Project 1

Overall Goal

Vignettes are explanations of some concept, package, etc. with text, code, and output interweaved. Here are a few examples of varying quality:

- Basic ggplot
- 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 functions you've created to read in some data from the web along with use cases and some exploratory data analysis using the data.

Before You Get Going

The first step is to create a github repo. All project work should be done within this repo so we can track your activity. You should have at least five solid commits on the repo.

Ideally, you have connected RStudio with github and can work from the command line within RStudio as done in the 'RStudio & Github Workflow' video.

Other than when you are creating your repo and getting it linked up to RStudio, your repo should remain private until the day the project is due.

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.

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 `knitr::render` function to output your .Rmd file to a file called `README.md` rather than using the menus to output the file. This file should be placed in the top level folder of your repo.

Code to create the README file should be included in your repo in a separate R script (.R file). (Including this in a code chunk of your .Rmd file can cause issues!)

Blog

Once you've completed your vignette you should write a brief blog post:

- explaining what you did in the project and any interesting finding
- You should also 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?
- **In your blog post, provide a link to your github pages repo and the usual repo as well!** The URL to this blog post is what you will submit at the project assignment link.

Vignette Content Details

You are going to create a vignette for reading and summarizing data from the National Hockey League's (NHL) API. Some documentation for the API is given in the instructions below. The following components must be present in your vignette:

- You should have a table of contents
- You should have a way of reaching your github repo from your github pages document (many themes provide this naturally, if not, make sure a link exists)
- 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 functions to contact the NHL records API for the endpoints listed below. The functions should return well-formatted, parsed data (usually a data frame). Where possible, the user should have the option to specify the franchise of choice by both name and ID number - you'll need to map the names to ID numbers yourself.
  - /franchise (Returns id, firstSeasonId and lastSeasonId and name of every team in the history of the NHL)
  - /franchise-team-totals (Returns Total stats for every franchise (ex roadTies, roadWins, etc))
  - /site/api/franchise-season-records?cayenneExp=franchiseId=ID (Drill-down into season records for a specific franchise)
  - /franchise-goalie-records?cayenneExp=franchiseId=ID (Goalie records for the specified franchise)
  - /franchise-skater-records?cayenneExp=franchiseId=ID (Skater records, same interaction as goalie endpoint)
  - /site/api/franchise-detail?cayenneExp=mostRecentTeamId=ID (Admin history and retired numbers)
- You should write a function to contact the NHL stats API for the ?expand=team.stats modifier. The function should be able to take a single team or return data from all teams.
- You should write a wrapper function that is essentially a one-stop-shop for the user to access any of the API endpoints you did above. That is, this function should simply call the appropriate endpoint as per the users request (including any modifiers, teamIDs, etc.)
  - Note: This article may help you with contacting the API.
- 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. I know many of you don't know hockey and that is ok! Just do your best and ask if you aren't sure what something means. A few requirements about your EDA are below:
  - You should data from at least two endpoints (possibly combining them into one)
  - You should create at least two new variables that are functions of the variables from a data set you use
  - 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

In the project submission, you should simply put a link to your blog post. (We'll be able to navigate to your repo and github pages version via the blog post).

Rubric for Grading (total = 100 points)

Item	Points	Notes
Use of proper markdown things such as headings, table of contents, chunk options, links, etc.	8	Worth either 0, 2, 4, 6, or 8
Required packages list	3	Worth either 0 or 3
Functions to query endpoints	24	Worth either 0, 4, 8, . . . , 24
Wrapper function	8	Worth either 0, 2, 4, 6, or 8
Creation of relevant new variables	6	Worth either 0, 3, or 6
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	10	Worth either 0, 3, 7, or 10

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 on the project.

- 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.