Introduction to BIOC8142

Bioinformatics Overview / *Unix for Smarties*

BIOC 8142
February 11, 2013
Bill Pearson wrp@virginia.edu 4-2818 Jordan 6-057

- Goals of today's lecture:
- Introduction to the course - programming for Bioinformatics and Genome Analysis
- Topics in Bioinformatics
- Introduction to Unix - file systems/editors/bash
- Connecting to franklin.achs.virginia.edu
- Transferring files

1

What should you do to reinforce the lecture material?

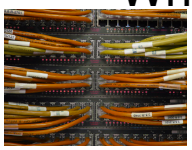
- More information on Unix (ignore the history, learn the filesystems, shell)
  www.ee.surrey.ac.uk/Teaching/Unix/ (tutorials one-six)
- More information on Editors (nano)
  mintaka.sdsu.edu/reu/nano.html
- More information on Perl
  www.perl.com/pub/2000/10/begperl1.html

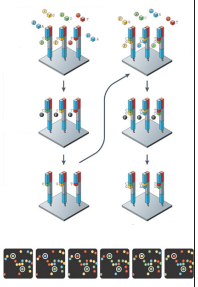
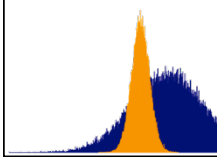
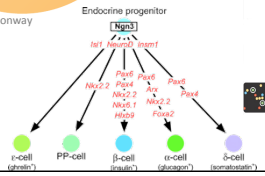
How will you be graded on this material:

- Homework on collab

2

What is bioinformatics?



Bioinformatics and Computational Biology

- Computer algorithms, pattern recognition, machine learning, statistics
  - Similarity searching/multiple sequence alignment (BLAST)
  - Mapping genomic reads/assembling reads/genomes
  - Gene prediction/structure prediction/function prediction
  - Phylogenetic tree reconstruction, co-evolution, ...
- Management of biological data
  - Databases: Sequence, ENSEMBL, ENTREZ Gene, protein domains, variation, mutation, protein structure, genome, taxonomy, active sites, transcription factor binding sites, ontologies (NAR database issue)
    - Where is the data
    - How reliable is it?
  - Data formats: FASTA, FASTQ, SAM, BAM, BED, GFF, Genbank flat file, Uniprot, PFAM
  - Management tools: SQL

UNIX for Smarties

•Aaron J. Mackey
•Bill Pearson

computing environments

- UNIX computing: the command line
  - "shell" environment, built-in tools
  - infinitely extensible: download/install tools
    - most bioinformatics algorithms/tools are implemented as UNIX command line utilities or libraries
    - or, write your own algorithms/tools from scratch
  - highly automatable by scripting (Perl, Python, etc.)
  - interoperation between tools only limited by your ability to glue together input/output formats
  - almost entirely free access to tools
- demo

UNIX concepts

- Linux (RedHat, Ubuntu, &), AIX, Solaris, & Mac
- shells: sh, bash, csh, tcsh, zsh, ksh, &
- commands: ls, cd, more, cat, echo, &
- flags and arguments: ls l, or: cd ~
- inputs and outputs: stdin, stdout, stderr
- redirecting input/output from/to a file
- piping output/input between commands
- environmental variables: $PATH, $PWD, etc.
- shebang (#!) scripts

Unix file editors

- UNIX newlines are "`\n`"
  - PC is "`\r\n`"; Mac is "`\r`" (sometimes);
- Use a UNIX editor on UNIX files:
  - `nano`
  - `emacs` vs. `vi/vim`
- When programming, use an IDE
  - `eclipse` (www.eclipse.org)
  - Komodo Edit (www.activestate.com/komodo-edit)
  - do not use: Word, NotePad/WordPad, TextEdit, etc.
- every editor has pros and cons, try a few

8

File system navigation

- `cd` – change directory
  - `cd ~` - change to "home" (~, tilde) directory
- `pwd` – print working directory (current dir.)
- `ls` – list files
- `pushd/popd` – `cd`, but remember stack
- `find` – search through filesystem

Unix filenames

- Case matters (always use lowercase):
  - `gstm1_human` ≠ `Gstm1_Human`
- Only use letters (lc), numbers, and '_'. (you can use other characters, but then files are difficult to specify/rename/delete)
- "." and "/" are special – never use "/"; only use "." for suffix (one per filename, not in directories/ folders)

File system manipulation

- `cp` – copy files
  - `cp file.name .` - copy the file to the current directory
- `mv` – move files
- `rm` – remove files
- `rmdir` – remove directories (must be empty), or "`rm -r -f`"
- `touch` – make a new, empty file
- `mkdir` – make a new, empty directory

File Inspection

- `more` – read/browse through a file/stidin
- `cat` – contents to stdout/ concatenate files
- `head/tail` – look at top/bottom of file
- `od` – look at bytes in file (`od -c file | more`)
- `sort` – sort lines in a file (`sort -n – numeric`)
- `cut` – extract specific columns
- `uniq` – report unique lines (remove duplicate accessions)
- `grep` – search/count matching lines
- `wc` – count words/lines/characters

Unix Permissions

- `chmod` – change permissions on file/dir
  - `u/g/o` – user (you), group, others
  - `r/w/x` – read/write/execute (look inside for directories)
  - `chmod +r data.file` – let others read the file
  - `chmod +x program.pl` – make script "eXecutable"
  - `chmod -R go+r .` – recursively let others read files below "." (your current directory)
- `chown` – change owner of file (rarely used)
- `chgrp` – change group of file (also rare)

Unix host (machine) status

- `top/ps` – what is running, how long, how much memory
  - `ps -fu wrp` – tells what programs wrp is running

```
UID        PID  PPID  C STIME TTY          TIME CMD
wrp      21128 21126  0 11:19 ?        00:00:01 sshd: wrp@pts/0
wrp      21129 21128  0 11:19 pts/0    00:00:00 -tcsh
wrp      23615 21496  0 14:54 pts/2    00:00:00 ps -fu wrp
```

- `kill` – force quit a process

```
kill -9 21129
```

  - `-3 -> ^C, -9 ->` nuclear option

- `df –h` - disk space available
- `du` – disk space usage

other UNIX commands

- `builtins` - list available shell commands
- `which/where` - find path of commands
- `time` - measure how long something take
- `echo/tee` - print/report text
- `wget/curl` - download files
- `gzip/gunzip/bunzip/zcat` - compressed files
- `ssh/scp` - login/copy to/from remote hosts
- `history` - what have I done previously
- `man` - get help

redirection, pipes, replacements

- `>` - redirect stdout into file, replace existing
- `>>` - redirect stdout into file, appending
- `|` - redirect/pipe stdout to stdin of next command

Unix programs rarely ask for file names, they assume you will send to "`>stdout`"

```
ls –l  > file.list
```

(is file.list in the list of files?)

- `` `backticks` `` - replace with captured stdout

globs (wildcards)

- `*` wildcard matching
- `{a,b,c}` multiple choice
- `[a-c], [1-5,9]` range/set choice
- `^` negation

- `ls -l *.bam`
- `ls chr[1-23,X,Y].bed`

Environment variables

- `$USER` - who you are
- `$SHELL` - what shell you are running
- `$PWD` - your current working dir
- `$PATH` - where the shell will go to look for commands
- `$EDITOR` - your default editor

- set in your `.cshrc/.bashrc`, `set/setenv`

UNIX editors: learn (at least) one

- nano
  - simple, easy
  - no mouse, use arrow keys
  - how to quit: ctrl-X (all commands at screen bottom)
- vi
  - not so simple to use
  - guaranteed to be on any UNIX machine
  - often the default $EDITOR
  - how to quit: [colon]q![enter]
- emacs
  - also not so simple to use
  - incredibly versatile, customizable, programmable
  - how to quit: ctrl-X ctrl-C

Using emacs

- `sh>emacs`
- `^x^c` *exit*
- `sh>emacs filename`
- `type some stuff`
- `^f,^b,^p,^n` *forward,back,prev, next*
- `^x^s` *save it*
- `^x^c` *exit*
- `sh>`

Intermediate emacs

- sh>emacs random.pl
- ^s, ^r *search forward, reverse*
- ^a, ^e *start, end of line*
- *esc* = M-
- M-<, M-> *start, end of buffer*
- M-% *query-replace*
- ^k *kill-line (and put in kill buffer)*
- ^k^k *delete line and linefeed (EOL)*
- ^y (yank – insert kill buffer)
- ^x 2, ^x 1, ^x o (multiple windows)
- ^u (repeat number)
- ^h (help,^h-t tutorial, ^h-a apropos)

alternative scripting languages

- Perl
  - once the mainstay of WWW/CGI programming
  - long history == lots of reusable packages
- PHP
  - mainly limited to dynamic WWW pages
- Python
  - extremely popular
- Ruby
  - compact, expressive
- …

Homework

1. Get an ITS unix account
2. On your computer, login to your account on franklin.achs.virginia.edu
   - Windows: download/install SecureCRT
   - Mac: open terminal

     `slogin unix-id@franklin.achs.virginia.edu`
3. create a file containing your `$PATH`

   `echo $PATH > path.file`
4. list the contents of the file
5. Make a copy of the file
6. Make a sub-directory (folder) called "data"
7. Move the path.copy file into the `data` folder
8. List the contents of the `data` folder
9. BLAST format sequence libraries are in /data/slib/bl_dbs
   - How many *.psq files are there?
   - How many files start with "swissprot"
   - Look at the "swissprot.nam" file
   - How many different libraries can you find (hint, there is often one .pal file per protein database)

Homework (cont.)

The blastp program is at: `/seqprg/bin/blastp`
The swissprot database (formatted for BLAST) is at:

`/data/slib/bl_dbs/swissprot`

1. Download a protein sequence from the NCBI (`www.ncbi.nlm.nih.gov`) in FASTA format (try `GSTM1_HUMAN`)
2. Transfer the sequence file from your computer to `franklin.achs.virginia.edu` using scp (Mac) or SecureFX.
3. Move the transferred file into the "`data`" folder
4. In the "`data`" folder, Run a `blastp` search:

```
blastp –help
blastp –db /data/.../swissprot –query gstm1_human >gstm1_human.blast
```

5. List the size of the blast output file
6. View the contents of the file
7. Count the lines in the file
8. Run the search producing "tabular" output
9. Use "`cut`" to isolate the query, subject accessions, bit score, and Expect from the tabular output file