STAT 19000 Project 3

Topics: lists, data.frame's, sapply, tapply

Motivation: Practicing with the suite of apply functions will allow us to continue to perform operations in a vectorized manner. Proper preparation prevents poor performance, you need to practice in order to become proficient.

Context: We've been working with the suite of apply functions in R to solve problems in a vectorized way. We continue to hone our skills with these functions in this project.

Scope: The entire suite of apply functions in base R, data.frame's, and lists.

You can find useful examples that walk you through relevant material here or on scholar: `/class/datamine/data/spring2020/s`
It is highly recommended to read through these to help solve problems.

Use the template found here or on scholar: `/class/datamine/data/spring2020/stat19000project03template.ipynb` to submit your solutions.

After each problem, we've provided you with a list of keywords. These keywords could be package names, functions, or important terms. Each keyword will point you in the right direction when trying to solve the problem, and give you accurate terminology that you can further look into online.

Don't forget the very useful documentation shortcut `?`. To use, simply type `?` in the console, followed by the name of the function you are interested in.

You can also look for package documentation by using `help(package=PACKAGENAME)`.

Question 1:

Load a dataset located at https://raw.githubusercontent.com/zygmuntz/goodbooks-10k/master/books.csv into a data frame in R, using the `read.csv()` function.

1a. *(1 pt)* Stop words are commonly used words that are usually filtered out from text analysis due to the fact that they provide very little information. Examples of stop words include "the", "and", and "on". Load the list of selected stop words provided for you in the file `/class/datamine/data/spring2020/stopwords.RData`. To do this run:

```
load("/class/datamine/data/spring2020/stopwords.RData")
```

this will load a variable called `stopwords` into your R environment. Try running `str(stopwords)` to take a look. Create a function that gets the number of stop words in a title. We've provided you with the "skeleton" of the function below. Simply fill it in. Test your new `count_stop_words` function by running the following:

`count_stop_words("This is the best of us")` (should be 4) `count_stop_words("Ok, I think we figured it out.")` (should be 3)

In the examples above, `x` inside our `count_stop_words` function will contain the texts: "This is the best of us" and "Ok, I think we figured it out.". Your job is to take those sentences and count how many stopwords in our variable `stopwords` occur in `x`, save that number in the `stopwords_in_title` variable, and then the end of our `count_stop_words` function will return that number.

```
count_stop_words <- function(x) {
    # write code that counts the number of stopwords in a string, x, below
    # assume that stopwords is a character vector where each element is a word,
    # and has already been loaded via load("~/path/to/stopwords.RData")

    # stopwords_in_title should be a number
```

```
    return(stopwords_in_title)
}
```

Hint: *Be careful when making comparisons. Note that all stopwords are in lowercase.*

Keywords: *strsplit, %in%, tolower*

1b. *(1 pt)* Create a vector containing how many stop words each original title has using the function we wrote in (1a). Which title has the highest number of stop words? How many stop words does it have? How many titles have 1 stop word?

Keywords: *which.max, sapply*

1c. *(1 pt)* In (1a) we create a vector with the number of stop words per title. Make a histogram of the distribution of number of stop words. Does this graphic surprise you or is it expected?

Keywords: *hist*

Question 2:

Read in the `/class/datamine/data/spring2020/rotten_tomatoes_reviews.csv` file into a dataframe called `reviews`. Read the `/class/datamine/data/spring2020/rotten_tomatoes_movies.csv` file into a dataframe called `movies`. Consider the `rotten_tomatoes_link` to be a unique movie indicator (i.e., each movie has a unique value of `rotten_tomatoes_link`).

2a. *(2 pts)* Use nested `sapply` to calculate how many people are a part of the movie's listed cast. What is the largest cast? What movie does the cast belong to?

Keywords: *strsplit, sapply, length, max, which.max*

Hint: *Take a look at movies$cast[1] to see what character you can use to split the vector into individual names.*

Hint: *Take a look at our examples to see how to pass function's arguments in `sapply`.*

2b. *(1 pt)* Split the `reviews` dataframe into a list of two dataframes using the `split` function. One dataframe with reviews from top critics (based on the column `critic_top`), and the other dataframe with the remaining reviews.

Hint: *After splitting, you may want to rename the resulting named list by doing something like: `names(my_split_list) <- c("not_top", "top")`.*

Question 3:

3a. *(2 pts)* Use the `reviews` dataframe to get a list of `rotten_tomatoes_link` for movies that have reviews containing the word 'soul'. Use `tapply` and the list provided to compare the average tomatometer rating between movies that have reviews containing the word 'soul' and the ones that don't.

Keywords: *grepl, %in%, tapply, mean*

3b. *(2 pts)* Use any of the apply functions to find one factoid you think is interesting from this data. Include your code and your factoid in your answer.

Keywords: *apply, sapply, tapply, lapply*

Project Submission:

Submit your solutions for the project at this URL: https://classroom.github.com/a/i7EMv_O- using the instructions found in the GitHub Classroom instructions folder on Blackboard.

Important note: Make sure you submit your solutions in both .ipynb and .pdf formats. We've updated our instructions to include multiple ways to convert your .ipynb file to a .pdf on scholar. You can find a copy of the instructions on scholar as well: `/class/datamine/data/spring2020/jupyter.pdf`. If for some reason the script does not work, just submit the .ipynb.