Texas A&M University

Department of Electrical and Computer Engineering
Dwight Look College of Engineering

BIOL 689 Digital Biology

Zhiyang Ong [1]

Notes from the class and revision notes

June 29, 2014

[1] Email correspondence to: 🖂 ongz@acm.org

Abstract

Notes from my BIOL 689 Digital Biology class at Texas A&M University in the first summer session of 2014.

To be completed. . .

Revision History

Revision History:

1. Version 0.1, June 3, 2014. Initial copy of the report.
2. Version 0.2, June 23, 2014. Added material for the first lecture.
3. Version 0.3, June 23, 2014. Added material about computing (SSH, rsync, and Git) for the second lecture.
4. Version 0.X, June ???, 2014. Added BLAH and modified BLAH:
   (a)
5.

Contents

Chapter 1

Introuctory Material

Prof. Rodolfo Aramayo is my class instructor.

This is a UNIX-based class.

My username is db00XX. See the comments of this statement for my username.

Ricardo is my (lab) teaching assistant (TA).

Genome assembly is still an unsolved problem.

GitLab (from GitHub, Inc.) will be used for the first time in this class. Three concepts will be covered in the introduction: Wiki for the class, which contains the standard class information; the code repository that the Wiki uses; and the *Git* version control system.

The outline of the class (i.e., syllabus and class schedule) will be modified as the semester progresses.

On Thursday, June 5, 2014, we will cover genome analysis, gene models, and gene files. Next week, we will cover next generation DNA sequencing. We will also look at library construction methodology and techniques, and associated challenges. Next Thursday, we will also look at "small reads." We will write small scripts to process small data sets, and organize the pipeline (or design the algorithm) for the program/script. That is, design the control and data- flow graph of the algorithm. Furthermore, we will look at genome mapping, genome assembly, data display (i.e., data visualization), and transcriptome mapping and transcriptome assembly. Subsequently, we will be given data sets from the professor to carry out (machine) learning for pattern classification, and explore read archives (with unknown outputs).

Chapter 2

Administration

This is a chapter about administrative details.

2.1 Tuesday, June 24, 2014

There were problems with the SRA files, which reads were not mapping; the reads in the SRA files were not mapping properly. There were also problems with the network/server.

Chapter 3

Software Engineering Basics

3.1 UNIX Basics

My TA, Ricardo, suggested using Guake (http://en.wikipedia.org/wiki/Guake) as a substitute for the common/normal `Terminal` application.

We will be using the `Terminal` to do a lot of our work in this class. Prof. Rodolfo Aramayo briefly talked about the history of UNIX and its derivatives, such as *Linux*, *BSD*, *Oracle/SUN Solaris*, and *Mac OS X*. UNIX was started at *Bell Labs*. He also talked about the UNIX philosophy.

We shall operate in the UNIX environment via text files. Everything (including directories) is a file in UNIX. Some files can be read visually (i.e., text files), while others (i.e., binary files) cannot.

The kernel is the heart of the operating system. The UNIX shell (accessed via applications, such as the `Terminal`) is an application that allows users to interact with the kernel indirectly.

Anatomy of UNIX commands: `command_name [options] [arguments]`. Double dashes for options of UNIX commands cannot be combined. However, for options for single dash lines, they can be combined.

The "`man`" page is the UNIX manual. To find documentation of a UNIX command, use the `man` command.

UNIX commands to learn:

1. alias: "alias ll"
2. apropos: "apropos copy" would search the UNIX "man" pages for the keyword "copy".
3. cat: conCATenate
4. cd
5. chmod: Change mode
6. clear
7. cp: cp –version
8. dir -l
9. date
10. du: "du -hd 0 ." list the size of the directory in KB, and "du -hd 1 ." list the size of the directory and its files. "df -h" indicates the size of the directory and its contents.

11. echo: “echo -e” refers to `echo` enhanced, which redirects the output in the UNIX pipeline to a file. *echo -e “ ‘date‘” > tata1.* “echo $PATH”
12. file
13. history
14. info cp
15. less
16. ls [-al]
17. more
18. mkdir
19. mv
20. pwd
21. rm
22. rmdir
23. rsync:
    (a) An example of how the command can be used is: “rsync -v username@host:∼/path/to/file .”. This commands copies the file at the specified path to the current working directory. The “-v” option runs the UNIX command in verbose mode.
    (b) Its “-vr” option runs the command recursively in verbose mode.
    (c) Prof. Aramayo mentioned something about an option that transfers files with automatic compression and decompression. Is this option “-a”, or something else? Use the “tar” command to compress/uncompress files.
24. script:
    (a) Use this UNIX command to keep a log of the terminal session. That is, use it to record the terminal session. Some `Terminal` applications allow people to save the terminal session as a text file.
    (b) Prof. Aramayo suggested redirecting the standard output stream to a file to keep a record of the commands that are executed and their standard output. However, this requires appending each UNIX command with the redirection symbols.
    (c) `unix-command` > zlog &. This creates a new file, if the file `zlog` does not exist, and redirects the standard output to the logfile (`zlog`). Using the ampersand symbol allows the command to be run in the background while allowing me to continue using the `Terminal` application.
    (d) `unix-command` >> zlog &. This creates a new file, if the file `zlog` does not exist, and redirects the standard output to the logfile (`zlog`). However, if the file `zlog` already exist, the redirected standard output would be appended to the end of the logfile (`zlog`).
    (e) Put `&> slog &` at the end of each command???
25. touch
26. tree
27. type: type zrio
28. whatis
29. which: which blastn

Use “tab” to autocomplete filenames and directory names. Avoid using spaces in filenames and directories to keep file and directory access simple.

Directory access: The “.” file is the current working directory, and the “..” is the parent directory. A directory can also be called a folder. By using the `cd` command, I can return to my home directory.

You cannot undo operations in UNIX. Hence, save and backup files before performing removal operations in UNIX. There is also no "trash can" or "recycle bin".

Microsoft Excel has a maximum limit of 65,000 rows in the spreadsheet. Hence, this limits the amount of information that I can process with Microsoft Excel. To process more data, such as GBs or TBs of data, I need other software applications or develop my own computer program.

Symbolic links in UNIX are like shortcuts or aliases in Windows. An example of creating a symbolic link is: "ln -s ../01/test01".

The human genome has been decoded into a file about 7 TB.

The colon ":" serves as a dummy placeholder to remove the contents of a file; ": > filename"

Standard output stream, `stdout`, is described along with exit signals of UNIX processes. Standard error output stream will write to the standard error output file. UNIX redirection for standard output and error streams are described.

Use *tree* to show contents of a directory as a tree.

Discussed UNIX path redirection, pipelining of UNIX commands, and separate execution of UNIX commands (using the semicolon ";" symbol).

Covered special/escape characters to use tabs and newlines to print information.

Covered information on how to go to the "home" directory. "∼" refers to the home directory.

Covered absolute paths and relative paths in UNIX.

Detailed explanation of the "ls" command. It indicates when the file has been created/modified. It also indicates the size of the file in bytes. It also indicates the username ("db0015") and the group ("student") that I belong to. Permissions to access files are determined by the group that I belong to. File permissions are indicated for read, write, and execute. They are set for individual users, groups, and everybody with access to the computer network/system. File types are indicated for directories ("d"), regular/normal files ("-"), and symbolic links ("l").

Most files have the file permissions set as 755.

Discussed how to create aliases in UNIX.

Configure my UNIX environment with the ".bashrc" (or ".bash_profile") file.

The `.profile` is used by `shell`, and is equivalent to `.bashrc` for Bash.

GUI-based *Galaxy* is used for this class.

Regarding file transfer, avoid unencrypted file transfer that can be accessed by others. People can listen or snoop on the packet transmission of files, and find out what you are doing. An aside: Email service providers, such as Google, transmit emails between their servers without encryption.

There are many applications for downloading files from the Internet. The applications curl and *wget* are more common for downloading files.

The UNIX command ifconfig gives you information about computer networking for your computer or computing account (if you are connected to a remote computer).

Further references in my research database about UNIX include the following: [?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?].

3.1.1 SSH Basics

SSH is an application that allows me to connect securely to another computer that is connected to the same computer network, or to the Internet. It uses encryption for network connection, including file transfer between computers in the same network, or between different networks. Its various levels of encryption correspond to various levels of simplicity in the encryption.

The actual/real SSH application requires paid subscription. However, its open source variant is FREE!!!

SSH key generation creates a pair of private and public keys. Keep the private key private to myself (only). Allow others to have the public key, so that a valid authentication of myself can be made.

rsync is an application for file copying and synchronization between different computer accounts. It does not copy all files in your directory, but copy modifications to existing files and copies only new files. It transfers files in compressed format. That is, it transfer files between different computers by synchronizing them via delta modifications. This is because copying entire directories of huge files take a lot of time. Hence, use rsync to carry out file transfer to save time. rsync uses the public key of SSH (from SSH key generation) to connect the local machine to the remote machine. For example, I can create the authentication file (SSH public key) and transfer the public key to the remote machine.

3.1.1.1 Shell Scripting Basics

To review the basics of UNIX shell script, see my internship report (and associated material) for my internship at the Institute of Microelectronics, Singapore [?].

3.1.1.2 Related Issues

Download data to group directory on “Geiger”, so that I do not corrupt the local machine.

Use “tree” to find out the directory structure of the specified directory.

For class on June 17, 2014, clone the repository from Prof. Aramayo, https://geiger.tamu.edu/gitlab/raramayo/digitalbiology_project_summer2014. Work on this directory to practise the UNIX sub-lesson for today.

3.1.2 Version Control (Or Revision Control)

Revision control is also known as version control or source control. It is an aspect of software configuration management (SCM). For this class, `Git` [?, ?, ?, ?, ?] will be our revision control tool.

`git status` tells me the status of my `Git` repository. `git diff` tells me the difference between different commits/stages of my repository. Watch videos about `Git` to learn more about `Git`, via hyperlinks provided on the class Wiki. Also, read "`Git` in the Trenches."

While adding files to my `Git` repository, use the `Markdown` language to provide some structure to the presentation of information for my project repository. Save files in the `Markdown` language as `filename.md`. `Markdown` is a document markup language, just like LaTeX, HTML, and XML.

3.2 Other Computing Issues

Launch system monitor to track how much CPU time are processes taking.

Chapter 4

Bioinformatics: Computational Genomics/Genetics

This is a chapter about bioinformatics (particularly computational genomics/genetics).

At the end of the class, I should be able to use `Bowtie` (or a comparable bioinformatics tool) to create my own `Bowtie` index/directory for read alignment, when it aligns short DNA sequences (reads) to the human genome.

4.1 Introductory Bioinformatics

4.1.1 Advice Regarding Bioinformatics

Some advice regarding bioinformatics are:

1. When doing a lot of mapping (e.g., genome mapping and transcriptome mapping), put aliases to all datasets in a directory. Don't make multiple copies of files/directories. This redundancy takes up a lot of storage space on the hard disks/drives. When operations/processes are carried out on these copies, they will bog/slow down the computer network.

4.2 Bowtie

`Bowtie` is a software application for sequence analysis that reads short reads from a DNA/genomic sequence [**?**].

Some flags for `Bowtie` (Version 1) are discussed as follows:

1. -v0. Do not allow any misalignment errors.
2. -m1. Do not report multiple (> 1) misalignment errors.
    (a) Therefore, `-m100` does not report > 100 misalignment errors. Reads can map between 0 to 100 times, but not more than 100 times (1–100 times), such as 1X, 3X, 49X, or 100X. That is, it suppresses reads that maps to > 100 times. It does not demand exact mismatches.
    (b) It does strictly not depend on the length of the reads (the size of the mapping, or mapping size).
3. `-n` versus `-v` is about global versus local alignment:
    (a) Regarding global alignment, one sequence is matched with another.
    (b) As for local alignment, compare one region of a sequence to another region.

(c) Local alignments are more sensitive than global alignments.
(d) Local alignments are less informative than global alignments.

4. `-v1 -m1` → `-v1 -m100` increases the number of reads by a small amount.
5.

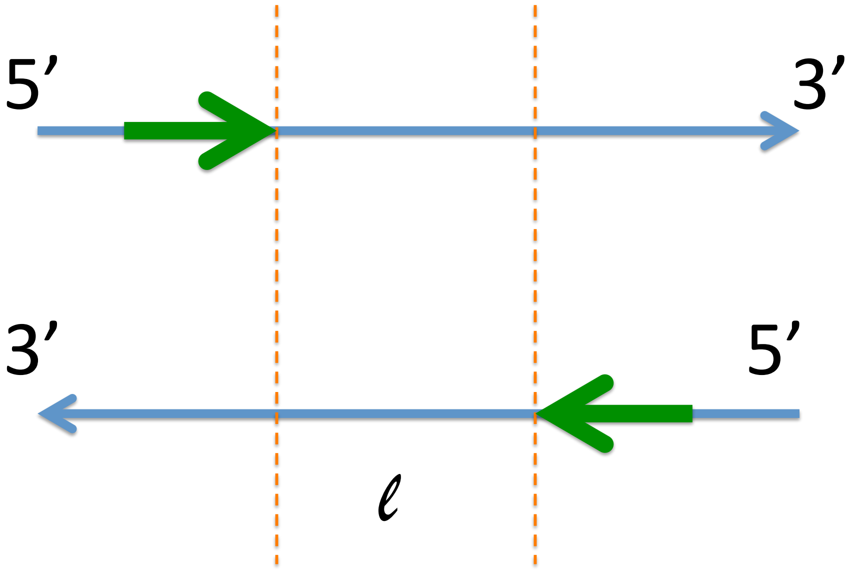

Figure 4.1: An example of Bowtie, as it tries to align a subsequence in a DNA sequence with another subsequence in another DNA sequence.

An example of DNA sequences being read by `Bowtie` is shown in Figure 4.1.

Separate the reads into different classes, especially for genomes with lots of single nucleotide polymorphism (SNPS) → lots of inversion that need to be sorted.

4.3 SAM File Format

The second field of the `SAM` file format indicates various information about the alignment and pairing. It indicates the call of a proper read, or align read. The majority of the reads don’t map well because we are mapping only to the smallest chromosome of the species. The *Neurospora crassa* species has 7 chromosomes... Without paired reads, the numbers are different.

Some calculations to determine such information is demonstrated as follows:

1. $77 = 64 + 8 + 4 + 1$

2. 141 = 128 + 8 + 4 + 1
3. 163 = 128 + 32 + 2 + 1. Properly aligned.
4. 83 = 64 + 16 + 2 + 1
5. 99 = 64 + 32 + 2+ 1

Convert `SAM` files to `BAM` files, which are faster to read. `SAM` files are written according to when they (i.e., the subsequences) are found/read; that is, they are written as the subsequences are found. On the other hand, the `BAM` files can be sorted according to their coordinates of the beginning (and, hence, the end) of the subsequences; *verify this!!!*. These `BAM` files are sorted from left to right. For scientific/experimental control, show that the `SAM` file that is re-converted from `BAM` is sorted; that is, check that the recovered `SAM`-to-`BAM`-to-`SAM` file is sorted.

Use `Fasta` (.fasta) and `.fa` files. Sorted `SAM` files have no header. Reads are mapped and primed in ordered fashion, without the header. They can't be displayed on their own, and require another file for its contents to be displayed. Specifically, an index `BAM` file needs to be provided with the sorted `BAM` file. When the data is viewed (with the script `./igv.sh`), it shows valleys and peaks. The peaks indicate a good probability of having contigs. The valleys indicate a good probability of not having connected contigs (or a high probability of not having a connection is high); use make pair to "weaken connection"??? ... The colored regions are repeated subsequences that indicate polymorphism.

Grab???, scrub???, clone, and sequence DNA. Extract the DNA, build its library, and sequence without cloning. Structure DNA ⟷ sequence, making the library and PCR for sequencing bad. Base-call qualities are rejected. There exists different levels of PCR, or different uses of PCR in this process. When errors are present in the displayed DNA sequence, it could be because something was wrong with the PCR or there were problems during sequencing. The centropid (or centromere)... exists in mitochondrial DNA (or mtDNA)? It repeats regularly.

There exists two `Bowtie` programs. `Bowtie` is different from `Bowtie 2`, and they require different input file formats. Hence, `Bowtie 2` files are for `Bowtie 2` only. #1 mates indicate forward reads, and #2 mates indicate reverse reads. Concordantly, read1 and read2 must be mapped, as expected. Else, if they are not mapped, they cannot be used for genome assembly [discordantly].

`Bowtie 2` has two mapping modes, which are different to separate. They are end-to-end mapping and local mapping.

For the end-to-end mapping mode of `Bowtie 2`:

1. Very fast global mapping
2. Fast global mapping
3. Sensitive global mapping
4. Very sensitive global mapping

For the local mapping mode of `Bowtie 2`:

1. Very fast local mapping
2. Fast local mapping
3. Sensitive local mapping
4. Very sensitive local mapping

Control the number to change the sensitivity, so that diversions can be dealt with. The computational time taken (i.e., performance of the procedure/software) changes with respect to sensitivity. That is, the performance of the `Bowtie 2` algorithm is proportional to sensitivity. As sensitivity increases, the time taken (or computational complexity, or performance) increases.

Genome assembly needs to be carried out, since we fragment DNA and now we have to assemble it. Copy the mapped files and look at the regions between n3m1 and n3m100, and display the results. Show how the `-n` and `-m` parameters affect mapping; that is, genomic mapping is affected by the parameters. Thus far, we are mapping DNA reads.

When mooing RNA reads, things are more different. RNA has lots of contigs, which form a library. Genomic maps are generated from transcriptome mapping, which has its equivalent genome (and, thus, its own contigs). They depend on contigs, and are products of the genomic assembly process. Here, we are constructing the DNA sequence from the contigs.

Reads exist in eons. They map to introns/exons. However, at the boundaries of introns and exons, they do not follow rules. Therefore, we cannot use `Bowtie` to resolve the mapping. Consequently, we need to carry out transcriptome mapping. Genomic assembly: randomly assemble 1 read with another read that overlaps it at the edges. This construction of contigs using contiguous DNA sequences builds up the genome. Again, look for overlaps in the DNA sequences to assemble the genome. Instead of reads, programs grab a k-mer (i.e., n-tuple and n-gram Nucleic acid sequence, amino acid sequence, or polymer)??? ... For a k-mer of 13, it refers to a k-mer of 13 bases.

Our reads are of 50 bases. `Velvet` is a Blue Gene graft-based assembler.

Do not print graphs with bases, because they are huge. Assemble fractions of chromosomes (linkage to Group VII of the *Neurospora species*

Chapter 5

Questions

Questions:

1. Are sequences in the NCBI database either DNA or transcript sequences?
2. Since the Group VII chromosome (i.e., “chromosome 7”) of the *Neurospora crassa* species is the smallest chromosome, is it possible that a given DNA sequence of the *Neurospora crassa* species would not map to chromosome 7?
3.
4.
5.
6.
7.
8.
9.
10.
11.
12.
13.
14.
15.