Programming for Genomics IB 501 (4 credits)

Instructor: Julian Catchen <jcatchen@illinois.edu>
Teaching Assistant: Shriram Bhat <sb65@illinois.edu>

Schedule: Monday and Wednesday, 1-2:50PM
Location: 2083 Natural History Building (NHB)

Office hours:

J. Catchen	Tuesdays	2-3PM	233B Morrill Hall
S. Bhat	Tuesdays/Thursdays	3-4PM	17C Burrill Hall

Course Description:
The goal of Programming for Genomics is to teach students to *think algorithmically*. By the end of this course, students should be able to construct a biological hypothesis, and implement code, or deploy an existing code implementation to test that hypothesis. Students will first learn to use the UNIX operating system, including the use of command streaming through pipes, and will learn to program in Python. We will use biological data sets, mostly from high-throughput sequencing genomics projects, as a base for learning UNIX and Python. As the course progresses we will cover major genomics approaches and the algorithms that underlie them, including K-mer analysis, whole genome and transcriptome assembly, databases and SQL, and visualization techniques. Students will use these analysis programs and supplement the analysis with data generated by their own Python code. This course will provide students with a strong introduction to the major bioinformatics analyses approaches as well as the ability to write their own code in Python, within a UNIX environment.

Student Learning Outcomes:
The main learning outcome is teaching students how to *think algorithmically*. That is, starting with a biological hypothesis, how to break down the problem of biological analysis into a series of algorithmic steps, and then how to use code to implement those steps and thereby test the hypothesis. The second learning outcome is to study the major algorithmic approaches to understand the landscape of genomic analysis. Finally, the third outcome is to understand the state of the art in the field of population genetic and comparative genomic analysis by studying the literature in this area.

Prerequisites:
No computational skills are assumed for this course, however familiarity with molecular biology and evolution is expected and necessary.

Textbook:
Downey, Allen. 2015. Think Python: How to Think Like a Computer Scientist. Green Tea Press.

Journal articles:
Readings from primary literature for discussion sections, and tutorials for computer labs, will all be made available online.

Grading:

Quizzes	20%
In-class Assignments	20%
Problem Sets	40%
Research Presentation	10%
Presentation Discussions	10%

Expectations:
My primary expectation is that students come into the class ready to learn new skills and develop new perspectives. The course does not assume any previous experience in programming, but different students are able to learn the abstract thinking involved in programming at different rates. So, if it is a brand-new skill, you will have to work harder to master it. Attendance is essential, as well as engagement with the projects and assigned readings, to get the most out of the course.

All students should follow University of Illinois Student Code, available online at: https://studentcode.illinois.edu/.

Class Attendance and Late Work:
Attendance in class is required. There will not be any lecture recordings or opportunities to attend the class remotely. Late work will be allowed at the discretion of the instructor potentially with a late penalty.

Class Format:
The general format for each class will be a lecture in the first hour followed by time to work on projects. Early in the course we will use lecture/lab time to work together on new skills but that will transition to a more open-lab format as we work on more substantive projects.

Computer Requirement:
Students are required to have a laptop computer to complete computational assignments. We will do our actual computing on the on-campus IGB BioCluster, but we will access the cluster remotely inside and outside of class using laptops.

In-class assignments:
Early in the course we will work on assignments together in class (and students will continue working on them after class). These assignments will serve as a scaffold for learning UNIX and Python. These assignments are worth 20% of the final grade.

Problem sets:
The main way we will engage with the material, and the bulk of the final grade in the course, is through applied projects. Each problem set will involve a bioinformatics analysis, which will typically include a mixture of running existing software – for example to assemble a genome or identify SNPs – combined with writing your own Python code or using UNIX to answer additional questions about the data. Problem sets make up 40% of the final grade.

Quizzes:
There will be several quizzes during the course in total worth 20% of the final grade. These quizzes will cover key concepts from the lecture material and particularly the assigned reading. Students with a valid reason for missing a quiz will be given an opportunity to take a make-up exam at the discretion of the instructor. Valid reasons include only medical reasons (with a Dean's note), tragedy in your immediate family, or religious observances and practices.

Research Presentations:
Each student will choose a methodological topic, select several research papers related to that topic, and then create a short presentation based on their research. Graduate students will select a topic related to their research work; undergraduates are free to choose a topic of interest. Students are encouraged to form *affinity groups*, that is to link up with other students interested in the same, or a similar topic. These presentations are worth 10% of the final grade and are linked to the Presentation Discussions.

Presentation Discussions:
Once a student's presentation has been approved by the instructor, the student will present their work to another member of the class (independently during the semester). Each successful presentation (up to five maximum) to another student will provide 2% of this part of the grade. In total, these discussions are worth 10% of the final course grade.

Academic Integrity
According to the Student Code, "It is the responsibility of each student to refrain from infractions of academic integrity, from conduct that may lead to suspicion of such infractions, and from conduct that aids others in such infractions." Please know that it is my responsibility as an instructor to uphold the academic integrity policy of the University, which can be found here: https://studentcode.illinois.edu/article1/part4/1-401.

No course materials, including, but not limited to, lectures and assignments, may be posted online on any website or in any forum, whether privately or publicly available. If materials are shared the student may receive a serious penalty, up to a failing grade in the course.

Collaboration:
Students are encouraged to help each other and collaborate to discuss projects, analysis approaches and algorithmic strategy. However, students may not share code with one another, not even pseudocode or whiteboard code. **All programming must be original for each student**.

Use of Online/External Resources:
Students may not use code obtained externally from the course, e.g. from online web sites, chat bots or AI models, or from people outside the class with programming experience. Code turned in for grading must be original and must be based on the fundamentals presented in class – that is submitted code must reflect only the elements we have learned in class (the instructor may make exceptions for students already experienced in coding).

Disability Accommodations
To obtain disability-related academic adjustments and/or auxiliary aids, students with disabilities must contact the course instructor and the Disability Resources and Educational Services (DRES) as soon as possible. To contact DRES you may visit 1207 S. Oak St., Champaign, call 333-4603 (V/TTY), or e-mail a message to disability@illinois.edu.

Weekly Schedule

Week	Class	Topic
1 (Aug. 26)	M	Introduction to Programming for Genomics; Introduction to UNIX, part 1
	W	Introduction to UNIX, part 2
2 (Sept. 2)	M	Labor Day Holiday, No class
	W	Advanced UNIX, part 1
3 (Sept. 9)	M	Advanced UNIX, part 2
	W	Advanced UNIX, part 3;
4 (Sept. 16)	M	Advanced UNIX, part 4; History of Sequencing, Part 1;
	W	History of Sequencing, Part 2 Open Lab
5 (Sept. 23)	M	Introduction to Python, part 1
	W	Python, part 2
6 (Sept. 30)	M	Open Lab
	W	Quiz #1 Open Lab
7 (Oct. 7)	M	Python, part 3
	W	Introduction to K-mers: Normalization, Error Correction, Repeat Estimation; Open Lab
8 (Oct. 14)	M	K-mers: the BLAST and BLAT algorithms, Needleman-Wunsch Alignment/Dynamic Programming; Open Lab
	W	Short-read Alignment Algorithms; GTF files and SAMTools; Open Lab
9 (Oct. 21)	M	Population Genomic Analysis Algorithms with RAD-seq, part 1; Open Lab
	W	Population Genomic Analysis Algorithms with RAD-seq, part 2; Open Lab
10 (Oct. 28)	M	Whole Genome Assembly Algorithms, Part 1 Open Lab
	W	Whole Genome Assembly Algorithms, Part 2 Open Lab
11 (Nov. 4)	M	Overview of Computer Hardware
	W	Python Revisited Open Lab
12 (Nov. 11)	M	Data Visualization with Python Open Lab
	W	Quiz #2 Open Lab;
13 (Nov. 18)	M	Gene Expression Quantification; Open Lab
	W	Open Lab
(Nov. 25)	M, W	Thanksgiving Holiday
14 (Dec. 2)	M	Introduction to Databases and SQL Open Lab
	W	Open Lab
15 (Dec. 9)	M	Open Lab
	W	Quiz #3; Open Lab

The following schedule and set of topics is subject to change, depending on the pace of class and other factors.