Reproducible and Collaborative Statistical Data Science

- Course numbers: **Stat 159**
- Truncated title: **REPRO COLLAB DATSCI**
- Units: **4**
- Semester: **Fall 2015**

Overview and Course Description

This is a project-based course that introduces you to reproducible and collaborative statistical research, applied to a real scientific question. You will gain experience acquiring, cleaning, and curating data; formulating scientific questions statistically; developing appropriate statistical methods to analyze the data to answer the scientific questions; implementing those methods in robust, testable, reusable, extensible software; applying the methods; visualizing the results; interpreting the results; and communicating the results to others. And you will learn to do this in a way that is computationally reproducible, which is increasingly recognized as key to scientific progress.

As if you were working in a scientific research group or in industry, you will be given a set of tools (a "software stack") and practices you are required to master and use. As if you were working in a scientific research group or in industry, you will be pointed to resources to help you learn the tools, and it is your responsibility to ensure that you have mastered them, which will require a substantial amount of time outside class. And, as if you were working in a scientific research group or in industry, you will be required to collaborate and to take responsibility for your role in the collaboration.

But the point is not merely to master the tools or to learn to collaborate effectively. Rather, the point is to do sound computational science, and to do it in a way that others can verify the data and statistical techniques behind your analysis, can verify that your code does what it's supposed to do, can run the code using the data and parameters you used, can verify that they get the same results you do, and can build on what you have done to advance science even further.

Format

3 hours of class per week in two 90-minute sessions, plus two hours of labwork. Class sessions are divided between lecture and collaborative work. To a large extent, the class is "flipped," meaning that you are responsible for watching video lectures and working through online tutorial materials—outside class—as well as collaborating with your team outside class, so that a substantial amount

of class time can be devoted to group work on the project, including frequent in-class presentations by students, code reviews, brainstorming, etc. Students are expected to work at least 8 hours per week outside class, making this a 4-unit course (13 hours of effort per week).

Prerequisites

Statistics 133, Statistics 134, and Statistics 135 (or equivalent).

Grading

10% quizzes, 20% labs, 30% homework, and 40% group project

For the first half of the course, there will be a number of homeworks to ensure the students are mastering the basic computational skills necessary to complete their final group projects. The group projects will be the primary focus for the last half of the semester. There will be regular milestones that will involve code submissions, code review, and short presentations.

Starting in week 8, the pattern of class will alternate weekly between producing and presenting and checking. There will be a week of science, code development, and occasional new material, followed by a week in which students make short presentations about what their team did on Tuesday, and then a cross-team code review and check for reproducibility on Thursday.

The scientific question and primary data will be provided to the students taking the undergraduate version of this course. The final deliverable for the group project will involve a poster session and project report.

All work will be done using the version control system Git. The Git repository is not merely a mechanism for submitting work, but the way you will work with code and text throughout the course. You will be expected to become a Git expert by the end of the course. You are expected to make consistent contributions. Procrastination is not rewarded.

Textbook

Readings will be assigned weekly and will mostly consist of articles and tutorials. Examples of the assigned articles are listed below. Assigned tutorials will vary per semester, but there will be an emphasis on using official project documentation.

Syllabus

Week 1

Introduction and course overview. What is the difference between reproducibility, replicability, verifiability, and auditability? Why is reproducibility important to the scientific method?

Overview of toolstack for the course. Introduction to the command line interface, the filesystem hierarchy, and working with text files. Introduce version control with Git focusing on conceptual model and basic objects.

Reading

- Buckheit, Jonathan B., and David L. Donoho. Wavelab and reproducible research. Springer New York, 1995.
- Preeyanon, Likit, Alexis Black Pyrkosz, and C. Titus Brown. "Reproducible bioinformatics research for biologists." Implementing Reproducible Research (2014): 185.

Week 2

Introduction to Python and the scientific Python software stack. Python data structures and control flow. Basic NumPy, SciPy, and matplotlib.

Introduce the Duke/Potti cancer genomics scandal. Basic exploratory data analysis of the microarray data from the Duke/Potti scandal.

Reading

- Perez, Fernando, Brian E. Granger, and John D. Hunter. "Python: an ecosystem for scientific computing." Computing in Science & Engineering 13.2 (2011): 13-21.
- Baggerly, Keith A., and Kevin R. Coombes. "Deriving chemosensitivity from cell lines: Forensic bioinformatics and reproducible research in high-throughput biology." The Annals of Applied Statistics (2009): 1309-1334.

Week 3

Introduction to functional programming in Python including iterators, generators, list comprehension, zip, map, enumerate, etc. Unit testing in Python. More advanced Git. Continue working with the Duke/Potti cancer genomics data.

Reading

- Millman, K. Jarrod, and Fernando Pérez. "Developing Open-Source Scientific Practice." Implementing Reproducible Research (2014): 149.

Week 4

Additional Python packages for scientific programming including for example statsmodels, scikit learn, and scikit image.

Reading

- Python Scientific Lecture Notes

Week 5

Refresher on the scientific problems and the statistical issues. Estimation, prediction, hypothesis testing, confidence sets, robustness, nonparametrics, abstract permutation tests, Monte Carlo, pseudo-random number generation. Framing the scientific question as a quantitative statistical question. All theoretical topics will be explored from a practical perspective using Python.

Reading

- Lecture notes

Week 6

Introduction to the scientific problem for the semester (the substantive scientific question the class will address will vary by semester) and the data sources for the project. Example topics: recidivism in Federal prisons, earthquake prediction, effect of packstock use on the Yosemite Toad population. (Note: Graduate students will be expected to come up with their own scientific questions for the final project. But they will also be required to understand the problem that the undergraduates are assigned as it will serve as a case study for future lectures.)

Reading

- Reading will vary every semester depending on scientific question

Week 7

Introduction to project organization and process automation including: Python packages and setuptools; makefiles; markdown, reStructuredText, and LaTeX; and automated testing and continuous integration. Collaborative workflows with Git.

Reading

- The Hitchhiker's Guide to Packaging

Week 8

Discussion of bottlenecks. Refine scientific and statistical vision. Outline analysis strategy. Mini-lectures on statistical and computational tools needed to complete the project.

Reading

- Reading will vary every semester depending on scientific question

Week 9

Introduction to software architectures and design patterns. What is software architecture? What are design patterns? Overview of key principles of software architecture. Review several architectural patterns and styles including streams and filters, clients and servers, split-apply-combine, and map-reduce.

Reading

- Garlan, David, and Mary Shaw. "An introduction to software architecture." (1994).

Week 10

Discussion of bottlenecks. Refine scientific and statistical vision. Outline analysis strategy. Mini-lectures on statistical and computational tools needed to complete the project.

Week 11

Parallel and cloud computing. Brief overview of computer and network architectures.

Tuesday–present progress. Thursday–code review and reproducibility review.

Reading

- Lecture notes

Week 12

How to give good talks and posters; examples of bad talks and posters. The good, the bad, and the ugly of data visualization.

Week 13

Tuesday–present progress. Thursday–code review and reproducibility review.

Week 14

Tuesday–Lightning talks in class. Thursday–feedback on the talks.

Week 15

Tuesday–polish poster presentations. Thursday– 3-hour session of public poster presentations.