Topic 2: Programming for biologists

Non-command line options

- Galaxy
  - Open-source, run much of the same tools
- Geneious
  - Commercial
- CLCbio Genomics Workbench
  - Commercial, includes assembly

Why use command line?

- Files too big to open fully
- All steps recorded and repeatable
- Powerful text editing tools
- Generally faster than GUI based methods
- Most scientific programs do not have a GUI.
- Free

Programming Languages

- C++
- Shell scripting
- Perl
- Python
- R

C++

- Very fast
- Few prepackaged tools
    - More control
    - Much longer code

Shell scripting

- Basically, taking command line arguments and putting them in a reusable script
- Easy to write.
- Generally for executing scripts in other languages
- Can include command line tools (sed, grep, awk)

Perl

- Slower than C++, but still fast.
- Some prepackaged functions.
- Lacks complicated data structures.
- Good for reformatting data.

Python

- Slightly slower than Perl.
- More prepackaged functions, including scientific methods and plotting.
- Popular
- Good for reformatting data.

R

- Slowest option.
- Many packages for specific scientific tasks and statistics.
- Great at plotting.
- Harder to use with big data (GB+ files), although work arounds exist.

Recommendations

- Genomic dabbler
  - Shell and R
- Genomic scientist
  - Shell, R and Python/Perl
- Bioinformatician
  - Shell, R, Python/Perl and C++

Coding Example!

Compute Canada servers

- Uses a Slurm scheduling system
- Tasks must be submitted in specific bash scripts and will run when they get priority
- You must specify how much CPUs, RAM and time you need.
- You can use ‘salloc’ to get an interactive job, which is like working on your server.

Tools we will learn:

- Unix shell
- grep and sed
- byobu