Sittyba
Scientific Programming and Computing for the Behavioral Sciences
Spring 2018

Description: Scientific programming and computing is an increasingly critical skillset in all of the behavioral sciences. This course is designed to develop these skills by covering computational and programming needs across the entire research cascade in the behavioral sciences, from data collection 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 behavioral 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 recording of data 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:	Tuesday, 12:30 to 3:00 pm
Space:	SILVER 620
Instructor:	Pascal Wallisch, PhD
Office:	6 Washington Place (Meyer Hall), Room 402
Phone:	(212)-998-8430
Email:	pascal.wallisch@nyu.edu
Office hours:	Mo/Tu/We 3:40 - 4:40 pm (walk-in (no apt) - first come, first served)
TA:	Deshawn Sambrano
Office:	Meyer 435
Email:	dsambrano@nyu.edu
Office hours:	Wednesday, 10.30-11.30 am
Labs:	Thursday, 12.30-1.45 and 2-3.15 in Meyer 565.
Materials:	Wallisch et al. (2013). Matlab for Neuroscientists. Academic Press. Matlab 2015a+ (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 and a half of joint coding that implements the same concepts practically, followed by 30 minutes of discussing carefully prepared "canned" code conceptually.

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 collect in the first session.

Grading: Student grades will be determined by completing weekly programming assignments as well as (for graduate students) 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 10 graded assignments and can do an 11^{th} one for extra credit. A third of the **grad student** grade is determined by the final project.

10 code creation assignments (7.5% each -> 75% of your grade)
10 code review assignments (2.5% each -> 25% of your grade)

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 *SCICOMBS2018* in the subject line will be answered. If emails don't have this line in the header, they will be ignored.

Labs: Are mandatory, and based on our experience necessary for you to attend in order to succeed in this class. Success is defined as acquiring a new skill (coding in Matlab) and for that, guided practice is invaluable.

Workload: To succeed in this class, you should expect to put in 18-20 hours a week of dedicated work outside of class. If you can't commit to this now, you should probably not take the class this semester. It will be offered every spring. This is part of the nature of skill acquisition – there are no known shortcuts. For the assignments, we will use Auftragstaktik (mission command) methods – we will tell you what to do and why, not how. This approach is known to foster creative problem solving skills.

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.
However, do not confuse the points from the code review with grade points. First, 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. In addition, the TA will closely monitor each weeks review process to make sure that it is fair and calibrated.

Assignments: There are 15 weeks in the course, but only 10 assignments, so there won't be one for every week. Particularly in the beginning, there won't be an assignment just yet. There are still plenty of things to do in the labs during those first couple of weeks. After that, 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 is due on Saturday at 11:59 pm, the code *review* assignment is due on Monday at 11:59 pm. We will send reminders. But don't wait until the last minute. We recommend getting started immediately after the assignment comes out so that you can ask informed questions during the recitation/lab.

Deadlines: This class is an immersive experience – it is important to keep in mind that there are 4 dates every week: The class, the due date of the code creation assignment, the due date of the code review assignment and the time of the lab section. This also implies that there can be no real extensions – otherwise things will snowball out of control over the course of the semester.

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 the TA 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 collect, analyze or visualize data. The possibilities are endless. But it has to be useful. To you.

Course schedule

Part I: Introduction

Week 1: Welcome to the Matrix

Week 2: Foundations of Programming I: Matlab as a scientific calculator (matrix manipulations)

Week 3: Foundations of Programming II: Matlab as a graphing calculator (introducing scripts)

Week 4: Foundations of Programming III: Matlab as a programming language (program flow)

Part II: Data recording

Week 5: Reaction time experiments (timing)

Week 6: Attention experiments (graphics)

Week 7: Psychophysics experiments (arbitrary stimuli)

Part III: Data analysis

Week 8: Classical statistics: Significance tests

Week 9: Time Series Analysis

Week 10: Frequency Space Analysis

Week 11: Principal Component Analysis

Week 12: Resampling Methods

Week 13: Machine Learning and Classification

Part IV: Deployment

Week 14: Graphical user interfaces

Week 15: Advanced (multi-figure) Graphical user interfaces