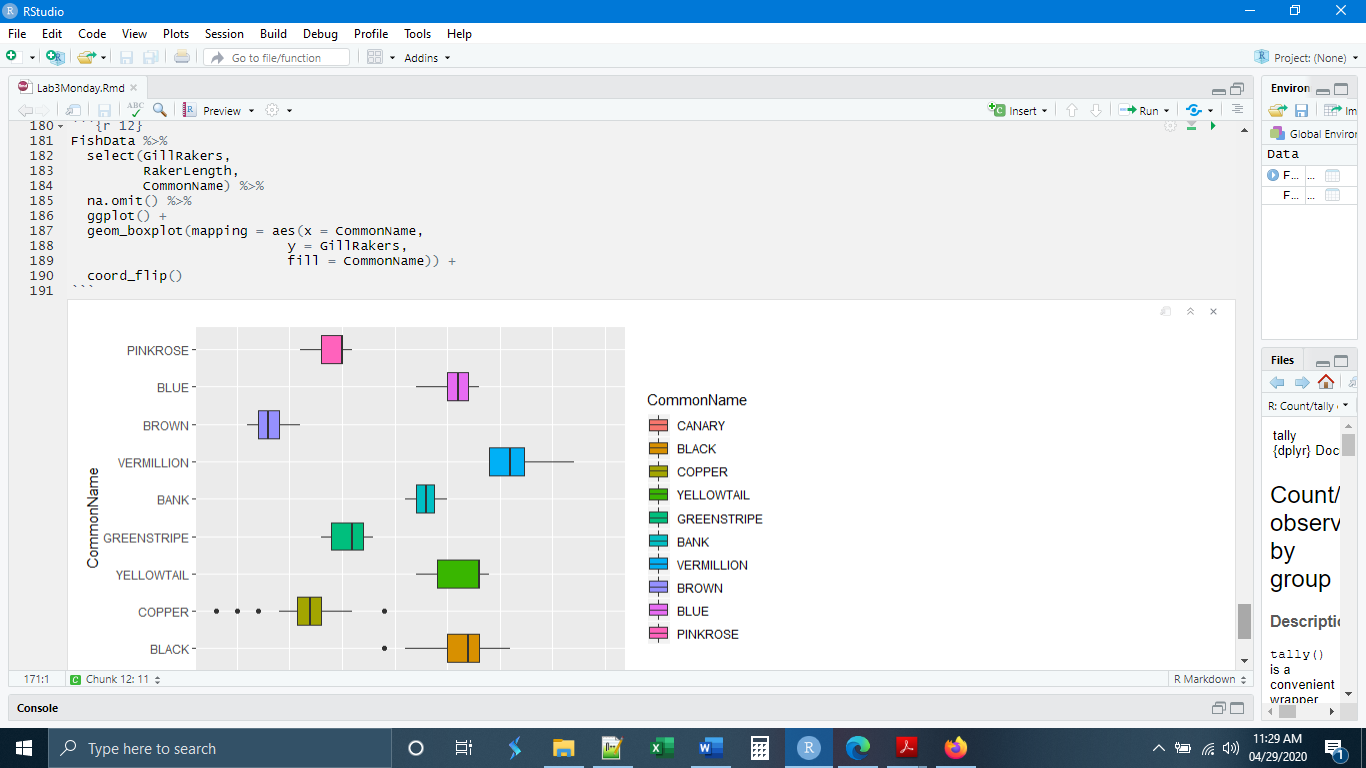
There are a number of ways that we can modify a plot without substantially changing the code we already have in place. One of these is the coord\_flip() command. As the name suggests, this command flips the coordinates (x and y) thus transposing the plot. We can try it on the plot we are already working with, but it does not make a huge difference!



The coord\_flip() command is often most useful when dealing with a two-variable plot where one of the variables is categorical and the other is continuous and numerical. Let’s try making boxplots of the number of gill rakers for the different fish. If we scroll down, we see that all of the names are overlapping at the bottom of the plot.



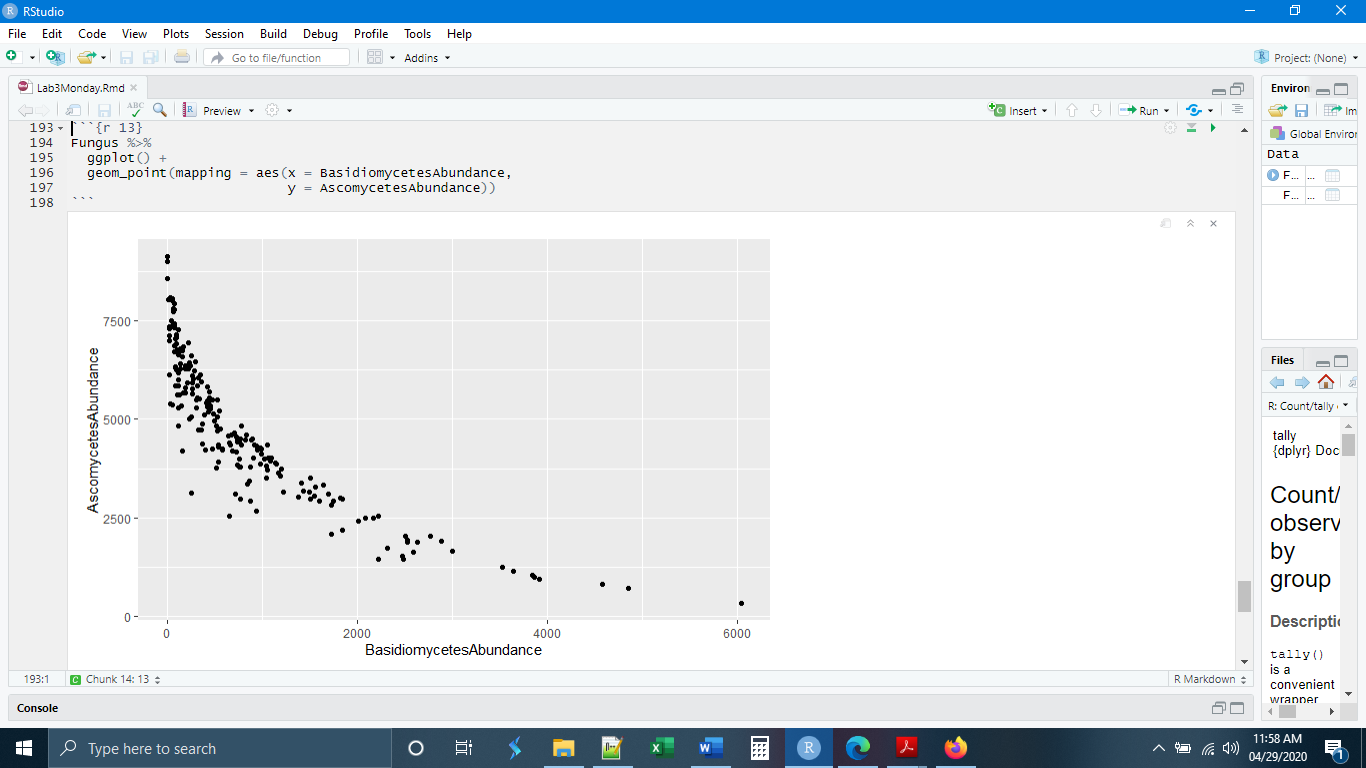
An easy way to fix this is simply to flip the x and y axis of the plot to give more room for the different fish names.



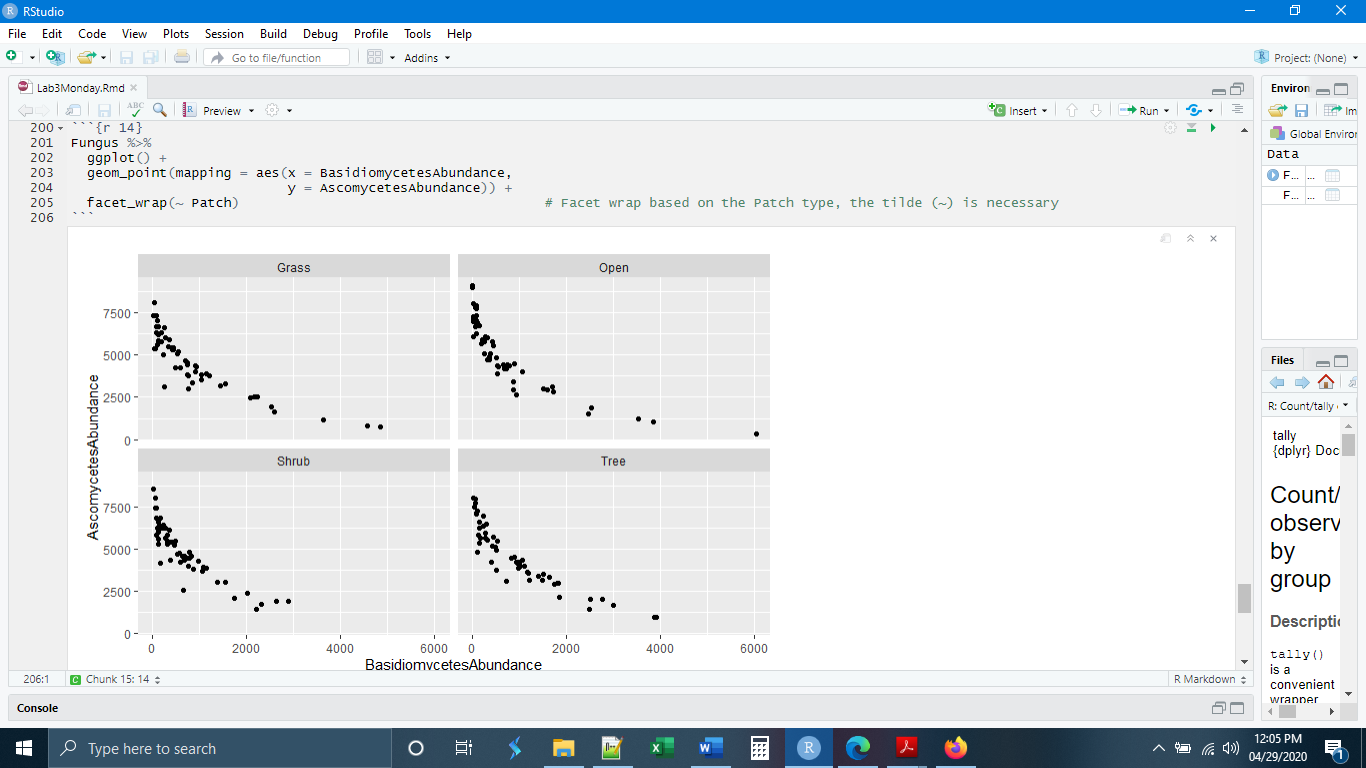
*Facets*

Up to this point we have learned how to visualize up to three different variables, one variable on each axis and a color variable. Sometimes we would like to examine the relationship between more than three variables simultaneously. This can become difficult and messy on a single two-variable plot. ggplot provides a way to get around this with the facet\_wrap() and facet\_grid() commands. Both of these commands generate a series of plots programmatically based on some categorical variable.

Let’s go back to our Fungus data and make a simple scatter plot of the basidiomycete and ascomycete abundances (basidiomycota and ascomycota are the two major groups of fungi).



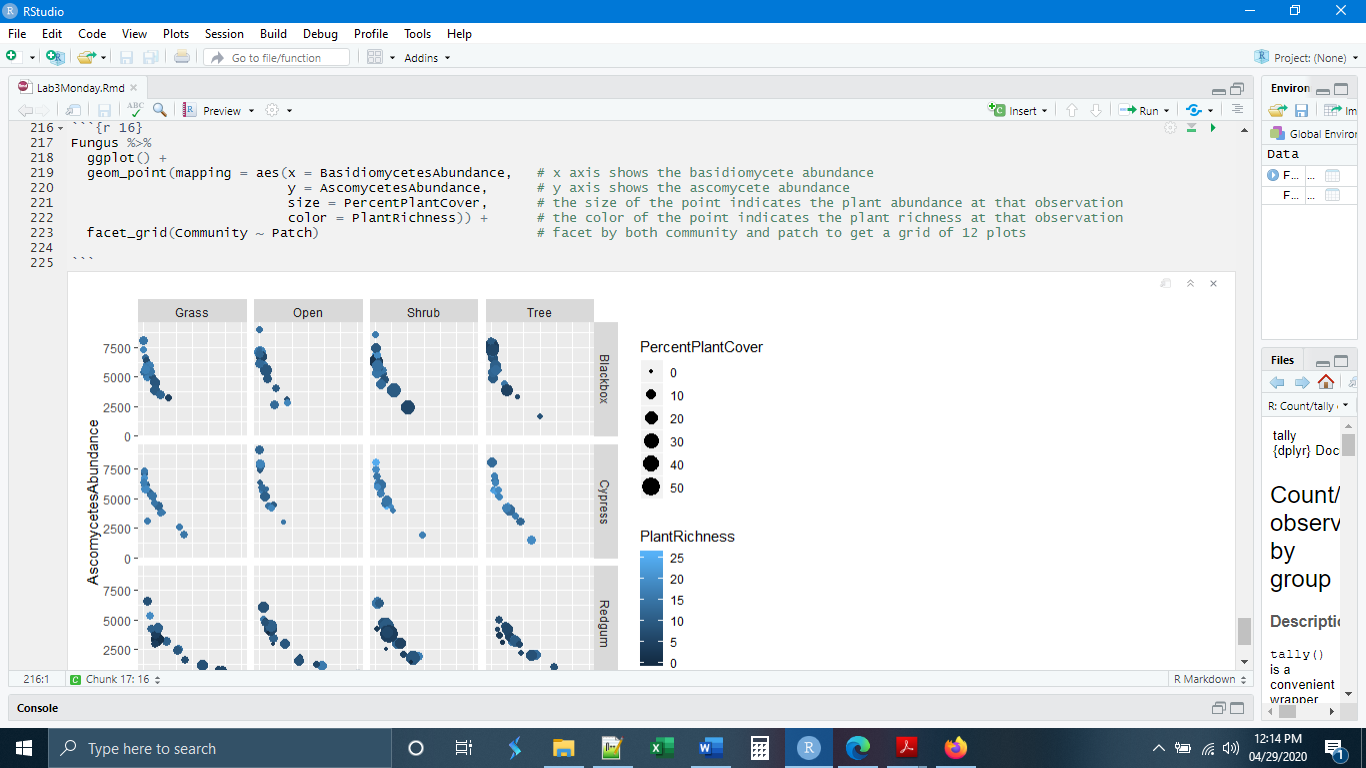
This is an interesting relationship, but what if we also wanted to see if the relationship was the same in each of the patch types? One way to do this would be to color the points based on patch type. Another option is to use a facet. The facet will create a new plot for each of the patch types including only the points for that patch type. The ggplot command is facet\_wrap().



It is also possible to facet by two different categorical variables using facet\_grid().

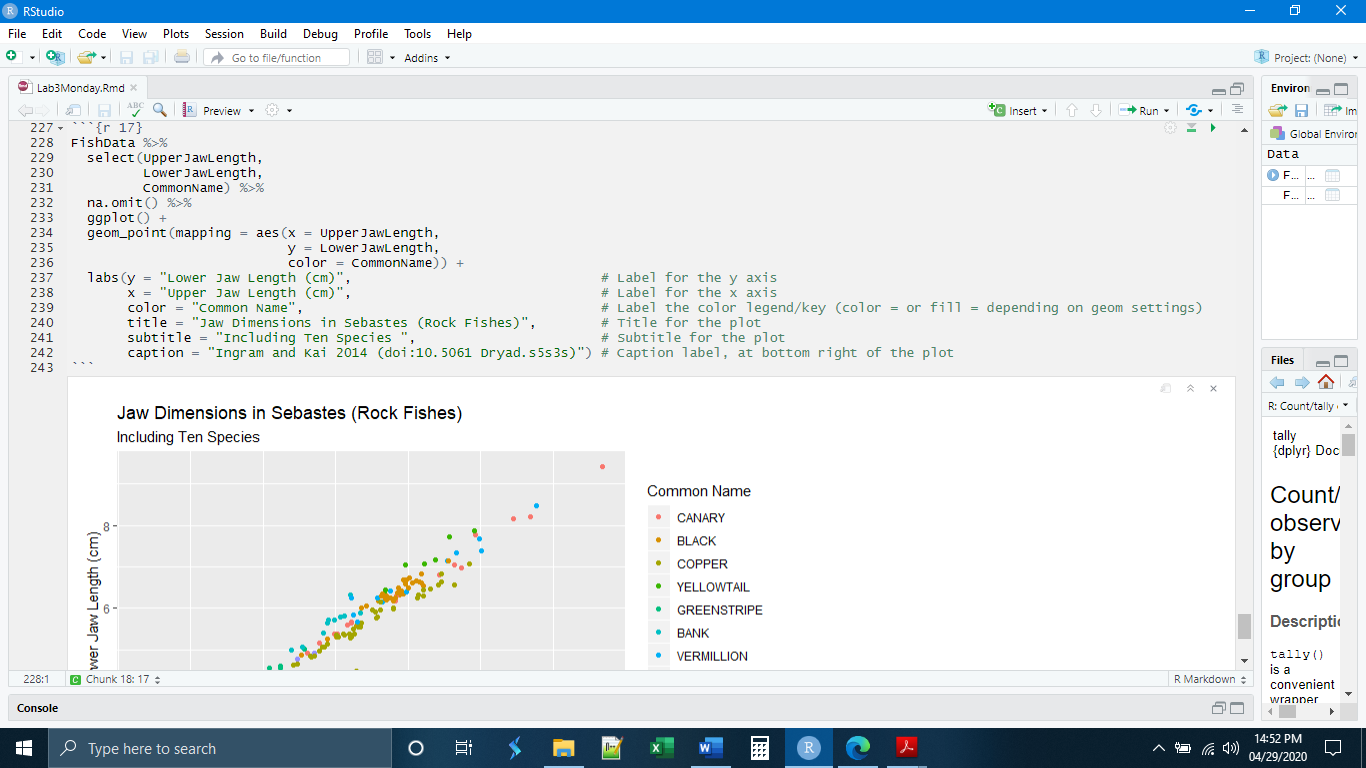


If we really wanted to go crazy, we could modify the points by color and size to display information from a total of six variables in only two dimensions.



*Labels*

The last thing we will do today is learn how to put custom labels on our plots. By default, ggplot will name axes and legends/keys by the variable name as it appears in the data object. There is a relatively easy way to change that using the labs() command. Within labs() you can choose to change existing labels (such as axes) as well as add labels that do not yet exist such as titles. Quotation marks around the text are necessary.



Notice that we changed the label for the color legend/key by using color = in the labs() command. If this had been the kind of plot where we had specified a fill, we would change the label using fill = in the labs() command.

The labs() code is consistent across most plots types and it is easy to copy-paste the labs() information as the last piece of any ggplot code and change things as necessary.

*Independent Exercises*

**1.** Make a new chunk called “chunk 1” and write code to

**-** Read in the Iris data (from Blackboard)

**-** Make a scatter plot of sepal length by sepal width

**-** Color the points by the species variable

(Hint: Always remember you need to load your libraries first)

**2.** Make a new chunk called “chunk 2” and write the code to

- Read in the ToothGrowth data (from Blackboard)

- Mutate the “dose” variable to make it a factor (see the bottom of page 2, as\_factor() function)

- Make a set of boxplots with the x variable as “dose” and the y variable as “len”

- Add a general fill color (you choose the color)

- Facet wrap based on the “supp” variable

- Add labels Title: Guinea Pig Tooth Length Growth

Subtitle: Based on Two Different Supplements

x: Supplement Dose

y: Tooth Length

Caption: R Data Set

- All of the above should be done with a single dplyr pipe into ggplot, no data object should be created

(Hint: Remember that the pipe symbols are different in dplyr and ggplot)

(Hint: Does your color fill = option go inside or outside aes()?)