Project 3 (Final)

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.

Project Work

All project work should be done in a github repo. Ideally, you have connected RStudio with github and can work from the command line within RStudio. All major updates should be made through github so we can track your activity.

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). You want to make sure anyone can run the app using the code!

Your README.md file should have the following things:

- 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.

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.
  - An `About` page. 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` page. The user should be able to
    - Scroll through the data set
    - Subset this data set (rows and/or columns) they can scroll through
    - Save the (possibly subsetted) data as a file (.csv is fine but whatever you'd like)
  - A `Data Exploration` page. This should allow the user to
    - Create numerical and graphical summaries. The plots should be downloadable with a button, at least one plot should utilize mouse input (hover, click to zoom, etc. - see the `plotly` package if you'd like, but this isn't required)
    - Change the type of plot and type of summary reported
    - Change the variables and filter the rows to change the data in the plots/summaries

- A `Modeling` page. You will fit three supervised learning models. Depending on your response you'll fit a multiple linear regression or generalized linear regression model, regression or classification tree, and a random forest model. This page should have three tabs to it.
    - `Modeling Info` tab: You should explain these three modeling approaches, the benefits of each, and the drawbacks of each. You should include some type of math type in the explanation (you'll need to include mathJax).
    - `Model Fitting` tab:
        - You'll split your data into a training and test set. Give the user the ability to choose the proportion of data used in each.
        - The user should have functionality for choosing model settings for each model. For all models, they should be able to select the variables used. Cross validation should be used for selecting models where appropriate.
        - When the user is ready they should be able to press a button and fit all three 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).
        - The models should be compared on the test set and appropriate fit statistics reported.
    - `Prediction` tab: You should give the user a way to use one of the models for prediction. That is, they should be able to select the values of the predictors and obtain a prediction for the response.
- You should have at least two different dynamic UI elements.

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.
- Remember that most people that have apps make their code freely available (often on gitHub). One thing you might do is search through apps to find someone that say has the option to download the data set. Check their code on gitHub to see how they did it and if it makes sense for you. The openness of the R community is really one of the great benefits of using R!
- `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
- 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
Tabs	5	Worth either 0 or 5
About page	10	Worth either 0, 3, 7, or 10
Data page	10	Worth either 0, 3, 7, or 10
Data exploration page	20	Worth either 0, 5, ... or 25
Modeling page	30	Worth either 0, 5, ..., or 30
Dynamic UI elements	10	Worth either 0, 5, 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 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 25 points on the project.**