Sittyba

Scientific Programming and Computing in the Behavioral Sciences

Summer 2019

Description: Scientific programming and computing is an increasingly critical skillset in all sciences. This course is designed to develop these skills by covering computational and programming needs across the entire research cascade, from stimulus creation to making publication quality figures.
This is a hands-on class with a focus on developing practical skills in scientific computing and programming. Language of instruction is Matlab.

Objective: Allowing aspiring researchers without any background in programming to acquire the necessary fluency in scientific programming and computing to implement these methods to pursue their own research projects, from data recording to publication and being in a position to successfully and confidently continue to explore scientific programming after the conclusion of the class, in short to become "code-safe".

Time: M/W 5.30 to 8:00 pm

Space: 12WV_L111

Instructor: Pascal Wallisch, PhD
Office: Meyer Hall, Room 402
Phone: (212)-998-8430
Email: pascal.wallisch@nyu.edu
Office hours: Before class (5 pm) and by appointment

Materials: Wallisch et al. (2013). Matlab for Neuroscientists. Academic Press.
Matlab 2017a+ (student version with toolboxes)
Laptop – if you don't own one, I'll give you an old one of mine

Workflow: Each class session consists of a 30 minute introductory lecture that introduces the topics of the day conceptually, then an hour of joint coding that implements the same concepts practically, followed by 60 minutes of discussing carefully prepared "canned" code conceptually and doing exercises.

Peer coding: You will be paired with a suitable peer, and you will do the in-class coding together. Pairing will happen after the first week, based on the calibration data we record after the first session.

Grading: Student grades will be determined by completing weekly programming assignments as well as a final project. Each weekly assignment is graded by outcomes (is the program doing what it is supposed to do?) and the quality of the code, for a total of 10 points per assignment. Students do 5 graded assignments and can do a 6th one for extra credit.

5 code creation assignments (12% each -> 60% of the grade)
5 code review assignments (3% each -> 15% of the grade)
Final project (25%)

Grade cutoffs:

A	95-100	C+	77-79.9	F	0-59.9
A-	90-94.9	C	73-76.9		
B+	87-89.9	C-	70-72.9		
B	83-86.9	D+	65-69.9		
B-	80-82.9	D	60-64.9		

Email policy: If you want to email the teaching staff, you can do so at any time. Every effort will be made to respond to your dispatch within 24 hours. However, ***ONLY*** emails that have *CODING2019* in the subject line will be answered. If emails don't have this line in the header, they will be ignored.

Code review: An essential part of learning how to code is to assess and learn from the code of others. Also – realistically – a large part of your duties as a scientific programmer will be to understand and modify the code of other people. So it is important to practice this now. It also trains you to write code that is palatable to others - think of it as community service. Therefore, you will be required to review the weekly assignments of your peers. To gain insight from multiple perspectives (as everything in coding can be done in multiple ways), you will be required to review two assignments of others. Review criteria are: a) How clear is the code (1-5), b) How functional is the code (does it do what it is supposed to do) and c) Compare and contrast in a few sentences how you solved the problem. These reviews will influence our assessment of the assignment for grading purposes so it is extremely important that you take them seriously.

We will normalize all ratings by the severity of the rater (so that you won't get penalized if you are unlucky enough to have a tough grader) at the end of the course. Also, discrepancies in ratings are adjudicated and resolved by TA review.

Workload: Given the compressed schedule, in order to succeed in this class, you should expect to commit a total of 20-30 hours of dedicated work per week (depending on your background). If you can't commit to this now, you should probably not take the class at this point. This is part of the nature of skill acquisition – there are no known shortcuts.

Assignments: There will be one assignment a week. You should expect to complete code creation and code review assignments on a weekly basis, so there will be two deadlines per week: The code *creation* assignment and the code *review* assignment. Don't wait until the last minute. I recommend getting started immediately after the assignment comes out.

Deadlines: This class is an immersive experience – it is important to keep in mind that there are several dates every week: The class, the due date of the code creation assignment and the due date of the code review assignment. This also implies that there can be no real extensions – otherwise things will snowball out of control over the course of events.

Late work: Due to the above, please try to avoid late work. In general, please just turn in something, even if it is not complete. Of course this will still happen in extreme circumstances. Now, late work is not worthless, but it is most definitely worth less. Specifically, we must deduct 20% of the maximal grade score per day it is late. After the deadline, send to me directly.

Final projects: Are anything that will help you with your research in some way. This could be a data browser or something that lets you record, analyze or visualize data. The possibilities are endless. But it has to be useful. To you.
Should spark joy.

Course schedule

(This is an aspirational schedule. Depending on how it goes, we might go faster or slower or do other stuff altogether, depending on the diverse needs of the student population. The point of the course is not to "get through" prepared materials but rather to teach the material that would be most useful to the student population at hand (both in terms of their background, needs and interest) in a way that allows for long-term, transformational change. The only way to achieve this is to continually calibrate and re-calibrate as the course goes on.)

Italics: We'll assign a project at the end of this session

Part I: Foundations

Session 1 (05/29): Foundations of Programming I: Matlab as a scientific calculator (+Welcome)

Session 2 (06/03): Foundations of Programming II: Matlab as a graphing calculator

Session 3 (06/05): Foundations of Programming III: Matlab as a programming language

Part II: Data recording (stimulus generation and experimental control)

Session 4 (06/10): Visual search and attention experiments (Reaction time and randomization)

Session 5 (06/12): Visual psychophysics experiments (flexible stimulus generation and deployment)

Part III: Data analysis (seeds)

Session 6 (06/17): Classical Statistics – Null Hypothesis Significance Testing

Session 7 (06/19): Resampling methods: Monte Carlo methods and permutation tests

Session 8 (06/24): Time Series and Fourier Analysis

Session 9 (06/26): Dimension Reduction (Principal Component Analysis)

Session 10 (07/01): Machine learning methods (clustering and classification)

Part IV: Deployment

Session 11 (07/03): Graphical User Interfaces