


Data Cleansing

Open RStudio and create a new project under your Module 4 folder and call it **Mod4Assignment2**. For this assignment, you will be creating an R Markdown document to present your results for the current assignment.

Once completed, all you need to do is submit the word document that is created.

Create the R Markdown Document

- 1.) In RStudio, select *File -> New File -> Text File*. This will create a blank text file in the same area that scripts were created in previous assignments (upper left panel). Save this file to your project as **Mod4Assign2Answer.rmd** (it is important to save with the .rmd extension as this saves the text file as an R Markdown file).
- 2.) Create a Header 1 with the title: **Module 4 - Assignment 2**
- 3.) Create a Header 2 with the title: **Last Name, First Name** (replace with your name)
- 4.) Create a Header 3 with the title: **Data Cleansing**
- 5.) Click on the dropdown arrow next to the Knit icon  at the top of the R Markdown Pane in RStudio and select Knit to Word.
- 6.) Notice that you now have a document in your files for the project named **Mod4Assign2Answer.docx**. This is the file you will be uploading later to Canvas.
- 7.) For this assignment, you will need to download the **CustomerChurn.xlsx** file from Canvas. Save this in the same folder that you created project in. This is a small sample of a much larger file but it provides a good overview of data cleansing.
- 8.) Add a chunk of code that will load both the tidyverse package as well as load the dataset you just downloaded. When you click on the .csv file, look at how the data set is being uploaded.

Specifically, you may notice columns that should be a double/numeric being imported as a character (e.g., MonthlyCharges and TotalCharges may be coming in as a character).

MonthlyCharges	TotalCharges
(character)	(character)

It is always good to make sure you are importing data correctly and I would encourage you to examine the import screen carefully as this is commonly the case. You can change this in the dropdown box below the title of the column and it will also change your code for importing. Copy the "Code Preview" and include it in your R Markdown chunk of code. See the code below on how I loaded the file (notice that I did have to change the way TotalCharges and MonthlyCharges was being imported):

```
library(readxl)
CustomerChurn <- read_excel("CustomerChurn.xlsx",
  col_types = c("text", "text", "text",
    "text", "numeric", "text", "text",
    "text", "text", "text", "text", "text",
    "text", "text", "text", "text", "text",
    "numeric", "numeric", "text"))
```

You will notice that there may be errors resulting from the import. This is a good indicator that something may be wrong with the data (i.e., missing data). We will be looking at this next.

- 9.) For this assignment, there are 2 parts that need to be completed within the R Markdown document. The first will be looking for missing or unavailable data. The second will be dealing with outliers.

Part 1: Dealing with missing data

- 1.) Within the R Markdown document, create a Header 4 with the title: **Cleaning Missing Data**
- 2.) The CustomerChurn dataframe should contain 19 rows (or customers) with 20 variables for each customer. However, some of the data is missing or may have come in as NaN or NA. When we imported the Excel file, R automatically changed missing data fields to NA. Let's take a look at what data we are working with.

PaymentMethod	MonthlyCharges	TotalCharges	Churn
NA	29.85	29.85	No
Mailed check	56.95	1889.50	No
Mailed check	NaN	108.15	Yes
Bank transfer (automatic)	42.30	NA	No
NA	70.70	151.65	Yes
NA	99.65	820.50	Yes
Credit card (automatic)	89.10	1949.40	No
Mailed check	29.75	301.90	No
Electronic check	NaN	3046.05	Yes

- 3.) Create a new chunk of code. Before adding any code, let's look at the data file by clicking on the filename in the upper right portion of the screen (the environment section).
- 4.) Add a new line of code that will show you a summary of the data: **summary(CustomerChurn)**

This will provide a summary of all the variables in your dataset. Notice how we can quickly see that MonthlyCharges and Total Charges have NA's within the data set. Other variations to find this include:

Module 4: Assignment #2

```
summary(CustomerChurn)
```

customerID Length:19 Class :character Mode :character	gender Length:19 Class :character Mode :character	Partner Length:19 Class :character Mode :character	Dependents Length:19 Class :character Mode :character	tenure Min. : 1.00 1st Qu.:16.50 Median :25.00 Mean :26.42 3rd Qu.:30.50 Max. :80.00	PhoneService Length:19 Class :character Mode :character
MultipleLines Length:19 Class :character Mode :character	InternetService Length:19 Class :character Mode :character	OnlineSecurity Length:19 Class :character Mode :character	OnlineBackup Length:19 Class :character Mode :character	DeviceProtection Length:19 Class :character Mode :character	TechSupport Length:19 Class :character Mode :character
StreamingTV Length:19 Class :character Mode :character	StreamingMovies Length:19 Class :character Mode :character	Contract Length:19 Class :character Mode :character	PaperlessBilling Length:19 Class :character Mode :character	PaymentMethod Length:19 Class :character Mode :character	MonthlyCharges Min. : 18.95 1st Qu.: 36.08 Median : 56.15 Mean : 62.78 3rd Qu.: 94.38 Max. :113.25 NA's :4
TotalCharges Min. : 29.85 1st Qu.: 320.57 Median :1919.45 Mean :2582.56 3rd Qu.:3875.04 Max. :7895.15 NA's :3	Churn Length:19 Class :character Mode :character				

- 5.) Following the readings from this week's module, let's first use `mutate` to change the NA's to numbers. This is one option among others that could be done with the data.

Other options include removing the rows altogether or following the `impute()` approach. You should really look at the data and what you are trying to find out to make decisions how you will handle missing data. At the very least, include some comments after the code in the R Markdown document stating how missing values were handled.

- 6.) Add the following code to the same chunk to replace missing values:

```
CustomerChurn2 <- mutate(CustomerChurn, MonthlyCharges = replace(MonthlyCharges,  
is.nan(MonthlyCharges), median(MonthlyCharges, na.rm = TRUE)))
```

- 7.) With this code, we are creating a new dataframe called **CustomerChurn2** and using the `mutate` function to replace only `MonthlyCharges` that have a value of `NaN` to the median of the `MonthlyCharges` (Note: you could also use `mode` or `mean` as well).
- 8.) Using similar code, write the lines of code that will use `CustomerChurn2` to mutate `TotalCharges` to be replaced missing values with the mean. Note: you will need to change `is.nan` to `is.na` as the missing values in `TotalCharges` are represented with an `NA`.

Your file should now look like this:

Module 4: Assignment #2

PaymentMethod	MonthlyCharges	TotalCharges
ElectronicCheck	29.85	29.850
Mailed check	56.95	1889.500
Mailed check	56.15	108.150
Bank transfer (automatic)	42.30	2582.559
ElectronicCheck	70.70	151.650
ElectronicCheck	99.65	820.500
Credit card (automatic)	89.10	1949.400
Mailed check	29.75	301.900
Electronic check	56.15	3046.050

- 9.) Finally, you can also replace missing values with text. In the image above, we see NA for payment method. We would like any missing values to read *ElectronicCheck* instead of NA. To do this, copy the code from step above where you replaced *TotalCharges* that were blank with the mean. You will need to replace the part of the code shown below:

So from the step above, the code:

```
mean(TotalCharges, na.rm = TRUE)
```

should now be replaced with:

```
"ElectronicCheck"
```

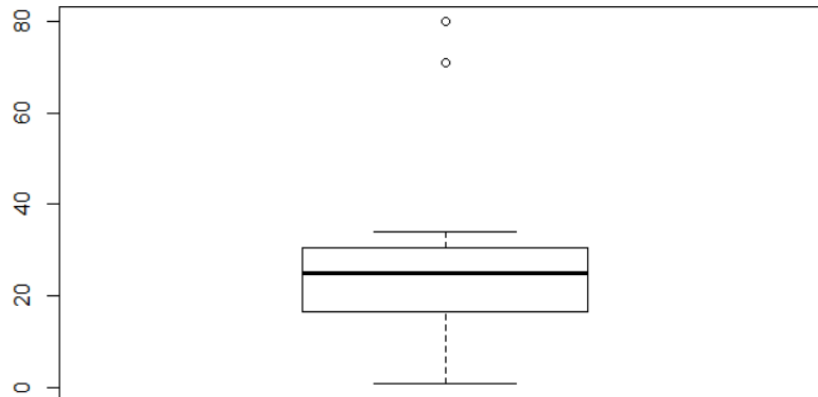
This will just replace blank entries (NA entries) with the text *ElectronicCheck*.

- 10.) Create a new dataframe called **CustomerChurn3** and using dplyr, add the *PaymentMethod*, *MonthlyCharges* and *TotalCharges* variables from *CustomerChurn2* (the ones you just removed the missing values from).
- 11.) Use the print data frame command, i.e. `print(CustomerChurn3)`, to show your results from *CustomerChurn3*

Part 2 - Outliers

- 12.) You also will need to deal with a number of outliers. Insert a new chunk of code and create a boxplot that will display the *tenure* variable. You should get a plot similar to that below:

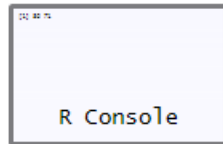
Module 4: Assignment #2



You will notice some outliers are present in our data represented by the points around 80 above.

13.) You can actually identify the outliers by including the following command in your boxplot (try it as a new line of code in your current chunk):

```
boxplot(CustomerChurn2$tenure)$out
```



```
[1] 80 71
```

14.) You can see the values of the outliers include 80 and 71. As a data analyst, you must decide if these are truly outliers but for now, let's remove these 2 as they may cause issues with our analysis.

15.) Using the following code, we can assign the outliers to a new dataframe and see which rows currently have these outliers:

```
outliers <- boxplot(CustomerChurn2$tenure)$out
```

```
CustomerChurn2[which(CustomerChurn2$tenure %in% outliers),]
```

Module 4: Assignment #2



customerID	gender	Partner	Dependents	tenure
6388-TABGU	Male	No	Yes	80
9959-WOFKT	Male	No	Yes	71

customerID <chr>	gender <chr>	Partner <chr>	Dependents <chr>	tenure <dbl>	PhoneService <chr>
6388-TABGU	Male	No	Yes	80	Yes
9959-WOFKT	Male	No	Yes	71	Yes

2 rows | 1-10 of 20 columns

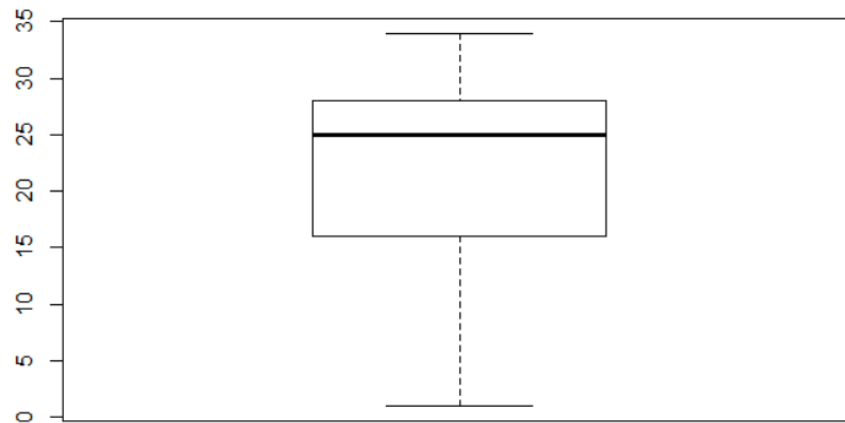
16.) Add the following code to remove these two outliers from the file:

```
CustomerChurn3 <- CustomerChurn2[-which(CustomerChurn2$tenure %in% outliers),]  
boxplot(CustomerChurn3$tenure)
```

This will create a new dataframe called CustomerChurn3 and the results will look like:

Global Environment ▼	
Data	
CustomerChurn	19 obs. of 20 variables
CustomerChurn2	19 obs. of 20 variables
CustomerChurn3	17 obs. of 20 variables
Values	
outliers	num [1:2] 80 71

Module 4: Assignment #2



Notice that our new CustomerChurn3 data has only 17 observations (the two outliers have been removed) and the resulting boxplot of CustomerChurn3 shows no outliers.