IPBR: Coding Project 2016

The centerpiece of the IPBR course is a group coding project that you will work on for the second half of the course, and present to your classmates during the last week of class. This project will incorporate the MATLAB skills and algorithm design principles that you have learned so far with topics from biology research. You will work in groups of two (or three, if needed), for four weeks total. Be creative and have fun!

Each group will be assigned an instructor who will assist you with your project. Every week, your group will check in with the instructor, and you can also reach out to us as you need with questions at the review sessions and by e-mail.

Your weekly check-in should include all of your project code, as well as a short description (a paragraph is fine) of your progress that week. This is also a good opportunity to ask questions. Please make sure to **comment your code**, so that we know what it does! We'll send feedback along, and answer any questions you ask.

At the end of the project, you should submit all of your commented code for the project, as well as a README document to the user with instructions on how to run it. You will also give a 5-10 minute presentation on your project to the class.

Schedule:

July 8 (Friday): E-mail Georgia (squyres@g.harvard.edu) with the names of your group members and your choice of project (see below). Georgia will respond approving your project, and will assign your supervising instructor.

July 13 (Wednesday): Send your first week progress report to your assigned instructor by the start of lecture. At this point, you should have designed an algorithm for your project, and have started coding.

July 20 (Wednesday): Send your second week progress report to your assigned instructor by the start of lecture.

July 27 (Wednesday): Send your third week progress report to your assigned instructor by the start of lecture. At this point, you should be close to finishing up your project so that you can work on your presentations.

August 3 (Wednesday): Final projects are due! Submit all of your code for the project, as well as a README document, by the start of lecture. Final presentations are given in class today.

Projects:

Below is a sample list of projects. You can choose from one of the list below, or design your own! We encourage you to draw on your own research interests and design a project that interests you. Please e-mail a description of your project to Georgia by Friday 7/8- she will approve your proposal and assign a supervising instructor.

Ultimately, you want to write code that you and other researchers could use in their research. Once you have finished writing the main algorithm, you can and should also think about ways to make your code easier to use. Checking for errors (especially in user inputs), writing helpful error messages, and documenting your code well in a README are all important aspects of programming!

Image Classifier: It is common in microscopy labs to want to classify images. For instance, you might want to look through a data set of images and classify them as "labeled" or "unlabeled". Write code in MATLAB that takes as input a directory full of images and shows them to the user one at a time. The user can then type in their classification for each image. MATLAB should save the user's input and, at the end, show some helpful information (for instance, what percent of images were "labeled".)

- Week 1: Outline algorithm, write code to read images from a directory
- Week 2: Code to show images to the user and get user classification
- Week 3: Output code, error handling
- Week 4: README and presentation

Protein Identifier: MATLAB has the ability to run BLAST searches with the Bioinformatics toolkit. In this project, your code should read in a nucleotide sequence, and translate it to protein. Then, BLAST the protein sequence and display the identity of the protein.

- Week 1: Outline algorithm, write code to read sequence file
- Week 2: Write code to translate the nucleotide sequence
- Week 3: Code to run BLAST and show the results to the user, error handling
- Week 4: README and presentation

Cell Counter: This project involves counting the number of objects in an image (for instance, cells in a microscope image). This can be done using MATLAB's thresholding toolkit: see http://www.mathworks.com/help/images/image-enhancement-and-analysis.html. The program should take images as input and tell the user the number of objects in the image, as well as any other statistics you think are important.

- Week 1: Outline algorithm, write code to read image files
- Week 2: Code to segment images and identify objects.
- Week 3: Output code, error handling
- Week 4: README and presentation

Worm Motion Simulator: This project involves simulating the movement of a *C. elegans* worm. There are two aspects to how *C. elegans* move: their overall motion (resembles a random walk), and the sinusoidal movement of their body along that trajectory. Your program should generate trajectories of *C. elegans* motion.

- Week 1: Outline algorithm, write code to simulate random walk
- Week 2: Add sinusoidal motion, draw plots of trajectories
- Week 3: Write code to have your plot update as the program runs and save as a movie (tip: use the pause command in your loop!)
- Week 4: README and presentation

Population Simulator: There are many different ways to model population dynamics- you can ask your instructor for help, or design your own! Even simple models can show interesting behaviors about competition between species, mutation rate, predator-prey relationship, and much more.

- Week 1: Design the simulation you want to run, outline algorithm
- Week 2: Write code to run your simulation, try multiple runs
- Week 3: Code outputs from simulation results (plots, etc.), and try running your simulation with different parameters
- Week 4: README and presentation

Presentation

At the end of the project, your group will present your results to the class. You should prepare a 5-10 minute presentation (we suggest using Powerpoint slides). Your presentation should describe the motivation for your project, how your code works, and what the results are when your code is run. You can also talk about your experience working on your project (what did and didn't work?), as well as how you would improve your code in the future.