# STAT 19000 Project 2

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

**Motivation:** When using R it is best to think about performing operations in a vectorized manner when possible. The suite of apply function provided in R are a key component in doing so. Practicing with the suite of apply functions will allow us to continue to perform operations in a vectorized manner. Sapply is one such function that performs a function on a list of inputs. Practicing with these functions will beneficial and soon you'll be much more proficient performing operations in R.

**Context:** We've previously worked with the useful `tapply` function in R, and have seen how it can conveniently divide our data into groups and perform functions on those groups. In many instances, what we really want is a way to apply a funtion to a whole list of inputs. For these situations, we will turn to another function in the apply suite called `sapply`.

**Scope:** `sapply` takes the data, X, a function, FUN, and applys the function to each element in the data. It has two extra arguments: simplify, and USE.NAMES. simplify dictates whether or not your result should be simplified to a vector, matrix, or higher dimensional array if possible. USE.NAMES defines whether or not to use X as names for the result.

The `/class/datamine/data/spring2020/stat19000project02examples.R` R script will provide you with useful examples that will help you solve problems given in this project.

Use the template found here or on scholar: `/class/datamine/data/spring2020/stat19000project02template.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.

This project will be done in the Jupyter notebook environment. Open a web browser of your choice (it can even be on your computer, rather than on Scholar), and go to this URL:

https://notebook.scholar.rcac.purdue.edu

The first time that you do this, after you login (using your Career username and Career password, without BoilerKey), please click "New" and make sure that the option "R 3.6 (Scholar)" appears there.

For questions that require writing a function, we've provided you with skeleton code that you will need to fill in and figure out how to use.

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:

**1a.** *(1 pt)* 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. How many rows and columns does the dataset contain?

**Keywords:** *read.csv, nrow, ncol, dim*

*Hint: If you set the option `stringsAsFactors` in `read.csv` as `FALSE` when you import the data, then the strings in your data will be imported as strings (rather than getting converted to factors). This might not seem important at the outset, but it will be helpful when you are searching for patterns in the strings during your work with the data.*

**1b.** *(1 pt)* Get a vector containing the word count of the `original_title` of each book. What is the title with the largest word count? Do not use functions such as `wordcount` from package `ngram` or similar. Consider contractions as a single word.

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

**1c.** *(1 pt)* Get a subset of the dataset called `ratings` that contains the following columns: `average_rating`, `ratings_count`, `work_ratings_count`, `ratings_1`, `ratings_2`, `ratings_3`, `ratings_4`, and `ratings_5`. Use `sapply` to find the minimum value from each column in your new `ratings` dataset.

**Keywords:** *sapply, subset*

## Question 2:

**2a.** *(1 pt)* Starting with the original data frame from question 1, use the `subset` function to make a new data frame that contains *only* the columns that have the words `rating` or `count` present in the header. (In other words, if you called your data frame `myDF` in Question 1, then we want to build a new data frame that only has the columns of `myDF` with the word `rating` or `count` appearing in `names(myDF)`.)

How many columns does your new data frame have?

**Keywords:** *names, grep*

**2b.** *(1 pt)* Use the `sapply` to make a `summary` of each column in your new data frame. Use the option `simplify=F` in the `sapply`, so that the first several lines of the output look like this.

```
$books_count
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00   23.00   40.00   75.71   67.00 3455.00

$average_rating
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.470   3.850   4.020   4.002   4.180   4.820
```

**Keywords:** *sapply, summary*

## Question 3:

**3a.** *(1 pt)* 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`. Take a look at information the dataframes contain.

Consider the `rotten_tomatoes_link` to be a unique movie indicator (i.e., each movie has a unique value of `rotten_tomatoes_link`).

How many movies do we have in each of these two dataframes?

(As in Question 1, please set the option `stringsAsFactors` in `read.csv` as `FALSE` when you import the data, so that the strings in your data will be imported as strings, rather than getting converted to factors.)

**Keywords:** *read.csv, stringsAsFactors, head, str, names*

**3b.** *(1 pt)* How many movies do the two dataframes have in common? Remember that the column `rotten_tomatoes_link` is present in both datasets, and uniquely identifies a movie. If both datasets have, for example, '/m/0814255' in the `rotten_tomatoes_link` column, that would count as 1 movie (so far) in common.

**Keywords:** *unique, length, %in%, sum, tapply, mean, na.rm*

**3c.** *(1 pt)* For each movie in our `reviews` dataframe, calculate the average critic score.

**Keywords:** *unique, length, %in%, sum, tapply, mean, na.rm, as.factor*

**Hint:** *Note that there are a lot of NA's in the* `critic_score` *column in our* `reviews` *dataframe. Make sure to take that into account when calculating the average.*

**3d.** *(2 pt)* We can use `tapply` to get the number of "Rotten" and "Fresh" reviews for each movie.

```
RotOrFresh <- tapply(reviews$critic_icon, reviews$rotten_tomatoes_link, table)
```

Follow up by using `sapply` on our new dataset, `RotOrFresh`, to calculate the difference between the number of "Fresh" and "Rotten" reviews for each movie. For example, let's say we have a movie with 10 "Fresh" reviews and 2 "Rotten" reviews. The difference would be 8. If we had 10 "Rotten" and 2 "Fresh", the difference would still be 8.

**Hint:** *There is no reason why you could not add or subtract the results of multiple* `sapply`*'s.*

## Project Submission:

Submit your solutions for the project at this URL: https://classroom.github.com/a/GWA9qQQn using the instructions found in the GitHub Classroom instructions folder on Blackboard.

**Important note:** Make sure you submit your solutions in both .ipynb and .html formats.