Genome 373: Genomic Informatics

Professors
Elhanan Borenstein and Jim Thomas
Genome 373

• This course is intended to introduce students to the breadth of problems and methods in computational analysis of genomes, arguably the single most important new area in biological research.

• The specific subjects will include:
  • Sequence alignment
  • Sequencing and next generation sequencing
  • Gene prediction
  • Micro-array analysis
  • Evolutionary relationships and phylogeny
  • Biological systems
Outline

- Course logistics
- Introduction to Bioinformatics
- Introduction to Python
Instructors

- **Elhanan Borenstein**: Weeks 2-6
- **Jim Thomas**: Weeks 1,7-10
- **Office hours**: Monday 11:20-12:00

- **Tanya Grancharova** (TA) will teach additional topics including programming and problem solving skills.

- Material covered in section is required, and will be on the exams.
Webpage

• Web site:
  
  http://elbo.gs.washington.edu/courses/GS_373_14_sp/

• Page has links to
  – Lecture notes
  – Handouts
  – Homework assignments
  – Many useful resources on:
    • Bioinformatics
    • Python
Programming

• Note: Historically, this course required prior programming experience.

• The first week in class and the first few weeks of section will focus on learning to program in Python.

• If you do not have any programming experience, that’s ok, but ... you will need to work hard to catch up.
Grading

- 50% homework
- 20% midterm exam (in class)
- 30% final exam, Mon, June 10

- Final exam is cumulative.
Homework

• Posted online each Wednesday and due the following Wednesday.
• Homework is a mix of written problems and programming.
• Homework assignments are to be submitted by email!
• Programming assignments should be implemented in Python. For other languages, please ask Tanya.
• More on home assignment submission in the quiz section.
Textbooks

References:

Electronic access to journals is generally free from on-campus computers. For off-campus access, follow the "[offcampus]" links or look at the library "proxy server" instructions.

3. [How dictionaries work (aka hash tables or hash maps)](https://www.example.com)

Python Resources:

**General**
- *[Free (legal) PDF text book](https://www.example.com)*
- *[Python quick reference guide](https://www.python.org/docs/)*
- *[Much more extensive python quick reference guide](https://docs.python.org/)*
- *[Python.org](https://www.python.org)*
- *[Python Documentation](https://docs.python.org/)*
- *[Tutorial](https://docs.python.org/3/tutorial/)*
- *[Howtos](https://docs.python.org/3/howto/)*

**Regular Expressions**
- *[Intro](https://docs.python.org/3/howto/regex.html)*
- *[Howto](https://docs.python.org/3/howto/regex.html)*
- *[Another Howto](https://docs.python.org/3/howto/regex.html)*
- *[Library Reference](https://docs.python.org/3/library/re.html)*
- *["RegExPal!" (For Javascript rather than Python, but similar and quite handy. Try it)](https://www.example.com)*

**Biopython**
- *[Biopython.org](https://biopython.org)*
- *[Tutorial and Cookbook](https://biopython.org/Tutorial/)*

**Python Books**
- *[Python for Software Design: How to Think Like a Computer Scientist] by Allen B. Downey. (Includes early drafts of our text book; cheaper than the published version, but less polished...)](https://www.example.com)*
- *[Learning Python] by Mark Lutz. O'Reilly (Very comprehensive. Much is accessible to beginners.)*
- *[Dive Into Python 3] by Mark Pilgrim. (Another online book. Based on Python 3, so some differences, and more advanced, but also free.)*

Bioinformatics Books

» *[Python for Bioinformatics] Sebastian Bassi, CRC Press, 2010. (A little too advanced as a programming book for beginners, but fine now that you're experienced.]*
» *[Python for Bioinformatics] Jason Kinser, Jones and Bartlett, 2009. (Ditto.)*
Background survey

Please write on the index card your:

1. Name and email

2. Major

3. Primary background (biology, computation, other)

4. Programming experience (how much, what language)

5. Registered/not-registered/waiting-list
Why Bioinformatics?
Find the binding sequence: caattatatgtaaa
Moore’s law

Computer processing power doubles every ~2 years.

Microprocessor Transistor Counts 1971-2011 & Moore’s Law

dotted line - 2 year doubling
Sequencing cost decreasing much faster than computing cost

>2-fold drop per year

? - changing so fast hard to be specific
**As of 2011 done or nearly done**

- > 2,000 viruses
- > 1,000 bacteria and archaea
- Hundreds of fungi
- Dozens of protists
- Dozens of nematodes and insects
- 6 fish, 1 reptile, 4 birds, 1 amphibian
- About 10 plants
- About 40 mammals (+multiple individuals)
- Microbial communities (e.g., human microbiome)
A computational bottleneck
Find the binding sequence: **caattatgttaaa**

... allowing for one mutation and one insertion
How well can the string

**GAATTTCAGTTA**

match the string

**GGATTCGA?**

(what is the best **alignment** between the two strings?)
Informatics Challenges: Examples

• Sequence comparison:
  – Find the best match (alignment) of a given sequence in a large dataset of sequences
  – Find the best alignment of two sequences
  – Find the best alignment of multiple sequences

• Motif and gene finding

• Relationship between sequences
  – Phylogeny

• Clustering and classification

• Many many many many more ...
Why Python?

- **Python** is
  - easy to learn
  - fast enough
  - object-oriented
  - widely used
  - fairly portable

- **C** is much faster but much harder to learn and use.

- **Java** is somewhat faster but harder to learn and use.

- **Perl** is a little slower and a little harder to learn.