Final Project

Creating a Shiny App

The goal of this project is to create a nice looking shiny app that can be used to explore data and model it! You'll get to choose your own data set.

Project Work

This is a solo project. 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 substantial commits on the repo or else you will lose credit.

You should have RStudio connected to github so you can work from the Git menu or command line within RStudio.

We will run your app from gitHub using `shiny::runGitHub()`. You should check that this works with an "empty" version of RStudio (that is, one that doesn't have objects you've already created existing in your environment - empty your environment!). You want to make sure anyone can run the app using just the code on the repo (and having the appropriate R packages on their computer)!

Your `README.md` file should have the following things (you can create this in R markdown or just update the .md file that you create with the repo):

- Brief description of the app and its purpose.
- A list of packages needed to run the app.
- A line of code that would install all the packages used (so we can easily grab that and run it prior to running your app).
- The `shiny::runGitHub()` code that we can copy and paste into RStudio to run your app.

You do not need to use github pages for this project (unless you want to).

Find a data set you are interested in

For this project I'm going to let you choose your own data set. You can either pull in a data set via a file, grab it from an API, scrape the data as the app runs, or pull it in from a database. You cannot use the data from project 3. If you want to incorporate your API material from project 2, that is fine (unless you used COVID related data - I don't want to read another COVID project)!

Choose something you are interested in and have ideas for investigating!

App Requirements

Your app should have multiple pages (tabs) to it. I don't care if you use the built in tabs for shiny or a package like shinydashboard - use the method you prefer. Tab information:

- An `About` tab The page should
  - Describe the purpose of the app
  - Briefly discuss the data and its source - providing a link to more information about the data
  - Tell the user the purpose of each tab (page) of the app
  - Include a picture related to the data (for instance, if the data was about the world wildlife fund, you might include a picture of their logo)

- A `Data Exploration` tab This should allow the user to create numerical and graphical summaries (at least four different kinds of plots).
  - The user should be able to change the type of plot shown and type of summary reported
  - The user should be able to change the variable(s) being looked at and filter the rows to change the data used in the plots/summaries
  - You should have plots that (at least have the option to) show multivariate relationships. For instance, coloring points on a scatter plot or creating side-by-side bar plots faceted across another variable (three variable relationships). **The user should be able to manipulate some of these things!**
- A `Modeling` tab You will fit two types of supervised learning models depending on your response type:
  - A multiple linear regression or generalized linear regression model
  - A random forest model
  - You should have three subtabs on this modeling tab:
    - A `Modeling Info` tab where you explain what the two modeling approaches are and the benefits/drawbacks of each. You should include some type of math type in the explanation (you'll need to include mathJax).
    - A `Model Fitting` tab where you let the user choose:
      - A percentage for a test/train split
      - The ability to choose the predictor variables for each model (they should be able to specify different predictors for each model)
      - For the random forest model, you should give the user the ability to specify the tuning parameter grid and the CV settings
      - When the user is ready to fit the models, they should be able to press a button and fit both models on the training data.
      - Fit statistics (such as RMSE) on the training data should be reported for each model along with appropriate summaries about the model (for instance, `summary()` run on your `lm()` or `glm()` fit, a plot showing the variable importance from the random forest model, etc.).
      - Appropriate comparison statistics should be produced for performance on the test set as well
    - A `Prediction` tab where you give the user the ability to use both of your models for a prediction. That is, they should be able to select the values of the predictors in the models and obtain predictions for the response (one for each model).

Submission

You should simply post your github repo URL as your submission.

Notes

- There are some nice shiny additions such as a status bar that you can add to display to the user that your model is running.
- `aes_string()` is useful when trying to use columns selected by the user in a `ggplot()` plot since the columns are often returned as a string
- Similarly, `!!sym(input$...)` can also be useful to deal with this type of issue generally
- `get()` comes in handy some times as well
- This project is pretty open ended! Have fun with it and make something that you can show off to others - Good luck :)

Rubric for Grading (total = 100 points)

Item	Points	Notes
Repo setup/requirements	15	Worth either 0, 5, 10, or 15
Tab structure	10	Worth either 0, 5, or 10
About tab	10	Worth either 0, 5, or 10
Data exploration tab	30	Worth either 0, 5, . . . or 30
Modeling tab	35	Worth either 0, 5, . . ., or 35

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 50 points on the project.**
- **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.**