Stat 535 - Midterm Project

Due: Wednesday, 10/18/2023, 6:00pm, EST

Type the following (up to and including the date) into your RMarkdown file:

"The Code of Honor will be strictly applied as described in The Academic Code of Honor Handbook (https://www.umass.edu/honesty/). I will not give or receive aid on this exam. This includes, but is not limited to, viewing the exams of others, sharing answers with others, and making unauthorized use of books or notes while taking the exam. Relying on solutions from other students, whether or not they are currently in the course, constitutes plagiarism. I will not seek help from any person, this exam will reflect only my original work. I acknowledge that failure to abide by the Code of Honor will result in disciplinary action."

Signature: [– embed a scanned image of your signature here–] Date: _____

This take-home midterm exam is open book. You may consult your homework assignments, lecture notes, reference books or other online resources (which DO NOT involve your posting of a question about the exam in any online forum). You MAY NOT talk about or receive guidance for this exam with anyone: neither classmates, other students or any other person. This work should be entirely your own. You must submit both an RMarkdown (Rmd) file and the HTML output of that file. It is your responsibility to make sure that the results are **executable** and **legible**. Beware of lines that are cut in the middle. Failure to submit the exam on the due date and time will result in a grade of 0.

Exam starts: Wednesday, October 11th, 2023, 9:00pm EST.

Exam due: Wednesday, October 18th, 2023, 6:00pm EST.

In the first part of this exam, we will work again with the XML file that contains 4,938 tweets that include the string "UMass" (this is **not** the same version that we worked with in Homework 4). You will study the XML file using the tools we learned and compare it to a DB version provided here using direct SQL queries. In the second part of the exam you will perform some exploratory data analyses on the Twitter data.

Part I - "Scraping"

In this part you may not copy the contents of the SQL file to the memory as a local data frame. Any data you extract in this part must be retrieved via direct SQL queries using `dbGetQuery()` unless explicitly specified otherwise.

1. [2 pt] Set up a database connection to the `UMass_tweets.db` file.
2. [3 pt] There is one table, named `Tweets`, in this database. Use a SQL query to show the first 5 records of the table.
3. [4 pt] The query

   ```
   PRAGMA table_info(Tweets)
   ```

returns a summary data frame the contains (among other things) the names of the fields and the data type (**SQL data type**) of each of the fields in the table `Tweets`. Use this and `str_replace` to create a **named character vector** in R that contains the **R data types** of each of the fields as its entries and the field

names as the names corresponding names. Print the contents of the vector, assign it to a variable named `types`. Here you may save the output of the "PRAGMA table_info(Tweets)" into an R local data frame.

Recall: you can assign names to a vector `v` using `names(v) <- a_vector_of_names`

4. [3 pt] Let's start working on the XML file. Our goal is to extract from it a data frame which is as close as possible to the DB table. First, read the contents of the XML file into a dataframe named `umass_tweets_xml_raw`. Secondly, use regular expressions to keep only entries that include actual tweets, save those to `umass_tweets_xml`. Do not use R functions from packages designed for reading XML files, the point here is to use the material we learned in class - treating the XML file as raw text and extracting information from it using regular expressions.

5. [3 pt] Verify that the number of messages is the same in both your XML version and the DB version (last reminder, use only direct SQL queries to get the number of rows in the database. I.e. don't use `nrow(dbGetQuery(...))`). This should be 4938. Assign this number (as a result of a computation, don't type `4938...`) as a numeric scalar to a variable called `N`.

Are the two data sets the same?

6. [45 pt] In order to answer the question, we will extract the data from the XML file using string manipulations and store it in a new data frame that has the same dimensions and variable names as the DB version.

On examining the XML structure, you'll notice that fields are defined between `<tags></tags>`. E.g. the database values of the field `created_at` are stored between `<created_at>` and `</created_at>` tags in the XML file.

Write a function called `getXMLfield` which takes a single input, a **named vector** of length one. Specifically, one element of the same one you created in Part I Question 3. The input then is a character vector of length one where the R type of data is the element, and its name is a name of a field from the database. The output will be a vector that contains the values of the corresponding `field` variable, cast according to the data type they should have.

Once you are happy with your function, you should create a new data frame for the XML data as follows:

```
DBfields <- dbListFields(con, "Tweets")
umass_tweets_xml_df <- as_tibble(matrix(NA, nrow = N, ncol = length(DBfields)))
colnames(umass_tweets_xml_df) <- DBfields
```

This initializes a data frame that has the same number of rows and the same columns (alongside their names) as the Tweets DB table.

You should fill all your columns using the function you created. You need to run it once for each variable. Write a `for` loop that will do this for you (you'll need to tweak the below example):

```
umass_tweets_xml_df[ , "created_at"] <- getXMLfield(types[1])
umass_tweets_xml_df[ , "id_str"] <- getXMLfield(types[2])
```

> *Extra credit [2 pt]: instead of multiple calls within a `for` loop to `getXMLfield` use a single call using a vectorized notation, e.g. using a function from the `apply` family, or another mapping operation from `dplyr` or `purrr`.*

Let us now check our new data frame that is based on the XML file.

Write a loop that examines each of the columns of `umass_tweets_xml_df` and compares those to the columns of the `Tweets` table from the database file (again, use SQL queries to obtain the latter, you may assign individual columns to local R variables). You should find that 2 of the columns do not perfectly match. Which are those?

What went wrong?

Compare the entries in the two mismatched fields. One of the fields should have 1709 mismatches, the other 4938 (all!).

Carefully compare the mismatched entries in the column that has 4938 mismatches. Using regular expressions, correct the column `umass_tweets_xml_df` that has 4938 mismatches so that it will be equal to the respective column in the database table. Substitute the corrected column for the mismatched one in `umass_tweets_xml_df`.

> *Up to [8 pt] extra credit: Copy the column that has 1709 mismatches from the DB table to a local R vector. Using regular expressions, correct this vector so that it is as similar as possible to the respective column from the XML data frame. You will receive the full score if all the elements of the modified vector will be equal to the respective XML column.*

This kind of data cleaning is very common in data science.

Part II - Data Manipulations

In Part II we will use the table from the database (DB file). We'll generate some additional quantitative variables, based on different summaries of the tweets and metadata. Here, you may copy the `Tweets` table to a local data frame and use base R and/or `dplyr` functions to analyze and transform your data.

```
umass_tweets <- dbReadTable(con, "Tweets")
```

[8 pt] 1. Create a new variable that counts the number of characters in a tweet (the `full_text` field), show the lengths of the 5 longest tweets for each of the 4 most common languages in the data (total of 20 entries) (hints for `dplyr`: `slice`, `seq_len`)

[8 pt] 2. Create a new variable that counts the number of words in a Tweet, show the number of words that correspond to the 5 longest tweets for each languages (in terms of characters, from question 1)

[8 pt] 3. Create a new variable that measures the average length of a word in a tweet (that is, for each tweet measure first the number of characters in each word, and average over those). Show the the average number of characters per word for the 5 tweets with the most words for each languages.

[8 pt] 4. Show the full text of the 2 most quoted tweets in the dataset (across all languages)

[8 pt] 5. What are the two most common hashtags and how many times each appears in the dataset? (hashtags are keywords that start with a # and then a letter followed by letters or numbers, do not include punctuation marks and are not case sensitive) (there could be more than one hashtag per tweet)

GOOD LUCK!!!