(Bio)python course at CSC

janne.ravantti@helsinki.fi
Today

1. Introduction
2. Python-programming warm up
3. Biopython-library
4. An example of Biopython’s usage: PhiNN capsid protein’s friends
5. CpG-island-exercise
6. Recap
Ask questions!
Bioinformatics & Python?

Comp.sci  Bioinfos’ methods development  Processing biological data  Biology
Statistics  Mathematics
Programming requires peculiar way of thinking 
(but it can be learned!)
Good* way to learn programming is to program!

*The Best?
Goodness of your program is (mostly) defined by the biological question
Opinionated tips for programming

- Start small (e.g. not aligning 1000-genomes humans!) and one step at a time
- Don’t worry (about errors) (too much - testing is important, but...)
- Think! What...:
  - is the **biological question**?
  - is the **data**?
  - the program is supposed to do (methods, algorithms, ...)?
  - input (DNA-sequence? Set of RNA-seq data, names of plants, …)
  - can go wrong => then what (disk full, memory full, bad methods, too little data, ...)?
- Learn to **save your code** (naming, locations, even something like git)
Caveats

- **Everything** changes...
  - Data (WXS => WGS => WGBS; RNA-seq, ...; HG37 vs. HG38...)
  - Methods (bowtie => bowtie2 => bwa mem => minimap2 => ...)
  - Links go stale (404 Not Found)
  - Python 2.7 => 3.6+
  - Python-libraries (Standard library, Biopython, ...)
  - Operating systems / platforms
  - System libraries

=> Do not get stuck with the old unless absolutely necessary, but don’t worry too much about newest trends!
EXERCISES - warm up

- Make a Python-program that reads a multi-FASTA-file, cleans up the header line to have only ID & gene-name and prints headers and sequences to standard output as an multi-FASTA-file again:

```bash
>lcl|NC_007217.1_prot_YP_271858.1_1 [gene=HPSH1_gp01] [protein=ORF 1] [protein_id=YP_271858.1] [location=164..421]
=>
> YP_271858.1_##HPSH1_gp01
```

Tips: you can use file SH1_prots.fasta for the exercise, about FASTA-format: https://en.wikipedia.org/wiki/FASTA_format
EXERCISES - advanced (, if you have time...)

Make a Python-program that reads all files in given directory and makes an multi-FASTA-file from protein-sequences ignoring the other files.

Tips:
● you can use directory example_data/sequences for the exercise
● use Python’s pathlib-library to list the files
Biopython - introduction

● An Open Bioinformatics Foundation project
  ○ https://www.open-bio.org/wiki/Projects
  ○ The idea is to provide common programming tools for various languages, including Python

● http://biopython.org
● http://biopython.org/DIST/docs/tutorial/Tutorial.html

● Can be called in Python by:
  import Bio
  or specific sub-library e.g.
  from Bio import SeqIO
  import Bio.SeqIO  # if import fails, install biopython-library
Biopython - capabilities

- [http://biopython.org/DIST/docs/tutorial/Tutorial.html#htoc2](http://biopython.org/DIST/docs/tutorial/Tutorial.html#htoc2)

- Mainly: dealing with biological sequences (DNA / RNA / proteins)

- *E.g.* nice ways to change sequence formats from command line:

```python
import sys
from Bio import SeqIO
SeqIO.convert(sys.argv[1], "fasta", sys.argv[2], "clustal")
```

*N.B.* here `sys.argv[]` takes filenames from a command line
(Bio)python - caveats

- Largish project based on **volunteers**
  - some parts might break ("API changes")
  - some parts might get much, much better

- Sometimes (Bio)python is not the best solution (hammer vs. nail)
  - sequences are strings => easy to manipulate with Python itself
  - other libraries exist (numpy, pandas, …)
    - e.g. data in tables, csv-files, …
  - other tools exist (e.g. EMBOSS)

=> learn to use also Linux & command line tools (CSC has nice courses!)
Biopython - sequences

● Sequences are everywhere in bioinformatics

● Biopython has many, many, many ways to work with sequences

● Sequences are string-like objects, with some additional information
  ○ all Biopython’s sequences have alphabet
  ○ alphabet defines type of the sequence (DNA / Protein)

● Biologically relevant methods per sequence-type
  ○ e.g. my_dna.reverse_complement(); my_protein.translate()

● Let’s try out! => http://biopython.org/DIST/docs/tutorial/Tutorial.html#htoc17
Biopython - sequences - exercise

- Make a Python-program that asks user to input a DNA-sequence and then translates the sequence to protein sequence

- Questions:
  - How to determine, if the input is actually DNA-sequence?
  - How to count number of STOP-codons?

- Modify your program to ask also FASTA-header and a filename and then write the protein sequence into the new file
Biopython - Blast

- Blast is arguably the single most important program in bioinformatics
- BioPython supports both WWW and local Blast-searches
- http://biopython.org/DIST/docs/tutorial/Tutorial.html#htoc87

Caveats
- Blast has multitude of options - you need to understand them too!
- Parsing Blast output is a bit complicated => see http://biopython.org/DIST/docs/tutorial/Tutorial.html#fig:blastrecord
Biopython - Blast - exercise

- Make a Python program that asks a filename and then Blasts the sequences found in the file (assume one sequence per line). Print the best hit’s title per sequence.

Tip: you can use “my_sequences.txt” file for testing.
Biopython - Entrez

- Entrez is an interface to NCBI’s databases such as PubMed and GenBank
- Biopython supports Entrez in similar manner to Blast (handles, XML-output)

  - [http://biopython.org/DIST/docs/tutorial/Tutorial.html#htoc111](http://biopython.org/DIST/docs/tutorial/Tutorial.html#htoc111)

- The output parsing can be confusing for a beginner (and for me…)
  - you can use retrieved sequences in BioPython with Bio.SeqIO
  - Pubmed entries can be parsed using Medline-parser
    - [http://biopython.org/DIST/docs/tutorial/Tutorial.html#htoc129](http://biopython.org/DIST/docs/tutorial/Tutorial.html#htoc129)
Biopython - Entrez - exercise

- TT-Seq is a recent RNA-seq technique that maps a transient transcriptome.
- Make a Python-program that will find all TT-seq articles in Pubmed and prints how many there are
- Advanced: print each article’s authors
Remember!

SAVE YOUR WORK FREQUENTLY!
Python at CSC

Kimmo Mattila’s presentation
Example: PhiNN* P1-protein with Biopython

1. Entrez => get PhiNN

2. Get P1 (capsid) protein of a related phage:
   ○ Blast & Process results

3. Align PhiNN and “the friend” - pairwise2

4. Find possible related structures

5. Visualize (needs nglview: [https://github.com/biopython/biopython/issues/1331](https://github.com/biopython/biopython/issues/1331))

http://jgv.microbiologyresearch.org/content/journal/jgv/10.1099/vir.0.000063
Links to further reading


http://www.dalkescientific.com/writings/NBN/

https://github.com/peterjc/biopython_workshop (uses Python v. 2.7!)
Other similar libraries to Biopython

http://scikit-bio.org/

https://pypi.python.org/pypi?%3Aaction=search&term=bioinformatics&submit=search

http://pycogent.org/

https://python-graph-gallery.com/ (for graphs)
Finally, CpG-islands

- Important in gene regulation => epigenetics
- Mutations in CpG-islands can cause diseases like cancers
- Simple(?) definition:
  - Region of DNA with at least 200 bp, in which:
  - GC percentage is greater than 50%
  - Observed-to-expected CpG ratio (OE) greater than 60 %
    - \[ OE = \left( \frac{(\text{number of C} + \text{number of G})}{2} \right)^2 / \text{length of sequence} \]
CpG-islander

● Make a Python-program that finds CpG-islands from a given DNA-sequence

● Test your program with sequences “finland_mtDNA.fasta” and “italy_mtDNA.fasta” - do you see any differences in CpG-islands?
Recap

- **Python** is well-suited for doing bioinformatics
  - easy(?) to learn
  - widely available
  - good standard library ("everything & kitchen sink!")
  - good / stable external libraries
  - performant with e.g. numpy

- **Biopython**
  - is strong especially in sequence manipulations
  - also WWW-searches (Blast, Entrez, …)
  - special sub-libraries to structure manipulations, phylogenetics, …

- However, things change, so plan accordingly
THANK YOU!

janne.ravantti@helsinki.fi