**Biostatistics**

**Lab 7**

*Tasks*

Confidence Intervals, t-Tests in R, Multiple Tests

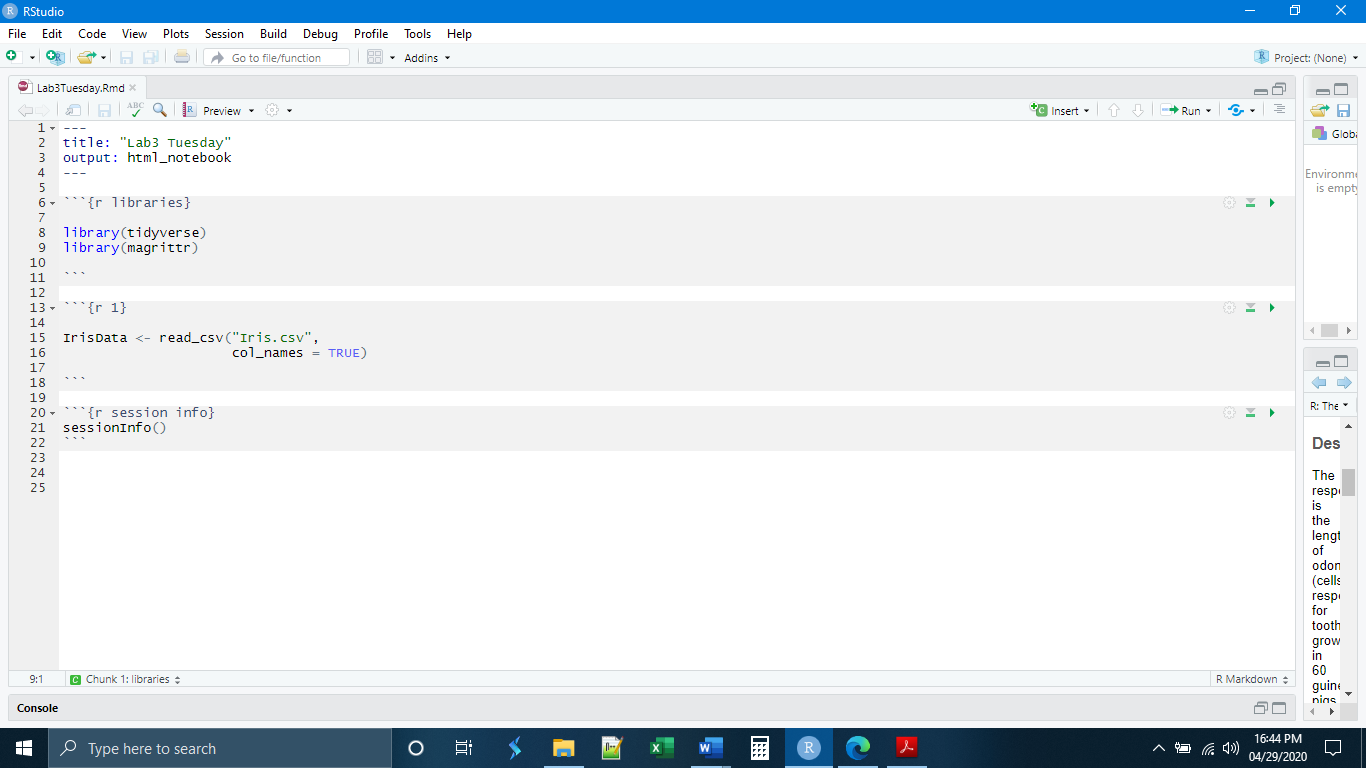
*Introduction*

Today we will be learning how to run a t-Test in R, as well as get confidence intervals and make corrections for multiple tests. Remember that when we do statistical tests it is always in the framework of explicit hypotheses. Our hypotheses are not random choices, however. Generally, we formulate hypotheses based on some bit of information that we already have. We may have seen a pattern in a particular data set and wanted to see if that pattern was statistically significant, or just due to chance. This process of looking for interesting patterns that we can then test is called “exploratory data analysis”. All of our R work up to this point, the ways we have combined the commands and functions in readr, dplyr, and ggplot, all falls into this practice of exploratory data analysis. Today, we will be taking the next step and testing some of the patterns we find.

*Set-Up and Data*

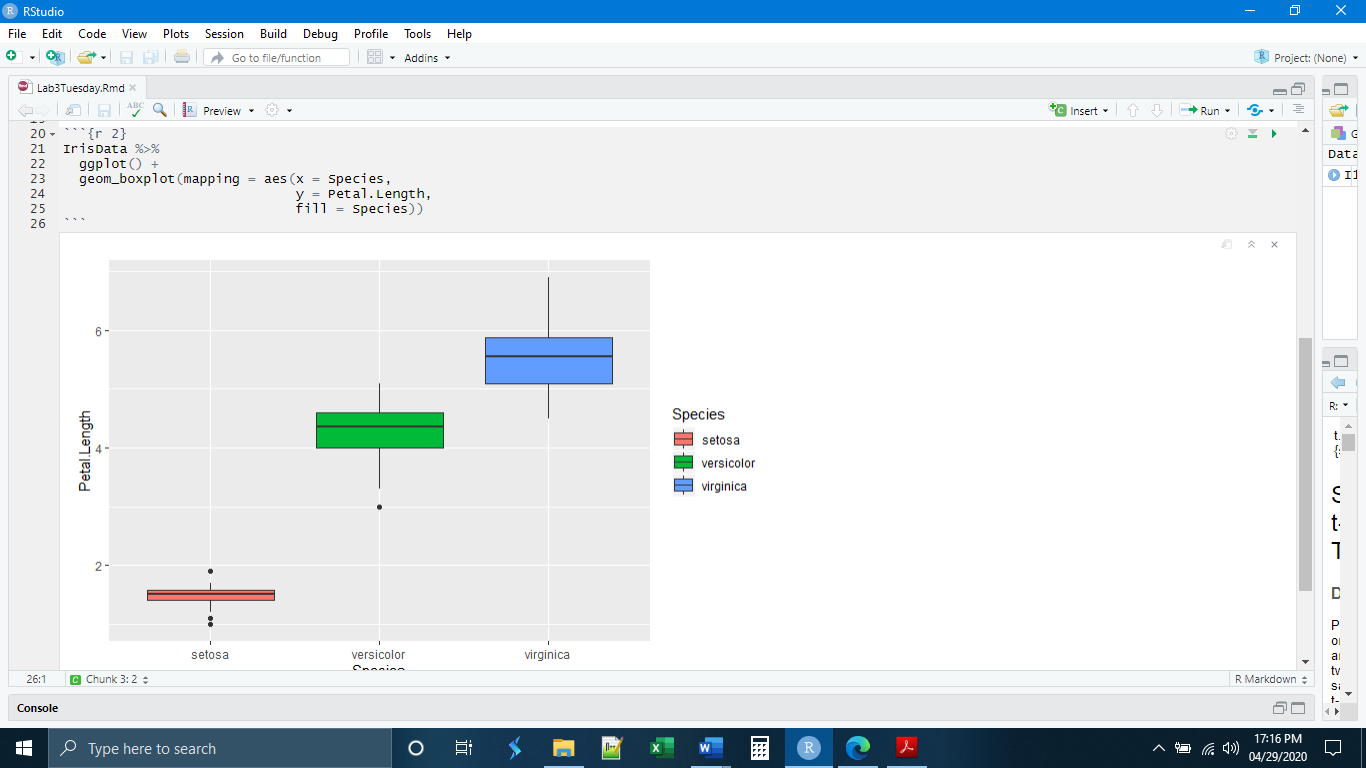
If you have not done so already, download all of the data files necessary for the lab this week and save them in the Lab3 folder you made yesterday. The file that will be most important for today is the Iris.csv data.

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. Make a chunk 1 to read in the data.



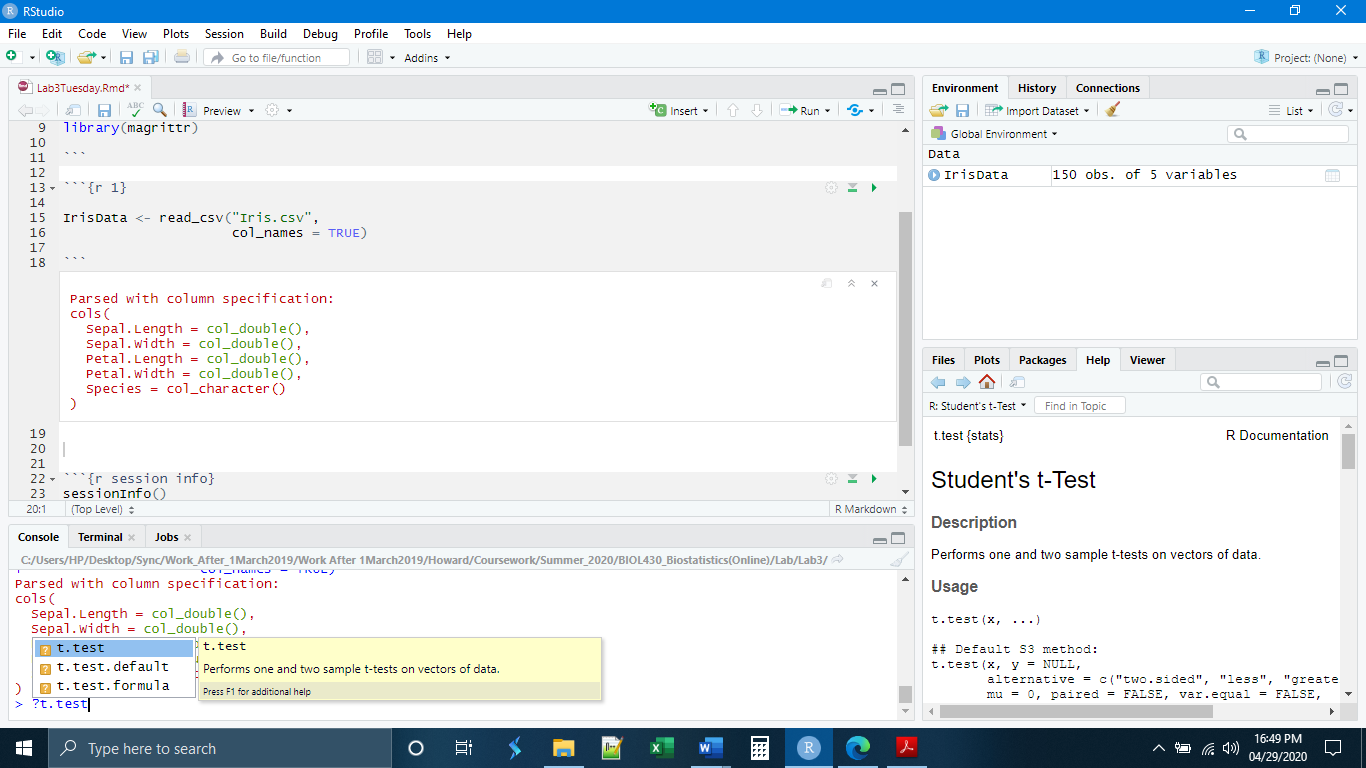
*Exploration and Hypothesis Formation*

The first thing that we need to do is find an interesting pattern that is worth testing. If you have started the exercises for yesterday, you will know that there are some interesting patterns in the Iris data. If we make a boxplot of the petal length by species, we see that the three species of Iris all appear to have different flower petal lengths. This is something we can test with our t-Test.



*t-Test*

Let’s try navigating R help to understand how it works. To do this, go to the console and type a question mark, followed by the command. Information will appear in the help window.



This t.test() function is part of the base R statistical functions and not the tidyverse. Stepping outside of the tidyverse, we will have to abandon some of the habits we have acquired, such as piping. Instead, we will have to subset our data and create new data objects specifically for each test. Our first test is a simple one-sample test that the mean is different than zero. This test is often used in medical scenarios to see if a particular drug changes some measurable aspect of a patient.

As an example, we will use the Iris data and test that the flower petal length is different from zero (hint: it will be). First, we create a new data object with measurements from only one of the species, here *Iris setosa*. Second, when supplying the data to t.test() we have to specify both the data object and the variable within that object using the $. The syntax for the t.test() function is:

t.test( x = DataObject$VariableName[SubsetIfNecessary],

y = DataObject2$VariableName2[Subset2IfNecessary], (= NULL if one mean)

mu = TrueMeanValue,

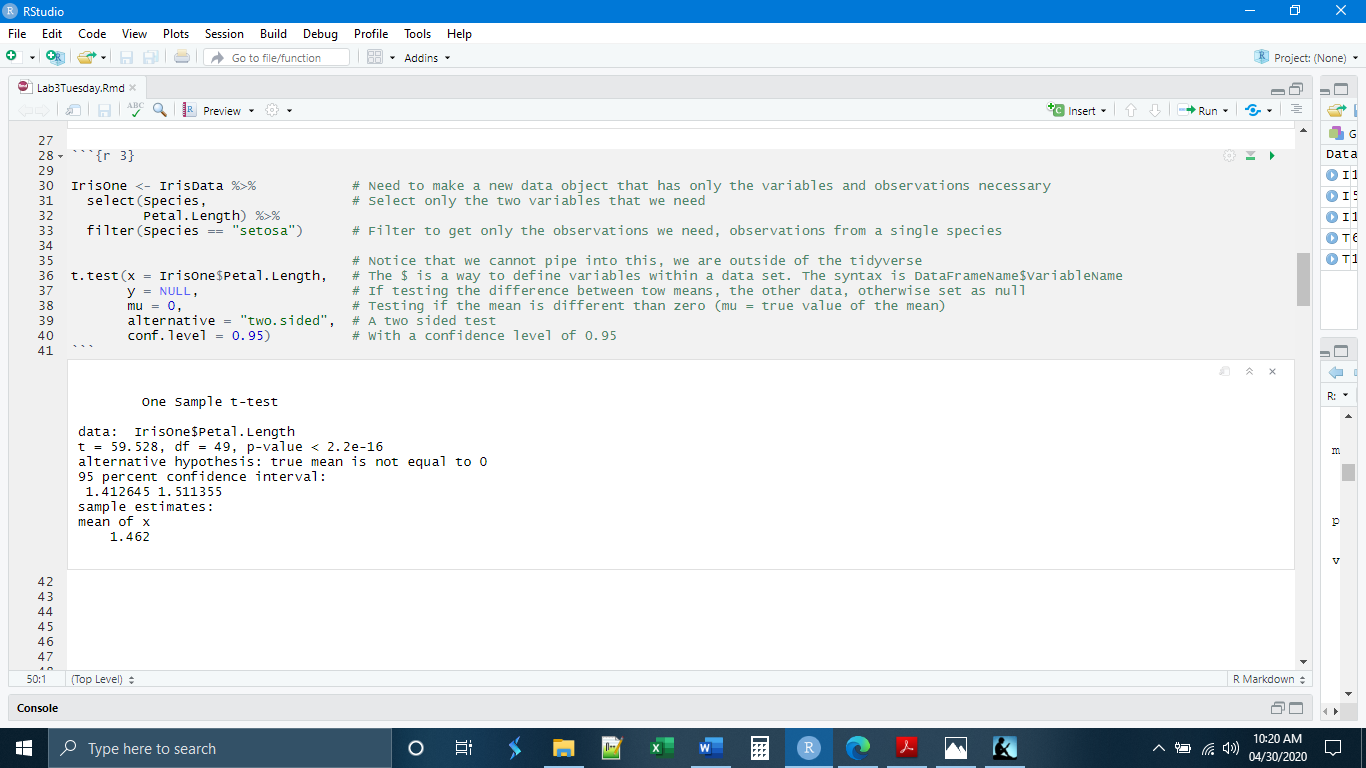
alternative = “KindOfTest”,

conf.level = significance level expressed as 1 – p)

In this case, our hypothesis would not be particularly relevant biologically, but as an exercise we should form it anyway.

H0: The petal length of *Iris setosa* is equal to zero.

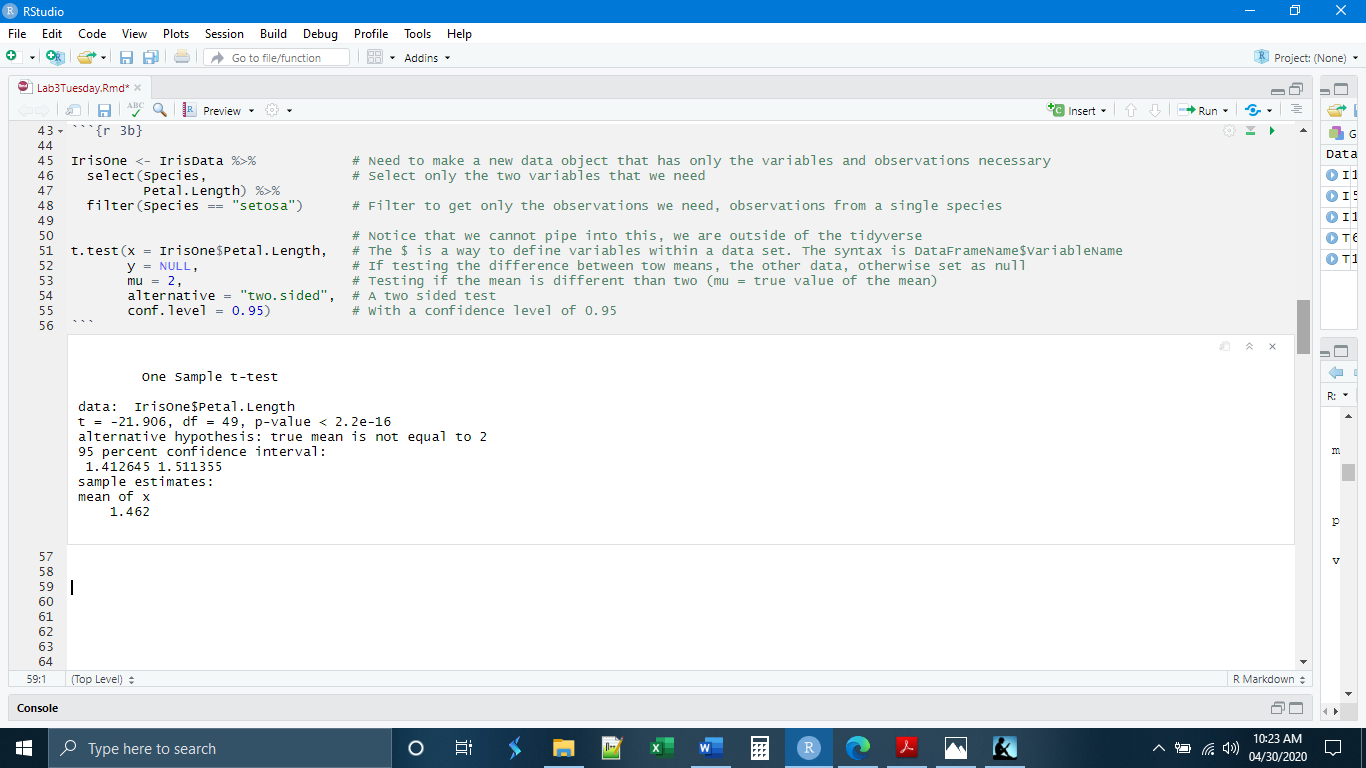
HA: The petal length of *Iris setosa* is not equal to zero.



We could also imagine a scenario that is more biologically relevant. Let us say that previous reports had indicated that the petal length of *Iris setosa* was 2. We have collected a number of plants that look like *Iris setosa* but have shorter petals. Maybe it is a new species or a new variety. We want to test to make sure. In that case, we could do this same test but set mu = 2, to see if the petal lengths of our plants are different than that. In that case the hypotheses would be:

H0: The petal length of *Iris setosa* is equal to two.

HA: The petal length of *Iris setosa* is not equal to two.



Another way to use the t-Test is to test the difference between two means. Technically, this is testing that the difference between the two means is not equal to zero. In the case of the Iris data, we can test whether the petal length from one species is different from the petal length of another species, or, if we thought we had collected two different species of plants but weren’t sure, we could test to see if they had different petal lengths. Here we will test if the petal length of *Iris setosa* is different from that of *Iris virginica*.

There are two ways that we can do this in R. One way is to make two different data objects, each with the petal length measurements for one of the species. These can then be called separately for x and y. The other option, which we will use here, is to create one data object with the data for both x and y, and then subset as necessary. The subsetting would look like:

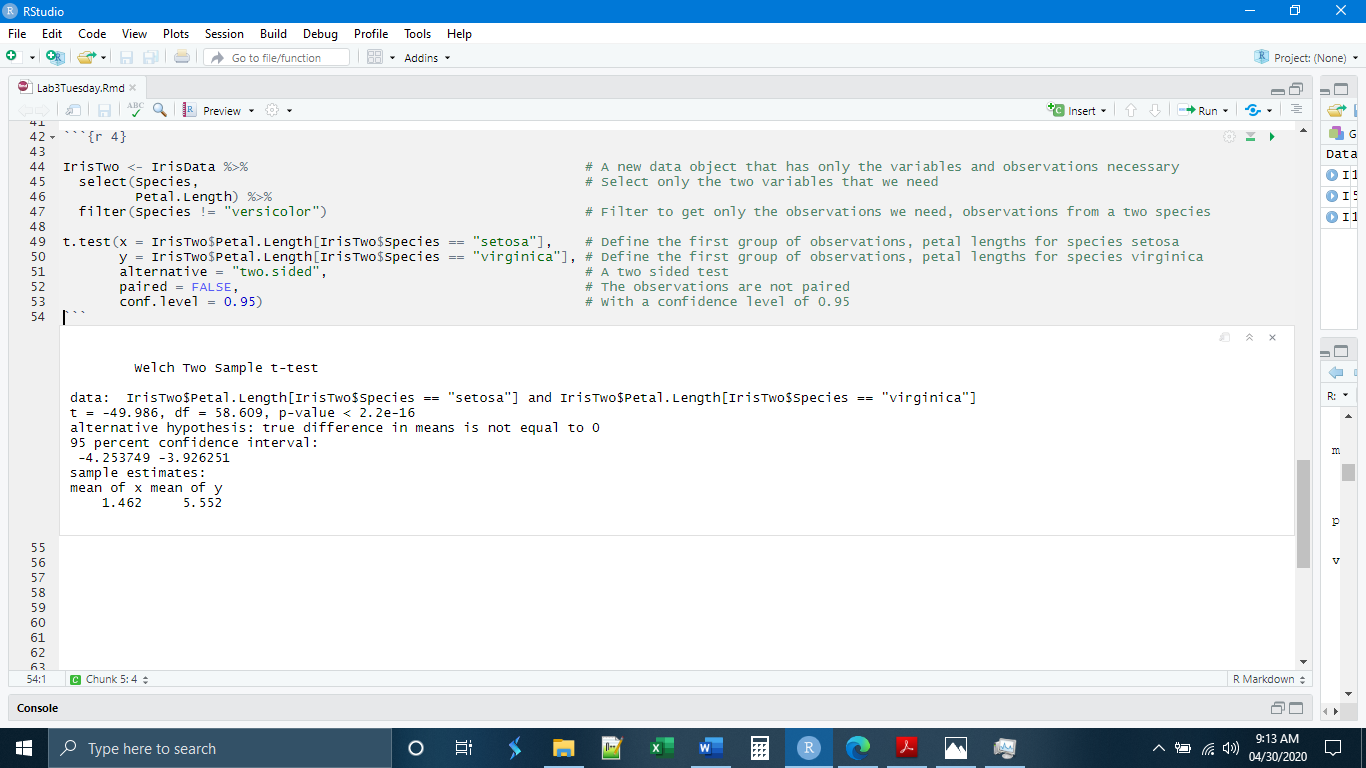
IrisTwo$Petal.Length[IrisTwo$Species == "setosa"]

This is code read in the following way: “the observations in the Petal.Length variable in data object IrisTwo [but only the observations where the Species variable in IrisTwo is equal to setosa”

In this case our hypotheses would be:

H0: The petal length of *Iris setosa* is equal to the petal length of *Iris virginica*.

HA: The petal length of *Iris setosa* is not equal the petal length of *Iris virginica*.



Notice that the output of the test is not simply the p-value. R gives us the t-statistic, the degrees of freedom, the p-value, the 95% confidence interval, and even tells us what the alternative hypothesis is for the test.

*Multiple Tests*

There are actually three groups in the Iris data, three species: *Iris setosa*, *Iris versicolor*, and *Iris virginica*. We may be interested in testing to see if all the petal lengths are different from each other. This practice is called multiple comparisons or multiple testing, and it comes with a problem. We set our p-value (the probability of obtaining the observed result, assuming that the null hypothesis is true) at 0.05, or 5 %. Thus, our probability of falsely rejecting the null hypothesis (type I error) is 5 %. Stated another way, our probability of mistakenly rejecting the null hypothesis when it is true, due to randomness associated with sampling is 5 % (p = 0.05).

But this 0.05 p-value is true for each of the tests individually. If we conduct two tests on the same data, we accept that probability for each test: 5 % + 5 % = 10 % (mathematically it is a little more complicated than that, but conceptually that is what we are dealing with). If we conduct 20 comparisons, we are virtually certain to commit a type I error in one of the tests.

This problem has been recognized and dealt with by a number of statisticians, who have determined ways to correct for this problem by changing the mathematical formulas. None of the corrections is perfect. The one we will use today is called the Bonferroni correction. Conceptually, the Bonferroni correction reduces the p-value for each of the individual tests so that the total p-value adds up to 0.05 (or whatever you set it to). For example, if I were doing five comparisons, the p-value for each of the comparisons would be reduced to 0.01 so that collectively the five comparisons had a p-value of 0.05.

The function in R that will perform multiple comparisons is pairwise.t.test(). It has the following syntax:

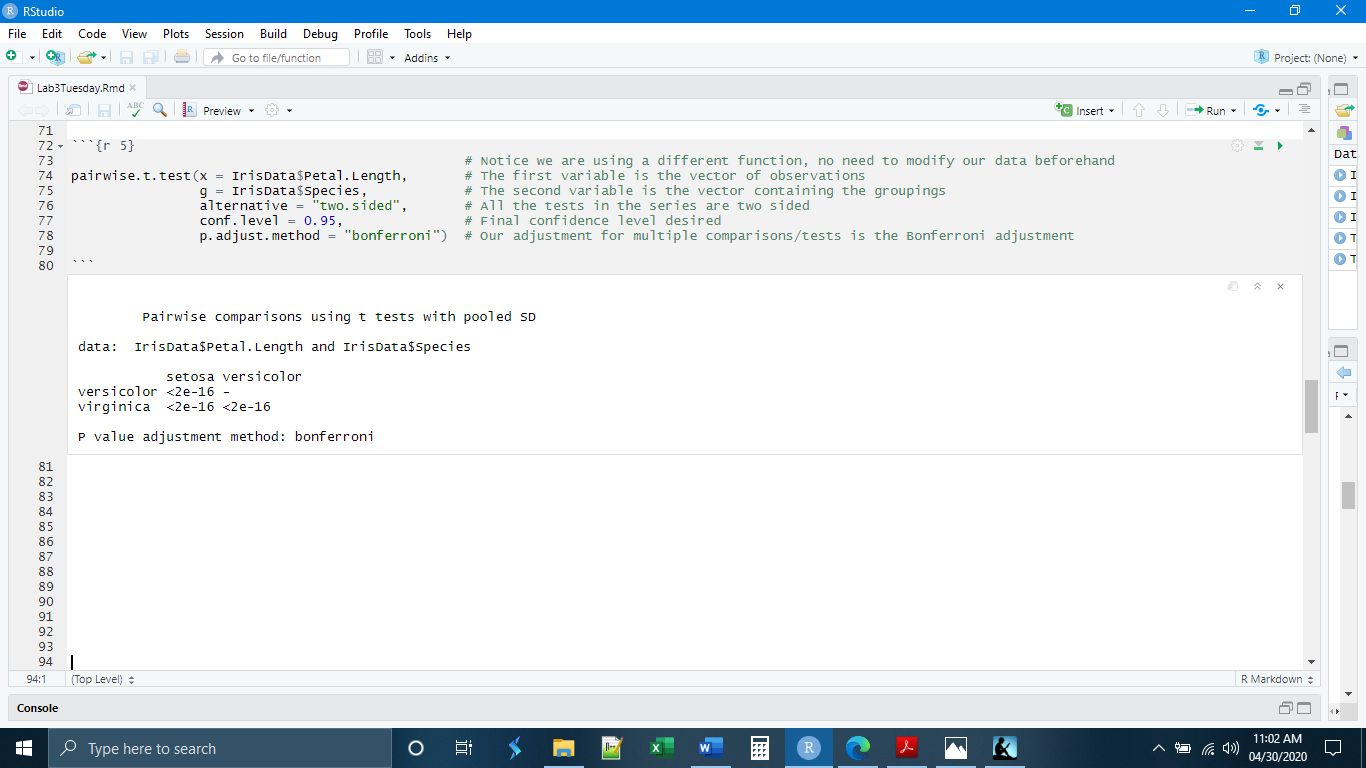
pairwise.t.test( x = DataObject$ObservationsVariable[SubsetIfNecessary],

g = DataObject$GroupingVariable[SubsetIfNecessary],

alternative = “KindOfTest”,

conf.level = final significance level expressed as 1 – p,

p.adjust.method = “AdjustmentMethod”)

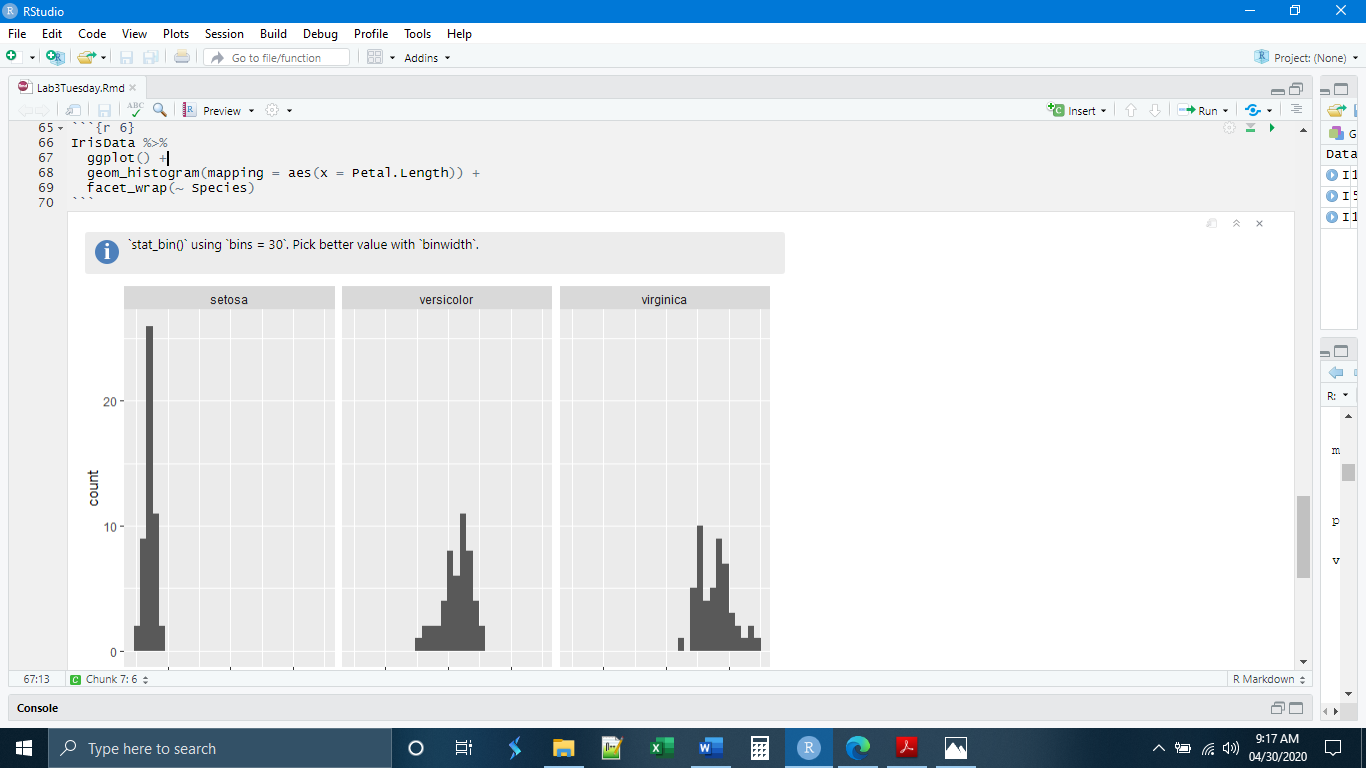


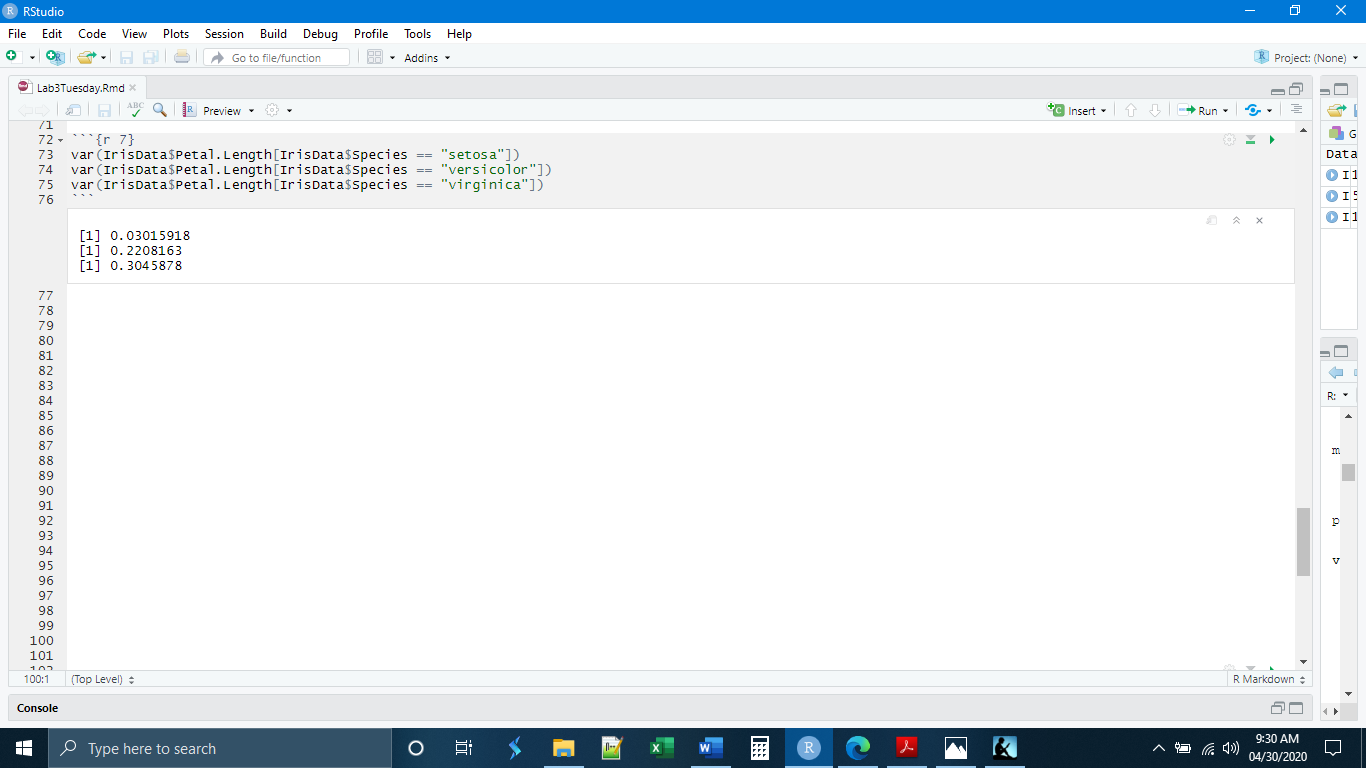
Note that here R does not give us the full output, only giving the p-values for each of the comparisons.

*Checking Assumptions*

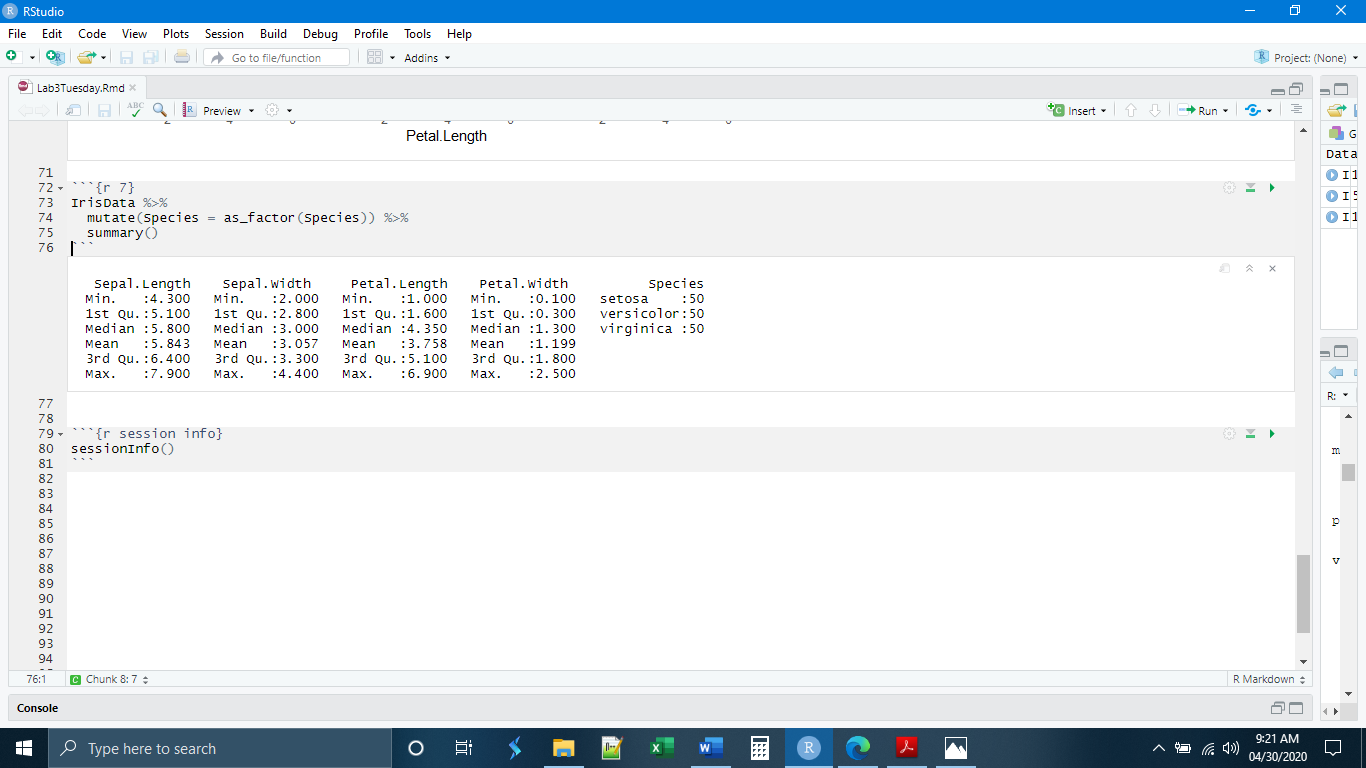
The mathematics of testing means requires that certain assumptions be in place. If the data do not meet these conditions, then the mathematical formulas will not accurately distinguish the groups. Some of these assumptions are assessed through the design of the experiment, for example the independence of the observations. Others of these conditions must be assessed from the data themselves after they have been collected. We are going to briefly look at these here.

The first is the assumption that the data are normally distributed. In this case we need to check if the petal lengths for each of the species are distributed normally. And this looks to be generally the case, though with some skew in versicolor and virginica.



Next, we will check if the variances are equal, and they are not. 

Finally, let’s look at the sample size. For sample sizes above 30, the t distribution and the normal distribution are very close. Thus, sample sizes above 30 are preferred. You can use the t distribution with sample sizes below 30, but the other assumptions must be more stringently observed, in other words, the distributions of points need to be normal and not skewed, and the variances need to be nearly equal.



Because our sample size is large, we can get away with inequal variances, but we have to admit that this is not the best test. There is a way to test if the variances are equal, called the f-Test. We will learn about this next week.

*Exercises*

**1.** Make a new chunk called “chunk 3” and write code to

- Read in the ToothGrowth data (from Blackboard)

- Create a new data object that contains only observations from supp = OJ and dose = 0.5

(Hint: try a mutate with two filters)

- Use this new data object to perform a one-sample t-Test that len is different than zero

**2.** Make a new chunk called “chunk 4” and write code to assess the assumptions necessary for the t-Test performed in chunk 3. You will need to:

- Make a histogram for the observations to check for normality

- Determine the sample size

- Answer the question, are the assumptions for a t-Test adequately satisfied? Record your answer as a comment in the chunk.