Zachary Tankersley

Southern New Hampshire University

DAT-375-T2962

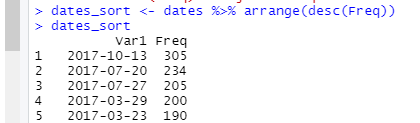
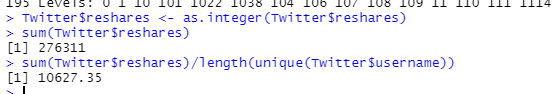
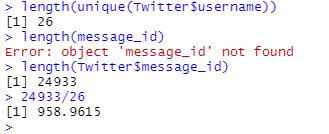
November 13, 2022

**Module Three Journal**

To analyze the social media data set which is in a relatively raw form we will first perform an exploratory data analysis via the data mining and analysis tool R Studio to achieve the following objectives:

* The average number of messages per unique username.
* The average number of reshares per unique username.
* The highest number of original (non-reshared) messages by time frame.

For the first task finding the average number of messages per unique user, we need to find the number of messages. Assuming that retweets and shares are included in the overall message count, the number of messages would just be a simple count of the rows or “message\_id” column. Once the number of messages is known (24,933 total messages), the number of unique usernames must be found which can be obtained via a simple script: “length(unique(Twitter$username))” which returns the value of 26. For the average number of messages per username, simply divide the total number of messages by unique usernames which end up being 958.9615 or about 959 messages per user. To achieve the second objective, the “reshares” column data must first be converted to integer types, then a simple script can be run to obtain the sum of all reshares which comes out to be 276,311 reshares, and then that number can be divided by the number of unique users using the following script: “sum(Twitter$reshares)/length(unique(Twitter$username)) which returns a value of 10,627.35 or about 10,627 reshares per unique user. The third and final objective is more complicated and requires a couple of stages to complete. Because the “time frame” is not specified (IE: half day, quarter day, week, month, etc.) it is assumed that the time frame means “day” for this objective. I first converted the “created\_at” column to dates from the time function type they were stored as using the following script: “Twitter$created\_at <- as.date(Twitter$created\_at)”. I then stored the dates in a new data frame and table using the following script: “dates <- table(as.data.frame(table(Twitter$created\_at))”. This gives a two-row table where the date would be held along with the frequency of messages for that date. I then sorted the table by frequency in descending order in a new table with the following script: “dates\_sort <- dates %>% arrange(desc(Freq))”. This returns the date with the highest occurrence of messages sent at the top of the new table which is 2017-10-13 with 305 messages sent. The following screenshots of the scripts and results will validate the findings:



The techniques I chose were very simple, mostly due to the simplicity of the objectives/directives and the non-specific nature of the requests. I chose R studio as the analysis tool as it makes importing and working with an already formed data set such as the excel file a very streamlined process. I do not know for certain if the techniques used are the *most proper,* however, they produced the desired results.

For choosing the individual scripts, I knew that I needed to isolate the number of *unique* usernames, as the sample size of tweets and usernames is rather large, and filtering through for minor differences and keeping count would be very time-consuming. I used the length function in combination with a “unique” modifier on the data set to obtain this count, and then simply divided the total number of tweets by this number to receive the average. A similar operation/script was used for the second objective. For whatever reason, the “reshares” data are stored as characters in the original data set so they needed to be converted into integers. Once that was done, the number of reshares could simply all be added using the sum function, and again, divided by the number of unique usernames to receive the average number of messages per user. The third objective threw me for quite a loop, candidly speaking. The non-specific nature of the desired “time frame” led me to choose my time frame, and in the end, I chose to go by individual day, rather than a half day, quarter day, month, week, etc. I had to convert the date/time data from “time function” to the more simple “date” type, and then I created an entirely new table where the messages were grouped by days and then I used an “arrange” function to sort them in descending order so that the highest message frequencies would be shown on top. All results were validated via Microsoft excel functions that did nearly identical operations.