MTH 324, Lab #2, 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.

Descriptive Statistics

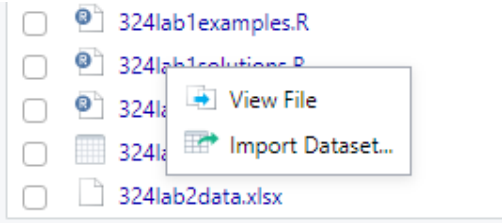
Last week we looked at how to access data sets already installed in base R. In this lab, we are going to read data from a file. To do that, we'll need to install and load some packages.

Install package: readxl

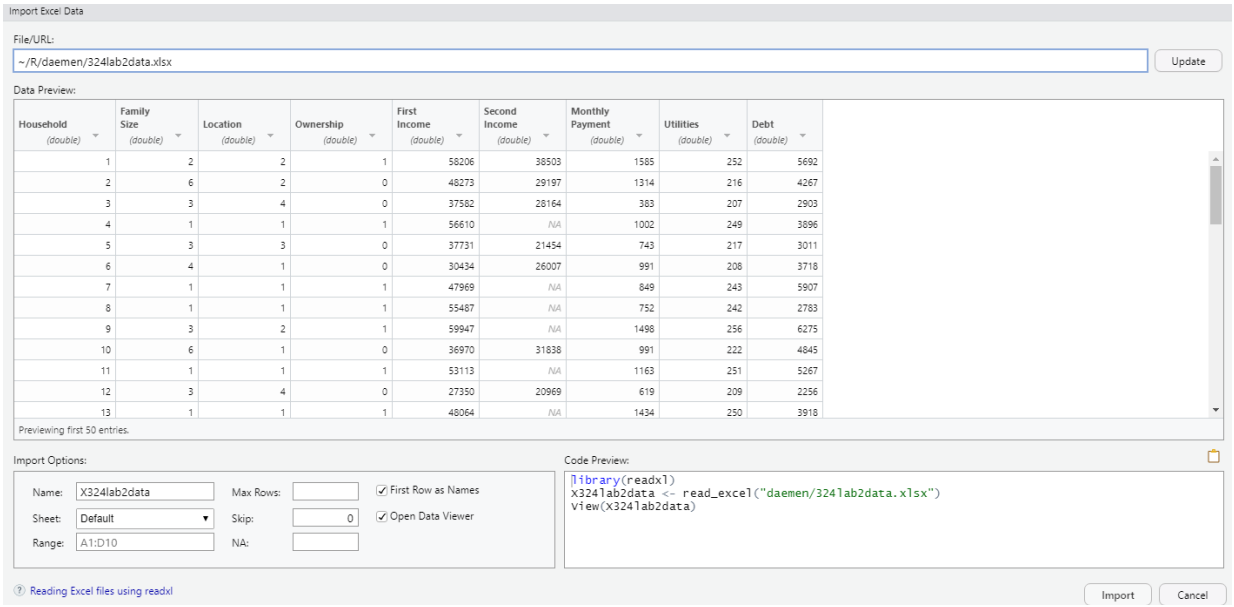
We can import csv files in base R, but we'll need the readxl package to read excel files. Load readxl into memory.

```
3  data1 <- read.csv('324lab2data.csv')
4
5  library(readxl)
6
7  data2 <- read_excel('324lab2data.xlsx')
8
```

I made two versions of the same data. One that is in csv format. One that is an Excel file. Load the data both way and view the imported data with View() to compare the results. Note, the examples above assume that the files are installed in your working directory. If they are not, you'll have to use a longer address. If you go to the file list (where we set the working directory last lab), you have move through folders to find it and if you click on it, you'll get an option to import the data.



If you click on Import Dataset, you'll get a menu that pops up with lots of useful information. You'll get a preview of the data, import options, and sample code you can copy and paste or click on import to run it in the console. (Copy and paste to save it for later.) You can change things like the name the dataframe is saved under (it normally just saves as the filename by default).



If you need to install other types of files, you will probably need to install other packages. Google can help you figure out which ones. If you use the Import Dataset option we described above, it may also direct you to specific packages for the given filetype. We'll mostly stick to datasets in R or R packages, or imported from Excel or csv files. If we do anything else, we'll go over how to import it. There are also packages for writing out to various file types.

We can check to see if the two data frames are the same.

```
12  identical(data1,data2)
13
```

We know these files are different versions of the same data, so if we get false, we might need to consider issues with importing it. We can check the data types of columns to see if that is the problem.

```
14  str(data1)
15  str(data2)
16
```

What we discover is that the Excel version imported all columns to the num datatype. The csv file imported some columns as int (which is fine, that is a numerical datatype), but it imported the money columns as strings. The problem is the dollar signs and commas. Fixing this is beyond the scope of our course, so we'll use data2 as the file we proceed with. The problem is common enough, though, so if you run into this issue in other cases, there may be some benefit in opening the file in Excel first, changing the file type and trying again. You may be able to remove the formatting in Excel so that R will process the data correctly.

As we saw last lab, we can refer to variables in a dataframe by their column names, or we can use indices.

We can refer to the first column of the dataframe using square brackets. R starts counting at 1, not 0. So, the first column is data2[1]. A nice feature of R is that we can remove a column using a minus sign.

```
57  data2<-data2[-1]
```

The first column of our dataframe isn't a variable (it's basically just a stand-in for the names of the families, or an index) so we can remove it since it doesn't measure anything.

Our main goal is to calculate some descriptive statistics.

Recall some common statistics are:

- Measures of center: mean, median, mode
- Measures of spread: standard deviation, variance, IQR, range
- Measures of location: quartiles, percentiles, rank

Check the mean, median and mode of the columns Family Size and Debt.

```
17  mean(data2$Debt)
18  median(data2$Debt)
19  mode(data2$Debt)
20
21  mean(data2$`Family Size`)
22  median(data2$`Family Size`)
23  mode(data2$`Family Size`)
24
```

Mean and Median produce values as output, but the mode returns only "numeric". To calculate the mode, we can write our own function (there is an example in a reference below) or you may be able to find it in a package. We don't use it often so we won't worry about it here.

If we try to find the mean or median from the Second Income column, we will run into problems because our data is missing some values. We can adjust the functions to overcome this.

```
25  mean(data2$`Second Income`, na.rm=TRUE)
26
```

The na.rm option allows us to ignore the missing values. The default is FALSE, and so the mean will result in NA if even one value is missing. The same modification works on the median.

Find the standard deviation, variance and IQR of a variable.

```
27  sd(data2$Debt)
28  var(data2$Debt)
29  IQR(data2$Debt)
30
```

To calculate the range, we have to find the maximum max() and minimum min() and then subtract them. There is no built-in range function in base R.

We use the same function to find quartiles and percentiles.

```
30
31  quantile(data2$Debt)
32  quantile(data2$Debt, 0.9)
33
```

By default, quantile will produce the 5-number summary: min, Q1, median, Q3 and maximum. We can tell it to provide other values. For one percentile value, just add the percentile you want, but if you want several different ones, you can use a vector. The default values as a vector would be c(0,0.25,0.5,0.75,1).

There is a rank function rank(). This function does not produce a single value. It produces a vector of the same length as the original data. We won't work with rank much until the end of the course when we look at non-parametric statistics.

```
33
34  rank(data2$Debt, na.last=TRUE, ties.method='average')
35
```

Sometimes it's useful to have the count of the data or the dimensions of a dataframe.

```
50
51  length(data2)
52  length(data2$Debt)
53  nrow(data2)
54  ncol(data2)
55
```

The Environment tab will display the dimensions, but we can also sometimes use these in calculations. The function length() gives the number of columns of a dataframe, or the number of entries in a vector. The function nrow() gives the number of rows of a dataframe, and ncol() the number of columns.

Another way to get the 5-number summary is the summary() function. It produces those 5 numbers and the mean.

```
35
36  summary(data2$Debt)
37
```

In fact, we can get this summary for every variable in the data set by leaving off the column name.

There are some packages that can give us more complete statistical summaries.

The package Hmisc has a function called describe() which provides the count, the number missing, the number of unique values, the mean, several percentiles of interest, and several of the largest and smallest values. The psych package also has a describe() function which includes the 5-number summary values, mean, count, standard deviation, standard error, skew, kurtosis (we'll talk about these last three later in the course) and MAD (mean absolute deviation). The package pastecs includes a stat.desc() function which produces various counts, min, max, range, sum, median, mean, variance, standard deviation, coefficient of variation, and some other values we'll discuss later in the course.

```
40
41  library(Hmisc)
42  describe(data2)
43
44  library(psych)
45  describe(data2)
46
47  library(pastecs)
48  stat.desc(data2)
```

Install the packages and try these out. Note, packages with function names that are the same get masked from packages installed earlier. You can call them anyway by adding the package name to the front. Hmisc::describe() will call the describe function from the Hmisc package even if you've installed the psych package on top of it. Or if the reverse is true, pysch::describe().

Another useful trick we can use is that if our data is Boolean (or takes values of 0 and 1), we can use the mean() function to find proportions.

```
59  mean(data2$Ownership)
60
```

Ownership has that property, so this function gives us the proportion of homeowners.

Another useful statistical function is calculating relationships between variables. We'll talk about this idea in depth next semester, but we can calculate correlations between numerical variables with cor().

```
60
61  cor(data2$Debt,data2$`Family Size`)
62
```

There is a strong relationship between variables if the correlation is close to 1 or -1, and weak if it's close to 0. Debt and Family Size don't appear to be closely related.

Tasks

1. Convert the Location variable in the dataset to a factor using as.factor() or factor() and then make a frequency table of it. Report the resulting table below. Why does the mean and other statistics for this variable not mean anything?

2. Calculate the mean, standard deviation and IQR of the Utilities variable. Report the values you find.

3. Report the following percentiles of First Income: 0, 0.05, 0.1, 0.25, 0.50, 0.75, 0.9, 0.95, 0.99, 1

4. Choose one of the descriptive statistics summary functions for the variables Utilities, Second Income and Location. Describe the results. Can the function handle missing values? What does it report for factor variables? Why did you choose this one?

5. Use the str() function to look at the structure of your descriptive statistics output. How easy is it to access the values in the summary table? For instance, you can access the standard deviation in the psych results using stats$sd if you save the output of the describe function as a variable stats. Do the other packages work the same way? Does this change your answer to the previous question?

6. Calculate the correlation between Monthly Payment and First Income. Report the value below. Does there seem to be a strong relationship between the variables or no? Is it higher or lower than the one between Debt and Family Size? Explain.

References:

1. Discovering Statistics Using R. Andy Field, Jeremy Miles, Zoe Field. (2012)
2. https://book.stat420.org/applied_statistics.pdf
3. https://scholarworks.montana.edu/xmlui/handle/1/2999
4. https://www.rstudio.com/resources/cheatsheets/
5. https://cran.r-project.org/web/packages/readxl/readxl.pdf
6. https://swcarpentry.github.io/r-novice-inflammation/11-supp-read-write-csv/
7. https://statisticsglobe.com/check-if-two-data-frames-are-the-same-in-r
8. https://datatofish.com/data-type-dataframe-r/
9. https://www.tutorialspoint.com/r/r_mean_median_mode.htm
10. http://www.r-tutor.com/elementary-statistics/numerical-measures/percentile
11. https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/rank
12. https://www.statmethods.net/stats/descriptives.html
13. https://stats.oarc.ucla.edu/r/faq/how-does-r-handle-overlapping-object-names/
14. http://www.sthda.com/english/wiki/correlation-test-between-two-variables-in-r