QLSC 600 – Programming Bootcamp Python and R

Mathieu Blanchette

Jack Guo

The right language for the right task

Programming is essential for nearly all quantitative analyses of life science data

Nearly any task can be done with any programming language, but…

- Development time: How much time will take you to code it and make it work
  - Depend on your experience, suitability of language for task, availability of suitable of libraries
- Execution time: May depend on language
- Debugging/maintenance
- Distribution and adoption

➔ One must be knowledgeable enough to identify the best language for the task, and nimble enough to learn it if needed

Overview of programming languages

Assembly languages: Direct instructions to the CPU

+ Can sometimes be highly optimized, if you know what you're doing

- Very painful to write
- Not portable (depends on computer architecture)

➔ To be used only to super-optimize the implementation of a critical piece of code.

Compiled languages: Compiler translates code to assembly language ahead of execution.

- **C and C++:**
  + Fast running time, good control over memory
  - Tedious debugging
  - Not too many Bio libraries
  - Poor support for visualization
- **Java**
  + Acceptable running time
  + Somewhat faster development/debugging
  - Limited choice of biology-related libraries
  - Little control over memory
  - Tedious IO and visualization

Overview of programming languages

Interpreted languages: No compiler → Easier to use, but generally slower

- **Python**
  + General-purpose, very flexible
  + Easy to learn, allows fast prototyping
  + Free, with tons of excellent libraries
  - Can sometimes be slow (but can be compiled to make it faster)

- **R**
  + Strong stats and biology packages (e.g. Bioconductor)
  + Free
  + Excellent visualization tools
  - Slow for iterative programs
- Matlab
  + Excellent for fast matrix computation , visualization
  - Not free

Benchmark [Aruoba et al. 2014]

Table 1: Average and Relative Run Time (Seconds)

	Mac			Windows		
Language	Version/Compiler	Time	Rel. Time	Version/Compiler	Time	Rel. Time
C++	GCC-4.9.0	0.73	1.00	Visual C++ 2010	0.76	1.00
	Intel C++ 14.0.3	1.00	1.38	Intel C++ 14.0.2	0.90	1.19
	Clang 5.1	1.00	1.38	GCC-4.8.2	1.73	2.29
Fortran	GCC-4.9.0	0.76	1.05	GCC-4.8.1	1.73	2.29
	Intel Fortran 14.0.3	0.95	1.30	Intel Fortran 14.0.2	0.81	1.07
Java	JDK8u5	1.95	2.69	JDK8u5	1.59	2.10
Julia	0.2.1	1.92	2.64	0.2.1	2.04	2.70
Matlab	2014a	7.91	10.88	2014a	6.74	8.92
Python	Pypy 2.2.1	31.90	43.86	Pypy 2.2.1	34.14	45.16
	CPython 2.7.6	195.87	269.31	CPython 2.7.4	117.40	155.31
R	3.1.1, compiled	204.34	280.90	3.1.1, compiled	184.16	243.63
	3.1.1, script	345.55	475.10	3.1.1, script	371.40	491.33
Mathematica	9.0, base	588.57	809.22	9.0, base	473.34	626.19
Matlab, Mex	2014a	1.19	1.64	2014a	0.98	1.29
Rcpp	3.1.1	2.66	3.66	3.1.1	4.09	5.41
Python	Numba 0.13	1.18	1.62	Numba 0.13	1.19	1.57
	Cython	1.03	1.41	Cython	1.88	2.49
Mathematica	9.0, idiomatic	1.67	2.29	9.0, idiomatic	2.22	2.93

Installing R and Python

- Python: https://www.anaconda.com/download/
  - Select Python 3.6
  - Follow installation instructions
- R:

  http://cran.utstat.utoronto.ca/
  - Select R 3.4.1
  - Follow installation instructions

Programming/execution environments

- Text editor + command line
  - Write your code in a text editor (vi, emacs, etc.)
  - Compile/execute code from command line
  - Useful for larger, longer running programs
  - Clusters (e.g. Compute Canada)
- Integrated Development Environment
  - Intelligent text editor (syntax error detection, auto-complete, etc.)
  - Integrated compilation/execution
  - Essential for large-scale projects
  - Examples: Eclipse, Spyder

Notebooks

- Designed to jointly keep track of code, outputs, and observations
- Great for small-to-moderate scale data analysis projects
- Jupyter Python notebook
  - Distributed with Anaconda
  - Runs within a web browser
  - Demo
- Rstudio's notebook

Learning Python

- How to Think Like a Computer Scientist in Python: Interactive Edition

http://interactivepython.org/courselib/static/thinkcspy/index.html#

➔ Get familiar with Chapters 1-12

Python Language reference:

https://docs.python.org/3/reference

Python Standard Library Documentation:

https://docs.python.org/3/library

Some key Python libraries

- Numpy: Fast array computation
- SciPy: Numerical computing
- Matplotlib: Data visualization
- Pandas: Data structures and data analysis
- SymPy: Symbolic computation

➔ All are available at: https://www.scipy.org/

- Scikit-learn: Machine learning
- JSON/Pickle: Data management

Learning R

- R is distributed automatically with Anaconda, under Rstudio
- R Notebooks (similar to Jupyter notebooks for Python)
  - http://rmarkdown.rstudio.com/r_notebooks.html
- Useful R tutorials:
  - Basics: https://www.datacamp.com/courses/free-introduction-to-r
  - Videos:

  http://lorainelab.org/statistics-and-r-for-the-life-sciences/
  - Cheatsheets:

  https://www.rstudio.com/resources/cheatsheets/
- Useful packages: Bioconductor for genomics data

Don’t reinvent the wheel

- How to search for relevant tools?
  - Talk to friends, colleagues
  - Look at tools used in publications
  - Google, youtube, etc.

- http://rmarkdown.rstudio.com/lesson-1.html
- https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf

- https://www.upwork.com/hiring/data/r-vs-java-vs-python-which-is-best/

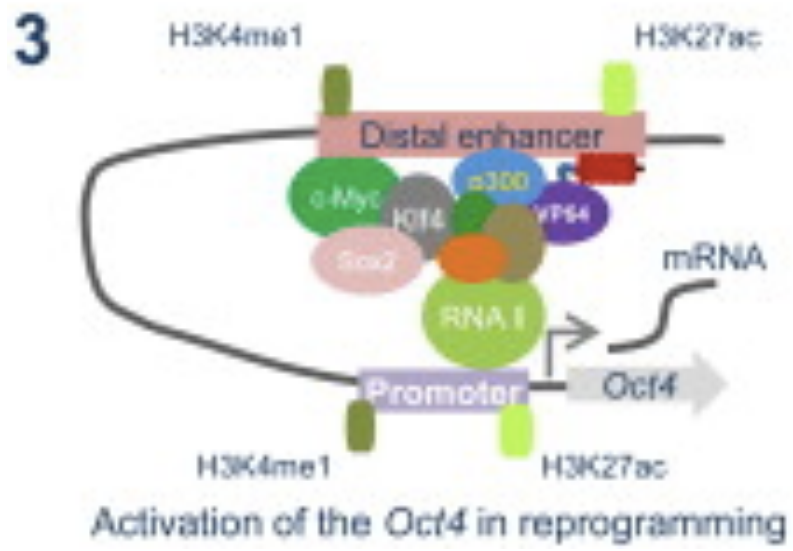
- https://www.datacamp.com/courses/free-introduction-to-r
- Introduction to Data Science with R, by Garreth Grolemund
  - https://github.com/rstudio/Intro/tree/master/slides

- http://www.r-tutor.com/elementary-statistics

Two challenge projects

- Goal: Get some programming experience in R and Python
- Two projects:
  1. Analysis of ChIP-seq data in Python
  2. Analysis of metagenomics data in R
- How?
  - Pick the language you are *least* familiar with, learn it, and start working on the project
  - On Thursday, sit with an expert (other student, TA) to review progress
  - Finalize project
- No need to submit anything – you are doing this for yourself

Python project – Transcription Factor Interactions



Activation of the *Oct4* in reprogramming

- Background:
  - Transcription factors are proteins that bind specific genome locations and affect the expression of nearby genes.
  - Humans have about 2000 different transcription factors.
  - TFs can interact with each other when bound to the DNA. Each TF has a specific set of other TFs it can interact with.
  - The binding of TFs to the genome can be mapped in a particular cell type using a Chip-seq experiment, which identifies regions of 100-500 bp that contain an occupied binding site for the TF under study.

Python project – TF Interactions

Goal: Identify potential interactors of the CTCF protein in GM12878 cells, by analysis of Chip-seq data

Data: Download ENCODE data Chip-seq at
http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeRegTfbsClustered/wgEncodeRegTfbsClusteredV3.bed.gz
and
http://hgdownload.cse.ucsc.edu/goldenPath/hg19/encodeDCC/wgEncodeRegTfbsClustered/wgEncodeRegTfbsCellsV3.tab.gz

and read about the data here:
http://www.genome.ucsc.edu/cgi-bin/hgTables?db=hg19&hgta_group=regulation&hgta_track=wgEncodeRegTfbsClusteredV3&hgta_table=wgEncodeRegTfbsClusteredV3&hgta_doSchema=describe+table+schema

Approach: If two TFs interact, their binding sites should often be co-located close (within 100bp)

1. Think of an reasonable statistical measure of "co-location" of TF binding sites
2. Write a Python to assess the degree of co-location between CTCF and each of the other TFs studied by Encode, in cell type GM12878
3. Identify the five TFs with the strongest evidence of interaction
4. Using Matplotlib, produce graphs that meaningfully summarizes your data

R project – Tara Oceans project

Background:

- Metagenomics: The species composition of environmental samples can be obtained by sequencing DNA (often 16S gene) and comparing it to a database.
- The Tara Oceans project has collected and sequenced hundreds of ocean samples from across the globe
- For each sample, they estimated the abundance of thousands of species (bacteria, protozoa)

R project – Analysis of the Tara Oceans project data

Goal: Study the distribution of microbes in the ocean
http://science.sciencemag.org/content/348/6237/1261359.long
http://ocean-microbiome.embl.de/companion.html

Data: Tara Oceans project:
http://www.cs.mcgill.ca/~blanchem/QLSC600/miTAG.taxonomic.profiles.release.tsv
And
http://ocean-microbiome.embl.de/data/OM.CompanionTables.xlsx

Approach:

1. Load the species abundance and geographic location data into R
2. Identify species whose abundance best correlates with latitude, longitude, depth
3. Cluster samples based on co-occurrence
4. Cluster locations based on abundance profile
5. Play with geographical visualization tools (try the maps package)
6. Formulate a question that interests you about this data, and answer it using R