Tools for Computational Biology — Course Syllabus

2018-03-19

Course overview and curriculum content

The wealth of publicly available biological data has transformed the nature of molecular biology research in today's post-genomic era. In addition to careful design and execution of experiments, molecular biologists routinely interpret their collected data in the context of rich, multi-layered annotations available from public data repositories such as NCBI and ENSEMBL. Many research projects even start with extensive analyses of public data sets for hypothesis formulation and experimental design. Performing these integrative analyses requires an ability to computationally organize, transform, and visualize biological data from different sources and in diverse formats.

Students in the UW Molecular and Cellular Biology graduate program come primarily from an experimental biology backgound. Many of them have little or no training in computational analyses of biological data. Students are often expected to pick up computational skills on their own as part of their graduate research. However, the lack of familiarity and comfort with computational data analyses impedes students' ability to critically examine published literature and formulate their thesis projects early in their graduate career. The unstructured learning of computational skills by many students often result in *ad hoc* data analyses methods that do not adhere to established best practices in computational biology for reproducibility and sharing.

As practicing computational biologists, the course instructors have found that they repeatedly teach a core set of cross-cutting computational biology tools to each new graduate student in their research group. The proposed course will formalize and expand this training to all students in the MCB graduate program in their incoming year. Students will learn to organize unstructured data into standard tabular and hierarchical formats, transform data for statistical analyses, and visualize the transformed data. The course will introduce students to computational best practices including version control, project organization, and code documentation. Students will gain hands-on experience working with Unix command line tools and the widely used Python and R programming languages. Each class will be structured as an active learning module where students will download a publicly available data set and learn the computational methods required to analyze and visualize that data set.

Overall learning goals and objectives

At the end of the course, the following learning objectives will be met:

1. The student will understand the concept of tidy data as a required step for many computational analyses. The student will be able to use Python and R programming languages to transform unstructured biological data into a tidy format.
2. Students will be able to transform raw data into various summary statistics for testing specific biological hypotheses. They will be able to graphically present their analyses using data visualization libraries such as Seaborn and GGplot.
3. The student will feel comfortable using the UNIX command line for accomplishing common

computational tasks such as connecting to remote servers, installing new software from source, and batch and parallel processing of multiple data sets.

4. The student will be able to apply best practices for reproducible computational research including version control, code documentation, and project organization.
5. Students will be able to combine the above skills for performing integrative, multistep analyses of publicly available data such as high throughput sequencing data and genomic annotations.

Evaluation and grading

1. Students will be assigned reading material on a specific topic before each class. At the beginning of the class, the instructor will take questions to clarify any points of confusion or difficulty. This Q&A session will be followed by 70 minute of hand-on learning where the students will work through a series of computational exercises that illustrate practical use cases of the topic for that class. Students will work in teams of 2 or 3 and frequently consult the instructor to troubleshoot their programming code. This interaction is critical for students to acquire a thorough understanding of the discussed topic, and will constitute 25% of the evaluation. The instructor will note down the level of participation of each student at the end of each class.
2. Students' understanding of each discussed topic will be evaluated through weekly homework assignments. Students will receive their assignments through an online course repository at `http://gihub.com`. They will submit their homeworks a week after assignment as code samples through the same online repository. A total of 5 homeworks will be assigned and each will account for 10% of the evaluation for a total of 50%.
3. Students' ability to integrate the individual topics into a holistic computational workflow will be evaluated through a final assignment. These assignments will take the form of designing a computational pipeline for reproducing the findings in published research papers. The instructors will pick a list of 20 published papers for students to choose from for their final assignment. Students will submit their assignments as code samples through the same online repository as above. This final assignment will constitute 25% of the evaluation.

In summary, grades will be calculated based on the following distribution:

Class participation	25%
5 homework assignments	50%
Final assignment	25%

Course Textbooks

- Python Data Science Handbook: Essential Tools for Working with Data by Jake Vanderplas
- R for Data Science: Import, Tidy, Transform, Visualize, and Model Data by Hadley Wickham

A primary goal of this course is to teach students to use online learning resources effectively. Hence in addition to the above books, online references will be provided for each class.

Course Instructors

The course will be jointly taught by 4 faculty in the Computational Biology Program at Fred Hutch. Each instructor will teach 2–3 lectures. The instructors will closely coordinate their lectures to ensure seamless transition and absence of redundancy.

Arvind Subramaniam
Assistant Member
Basic Sciences Division and Computational Biology Program, PHS
Fred Hutchinson Cancer Research Center
rasi@fredhutch.org (Primary contact)

Trevor Bedford
Associate Member
Vaccine and Infectious Disease Division and Computational Biology Program, PHS
Fred Hutchinson Cancer Research Center
trevor@bedford.io

Jesse Bloom
Associate Member
Basic Sciences Division and Computational Biology Program, PHS
Fred Hutchinson Cancer Research Center
jbloom@fredhutch.org

Phil Bradley
Associate Member
Computational Biology Program, PHS
Fred Hutchinson Cancer Research Center
pbradley@fredhutch.org

Course schedule

Class 1: Introduction, code and data organization, version control

- Introduction to online resources for learning
- Organization of data in directories and in tabular format
- Writing in plain text using Markdown
- Using Jupyter Notebooks for interactive data analysis
- Basics of Git version control: to be able to submit assignments (add, commit, push); web interface
- GitHub (commenting on commits, various views)

Online Resource: StackOverflow

Class 2: Tabular data analysis using Python and Pandas

- Using the Python package, `pandas`, to read in tabular data
- Introduction to tidy data; data reshaping using `pandas`
- Basic plots with Matplotlib and Seaborn

Online Resource: Python cheat sheets

Class 3: Writing multi-functional programs using Python

- Read in various file types (`biopython`, `htseq`)
- Teach which types of data structures are appropriate for what tasks
- Text analysis using regular expressions
- General debugging: `try/except/assert`, checking types.
- Using `pdb` and `ipdb` to debug Python code
- Formatting code for easy reproducibility according to `PEP8` standards

Online Resources: `https://regex101.com/`, `https://pythex.org/`

Class 4: Project organization and advanced version control

- Using terminal emulators
- Separating data and code in projects
- How to work in a collaborative project using git (branching, merging, pulling, conflict resolution, .gitignore), pull requests (GitHub)

Online Resources: A Quick Guide to Organizing Computational Biology Projects, How to name files

Class 5: Introduction to Shell and command line

- How to set environment variables, e.g. `PATH`, `LD_LIBRARY_PATH`, get and set locations of programs using `export` and `which`
- Customizing command line using `.rc` files (or `.bash_profile`)
- Remote access using SSH: Setting up keys; ssh config; ssh forwarding; file transfer; and tmux for saving session
- Setting up Python environment using Conda
- Using `wget` to download to grid
- Text editing using `vim`
- Tracking changes using a text editor, `vimdiff`

Online Resource: Getting cozy with the command line

Class 6: Command-line tools for biological data analysis

- Download raw data from NCBI and ENSEMBL
- Tools for preprocessing large data files: `sratools`, `fastxtoolkit`
- Align high-throughput sequencing data using `bowtie`, `HISAT2`

Online resources: NGS sequence preprocessing

Class 7: Advanced Python

- Introduction to object-oriented programming
- Running external commands from Python
- Ensuring coding standard using `pylint`
- Concepts of mutability and thread pools
- Variable arguments using `kwargs`
- Using function and class decorators
- Creating command line interface to programs using `argparse`
- How to create a Python package
- Unit testing

- Writing documentation for your Python package using `Sphinx`

Online Resources: Argparse tutorial, How to package your Python code, Sphinx introduction

Class 8: Advanced Shell

- How to use a distributed cluster; Using `parallel` and `xargs`
- How to use command line to hack sequence data: `seqmagick` and `emboss` toolkit
- Using Shell scripting and command-line history
- Compiling programs from source using `Make`

Online Resource: Advanced Bash-Scripting Guide by Mendel Cooper

Class 9: Reading and plotting data using R

- Introduction to Rstudio programming environment
- Using `tidyverse` packages to analyze flow cytometry data
- Concept of tidy data and annotations using R
- How to read and write tab-delimited files
- Using `dplyr` verbs effectively: `select`, `mutate`, `filter`, `group_by`, `summarize`, `join`, `spread/gather`
- Concept of `pipe`
- Using `ggplot` to plot data in multiple formats. Introduce `geoms`, `faceting`

Online Resources: Enter the tidyverse by Stephanie Spielman, Fundamentals of Data Visualization by Claus Wilke

Class 10: Using R for genomic data analysis

- Use `bioconductor` packages to analyze RNA-seq data
- Use `Biostrings` for Fasta IO, sequence manipulation, motif counts
- Work with high-throughput sequencing alignments and genomic annotations using `GenomicAlignments`, and `GenomicFeatures`
- Retrieve standardized annotations using `AnnotationDbi`

Online Resource: Bioconductor for Genomic Data Science by Kasper Daniel Hansen