CS 638: Data Science in the City of Madison

Website: https://wisc-ds-projects.github.io/f19/

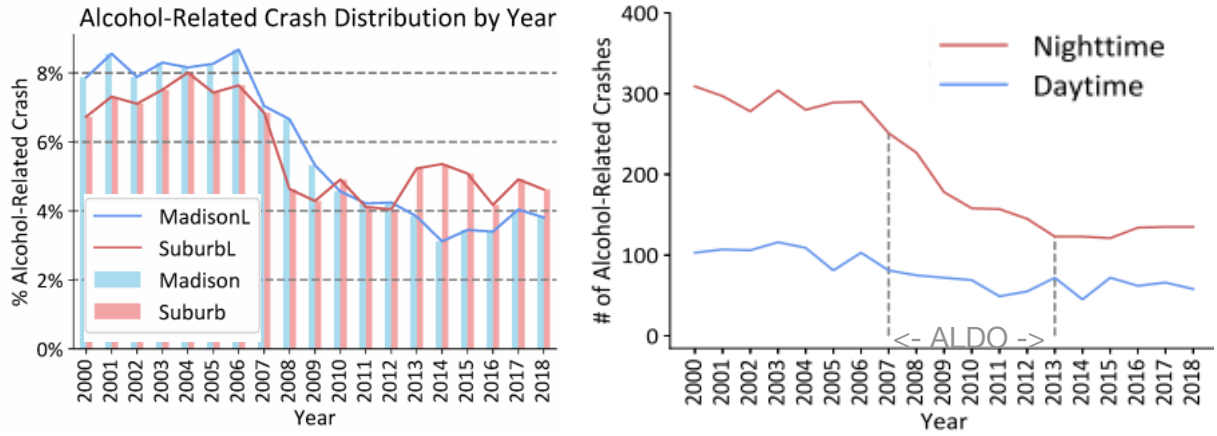
Email: compsci638-2-f19@lists.wisc.edu (feel free to send here; introduce yourself this week)

Optional text: *The Visual Display of Quantitative Information* by Edward R. Tufte

Note: 2-credits, letter grades, doesn't count for CS elective

Logistics:

- I won't "teach" much per se, but I'll give detailed feedback on your work
- Form teams of 2-4, choose topic (due by end of the weekend!)
- Every week, each person emails me Jupyter notebook with 2-3 plots (starts next Thu)
- Each plot should preceded with question and followed by conclusion (in notebook)
- If team is called up, present plots to the class (buy/bring adapter for the projector)



What makes a good plot? (above examples adapted from student plots)

- Obvious stuff: legible font, labeled axes, correct data (use common sense!)
- Lots of **context** (annotations, interesting subcategorizations); never show a single line or set of bars. Rule of thumb: maximize how many numbers are represented (https://trailsofwind.figures.cc/)
- Audience is **intelligent, but busy.** Minimize cognitive load w/ consistent line/legend order
- Minimize non-data ink: remove gridlines, top/right borders, box around legend (me **minimalist**)
- **Design before code**: think about how it should look, then write necessary code to make it. Don't start with the things that are easy to do with matplotlib and force your data to conform
- Interesting data: you don't know until you plot it. You should make MANY plots each week, then **only show us the best of the best**. Be ready to verbally summarize other results.
- Communicate about the plot verbally and with text. What are you asking? What can you conclude? What new questions arise? What is your **methodology**; what are threats to validity?
- Narrative: the plots should be sequenced in a meaningful way to **tell a story**
- **Printer friendly**: colors are OK, but only if they work if turned to gray

Real data is not like "toy" data used in classes

- Missing, different people collect different ways, practice change over time, incorrect entries
- Is dirty data better than no data? Data quality can itself be an object of study (e.g., heartbeat data).
- Lacking data should never stop you (conduct a survey, https://www.reddit.com/r/madisonwi.json)

Evaluation

- I'll "grade" each plot on a 0-5 scale, roughly as follows. **0:** no plot; **1:** plot hard to read or has mistakes; **2:** data is "thin" and lacks context; **3:** well-designed data-dense plot; **4:** the plot is used to make a convincing argument; **5:** the plot is so compelling I'll want to share it with others. Other factors include the methodology, communication, how will it fits with other plots of the team, etc.
- **Quality over quantity**: can you produce a visualization that everybody in Madison will want to see and talk about? https://paulbutler.org/2010/visualizing-facebook-friends/
- If your final presentation is amazing, I'll just you solely on that, regardless of the plot grades throughout the semester.

Version Control

What is version control?

- imagine 100 people working on the same program. How do they collaborate?
- answer: version control tools. We what to keep track of what each person wrote, when they wrote it, and what feedback received. We want a history so that we can go back in time if somebody messing things up.
- there are many version control tools: svn, mercurial, git, others. We'll use git.
- git is open-source, and there are many providers. We'll use github.

Git vocabulary

- commit: a checkpoint of the code in a certain state. "git commit" command creates a new commit
- branch: an easy-to-remember name for a commit (automatically follows latest)
- repo: collection of commits/branches for a given project
- clone: git command to make a copy of of a repo
- pull: git command to pull new commits from another repo to your own
- push: git command to push new commits from your own repo to another
- pull request: suggestion that the people managing a primary repo pull commits you made in your repo into the main codebase
- remote: local name for another repo

Practice

- do the introduction sequence on the git visualizer: https://learngitbranching.js.org/
- create a github account
- install git on your laptop
- create a team repo and fork it
- clone the same repo to your laptop
- add your fork as a remote
- push a commit directly to the main repo
- in another commit, make a change to your file
- push this second commit to your fork
- do a pull request
- merge another student's pull request

Note: don't put private datasets on github!!!