Project 2

Overall Goal

Vignettes are explanations of some concept, package, etc. with text, code, and output interweaved. Here are a few examples of varying quality (the first two being especially relevant):

- NHL API
- COVID API
- Basic logistic regression
- Cross validation

We already know how to make them with R Markdown!

Our goal with this project is to create a vignette about contacting an API using functions you've created to query, parse, and return well-structured data. You'll then use your functions to obtain data from the API and do some exploratory data analysis. This is a group project (see the project 2 page for details).

Before You Get Going

The first step is for the first listed group member to create a github repo and add the second listed group member as a collaborator. The second group member then needs to accept the membership request. This gives everyone access to push changes up to the same repository. All project work should be done within this repo. There should be multiple commits from each group member (this will be used to determine any group particpation issues).

This repo should be connected with R Studio (see the 'Git, Github, & RStudio video). Each time you go to work on the project, you should pull down any of the latest changes using `git pull` (via menu or terminal). You should then upload any changes you've made via the workflow described in that video. There may occasionally be merge conflicts that have to be dealt with. This can be done with the Git tab in RStudio. Let us know if you are having issues with conflicts that you can't resolve!

Other than when you are creating the repo and getting it linked up to RStudio, your repo should remain private until the day the project is due. You can of course make it public to check how the site looks and all of that. Just don't leave it public for an extended period of time until the due date has passed!

Other Repo Things

On your project repo you should go into the settings and enable github pages (feel free to select a theme too!). This will make it so your repo can be accessed like your blog (username.github.io/repo-name). Be sure to choose the master or main branch as the one to use if you have choices there. You can't do this unless your repo is public. Feel free to make it public to set this up and check that it looks fine.

When you knit your .Rmd file, use the output type `github_document`. This will create a .md file which will automatically be rendered by github when used appropriately. You should write code using the `rmarkdown::render` function to output your .Rmd file to a file called `README.md` rather than using the menus to output the file. This `README.md` file you create from R Markdown should be placed in the top level folder of your repo. Github pages will then create an HTML file from it using the theme you selected. (To reiterate, you do not need to output an HTML file yourself. You just need to create the .md file using the `github_document` output style. By naming it `README.md` it will then be converted into an HTML file that will act as the 'landing' page for your username.github.io/repo-name site.)

Code to create the `README.md` file should be included in your repo in a separate R script (`.R` file).

Note: You can make sure all of the above works before beginning on the project using the template .Rmd file they give you when you create a new R Markdown document. We recommend getting everything set up and then working through the project!

Vignette Content Details

You are going to create a vignette for reading and summarizing data from one of the APIs listed below. Please check the required components below before choosing your API and endpoints to return. Recall you will need to do summarizations as noted below. You must be returning at least some data that will work for the required summarizations.

- Financial data: https://polygon.io/docs/getting-started
- NASA data: https://api.nasa.gov/index.html
  - There are a lot of APIs here. Some easier to deal with than others.
- Beer data: https://www.openbrewerydb.org/documentation
- Pokemon data: https://pokeapi.co/
- Marvel comics data (I'm not sure how much numeric data is here, but there may be some): https://developer.marvel.com/docs
- Movie data: http://www.omdbapi.com/
- Food data: https://spoonacular.com/food-api/docs

The following components must be present in your vignette:

- You should have a section that notes the required packages needed to run the code to create your vignette near the top of your vignette
- You should write function(s) to contact your chosen API and return **well-formatted, parsed data in the form of data frame(s)**.
  - Your function(s) should allow the user to customize their query to return specific data.
  - You do not need to allow the user to query all parts of the API. Use should allow for connection to multiple end points and/or multiple modifications on a single end point. In total, your functions should be able to contact at least six endpoints or modifications (can be two endpoints with 5 modifications on one or three end points with two modifications on each, etc.)
    - For instance, if there is a date modification you allow the user to use with your function, that would be one modification of that endpoint.
  - The function(s) should be user friendly. That is, it should be easy to specify the options. For example, the ticker types for the financial API has options you can specify such as:
    - ADR (American Depository Receipt)
    - CEF (Closed-End Fund)
    - CS (Common stock)

  The user shouldn't be required to use the abbreviation and should be able to use the quoted string for use-ability. (You'd want to give the user the option of specifying the abbreviation or the quoted string. For the string, you may want to check the string after converting it to all lower-case and then map it to the abbreviation for querying).
- Once you have the functions to query the data, you should perform a basic exploratory data analysis (EDA). Not all things reported need to show something interesting or meaningful (i.e. graphs that show no relationship are fine) but each graph should make sense to look at and **each graph should be discussed. There should be a narrative throughout your EDA!**
- A few requirements about your EDA are below:
  - You should pull data from at least two calls to your obtaining data function (possibly combining them into one)

  - You should create at least one new variable that is a function of other variables
  - You should create some contingency tables
  - You should create numerical summaries for some quantitative variables at each setting of some of your categorical variables
  - You should create at least five plots utilizing coloring, grouping, etc. All plots should have nice labels and titles
    * You should have at least one bar plot, one histogram, one box plot, and one scatter plot

- Your code chunks should be shown in the final document unless they are set up chunks or other behind the scenes things that aren't important.

Submission

Once your group has completed their vignette, each group member should write a brief blog post on their own github.io blog site. The blog post should:

- explain what you did in the project and any interesting findings
- reflect on the process you went through for this project. Discuss things like:
  - what was the most difficult part of the logic and programming for you?
  - what would you do differently in approaching a similar project in the future?
- **provide a link to the rendered github pages repo and the regular repo (non-github pages site) as well!**

The URL to this (rendered) blog post is what each individual will submit at the project assignment link.

Rubric for Grading (total = 100 points)

Item	Points	Notes
Use of proper markdown things such as headings, chunk options, links, etc.	8	Worth either 0, 4, or 8
Required packages list	3	Worth either 0 or 3
Functions to query endpoints	28	Worth either 0, 7, 14, 21, or 28
Creation of relevant new variables	8	Worth either 0, 4, or 8
Contingency tables	6	Worth either 0, 3, or 6
Numerical summaries across variables	10	Worth either 0, 3, 7, or 10
Plots	25	Worth either 0, 5, 10, . . . , or 25
Blog post and repo setup	12	Worth either 0, 4, 8, or 12

Notes on grading:

- For each item in the rubric, your grade will be lowered one level for each each error (syntax, logical, or other) in the code and for each required item that is missing or lacking a description.
- If your work was not completed and documented using your github repo you will lose up to 50 points.
- If your github pages setup doesn't work or doesn't render properly, you will lose up to 25 points.
- You should use Good Programming Practices when coding (see wolfware). If you do not follow GPP you can lose up to 50 points on the project.
- If there are any group related issues, the commit history and any emails to Dr. Post will be used to determine any possible penalties (up to full credit possible).
- If you don't have a narrative or the narrative is lacking you may lose up to 30 points.