Statistical Genomics Exercise 1
Working with Linux

WS 2022/2023

Unix-like operating systems are the most popular OS in bioinformatics. Most tools used by bioinformatitions such as software for genome assembly/mapping, variant calling or motif discovery are only available in Linux. In this exercise you will gain experience working in unix-like operating systems (eg GNU/Linux, MacOS, Android) which all share the same general design including the hierarchical filesystem and basic commands. Note that you can print the help page of any command in the terminal with `<command> --help`.
A *shell* provides an interface between a user and the OS either using a command line interface (CLI) or graphical user interface (GUI). We will be using CLI *Bourne Again SHell* (`bash`) to implement the tools needed to work with genomics data and write code.

- Linux is free and open-source that anyone can install, modify and redistribute
- Biological data are very large and much more easy to handle and manipulated in the Linux terminal
- Bash scripting allows simple way of generating automated biological analysis pipelines. You don't want to have to call a tool individually on each file.

Running Linux/Unix

- Windows Subsystem for Linux (WSL) enables you to run a GNU/Linux environment – including most command-line tools, utilities, and applications – directly on Windows, unmodified, without the overhead of a traditional virtual machine or dual-boot setup.
- VirtualBox is a virtualization tool that runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD

1 Navigating the keyboard

There are some keys that are used very often in Unix commands but can be difficult to find on some keyboards. Open a text editor (eg. gedit, nano, Vi/Vim (if you want to learn try calling `vimtutor` in your terminal) and type the following keys:

- ~ tilde
- / forward slash
- \ back slash or escape
- | vertical bar or pipe
- # hash or number or gate sign
- $ dollar sign
- * asterisk
- ' single quote
- " double quote

1.1 Useful bash terminal shortcuts

With a little practice, you can become very efficient navigating your computer in a CLI. Keep these following shortcuts in mind when

1. `tab` can be used to autofill commands or file names. Try `touch -he` – hit the enter key. This will display the help page of the command `touch`
2. up/down arrow to navigate through used commands.
3. `history` to list last used commands
4. `Ctrl+A` Move to the start of the command line
5. `Ctrl+E` Move to the end of the command line

2 Navigating the file system

Launch a terminal window and implement the following using basic `bash` commands.

1. Create a new directory in your home directory called “StatGen”.
2. Navigate to that directory.
3. Create a new directory in “StatGen” called “Lab1”.
4. What is the absolute path to “Lab1”?
5. What is the relative path from “Lab1” to “StatGen”?
6. Go back to your home directory.
7. Navigate from your home directory to “Lab1” using only one command.
8. Check the current directory (print its absolute path).
9. List all files in the current directory in human readable format (print sizes like 1K 234M 2G etc.).

3 Managing your files

Implement the following using basic `bash` commands

1. Use your favorite text editor to create a file called “file1.txt” in your home directory.
2. Create a directory called “test” in “StatGen”.
3. Copy “file1.txt” to the “test” directory. Name this copy “file2.txt”.
4. Rename “file1.txt” to “myfile1.txt”.
5. Move “myfile1.txt” to “test”.
6. Go to “test” and delete “file2.txt”.
7. Go to the “StatGen” directory and delete “test”. Note that there is no *undo* or *trash folder* in the terminal, so be very careful when deleting files or directories!

4 Working with text files

In this assignment, we will work with a GTF file from the Ensembl genome browser that describes all protein-coding and noncoding genes that have been annotated in the human genome.GTF/GFF formats are 9-column text formats used to describe and represent *genomic features* such as exons, introns, and UTRs. The formats have quite evolved since 1997, and despite well-defined specifications existing nowadays, they have great flexibility and contain a wide variety of information.

1. `curl` transfers data from or to a remote file (i.e., on an FTP or HTTP site). In other words, it is the command line way of going to a website and downloading something. Download the GTF file using the "`curl`" command. Note that we use the ">" to redirect the data retrieved by "`curl`" into a new file called "human.genes.gtf.gz":

   ```
   curl http://ftp.ensembl.org/pub/release-104/gtf/homo_sapiens/Homo_sapiens.GRCh38.104.gtf.
       gz > human.genes.gtf.gz
   ```

2. The filename ends in "*.gz*". This means that it has been compressed with a program called "`gzip`". The file can be decompressed into a plain text file using the "`gunzip`" command:

   ```
   gunzip human.genes.gtf.gz
   ```

3. After decompressing the file, the file will be called "*human.genes.gtf*" – `gunzip` automatically removes the "*.gz*" extension once it finishes decompressing the file.

   - Use the "`head`" command to print the **first 5 lines** of each file as output.
   - Use the "`cat`" command to print all contents of the GTF file.
   - Use the "`less`" command to inspect the GTF file. Can you interrupt the process?

   Note that the first few lines in the file beginning with "#" are so-called *header* lines (as opposed to data lines), and describe the creation date, genome assembly version, etc. Header lines are not data records.

5 The "`grep`" command

The "`grep`" (global regular expression print) command is a small family of tools that search text files for a pattern and print any lines that match it. The basic syntax of `grep` is:

```
grep <OPTIONS> PATTERNS <FILENAME>
```

1. Print all lines that start with "19" in the file "*human.genes.gtf*".
2. Print three lines that come after the pattern "`\sgene\s`" in the file "*human.genes.gtf*".
3. Print all lines that do not contain the string "*protein_coding*" in the file "*human.genes.gtf*".

6 The "`cut`" command

The "`cut`" command allows you to extract a specific column from a file. By default, the column delimiter is TAB. You can change this using "`-d`".

1. Print the 5th byte of "*human.genes.gtf*".
2. Print the 2nd column of "*human.genes.gtf*".

7 The "`wc`" command

The "`wc`" counts the number of lines/bytes/characters/words in a file.

1. How many words does "*human.genes.gtf*" contain?
2. How many lines does "*human.genes.gtf*" contain?

8 Piping

A pipe (|) is a form of redirection (transfer of standard output to some other destination) that is used to send the output of one command to another command for further processing. You can redirect output to a file using >; to append use ≫. Use a single command line for each of the following:

1. To print the headers of "*human.genes.gtf*" we can pipe the output of `cat` and use `grep` to match a hashtag

   ```
   cat human.genes.gtf | grep \#
   ```

   Can you think of a quicker way to get this result?

2. How many GTF records (i.e., data lines) are in "*human.genes.gtf*"?
3. How many GTF records (i.e., data lines) in "*human.genes.gtf*" correspond to protein-coding genes?
4. Create a new file *temp1.txt* containing the last 5 lines of "*human.genes.gtf*", but only showing the 1st and 3rd columns.
5. Create a new file *temp2.txt* containing the first 3 lines of 2nd column of *temp1.txt*
6. Add the first 10 lines of "*human.genes.gtf*" if they contain the string "*protein_coding*"
7. How many GTF records (i.e., data lines) in "human.genes.gtf" correspond to exons from protein-coding genes?
8. How many GTF exons from protein-coding genes are on the forward (+) strand?
9. How many GTF exons from protein-coding genes are on the reverse (-) strand?
10. How many CoDing Sequence (CDS) exons (records where column 3 is "CDS") from protein-coding genes exist on each chromosome (column 1)?
11. Print recent commands you typed for this exercise that contained the command `cat`

Explain how you might design an analysis of "*human.genes.gtf*" that would reveal how many distinct protein-coding genes there are in the human genome.

Hint: you may not have learned all of the command you might need – the point is to think about what what you *could* do with the commands you know of and what limitations would have to be addressed.

9 Writing a bash script

When you start dealing with multiple files, manipulating them one by one will become tedious. We can automate the execution of our commands by saving them in a file `<file.sh>`.

1. Using any text editor create a new file named "*hello-world.sh*". The first line will contain a shebang `#!` which defines our script's interpreter as bash. the following code:

   ```
   #!/bin/bash
   echo "Hello World"
   ```

2. Save the file and make sure your script is executable. This is indicated by the first column in the `ls -l <filename.sh>`. Each permission is represented by a single letter- `r` (read), `w` for (write), and `x` for (execute) for each ownership level (user, group, others). Use `chmod +x <filename.sh>` to add executable permission and recheck file permissions.
3. Execute the bash script either with `bash hello-world.sh` or `./hello-world.sh`.

If you want to learn more about unix, this intro course is very helpful