Introduction to R

Make sure you have downloaded the "underc2019-Rworkshop" folder from Brittni and put this folder in a reliable place on your computer (Documents, Dropbox, etc). Open the R project "underc2019.Rproj".

Open a new R script (File -> New File -> R script).

Before we get going, we want to set up Rstudio so that all our coding goes smoothly. This is good practices that you should do every time you open R. First, we want to clear our Global Environment so that we're starting from a fresh template. Second, we want to check what working directory we're in, and if it's not correct, we'll set the working directory. Here is an example of what I would use on my computer, but the setwd() command will be different for everyone depending on where in your computer you've saved this directory.

```r
rm(list = ls())  #clears global environment
getwd()  #gets the present working directory
setwd("/Users/brittnibertolet/Documents/underc2019-Rworkshop/")  #sets a new working directory
```

We also want to look around at our file system and see where our data is located.

```r
list.files()  #lists the files or directories in the present working directory
```

```
## [1] "data"              "guides"            "markdowns"
## [4] "underc2019.Rproj"
```

```r
list.files("data")  #lists the files in the directory 'data'
```

```
## [1] "irisData.csv" "lakeData.csv"
```

We have a couple files in our data directory. Today, we're going to work with the lakeData.csv data. This is water chemistry data from 32 lakes in Wisconsin, Michigan, and Indiana. Read this data into Rstudio and look at the first few lines of data to get a feel for what is here.

```r
lakesDF = read.csv("data/lakeData.csv", stringsAsFactors = F)  #read in data
head(lakesDF)  #print the first six rows
```

```
##   lakeID location   long   lat  pH  DOC   TP   TN methanogen_perc
## 1     BA   UNDERC -89.49 46.24 7.0  6.5 22.8  840      0.03061616
## 2     BC Michiana -86.19 42.07 7.8  5.0 16.5  333      0.00337450
## 3     BE   UNDERC -89.51 46.24 5.2  9.4 21.5  531      0.02628239
## 4     BO   UNDERC -89.49 46.23 5.4 19.5 48.7 1389      0.05305554
## 5     BR   UNDERC -89.47 46.21 8.3  6.6 86.9 1187      0.02242244
## 6     CB   UNDERC -89.57 46.23 4.1 18.2 33.0  990      0.03677840
```

It is also often useful to look at the structure of your data.

```r
str(lakesDF)
```

```
## 'data.frame':    32 obs. of  9 variables:
##  $ lakeID         : chr  "BA" "BC" "BE" "BO" ...
##  $ location       : chr  "UNDERC" "Michiana" "UNDERC" "UNDERC" ...
##  $ long           : num  -89.5 -86.2 -89.5 -89.5 -89.5 ...
##  $ lat            : num  46.2 42.1 46.2 46.2 46.2 ...
##  $ pH             : num  7 7.8 5.2 5.4 8.3 4.1 7.9 7.9 5.9 7.8 ...
##  $ DOC            : num  6.5 5 9.4 19.5 6.6 18.2 5.5 4.6 4.5 4.9 ...
##  $ TP             : num  22.8 16.5 21.5 48.7 86.9 33 18.8 18.2 11.1 16.7 ...
##  $ TN             : int  840 333 531 1389 1187 990 508 508 1111 878 ...
##  $ methanogen_perc: num  0.03062 0.00337 0.02628 0.05306 0.02242 ...
```

This tells us a little bit more about our data. We have 32 rows and 9 different columns. Each row is a different observation of these 9 different characteristics. It also tells us the data type of each column (chr, num, int, etc). Dataframes are 2-dimensional data structures that can hold many different data types.

We can access different columns of the data using the notation "dataframeName$columnName". For instance, if we wanted to see all the lakeIDs for which we have data for. We could do:

```
lakesDF$lakeID
```

```
##  [1] "BA" "BC" "BE" "BO" "BR" "CB" "CH" "CO" "CR" "DI" "DO" "FO" "GA" "GV"
## [15] "HB" "HU" "JU" "LW" "MA" "MO" "NC" "NG" "PA" "PE" "PL" "RO" "SN" "ST"
## [29] "TR" "TU" "UF" "WL"
```

We can also access different columns or rows using square brackets and logic statements. This is often referred to as indexing. Square brackets are read as "[row, column]", in which any set of coordinates will reference a certain data cell in the dataframe. For example, if we want to know all the lakeIDs that have a DOC concentration greater than 10, we would do:

```
lakesDF[lakesDF$DOC > 10, 1]
```

```
##  [1] "BO" "CB" "FO" "HB" "LW" "MO" "NC" "NG" "PL" "TR" "TU" "UF"
```

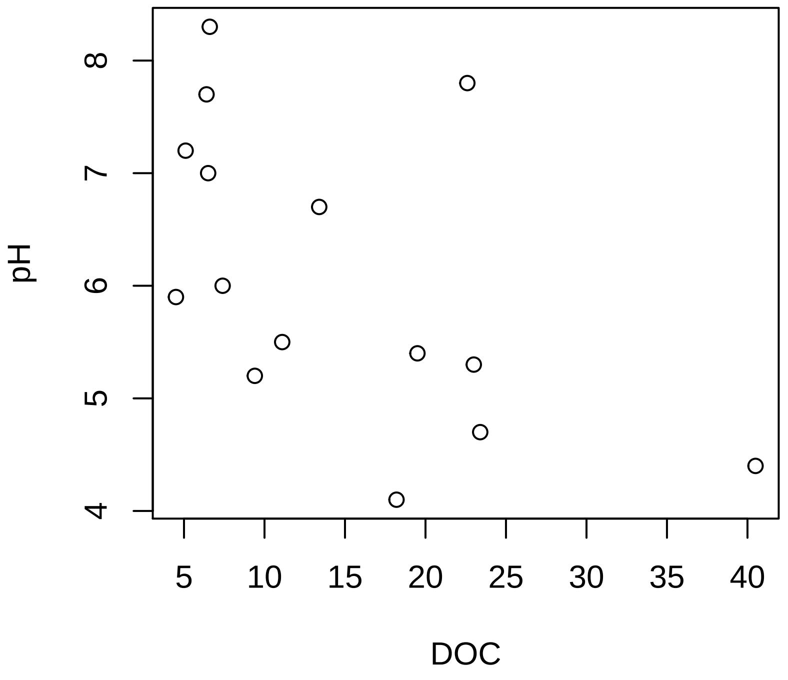
This can also be written using a combination of the "dataframeName$columnName" notation and the square brackets.

```
lakesDF$lakeID[lakesDF$DOC > 10]
```

```
##  [1] "BO" "CB" "FO" "HB" "LW" "MO" "NC" "NG" "PL" "TR" "TU" "UF"
```

This indexing can be embedded in functions and scripts in very powerful ways. For example, if I wanted to plot lake DOC versus pH, but only wanted lakes at UNDERC, I could use the following commands:

```
plot(x = lakesDF[lakesDF$location == "UNDERC", 6], y = lakesDF[lakesDF$location ==
    "UNDERC", 5], xlab = "DOC", ylab = "pH")
```



Challenges!

Now try some problems on your own. Feel free to mess around without worry. When you read in data into Rstudio, you are not permanently changing any data on your computer.

1. Seperate the data into two seperate dataframes: one with all the lakes that have TP concentrations less than 20, and one with lakes that have TP concentrations greater than 20.
2. Pick your favorite nutrient and plot it against DOC for just Michiana lakes. Determine whether they are correlated. Use the lm() function!
3. Determine the mean pH for both the UNDERC lakes and the Michiana lakes. Plot them using the boxplot() function. Determine whether there is a difference between UNDERC lakes and Michiana lakes using the t.test() function.