MTH 324, Lab #1, Fall 2022 Name _____

Instructions: Follow along with the tutorial portion of the lab. Replicate the code examples in R on your own, along with the demonstration. Then use those examples as a model to answer the questions/perform the tasks that follow. Copy and paste the results of your code to answer questions where directed. Submit your response file and the code used (both for the tutorial and part two). Your code file and your lab response file should each include your name inside.

Tutorial: Introduction to R

To get started, we first need to make sure everyone is set up to work with R. You can use school computers, if you wish, which should already have R installed (check with the technology people to see which computer labs it is installed in), but it may be more convenient to have it installed on your own computer. So the first thing will be to install R if it isn't already.

1. Install R. Go to the website https://www.r-project.org/ from the computer you want to install R on. Then click on the **download** link. Install the latest version of R.

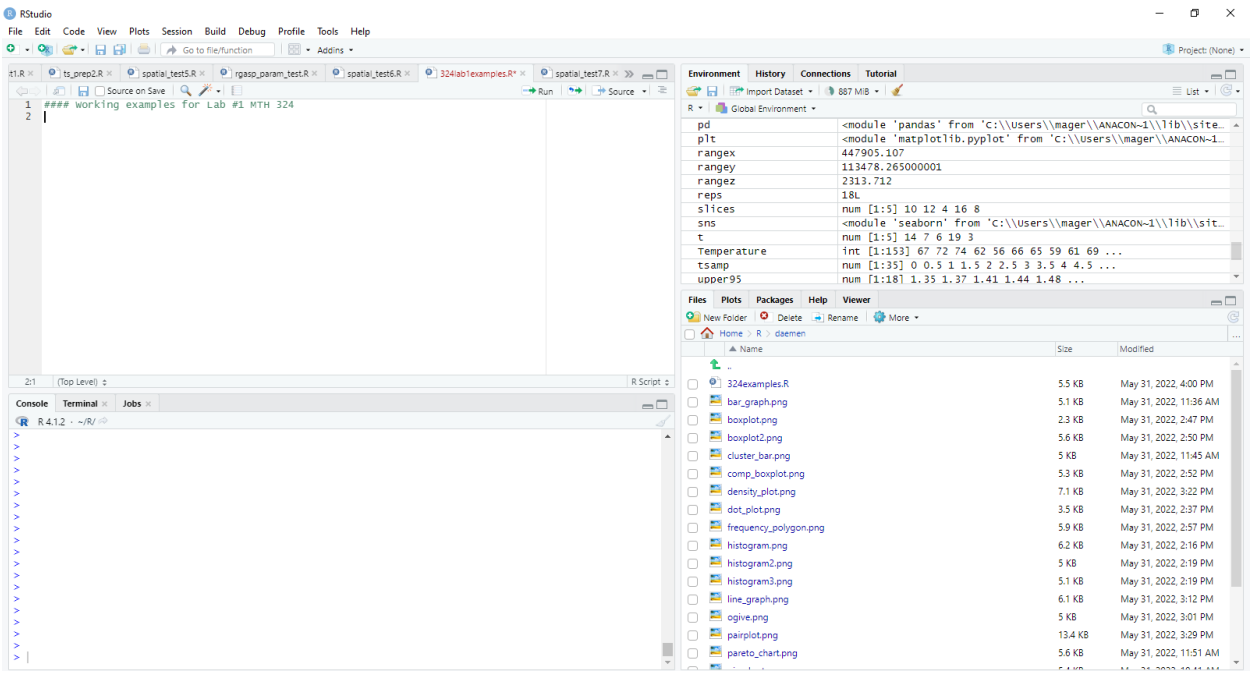
While it is possible to work directly from the R console, it is a little better to work from R Studio. You can also use Jupyter notebook to use R with some adjustments to Anaconda environment. If you have experience working with Python in Jupyter notebook, this may be the best way. If you are more accustomed to working with an IDE program like PyCharm or Spider, RStudio may be the way to go.

2. Install your choice of working environment if it's not installed already.
    a. Download RStudio Desktop (open source) https://www.rstudio.com/products/rstudio/#rstudio-desktop
    b. Alternatively, to use Jupyter notebook:
        i. Download and install Anaconda if it's not already installed. https://www.anaconda.com/products/distribution
        ii. Add R to the Anaconda environment. https://docs.anaconda.com/anaconda/navigator/tutorials/r-lang/

In making your choice of platform, there will be some differences in certain specific functions, such as installing packages (it's much easier in R Studio than in Jupyter notebook), and I will generally use RStudio as a platform for demonstrations. If you use Jupyter notebook, expect that you may have to use Google to resolve issues that differ from the usage in R Studio. While both platforms will work for this class, there are advantages to everyone using a common platform.

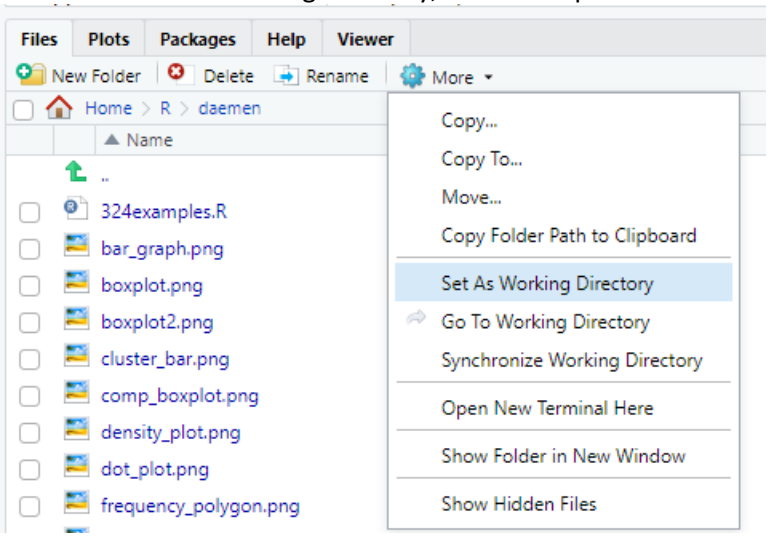
When you open RStudio, your screen will typically look something like this the image below.

At the top left is script file you are working on. At the bottom left is the command line. You can run commands from the script file or directly in the console. At the top right is the environment tab that displays variables and data in memory. At the bottom right you can see files in your working directory (or other folders). This is also where images will display. And the help tab is also in this area. I've got a lot of stuff in my program. Yours will look emptier of course, but take a minute or two to poke around and get familiar with where common function are.



I suggest creating scripts, not so that you can run the whole thing at once, but so that you can save your commands and comment on them so that they will be available to you later. Building a library of commands you can return to an modify saves you the trouble of googling for a particular line of code every time you need them. You will need to submit the commands you used for assignments, so the scripts can help you preserve the commands that work and discard the errors that got you there.

Set a working directory
Where do you want to save your files?
In RStudio, you can set the working directory from the Files tab. Click on More and from the folder you want to set as the working directory, click that option.



You can also do it with the setwd() function. You will need to full address of the directory and enclosed in quotes. https://r-lang.com/setwd-function-in-r-with-example/

Start a new R script file. In comments, add your name, date and assignment number to the top of the file.

We can work with data in R in a number of ways. We can assign a single value to a variable and do basic math with it. We can assign a set of values, either numeric or string, to a vector and perform operations on that vector. Or we can combine vectors into an array or dataframe.

Note: in RStudio, output will display automatically if you run the command in the console. In some programs you may need to surround the command by print() to see the output.

Create a variable for your age called age.

```
2  age <- 50
3
4  #or
5  age = 50
```

You can assign values using either <- or =. In R it's more common to use <-

To create a vector use c()
Create a vector of 4 names, and another vector of 4 ages.

```
6
7  names <- c("Betsy", "Cash", "Ann", "David")
8  ages <- c(50, 49, 74, 77)
9
```

String data needs to go into quotes (single or double). Numbers do not.

We can combine vectors into a dataframe, which functions a lot like a table in a spreadsheet.

```
10
11 people <- data.frame(Name=names, Age=ages)
12
```

You can also create an array.

```
12
13 heights <- c(60.5, 70, 59, 71.5)
14 numbers <- array(c(ages,heights),dim = c(4,2))
15
```

Creating arrays are a little trickier. You can create them from a single vector and use this command to reshape it into an array. If you have too little data to fill the array, it will copy values to fill the shape specified. See what happens if you change the dim specification to c(4,4).

R is mostly designed to work on dataframes, and vectors, but we'll see some applications where other formats are needed.

A special type of dataframe some packages use is called a tibble. We'll discuss it in more detail when we encounter it later on.

We can also create numerical vectors of evenly spaced valued using the sequence function.

```
16  vector1 <- seq(4, 10, by = 0.4)
17
```

This produces a vector that starts at 4, ends at 10 and includes all numerical values in between incremented by 0.4, so 4, 4.4, 4.8, 5.2, etc.

R can combine vectors in other ways. We can bind them as columns, or as rows.

```
17
18  array1 <- cbind(names, ages, heights)
19  array2 <- rbind(names, ages, heights)
20
```

The cbind function binds vectors together as columns. The rbind function binds vectors together as rows. Both can be useful functions for combining smaller calculations into a single object, particularly if we want to save the data for later use.

We can do basic arithmetic in R if we need to.

R Arithmetic Operators	Operation	Example
+	Addition	15 + 5 = 20
–	Subtraction	15 – 5 = 10
*	Multiplication	15 * 5 = 75
/	Division	15 / 5 = 3
%/%	Integer Division – Same as Division. but it returns the integer value by flooring the extra decimals	16 %/% 3 = 5. If you divide 16 with 3 you get 5.333, but the Integer division operator trims the decimal values and outputs the integer
^	Exponent – It returns the Power of One variable against the other	15 ^ 3 = 3375 (It means 15 Power 3 or 10^3).
%%	Modulus – It returns the remainder after the division	15 %% 5 = 0 (Here remainder is zero). If it is 17 %% 4, then result = 1.

Perform the simple arithmetic operations using these operators.

$$3 + 4$$
$$10 - 17$$
$$age * 2$$
$$age/6$$

Do this division with each of the division operators

$$ages\text{^}2$$

This is the vector of age values. What is the result of applying a math operator to a vector?

What happens if you use two vectors like $ages * heights$?

```
21  3+4
22  10-17
23  age*2
24  age/6
25  age%/%6
26  age%%6
27  ages^2
28  ages*heights
29
```

Some math functions are designed to be done on vectors, such as max(), min(), and sum().

```
27  min(ages)
28  max(ages)
29  sum(ages)
30
```

It's worth noting here that there are different naming conventions out there for variable names. Aside from not using the names of special variables or functions, I encourage you to be consistent with whatever convention you choose, but in general, your variables should have somewhat meaningful names to make your code easier to read. In general, don't use single letter names. These are sometimes used to specify syntax, to connect them back with the math formulas they are derived from, but it is generally considered back programming practice to do this for working programs.

R comes with many built-in data sets that we can also experiment with and which are often used in tutorials you find online.

We can call the built-in data with the data() function, and we can open it up to view the dataframe like a spreadsheet with the View() function.

```
34  data("mtcars")
35  View(mtcars)
```

ts_prep2.R | spatial_test5.R | rgasp_param_test.R | spatial_test6.R | 324lab1examples.R | mtcars

Filter

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3

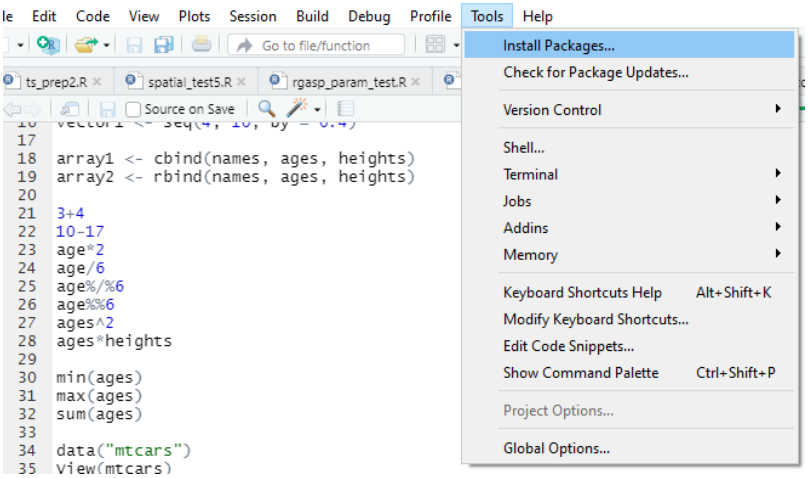
Showing 1 to 14 of 32 entries, 11 total columns

To refer to just one column of the dataframe, like mpg, use the name of the dataframe, a dollar sign, and then the name of the column.

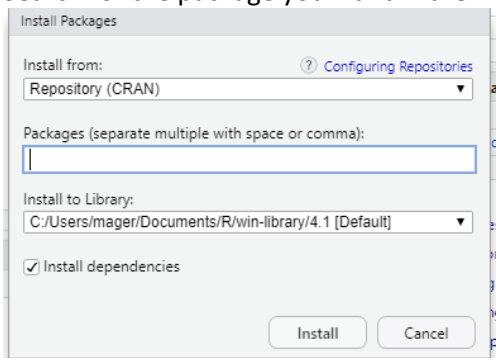
```
36
37  mpg <- mtcars$mpg
38
```

We can perform operations on it just like we did with vectors we created by hand.

Many packages in R exist to perform specialized tasks and many include additional datasets. One important package we will use in our next lab is ggplot2. To install a new package from RStudio Importing packages, go to the Tool tab at the top menu bar and select Install Packages.



Search for the package you want in the menu that pops up.



Most of the packages you want will be in CRAN, so you should not have to change any other settings most of the time. Once you find the package you want, select Install.

A full list of available packages is here:
https://cran.r-project.org/web/packages/available_packages_by_name.html

This website includes links to documentation for each package, how-tos, examples, creator and maintainer information, etc.

Cheat Sheets for commonly used packages:
https://www.rstudio.com/resources/cheatsheets/

These will come in handy as we get deeper into R.

Next week, we'll want to create graphs of data and sometimes it will be handy to have a frequency table to work from. To create a quick frequency table from the mtcars data set, we first have to covert one of the numerical variables to a string. In a dataframe, we convert it to a factor.

```
38
39  factor(mtcars$gear)
40  table(mtcars$gear)
41
```

```
 3  4  5
15 12  5
```

We don't need to apply the factor() function if the data in the column is already a string.

The last thing we are going to do is a simple random sample. Suppose that we want to generate a random sample of 10 rolls of a standard die but we don't actually have a dice. We can generate it from R.

```
42
43  sample (c(1:6), size=10, replace =T)
44
```

We can do the same thing from a column of our cars data set.

```
44
45  sample (mtcars$mpg, size=10, replace =T)
46
```

If you don't want to repeat any values, then set replace=F. But then size can only be the same size as your original vector or shorter.

R uses a typical pseudorandom number generator to generate its random samples. If you want to replicate your "random" sample, then you need to set a seed value.

```
47  set.seed(17)
```

You can use any number you want in the seed. Some people use their birthdays. There are many times when we are generating data when we might want to reproduce a specific result and setting the seeds allows us to do that. Otherwise, we will get a slightly different outcome each time we take our sample.

Tasks

Use the previous examples, any of the linked resources below or a Google search to complete the tasks. You will need to submit both the code used and the output of each command. Answer questions directly in the lab. Be sure to provide any explanations requested.

1. Create a vector of 20 elements using either seq() or sample(). Convert the resulting vector into a 4×5 array using any method we've discussed. Copy your array below.

2. Pull up the iris data set using the data() function. Create a frequency table of Species. Copy your frequency table below.

3. Calculate the value of the following expression in R. Report the final answer.

$$9^2 + 3 \bullet (9-5)^2 / 4$$

4. Find the average Petal length in the iris data set. First add up all the values, then divide by the number of operations. Do not use the mean() function. Report your answer below.

5. Sometimes we want to rescale a data set to be values between 0 and 1. We do this by finding the minimum and maximum values. Then we subtract off the minimum from each value and then divide everything by the difference between the minimum and the maximum. Perform this operation on the Sepal.Width column in the iris dataset and then save the result as a new column called Sepal.Width.Scaled. Check the values and confirm that all the values are between 0 and 1. Print a sample of the values below.

References:

1. Discovering Statistics Using R. Andy Field, Jeremy Miles, Zoe Field. (2012)
2. https://www.r-project.org/
3. https://www.rstudio.com/products/rstudio/#rstudio-desktop
4. https://www.anaconda.com/products/distribution
5. https://docs.anaconda.com/anaconda/navigator/tutorials/r-lang/
6. https://book.stat420.org/applied_statistics.pdf
7. https://scholarworks.montana.edu/xmlui/handle/1/2999
8. https://www.rstudio.com/resources/cheatsheets/
9. https://r-lang.com/setwd-function-in-r-with-example/
10. https://www.tutorialspoint.com/r/r_vectors.htm
11. https://www.tutorialgateway.org/r-arithmetic-operators/
12. https://www.datamentor.io/r-programming/operator/
13. https://www.w3schools.com/r/r_math.asp
14. https://cran.r-project.org/web/packages/available_packages_by_name.html
15. https://www.programmingr.com/statistics/frequency-table/
16. https://r-coder.com/set-seed-r/
17. https://www.programmingr.com/examples/neat-tricks/sample-r-function/