Week 1 Assignment: R

(Extra help with R: http://drgates.georgetown.domains/SummerClassificationRMarkdown.html)

NOTE: Whenever you do any assignment or task, always think about how to exceed the expectations. For example, if you are not sure if you "need" to comment – then comment. Comments should be smart and useful. Do not comment because it is "required", comment because it is a good coding practice that will make your code more readable. Avoid "picking apart" assignments ☺ Follow the requirements to the best of your ability – when it doubt – be creative – do more – and do it well. Never think about how to do the minimum amount of work, but rather, think about how to gain the maximum amount of knowledge.

Before you Start: Be sure that you have R and RStudio installed and ready to use. Before you start with this assignment, make sure you can print "Hello World" from RStudio. This will confirm that R/RStudio is ready to use.

For this Assignment, write an R program that does the following:

1) You will use the dataset called **TitanicDataset.csv**. This dataset is in class.
2) In your R code, be sure to **clearly comment** each area so that a reader can quickly see where you did each thing.

Example of good and easy to read commenting:

```
###### Read in the data into a dataframe and view the head of the data
#-------------------------
filename="TitanicDataset.csv"
TitanicDF <- read.csv(filename, na.string=c(""))
##print the head of the TitanicDF
(head(TitanicDF, n=5))
#-------------------------

##### Look at the data types for the variables in the dataset
(str(TitanicDF))
```

*** The easier your code is to read and understand – the better. Organize your code so that a reader can see where each section or function etc. is located and what it does. On the job people always need to read and use each other's code. You will also need to go back and read your own code. It is easy to forget what you were thinking at the time you wrote the code. Therefore, using good commenting is essential and required.

3) Once you have read in your data, **print out the structure/types of the data using str.** This will show the data types of each variable. Data types are very important, especially when performing supervised learning and model training. For example, the label in a dataset MUST be type "factor". If it is not, problems can occur.

4) **For every variable**, change the type (if necessary) to a more appropriate type. Some variables will not need to be changed. You decide. For example, numerical values should be type "num". Nominal values, such as gender should be type "factor". While you do this – think about each variable – think about what it represents – read about types in R (int, num, factor, ordered factor, char…) and then determine what each variable type should be. Write code that corrects each variable to the right type as needed.

For example, you will find that PClass probably reads in as num or int. This is not correct. The PClass is the price class or category. "1" means first class, "2" means second class, and "3" means third class. This is ordered factor data. So – write code to update your dataframe so that this variable is changed to ordered factor. It is OK to use Google to learn how to do this. You will use Google a lot!

Example of changing to factor – you can learn how to change to ordered factor:

```
TitanicDF$Survived=factor(TitanicDF$Survived)
```

5) Once you have corrected/updated all the types in your dataframe, print the str again.
6) Next, you will clean up the data – comment clearly in your code for each cleaning stage.

Follow these steps:

Cleaning Part 1:

(a) Remove (delete from your dataframe) the variable/column called, "PassengerId".

Why? Explain why you are doing this in your comments. Do you think that PassengerId would be useful for training a model? Comments can and should include notes about why things are being done. This will help you later and any readers now.

(b) Remove the "Name" column from your dataframe.
(c) Remove "Cabin" and "Ticket" from your dataframe.
(d) Round "Fare" to two decimal places.
(e) Now, print out the current dataframe head (n=5 or 10) and see where you are so far.

Cleaning Part 2:

(a) Print out the number of rows and columns currently in the dataset. When you print, use "cat" and print in a way that makes sense to humans. For example:

```
cat("The Titanic dataset has a total of ", nrow(TitanicDF), "rows.")
```

(b) Next, print out the number of "complete" rows. These are rows that do not contain any NA values. Again, use cat and print a human-readable note. **Moving forward, whenever you are asked to print something, assume that you must print it in a way that it makes sense to the viewer.**
Example:
`(nrow(TitanicDF[complete.cases(TitanicDF),]))`
(c) Next, remove/delete all rows that contain NA values. Print the **before and after** dataframes so you can see that it worked.

NOTE: There are many ways to do things in R. Hint for the above:
TitanicDF <- TitanicDF[complete.cases(TitanicDF),]

(d) **Aggregate** SibSp and Parch using SUM into one column. This will mean that you create a new column called FamilyNum which is the sum of SibSp and Parch. Then, you will remove SibSp and Parch.

(e) From this point, I am going to stop giving hints ☺ Part of coding is knowing what you need to do and then figuring out how to do it. At this point, you should have a dataframe with no NA values and with the correct data types. Next, have a look at your dataframe and determine if any other areas need cleaning or attention.

Exploratory Data Analysis (EDA) Part 1:

(a) **Create a function** that takes as parameters all the variable names in your dataframe. The function will loop through each variable and will create and print out a *table* for each of the variables.

To do this, you will need to learn how to write a function in R. You will need to understand how to get the names of the variables and how to call the function with those names. Inside the function, the function will loop through each variable name and will build and output a table of the data for that variable. Recall that each variable represents a column of data.

Build this SLOWLY. If you are new to functions – now is a good time to learn about them. Write a small function first – perhaps one that prints hello. Then, write a function that adds two functions. Once you get comfortable, think about the other parts you need to build.

Think about how to get the variable names. **You CANNOT type them in or hard-code them!** This code should work on ANY dataset with any variable names. So, you will need a method that gives you the variable names.
Hint: (names(TitanicDF))

Next, you will need to make sure that using the variable name will allow you to access the data in the column. For example,

```
(names(TitanicDF))  ## get the column names
ColNames<-(names(TitanicDF))  ## save the column names in a LIST
(ColNames[4])  ## Look at the 4th column name
(TitanicDF[ColNames[4]])  ## Look at the data under the 4th column
(table(TitanicDF[ColNames[4]]))  ## Create a table of the data in the 4th column
```

Always code in parts and one step at a time. Think about what you want to do and then think about all the "pieces" you will need to do it. Then, build and test each piece. Then, put them together.

(b) Create another function that takes all the column names. This function will determine if each variable (data column) is numeric. For each numeric variable, the function will print out the mean, median, mode, variance, and range. If the variable is not numeric, it will print out the

mode only. Call the function with your column names and make sure the function offers output that a viewer can understand.

(c) Create a function that takes Age and Sex and performs an independent samples t-test. Be sure the function produces proper and readable output.

Visual EDA

(a) Create a visualization using ggplot **for each variable in the dataset**. (The ggplot2 library will be required). Use color. Each plot must have a title and proper labels. Create a DIFFERENT plot type for each variable. So, you can have a boxplot, but only one. You can have a bar graph, but only one. etc.

(b) Is Fare correlated to PClass? Create a vis that answers this question.

(c) Create a visualization (ggplot) that shows **Survival by Gender**. Is survival affected by Gender? The vis should answer this question. Again, be sure that all visualizations use color, have a title, and are labeled.

Feature Generation:

(a) Create a new feature via binning. Create a new column (variable/feature) in your dataframe called AGEBIN. This column will have 4 categories: Child (up to 11 years old), Teen (12 – 19), Middle(20-45), Late(46-max age). Use the Age column to build this new variable and data. Keep Age – here you are not removing Age, you are generating an additional feature that is a "factor" type and based on Age.

For Example: Suppose I have a grades dataset with number grades. However, I want to create a variable that shows the letter grades. Using the number grades, I can build the letter grades. Here, LetterBin is my NEW feature that I created using NumGrade. You will do the same thing – but will use Age to create AGEBIN.

NumGrade	LetterBin
89.5	B
78.2	C
67	D
92	A
71.5	C
88.5	B

File I/O

Part 1: Writing output to a file

The last part of this assignment will help you to practice with using, reading, writing, and appending to files in R.

1) At this point, you have created a lot of code from all of the requirements above. Go back into the code. Create a text (.txt) file. Open the file for append (after you create it). Write into the file

all of the items that you printed to the output console. This will include all of the "output" that your code generates. It will not include the images – as they will not print to a text file.

2) Add code so that you not only print all output to a text file, but you also print it in a nice and readable way.
3) NOTE: This does not mean that you should remove the print or outputs that you currently have – leave those. This only asks you to *also* write all output generated by your program into a text file. You will find that this is very useful and allows you to look at all of the output in one place.
4) Name this file smartly.

Part 2: Saving a CLEAN dataframe to a csv file

1) Create a new .csv file. Write to that file your fully cleaned dataframe data. This will include the new features you generated as well.
2) Name this file in a smart way so that when the TA runs your code, they will see this file get generated and its name will identify it as your clean Titanic dataset.

WHAT TO SUBMIT:

1) You will submit four (4) things:
    a. You will submit your fully functional .R code. (Always and only submit code that runs and can be run from any folder. Do not use personal paths in your code. For example, this is a PATH on my computer:
    "C:/Users/profa/Documents/R/RStudioFolder_1/DrGExamples/RExamples"

    However, I cannot place this path into code that I give to someone else because they will not have the same folders on their computer ☺

    b. You will submit a .docx Word document that contains
        i. A copy of your raw dataframe – before you clean it at all – just the first 10 rows!
        ii. A copy of your final dataframe after all the cleaning and after the feature generation. Just the first 10 rows.
        iii. A copy of all tables that you created.
        iv. A copy of all visualizations you created.
    c. You will submit the .txt file that you created.
    d. You will submit the .csv file you created that contains your fully clean and augmented dataframe.

Deliverables:

Do not TAR for any reason.

Do not ZIP unless you must.

FOLLOW THE HOW TO SUBMIT RULES -