Assignment 04: Data Visualization, Quarto, and Git

Data Science for Public Policy

Aaron R. Williams and Alena Stern - Georgetown University

Deliverable: There are three deliverables to submit for this assignment:

1. index.html
2. the .qmd file with your R code
3. the URL of the GitHub repository

You may need to place the .qmd and .html file into a folder, and compress (or zip) that folder, in order to submit it to Canvas.

Points: 9 or 14 points

Lab partner: Each student can pick one lab partner from their section and submit one assignment between the two of them. Lab partners are expected to complete every exercise together and should spend time together in-person or on Zoom. Dividing and conquering is not an acceptable approach.

Plagiarism on homework or projects will be dealt with to the full extent allowed by Georgetown policy (see http://honorcouncil.georgetown.edu).

Setup

Read the entire assignment before beginning the assignment. Note that the stretch exercise requires a partner and is easier to complete while working on the normal assignment than after submitting the assignment.

Go to GitHub.com and log into your account. Click on the green button that says "New" to create a new GitHub repository and a create a repository called assignment04. Clone your repository to your local computer using `git clone`. Use `cd` to change your working directory to assignment04.

In GitHub, go to Settings > Collaborators and add your lab partner as a collaborator.

Create an RStudio project like in prior assignments. Next, create a new Quarto file (.qmd) in that new folder called index.qmd. Next, create a file called README.md and a title and authors in the README.

Please note: Setting up and using Git and GitHub for the first time can be a challenge. I strongly recommend you start this assignment very early and come to office hours or otherwise reach out with any issues.

Assignment Description

This assignment differs from previous assignments, in that you are tasked with finding and analyzing a dataset of your own choosing. This will result in you creating and submitting a single .html file, containing your code, a series of four visualizations made with ggplot2, and a brief discussion of what those visualizations mean.

First, find a public dataset available from the web, relevant to one of your policy interests. Add this data to a folder called 'data' in the same folder as your markdown file. You should also create a .gitignore file and add the file suffix of your data (like .csv, .xlsx, or .sas7bdat) to the .gitignore, so git will ignore your data. You can list the full file name to ignore individual files or use code like `*.csv` to ignore all .csvs. Properly ignored files will not show up in `git status`.

Then analyze the data, storing all your code and writing in your .qmd file, following the requirements described below. There are three graded components of this analysis (Read all instructions before starting, as you must use git as you create the graphs).

1. Git Repo and Commits (3 Points)

Use `git` thoroughly for this assignment. This means, at a minimum, you must commit when you first start the project and after the completion of each graph (a bare minimum of five commits). When you submit the assignment, all commits must have been pushed to GitHub. This means there will be a record of the commits on the public GitHub repository. You must submit the GitHub URL as part of the assignment, and will be graded on having committed your code as you worked on the assignment.

2. Quarto document (1 points)

Create and clearly format an Quarto document for this analysis. Specifically:

- Include a title and authors within the opening YAML;
- Use a hyperlink to link to the source of your data;
- Use appropriate headers to signify each visualization;
- Include code chunks for data manipulation and visualization so I can understand the code and analysis you ran to create the graphs;
- Hide warning messages and unnecessary printing of data.

3. Four ggplot2 graphs (4 points)

Your analysis should have four data visualizations of distinct graph types, made with ggplot2. Across all four graphs, use a total of:

- Six different aesthetics (i.e. set inside `aes()`);
- Six different non-aesthetic options (i.e. set outside `aes()`);
- Five different geoms;
- Two different scales (meaning change the default scale used for at least two aesthetics).

Further, each graph must include:

- Correct usage of all visual encodings;
- Appropriate data sourcing (hint, check out the `labs()` function);
- Proper labeling of all visual encodings;
- An appropriate title and subtitle;
- The code you used to generate each graphic, right above the graph (Quarto should make this easy).

The data visualizations should be clear and polished enough to go in a report. Checkout fivethirtyeight or the Urban Institute for inspiration.

4. Written narrative & interpretation of visualizations (3 Points)

Write at least three to five sentences about each graph in your document, describing what it says and how it informs relevant policy topics. You should (to the extent possible) write this as a narrative that ties together all of visualizations. You will be graded on the interpretation of the visualizations, and their relevance to the policy topic.

It isn't required, but consider learning more about figures and cross references to improve your Quarto document.

Submission

Upon completion of the assignment, render the .qmd file to .html, and submit both, along with the URL of your GitHub Repository.

Tips

- Quarto can be challenging at first. Render early and render often.
- Git/GitHub can be challenging at first. Commit early and commit often.
- If you forget `-m` with `git commit`, use this highly viewed Stack Overflow post to escape VIM.

Stretch (5 points)

This stretch exercise asks you to develop and demonstrate additional Git and GitHub skills that will prove valuable for collaboration.

Part 1: .gitignore (1 point)

A gitignore file specifies intentionally untracked files that Git should ignore. Files that you add to the gitignore file cannot be pushed to GitHub. This is useful to avoid accidentally pushing files that you don't want to GitHub - like data files or files storing passwords and other credentials.

1. Create a new text file by selecting `File -> New File -> Text File` in the top menu.
2. Save the file as `.gitignore` in the root directory of your repo (this is the same folder your RStudio project is located in). You may get a message that "Names that begin with a dot "." are reserved for the system." - click 'Use "." ' to save the file.
3. GitHub has a set of gitignore templates for different programming languages that are prepopulated with common files to add to gitignore. Copy the contents of the R gitignore template into your `.gitignore` file. Add the file path to the data file you used for assignment 04 to your gitignore file and save the file.
4. Try to `git add` your data file. What message do you receive in the terminal?
5. Push your `.gitignore` file to GitHub.

Part 2: GitHub issues (1 point)

GitHub issues are useful for managing to-dos. Open a GitHub issue for each of the four data visualizations before coding up your data visualizations. This only requires pointing and clicking in the web browser (instructions). You can also close issues in commit messages.

Part 3: Branches (2 points)

Branching is useful for collaborating because it allows for parallel development and avoids conflicts between programmers. The final figure in the Reproducible Research with Git notes documents this workflow.

1. Create a Git branch for each lab partner, for example `aaron` and `alena`. To do this, submit something similar to `git checkout -b aaron` to the command line. Then submit `git branch` to see all branches.
2. Make changes to your code. Then add and commit those changes. From time-to-time, push those changes to GitHub with something similar to `git push origin aaron`. You should now see multiple branches on GitHub.
3. To get your changes from your branch (e.g. `aaron`) to the `main` branch, open a pull request on GitHub and flag your partner for review.
4. Your partner should review the pull request and request changes or approve. After approval, merge the pull request. (merge is a big green button)
5. Be sure to run `git pull origin main` at the command line to pull remote changes from your partner to your branch (i.e. from branch `main` to branch `aaron`).
6. Close out GitHub issues when you complete exercises.
7. Repeat this workflow throughout the assignment.

The grader will review your commit history and pull request history. There should be at least one pull request from each branch. Saving these steps until the end will result in a reduction of points.

Part 4: GitHub pages and plot annotation (1 point)

GitHub pages offers free web hosting.

1. Call your .qmd and rendered .html document index.qmd and index.html.
2. In your GitHub repo, go to Settings > Pages.
3. Set the source to `main` and click save. You now have a website!
4. Read the section of the Urban Institute's R Data Viz Guide. Use `geom_text()`, `geom_text_repel()` or `annotate()` to add text to one of your plots from assignment 04 that enhances the clarity or interpretation of the plot. Push your updated `index.qmd` and `index.html` files to GitHub. Go to your GitHub page and observe that your changes are automatically reflected in your page (note that it may take a moment for the GitHub to deploy the latest changes).
5. Add the URL to README.md and make sure the updated README is on your main branch on GitHub.