Some useful information for bioinformatics projects

Max Carter-Brown
Anglia Ruskin University, Wellcome Sanger Institute
max.carter-brown@aru.ac.uk

Contents

1. Outcomes

By the end of this document you should be able to:

- Open a terminal on your computer
- Understand and navigate the file system on your computer
- Download sequence data from the internet
- Download bioinformatic software from the internet and run it

2. Introduction

Bioinformatics is the study of biological systems using computers. This definition is very broad and encompasses not just biological sequence analysis, but other areas too, such as gene expression and image analysis. We will concentrate on sequence analysis here, which encompasses genomics, population genetics, and parts of evolutionary biology.

Something important to remember that almost all data we deal with is text data. Sequences are no exception. The most widely used file type in bioinformatics is the fasta file, a standard plain text UTF-8 encoded file (Figure 1).

```
>sequence_one
AACCCTTACCCTTCCAACT
```

Figure 1: The fasta file, a header starts with a '>' and it is followed by a sequence on the next line. The sequence line is optionally wrapped, usually at 80 characters.

This is a very basic file type, and widely used. For example on the NCBI website, the National Center for Biotechnology Information, most data you can find will be available for download as a fasta file. See this example of a file - *Euphrasia confusa*. Look at that telomeric repeat (AAACCCT)!

In bioinformatics we edit, modify, or convert from one file type to another, changing the files and data as we go. This core concept is actually not unique to bioinformatics - it is what computers *do*, but in bioinformatics we call this process a **pipeline** (Figure 2). How do we get from sequence data to a phylogenetic tree of relatedness? This is a question of stringing the correct pieces of software

together in the right order (and if necessary writing some software ourselves), changing the data until we have the result that we want.

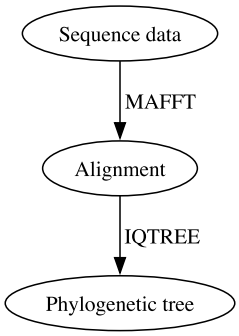

Figure 2: A very simple pipeline going from sequence data to a phylogenetic tree. The edge labels indicate popular software used in bioinformatics. MAFFT generates a multiple sequence alignment from an input of *homologous* sequences. IQTREE uses algorithms to infer a phylogenetic tree from this alignment.

3. Setup

To make best use of bioinformatics tools we need to set our computers up properly. We need to interact with the computer on a more basic level - we need to use what is called a terminal. A terminal is a text based prompt where a user can input commands and get results back. Simply put, you type text and the results of the command you executed are either written to the screen or to a file.

3.1. If you have a Mac...

You are in luck for this bit - all you need to do is find your terminal application.

3.2. If you have a Windows...

You need to install Windows Subsystem for Linux (WSL). Note that Windows PowerShell is *not* what we want. PowerShell, although it sounds great, is not widely used at all in bioinformatics and you will find yourself running into issues very quickly if you use it. Having said that, you will need to install WSL via PowerShell.

Shell 1: Install WSL via PowerShell on a Windows machine

```
wsl --install
```

3.3. If you have something else...

Let me know. Linux machines will already be set up.

4. The terminal and the filesystem

Go ahead and open up your new terminal. Depending on your machine you will be running a certain kind of shell. The shell is the program you use which interprets the commands you will be writing. It is usually BASH or ZSH, though many shells exist (Figure 3). Your shell will have a bit to the left of a `$` or `>`, which might include the date, and/or your location in your system. To the right of the `$` / `>` is where you type your commands.

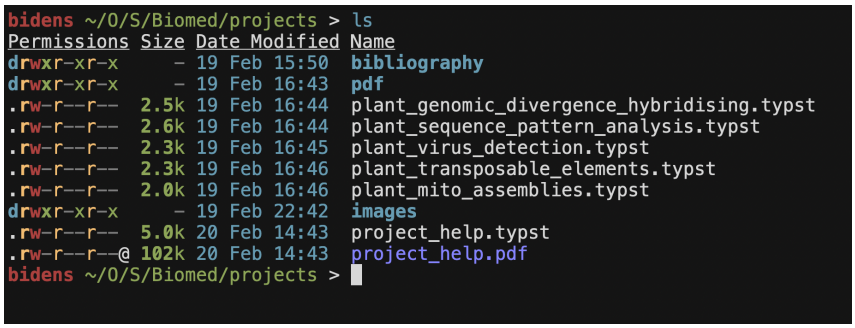

Figure 3: My shell set up. It will be a bit fancier than yours, but it does the same thing.

When you are starting out, it is good to know where you are in your system. By that I mean your file system. When you open up your terminal, it places you inside your file system somewhere (usually your home directory). This is equivalent to you opening up Finder (on a Mac) or File Explorer on Windows. I hope you have used these applications before. They are visual guides to where all the files on your computer are stored. The terminal reflects this too, but uses text and not fancy graphics. Now go ahead and list the files in your terminal - just type what is in the code boxes.

Shell 2: Listing files in the directory you are in.

```
ls
```

This command should output some text which shows you what is in your directory. It will not look like Figure 3, but it should display the same information that is in the 'Name' column.

Shell 3: Adding flags to our `ls` command modifies its behaviour. Hashed lines are comments, not code to be executed.

```
# these are the same
ls -t -h -o -r
# a friend of mine told me about these flags whilst doing my PhD
ls -thor
```

You notice that I added some text there. `-t` sorts output by time. `-h` makes the output human readable (i.e. adds K for kilobytes and M for megabytes etc). `-o` outputs list in long format. `-r` reverse sorts, so with `-t` I am reverse sorting by time - neat right?

So `ls` is a program which has the simple task of listing files. It can list files anywhere though, not just where we happen to be. All we have to do is point `ls` to the directory where we want it to list files. To do this, you have to be familiar with what a file system is a bit more. In Figure 4 I have made up a toy example which starts at the root of our file system and contains different directories.

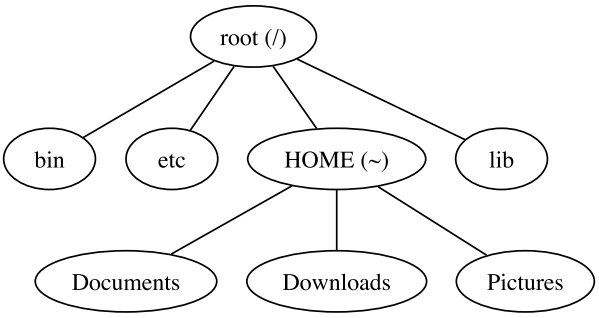

Figure 4: My made up file system. The root (denoted by a forward slash `/`) is the origin of the file system, and it usually has lots of stuff in it we won't worry about. Under the root, we have HOME, which is our home directory (denoted by a tilde `~`). Under HOME are our familiar directories - where my documents live, and where my downloads are.

As I mentioned earlier, when we open our terminal we are usually placed in our home directory. But now I want to list the files in the directory 'above' where I am. I want to ascend the file system tree up one level. How do I do this? There are actually two ways.

Shell 4: Two ways of doing the same thing. The `..` is a code for 'go up one level'. The `/` is the symbol for root. It so happens that when we go up one level from HOME, we reach the root! So these two lines of code do the same thing here. If you run these on your machines however, you are likely to get a different result.

```
ls ..
ls /
# note you can also pass the flags like you did before
ls -thor ..
```

I will introduce one more command here, `cd`. `cd` changes the directory we are currently in. We use it if we want to 'physically' travel around the file system. Running `cd` by itself will return you to your home directory (which may not be what you want). Usually we run `cd` with a path - that is, a pathway to where we want to go to.

Shell 5: Changing directory. The first line takes us home. The second line takes us up one level, and the third line takes us in to the Documents folder (provided the Documents folder is in the directory you are currently in!)

```
# change directory
cd
cd ..
cd Documents
```

By using these two commands (`cd` and `ls`) we can explore any part of our file system that we have permission to view.

Shell 6: Exploring our toy file system shown in Figure 4.

```
# this might display:
# Documents  Downloads  Pictures
ls
# go into Documents
cd Documents
# have a look around
ls
# go back to where you were
cd ..
# go up again
cd ..
# have a look around
# this might display:
# bin  etc  HOME  lib
ls
```

5. Downloading some data

`ls` and `cd` are not the only commands at our disposal, there are hundreds which are built in to our computers. Most manipulate text data in some way, or view files. Some create files, or download files. We will look at one in particular called `curl` which is a tool primarily to fetch data from the internet (amongst other things). We will download some data first. Note I added a flag `-o` to `curl` which saves the output as a file as the name which you pass after this flag (which is 'OW119596.1.fa').

Shell 7: Make a directory, go in to that directory and then download chromosome 1 of *Arabidopsis thaliana*.

```
# first make a directory to store the data
mkdir genome_data
# go in to that folder
cd genome_data
# execute the curl command
curl -o OW119596.1.fa 'https://www.ebi.ac.uk/ena/browser/api/fasta/OW119596.1?
download=true'
```

Now you have a directory called `genome_data` with some genome data inside.

6. Investigating our data

We will do some very basic investigation of the data using `seqtk`. `seqtk` is a tool written by the wonderful Heng Li, to whom we in the bioinformatics community owe a great deal. We need to download that first though. I introduce a new bit of software here called `git`. `git` is really important, and something that you should end up using later on in your projects. For now think of it as software which tracks changes in groups of files (aka repositories). You can also use `git` to fetch repositories from the internet, which is what we do here.

Shell 8: `git` is a tool to version control files. We can use it to download 'repositories'. Here we are cloning (i.e. downloading from the internet) `seqtk`. By default it creates a directory called `seqtk`. We go in to that directory using `cd`. We then run the `make` command which compiles the software for us.

```
# fetch the software
git clone https://github.com/lh3/seqtk.git
# go into the downloaded directory
cd seqtk
# compile the software before we can use it
make
# now list the files in the directory
# there should now be one called `seqtk`
ls
# go back up one directory
cd ..
```

After running these commands you will see that there is another file inside your system now called `seqtk`. This is an executable file. This means we can run it like any other program on our system. In general you should be careful downloading code from the internet, compiling it, and then running it as it could contain harmful code. We are safe here though. I promise.

The last part here is to run our new software on the data we just downloaded. It turns out that `seqtk` has a bunch of commands, but we will focus on one called `size` - `seqtk size` in full. To run `seqtk` we can't just type `seqtk` - you will get a command not found error on your computer. We have to direct to the `seqtk` executable. If you followed the commands above, we now reside in the directory above the `seqtk` directory.

Shell 9: Run the executable we downloaded. The first command brings up the help. The second command runs the `size` module of the software which reports the sequence number, and the number of bases counted.

```
# the ./ means 'from the current directory'.
# . by itself means the current directory
# this command shows the help
./seqtk/seqtk

# and this runs our command
./seqtk/seqtk size OW119596.1.fa
```

If you ran that correctly you should see this output:

```
1    33257523
```

Which is 33,257,523 base pairs or ~33Mb pairs.

7. Over to you

1. Run `head` on your data. What does it do?

Shell 10: The head command

```
head OW119596.1.fa
```

2. How would you modify the number of lines shown in `head`?
3. Look up the help for `cat` - what does `cat` do?

4. How would I copy a file on the terminal? (Hint: look up `cp`)
5. What are the two ways to jump to your `Documents` folder on your computer? One will use a relative path, and the other the absolute path (i.e. one will contain a `/` - the root directory, and one will not).