Scientific Computation for Social Scientists

Michael Colaresi
mcolaresi@pitt.edu

PS 2705
Fall 2020
Tuesday 1:45pm-4:10pm
5502 Sennott Square and Zoom via Canvas

1 Description

Social relations are incredibly complex, including dynamic, nonlinear, and interactive behaviors across multiple scales from neurons to nations. Yet, progress within the social sciences have been limited by the computational resources researchers feel comfortable utilizing and integrating into their research. While the use of aggregate measures at low resolution, and simple linear approximation may have been more defensible on feasibility grounds in the late 20th century; today there is the unique opportunity for applied researchers to use computational tools (both old and new) to increase the depth, breadth, and credibility of social science research as well as work in the interdisciplinary domain of computational social science.

This class aims to make a small contribution to narrowing this gap between the deep and shifting context and diversity that drew so many of us to social science in the first place and the abstract representations and tools that allow our community to understand, encapsulate, and communicate patterns in systematic and credible ways.

There are three guiding principles for this class.

- Computers are our co-workers, but not our replacements (yet). The computational and digital revolutions have fundamentally altered the information available to social scientists (big data), but also our ability to make inferences from heterogeneous, if sparse, data (big models). It is also crucial to remember that just because you see Big Data, that does not mean what you observe is representative or that the amount you are unable to see is not even bigger. With that in mind, computers can help us collect, organize, process, analyze, visualize, criticize, and contextualize a wide variety and spectrum of signals that we receive from the world around us. In order to accomplish these goals we need to understand theoretically how computers can be useful to the research enterprise as well as how to practically deploy that usefulness today (and tomorrow). Therefore this class aims to help you use computation to augment your science, not limit it to data rich topics.

- Research is a social activity. It is crucial for the integrity of research that other teams can replicate your work with minimal cost, this includes your future self. Further, you want others to collaborate and build on your work. There are significant obstacles to collaboration and replication that need careful thought and planning to overcome. The fact that computers are literal and can repeat the same instructions (almost) identically across researchers means that they can play an important role in cheaply mediating (if done correctly) scientific communication on complex topics. It is nearly impossible to directly tell another person all of the detail of how to replicate your research, but it is possible for you to tell your computer, and your computer to tell thousands of other researchers' computers exactly what do to. A corollary to this guiding principle is that open source tools are superior to proprietary and/or closed tools. Not every researcher with important insights to offer has access to a research account or expensive software licenses. Further, the software tools that have innovated most quickly over the last several decades have fostered open-source communities that together pushed new features and bug fixes.

- You will have to slow down now, in order to speed up and improve your workflow in the future. This class is about learning more efficient habits and workflows that leverage computation for social science research. Therefore, students will be asked, repeatedly, to avoid doing what they are comfortable with – what they are speedier at – in the here and now, in order to learn techniques that will lead to dramatic improvements in their replicability, impact, and efficiency in the not too distant future. These techniques will necessarily involve typing text and the command line (in contrast to a trackpad and GUI). Students need to be patient while working up these learning curves. Every time you do the easy, and inefficient (eg using your trackpad to create a directory or save a file), you miss an opportunity to practice and improve your skills at the harder but more efficient (and often more replicable) method.

2 Learning Objectives

This is a task-based class. Therefore the learning objectives involve organizational and computational tasks that have associated tools. Learning to use the tools should not be confused with learning why the tools are important and how to deploy them or other tools to improve your research.

1. Control a computer (either your own or a server) in ways that are reproducible in the future and to others. For this task we will mainly use the zsh (z-shell), which is mostly an enhancement of bash (the bourne-again shell) and make, a utility for defining dependencies between files. Students should be able to understand directory structures, understand the difference between absolute and relative paths, load a shell prompt, write shell scripts, set up a new project, define external and internal dependencies, copy an existing file or directory, create and organize files, and execute code in a pre-defined sequence, as well as communicate how others can repeat these steps with a low investment in time and resources.

2. Keep track of different versions of code and dependencies over time in a clear and organized way, that allows for collaboration. For this task we will mainly use git and github. Students should be able to understand the difference between a file and different versions of that file in a repository. They should be able to understand how to track changes, commit changes (with informative messages), use branches to explore adding new features to projects, how to diff files to understand changes between files, and how to track issues and resolutions when working as part of a team.

3. Solve complicated problems by breaking up challenges into related pieces. Students will use external functions, as well as their own, in addition to other abstractions (such as classes of objects with attributes and methods), to build pipelines and code bases that can be used across projects. Students will also understanding why type hinting and linting is important for building up code bases. For this set of tasks we will reply on Python, and in particular Python > 3.7.

4. Visualize vectors, data, simulations, and inferences using open source tools. For this task we will rely on matplotlib and altair. One benefit of moving from paper to pixels is the possibility of animations and user-initiated interaction. Students will be introduced to the tools and workflow for creating and sharing animations and interactive visualizations.

5. Understand the possibility of running jobs, when necessary, on computer clusters (often vaguely referred to at high performance computing centers/clusters). For this set of tasks, students will be introduced to ssh and scp as well as the the basics of the slurm scheduler. If there is time, students will also become familiar with different forms of parallelization.

6. If there is time, we will also curate versions of data and models. While the tools and ecosystems for code version control are mature and widely used (at least outside of the social sciences), versioning of data and model objects is still being developed. We will discuss the existing holes in reproducible workflows, as well as introduce dvc, built on top of git, as some emerging solution.

3 Class Format

Because of the ongoing COVID-19 situation most of the work for this class will be completed in a hybrid format. Students have the choice of engaging with the content solely online or coming to the physical class at the designated time, or some combination. Attendance will not be taken at the physical meeting.

Lectures and virtual walkthroughs will be posted online to the Canvas site for this class. There will be weekly assignments, based on these videos and the associated reading/material. As you will see below, these weekly assignments make up the majority of your grade.

During our scheduled class time, I will be available to work with students virtually. Students can use the classroom for internet access, or to use the white board for problem solving. This is a time to ask questions, share code with me and/or others, and run through

conceptual or software problems/questions. To be clear, I will not be physically in the room, but will be available by zoom during these times.

Since this is a hands-on class, it is vital that you work through all of the suggested problems in the lectures yourself, and not simply rely on my examples. Scientific computation is not a spectator sport.

In addition, it is likely that in your explorations related to this class, you will find useful material. Please feel free to share that material to me and I can share it with the class if it would be helpful to others.

4 Readings

The material I will be covering is largely taken from three books and other online tutorials and information. The online resources are listed under the specific weeks. The three books are listed here. Note that 2 are available for free online. The third can be purchased cheaply as a pdf.

- Bruce Shapiro. 2018. Scientific Computation: Python 3 Hacking for Math Junkies. CreateSpace Independent Press. Fourth Edition. [P3H4MJ]
- Frank T. Willmore, Eric Jankowski, Coray Colina (Editors). 2016. Introduction to Scientific and Technical Computing. CRC Press. First Edition. Available from the Pitt Library when logged in here. [I2STC]
- Joel Grus. 2019. Data Science from Scratch. O'Reilly Press. Second Edition. Available from the Pitt library when logged in here. [DSFS]

While there are a wide variety of books and online resources on both shell programming and python, one resource that has a decent depth that is often lacking in other treatments is Sandeep Nagar. 2018. Introduction to Python for Engineers and Scientists. APress, available from the Pitt library when logged in here.

5 Assignments

5.1 Weekly Assignments

The goal of this class is to help you build and/or improve the skills needed to effectively utilize your computer or a cluster in research. Eventually, these skills should expand your horizon for what research is possible and propitious for you to explore. This will take iteration. As such, while there are weekly assignments – approximately each week – except the first and a few other weeks where I hope you are working on other things, you have the chance to redo a past assignment or tackle a new assignment. Once you get a perfect score on an assignment, you cannot redo it. You can get full credit from the weekly assignments by either completing 6 assignments perfectly (10 points is the max for each) or through adding up the scores of your top 8 assignments and have then equal 60 or more points. For example, if you completed 9 assignments, with grades of 9, 8, 8, 8, 7, 7, 7, 6, and 4, you would receive

a 60 out of 60 on this portion of the class because the top 8 assignments add up to 60. You cannot get more than 60 points for these weekly assignments. In addition, if you do not have 6 perfect scores, your total points here will be the sum of your top 8 assignments. This includes one score for any assignment that was turned in multiple times. Just to be clear, this means that if you turn in one assignment 3 times, and get a 4, then a 6 then a 9 on it, only the 9 will count towards the final grade. When assignments are turned in they must be turned in as the correct format for the assignment. This will usually be a flat text file (for example for a zshell script) or a jupyter notebook for python based assignments. It will be helpful to complete the assignments in order. Note that you are welcome and even encouraged to complete all of the assignments as even if you do poorly on some you can, a) redo them, or b) discard the grade if it is not in your top 8 assignments. Because grading these assignments will be quite the job, most assignments will be short but of with some tricks.

5.2 Virtual Hack-a-Thon

Near the end of the semester, we will have a virtual hack-a-thon, where I will release a new dataset. The class can work together as one team or split into smaller teams to produce a research output from that data. This output could be an interactive visualization, a library for importing and working with the data, or something else. Teams will be assigned one grade and there will be a prize for the winning entry. This will be worth 15 points total. The hack-a-thon will be virtual, although it will be held in our regular classroom so there is the chance to use that space if circumstances make it safe.

5.3 Stretch Presentation Instructions and Topics

The final 15 points for the class come from a video presentation/walkthrough introducing a new tool or code-base to the class. These are all extensions from the class. The video is not expected to be professionally produced, or to have perfect grammar, etc. The goal is simply for you to introduce these tools so that students in the class can extend their knowledge of the computational ecosystem that exists to enhance research after the class. The possible topics are:

1. cython for speeding up calculations
2. Networkx for network calculations and visualization
3. pystan for using Stan within python
4. Docker for containers and replicable virtual environments
5. Pick one of support vector machines, random forests, or t-sne and present the algorithm and its scikit-learn implementation
6. Introduce scikit-learn pipelines and how they are useful
7. Scrappy for web-scraping

8. SpaCy for natural language processing
9. Tweepy for Twitter collection
10. Unit testing practices and tools
11. Mapping and projections with Altair
12. An introduction to Julia
13. An introduction to Vega-lite
14. A project that you pitch to me, must be of relatively general interest (must alert me by September 15).

The presentation should cover: a) what problem does this tool help solve, b) what are the research applications in social science for the tool, c) how to you use the tool, including providing a worked example of its use, d) what are some potential tricks or problems with its use that students should be careful about (if any), and e) where can students go to learn more about the tool, including weblinks, docs, tutorials, books, etc.

6 Adapting to Changing Risk Levels On and Off Campus

The University of Pittsburgh has three echelons of risk, ranging from guarded, to elevated, to high. As I write this, we are in the elevated posture. There is hope that we will move downward from this stage at some point in the semester. However, even if this occurs, there is the possibility of renewed outbreaks and of the risk posture moving upwards again. Therefore, I want to be clear about what the different risk levels entail for this hybrid class.

- **High Risk:** Within the high risk posture, there will be no in-class meetings. Thus, the classroom will not be available to students during the listed time. Asynchronous and synchronous learning will continue. There will be videos for you to watch, assignments for you to complete, and zoom classes where I will be available to help with assignments and problems.
- **Elevated Risk:** Within the elevated risk posture, there will be no in-class meetings. Thus, like in the high risk posture, the classroom will not be available to students during the listed time. Asynchronous and synchronous learning will continue. There will be videos for you to watch, assignments for you to complete, and zoom classes where I will be available to help with assignments and problems.
- **Guarded Risk:** Within the guarded risk posture, there will be optional in-class meetings. The classroom will be available for students to use during the listed time. Students are not required to be present in the class at any time even when we are in this risk posture. Asynchronous learning will continue. There will still be videos to watch, assignments for you to complete, and now the zoom classes will also be streamed in the

class room. I will be on those zoom meetings and available to help with assignments and problems.

7 COVID-19 Information, Related Campus Resources, and Dealing with the Pandemic in General

This is a unique semester and set of extremely challenging circumstances. To find out information about COVID, including symptoms and resources available to you, please see the documents here.

It is important to point out that given we are trying to teach and learn during an ongoing pandemic, we need to all show each other empathy and compassion. These are historically challenging circumstances. Some of us have lost loved ones during this time, many are scared, all of us have had our routines and support systems upended. It is impossible for anyone to do their best work under these circumstances. I will be very flexiblpate in the procedural process, initiated at the instructor level, as outlined in the University Guidelines on Academic Integrity. This may include, but is not limited to, the confiscation of the examination of any individual suspected of violating University Policy. Furthermore, no student may bring any unauthorized materials to an exam, including dictionaries and programmable calculators. To learn more about Academic Integrity, visit the Academic Integrity Guide for an overview of the topic. For hands-on practice, complete the Understanding and Avoiding Plagiarism tutorial.

8 Disability Services

If you have a disability for which you are or may be requesting an accommodation, you are encouraged to contact both your instructor and Disability Resources and Services (DRS), 140 William Pitt Union, (412) 648-7890, drsrecep@pitt.edu, (412) 228-5347 for P3 ASL users, as early as possible in the term. DRS will verify your disability and determine reasonable accommodations for this course.

9 Weekly Schedule and Assignment Due Dates

Week 1: August 25

Setting up for computer-augmented research

- If you have a mac or Linux computer, you should have a terminal command line interface (CLI) already. Recent mac's use zsh as the default shell, older mac's use bash. The commands we will be using should work in both bash and zsh but I prefer zsh. If you are using zsh (and I recommend it), I suggest also using oh-my-zsh, installation and discussion is available here. If you have a mac and need to switch to zsh from bash, see these directions.

- For mac, the default terminal app can be usefully replaced with iterm2, see the information here.

- Mac users should install Xcode, Xcode command line tools, and homebrew. Xcode and Xcode command line tools are available from the App store. Directions to install Xcode are here. For homebrew, see this link. Homebrew provides a way of getting extremely useful linux tools on macs. Xcode provides other tools that are useful for building code, compilers, etc.

- If you have a windoze computer, you will need to follow the steps here to get access to a command line interface that will work. While I do not have a windoze machine to test it out on, I believe these direction will allow you to switch your shell to zsh if you want to. In addition, you will need to install vim, direction are here.

- Everyone: skim Chapter 1 of this guide

- Everyone: You are going to need a robust python installation. So everyone should install either minicoda or the full anaconda install. The full anaconda install is here (you want the Python 3.8 or higher version). If you have plenty of space on your hard-drive and a good internet connection, go with that one. If your computer does not have a lot of memory or hard drive space, and/or your internet download speed is not great, then get miniconda. This is a lightweight version of the anaconda install of python, where you will need to install other libraries/modules as you go. This is no big deal. Many people actually prefer miniconda. You can install miniconda from here.

- Everyone: You need to install git, please follow the direction from here.

Week 2: September 1

Gaining reliability, repetition, and replicability: Intro to zsh

- I2STC Chapter 1 and Chapter 2

- P3H4MJ Chapter 1, 7-9

- DSFS Chapter 1
- Work your way through 1-5 (replacing bash with zsh if you that is your shell) here.

Week 3: September 8

More reliability, repetition, and replicability: zsh scripting

- I2STC Chapter 4
- Work your way through 6 and 7 (replacing bash with zsh if you that is your shell) here.
- Skim, but you do not have to deeply understand, all of the material here (again replacing bash with zsh if that is your shell).

Week 4: September 15

Writing messages to your computer beyond the shell prompt: Introduction to vim

- Read through this walkthrough of vim.
- Work your way through vimtutor, this should have come with vim. Just type vimtutor at your shell prompt. If you have vimtutor, see here for a short description of it. If you do not have vimtutor for some reason, see here for Windows specific troubleshooting.
- Play with this
- Play the free rounds of this game.

Week 5: September 22

Writing messages to your computer beyond the shell prompt and getting messages back too!: autocompleting, linting, yaml, json, and markdown with vim

- Install the vim-markdown plugin for markdown syntax highlighting and previews as described here. You will also need Vundle or another vim bundle/package manager. See installation instruction at vim-markdown page.
- Install Kite or another autocompletion library with vim.
- Use conda to install the Flake8 linting software, see command here.
- Read this on yaml and json.
- Run through the markdown tutorial here

Week 6: September 29

Organizing projects and changes to files: Intro to git and make

- I2STC Chapter 3
- I2STC Chapter 6
- Read Part 1 of Beginning Git and GitHub: A Comprehensive Guide to Version Control, Project Management, and Teamwork for the New Developer, available here from the Pitt Library if you are logged in.
- Read the Beginner and Getting Started tutorials from Atlasian available here

Week 7: October 6

Organizing collaboration and sharing: From git to github

- Read Part 2 of Beginning Git and GitHub: A Comprehensive Guide to Version Control, Project Management, and Teamwork for the New Developer, available here from the Pitt Library if you are logged in.
- Read the Beginner and Getting Started tutorials from Atlasian available here

Week 8: October 13

Communicating with your computer about research: Intro to python

- P3H4MJ] Chapter 2-6, 10-17, 24, 25
- I2STC Chapter 9
- Check out the JupyterLab interface here to compare iPython, with Jupyter notebooks to JupyterLab
- DSFS Chapter 2 until but not including "object-oriented programming"

Week 9: October 20

Communicating with your computer about research continued: Python classes, methods, and iterables

- P3H4MJ 18, 19, 21, 22, 26-30
- Read the official Python tutorial, available here
- Look at IPython tutorial here
- Read the "introduction" and "getting started" sections of the mypy docs, available here
- DSFS Chapter 2 from "object-oriented programming" until the end of the chapter.

Week 10: October 27

Using your computer to see patterns in data: Python visualization and maps

- P3H4MJ 23, 48
- DSFS Chapter 3
- Explore the docs for Altair, here and run through Case Study: Exploring Seattle Weather, available here

Week 11: November 3

Using your computer to count, average, and solve: practicing python with linear algebra and statistics

- P3H4MJ 31-33, and just read the "Newton's method" section in Chapter 36
- DSFS Chapter 4, 5 and 6, pick one of the "Further Exploration" links in each chapter to skim and get a little more background/remind yourself of some basic concepts

Week 12: November 10

Using your computer to find a maximum or a minimum and integrate distributions: python does calculus

- Note: we are skipping chapter 7 of DSFS, you are welcome to read through it; but it is generally a waste of time... not because of the treatment in this book (it is fine, even efficient), but because when you have a computer at your disposal, p-values are deeply silly.
- DSFS Chapter 8

Week 13: November 17

Tell your computer to collect data: python scraping and munging

- DSFS Chapters 9 and 10
- Read through the "Getting Started" with pandas doc, available here including the getting started tutorials.

Week 14: November 24

Teach your computer to learn from data: basic machine learning in python

- DSFS, Chapters 11-13
- P3H4MJ Chapters 43-44

- Look at Jake Vanderplas' jupyter notebooks introducing scikit-learn here, especially the "basic principles of machine learning" notebook, look in the notebook folder.
- Read about scikit-learn's nearest neighbors implementation here and scikit-learn's naive Bayes implementation here
- **Stretch Presentation Due November 31**

Week 15: December 1

Hack-a-Thon

- I2STC Chapter 11
- Look at the documentation for dvc, available here here
- See this short walkthrough about dvc and replication with large models/data