CIS 4930/6930-002: Data Visualization (Spring 2019)

Project 2: Drawing Basic Charts in Processing

(Adapted from an assignment created by Carlos Scheidegger)

1 Objectives

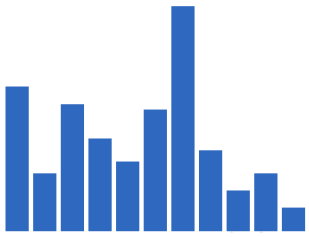
In this assignment you will learn the basics of Processing, and create a few Processing sketches that implement basic visualizations. These will be building blocks for future assignments, so take care to use good software engineering practices. You will create 3 sketches.

2 Ground Rules

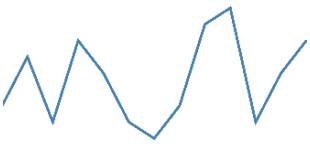
This assignment is intended to be done alone. You may ask others for help with figuring out how details of Processing. However, code must be your own (MOSS will be used!). Furthermore, NO additional libraries (such as giCentre utilities) may be used. Doing so will result in a 0 for those sketches.

3 Assignment Instructions

- Download and familiarize yourself with Processing (http://processing.org/download/). Use the available tutorials (http://processing.org/tutorials/) and examples (http://processing.org/examples/) to help you understand how Processing works.
- Create a data file. Your data file should be a csv format file (http://www.computerhope.com/issues/ch001356.htm). The file should contain the following headers {YEAR, VALUE0, VALUE1, PARTY} and values $\Big[$ {1992, 7.30, 88, DEM}, {1996, 5.60, 43, DEM}, {2000, 4.00, 76, REP}, {2004, 5.70, 90, REP}, {2008, 5.00, 12, DEM}, {2012, 8.30, 14, DEM}, {2016, 4.90, 50, REP}$\Big]$.
- Create a sketch with 600x600 resolution that opens a file dialog box (http://processing.org/reference/selectInput_.html) and loads a selected csv data file. You can start with the skeleton code provided on Canvas.
- With the given data file, draw a BAR CHART for the data (you get to pick which variables to use).



- Create a 2nd sketch (same resolution and dialog box), and draw a LINE CHART for the data (again you get to pick which variables to use).



- Modify your sketches such that they use additional visual channels to encode additional variables. Consider using color, size, shape, depth, etc. Your selection and their implementation will have an impact on your grade.
- Add embellishments of your choice. These can include but are not limited to: axis lines, labels, and tick marks. Consider the margins for your embellishments (try to pick good values for the tick marks and a good number of them—not too many and not too few). Your selection and their implementation will have an impact on your grade.
- Make your visualizations robust by designing them to support any data (number of elements or value range) and by designing them to support any size or aspect ratio of canvas. You will reuse this code in future assignments, so functionalities will be particularly important.

4 Submission

We will be using a git repository for submissions throughout the semester. A script will automatically scrape your solution at the deadline. **Please follow these instructions carefully.** If done incorrectly, we won't get your submission, and you will get a 0.

- **Install git** on your local machine.
  - For Windows, I recommend TortoiseGit (https://tortoisegit.org/). You will also need the Git for Windows tools (https://git-scm.com/download/win).
  - For Mac, install xcode (https://developer.apple.com/xcode/) and then the command-line tools (http://railsapps.github.io/xcode-command-line-tools.html).
  - For Linux, git should already be installed. If not, use the appropriate package manager (e.g. apt or yum) to install it.
- **Create Account:** Create a GitHub account, if you don?t already have one (https://github.com/). A GitHub Education account is optional (https://education.github.com/discount_requests/new).
- **Setup Repository:** Visit ¡https://classroom.github.com/a/0JcYQJsF¿ to setup your repository with the skeleton code already added. It is important that you use this url. It attached your repository to our class.
- **Clone the repository:** Once the repository is created (this can take a few minutes) determine the remote path and pick a local directory for code. Clone the project locally (i.e. *git clone* <*url_to_repository*> <*path_to_local_directory*>).
- **Edit your code:** Use existing **project2** directory for storing/editing your code. If you put it anywhere else, our scripts will fail (and so will you). As you work, make sure you add the files to the repository (i.e. *git add*), commit the changes (i.e. *git commit*), and push changes to the remote server (i.e. *git push*). Be sure to frequently commit and push your work!
- **Submission:** We will pull from the last commit before the deadline. Make sure to push your code one last time before you're done. If you fail to do this, we won't get your files.
- **Verify:** You can verify that your files have been properly uploaded by checking the Code page on the GitHub website.

5 Grading and Feedback

Your grade will be combination of objective measures (based on the assignment instructions) and subjective grading by the instructor.

Peer Review will be used to provide feedback. You will review 3 of your peers' submissions, and 3 of your peers will review your work. This should be taken very seriously as it is the primary form of feedback you'll receive.

- Your grade will be combination of objective measures (based on the assignment instructions) and subjective grading by the instructor.
- Breakdown
  - 3 Visualizations - 6 points (basic implementation 2 points each)
  - Additional Visual Channels - 1.5 points
    - 0.5 points for none used
    - 1.0 point for a few
    - 1.5 points for many
  - Embellishment - 1.5 points
    - 0.5 points for none used
    - 1.0 point for some
    - 1.5 points for a lot
  - Code Professionalism - 1 points
    - 0.5 no comments, no classes, "hard coded" values
    - 0.75 minimally commented, few "hard coded" values
    - 1.0 commented, properly used classes, few "hard coded" values
- Peer Review will be used to provide feedback. You will review 3 of your peers' submissions, and 3 of your peers will review your work. This should be taken very seriously as it is the primary form of feedback you'll receive.