Exercise: Sequencing Data

Nick Bartelo

2/5/2021

Q1

1. Write a for-loop to download all fastq files of WT biological replicate no. 2 of the Gierlinski RNA-seq data set (UNIX). Try to have a solution that's as generally applicable as possible.

To begin solving this question, we navigate to the website https://www.ebi.ac.uk/ena/browser/home. We are given the accession number, ERP004763, which is a unique identifier for the project PRJEB5348. We paste this accession number in the top right box that says "Enter accession" and hit search. This brings us to https://www.ebi.ac.uk/ena/browser/view/PRJEB5348. Once on this page, we scroll down and click on TSV, which is to the right of 'Download report:'. This downloads the TSV file to our computer, which contains URLs for the fastq files.

After unsuccessfully using the wget command to upload the TSV file to the Scientific computing unit, we used WinSCP as an alternative way to transfer the TSV file from the computer to the SCU. To do this, we open WinSCP, type `aphrodite.med.cornell.edu` into the Host name line, our username for the SCU in the User Name line, and our password for the SCU in the Password line. This allows for the transfer of files from our computer to the SCU in a drag-and-drop fashion which is extremely easy.

We then log into the SCU using the command line, cd into `/home/nib4003/ANGSD_2021_hw` and create a folder for this exercise entitled "sequencing_data" using the command `mkdir sequencing_data`. We then go to the location using WinSCP where the TSV file is on our computer and go to the location of the newly created directory in the SCU, and we transfer the TSV file from the computer to the SCU. This copies the file so that it is now on my computer and the SCU.

(As an aside: Merv did help me download the TSV using the wget command `wget 'https://www.ebi.ac.uk/ena/portal/api/filereport?accession=PRJEB5348&result= read_run&fields=study_accession,sample_accession,experiment_accession,run_accession,tax_id,scientific_name,fastq_ftp,submitted_ftp,sra_ftp&format=tsv&download=true'`. Notice the extra quotes around the URL.)

Now that the TSV file was in the SCU, we used the command `less filereport_read_run_PRJEB5348_tsv.txt` to look at the TSV file. We notice this file contains 9 columns.

We then navigated to `http://dx.doi.org/10.6084/m9.figshare.1416210`. Here, we right clicked on the arrow to download the file containing all rows for the "ERP004763_sample_mapping.tsv" file shown on the page, and clicked "copy link address". In the SCU, we used the command `wget https://ndownloader.figshare.com/files/2194841` to download this file. We then renamed this file using the command `mv 2194841 info_for_accession_numbers` so that we have more information on what the file contains if we come back to this project in the future. We then look at the file using the command `less info_for_accession_numbers` and since the file is not too long, we inspect the whole thing and find the rows of interest to us in the document. We notice that this file contains the run accession values which match with the run accession values for the "filereport_read_run_PRJEB5348_tsv.txt" file. Therefore, we need to generate a file with the individual run accession IDs and then use that list as input to the for-loop of the "filereport_read_run_PRJEB5348_tsv.txt" file in order to download all fastq files of interest.

To parse the run accession values we need, we use the following command:

```
egrep 'WT' info_for_accession_numbers | egrep -w '2$' | awk '{print $1}'
```

In this command, `egrep 'WT' info_for_accession_numbers` first locates all lines with 'WT' in them, representing the wild type samples. `egrep -w '2$'` then prints only those lines that contain the whole words. The `-w` command provides a differentiation between 2, 12, 22, etc. which all end in 2, which makes it possible to only keep the wild type samples we are searching for. The `awk '{print $1}'` command prints the first word from all the remaining lines, which corresponds to the run accession values we are after. We assign these accession values to a variable using the following command:

```
accession_numbers=$(egrep 'WT' info_for_accession_numbers | egrep -w '2$' | awk '{print $1}')
```

This results in all accession numbers being conveniently stored in a single variable. Next, we loop over these accession numbers using a for loop to extract each fastq file for download and use wget to download the corresponding file using the command:

```
for i in $accession_numbers;
do link=$(egrep "$i" filereport_read_run_PRJEB5348_tsv.txt | cut -f 7);
wget ftp://${link}; done
```

In this command, i represents one of the accession numbers from those stored in the variable `accession_numbers`. The command `do` tells the for loop to commense. The command `egrep "$i" filereport_read_run_PRJEB5348_tsv.txt` searches for all lines in which the current accession number appears. Notice how it is necessary to use the double quotes instead of the single quotes when searching for one of the elements in the for loop. In each case, each accession number only appears once, and so we are output with 7 lines. The command `cut -f 7` extracts the contents in the seventh field, which represents the column fastq_ftp. Therefore, this outputs the fastq link we need to download. The link is then downloaded using the `wget ftp://$link` command where the ftp:// is necessary not to receive a 504 error:gateway timeout. The `done` command finishes the for loop. The result is seven new files labeled accession_number.fastq.gz for each corresponding accession number.

We can also get the same result using a bash script. This involves creating a file using the command `vi importing_wt2.sh`. Once in this document, we have to type #! /bin/bash in the first line. We then write the following code, which we have shown above, in the document such that the resulting document contains the code:

```
#! /bin/bash

accession_numbers_wt=$(egrep 'WT' /home/nib4003/ANGSD_2021_hw/
sequencing_data_qc_and_preprocessing/info_for_accession_numbers
| egrep -w '2$' | awk '{print $1}')

for i in $accession_numbers_wt;
do link_wt=$(egrep "$i" /home/nib4003/ANGSD_2021_hw/sequencing_data_qc_and_preprocessing/
filereport_read_run_PRJEB5348_tsv.txt | cut -f 7);
        wget ftp://${link_wt}; done
```

We then press the Esc key and ZZ to save the changes and exit the file. To call this script so that all the files are imported, we type the command `./importing_w2.sh` and this results in all fastq files being downloaded.

Q2

2. Why are there multiple fastq files for sample WT_2? What does each file represent?

There are multiple fastq files for sample WT_2 because each replicate was sequenced in a different lane 1-7. Each file represents the biological replicate relating to the specific lane which the sequencing occurred.

Q3

3. Count the number of lines in each FASTQ file and keep a note of the results (UNIX). The zcat command allows you to see the contents of a gzipped file. A fastq file has 4 lines per read. Do a second for-loop where you determine the number of reads per file.

We solve this question using the following for loop:

```
for file in *.gz; do number_of_lines=$(zcat $file | wc -l);
echo $file; echo 'number of lines:' $number_of_lines;
echo 'number of reads:' $(( $number_of_lines / 4 )); done
```

In this code, we loop through all seven fastq.gz files we created in Q1 using `for file in *.gz` where `*` is a symbol meaning to take all files that end in what comes after it. The command `number_of_lines=$(zcat $file | wc -l)` creates a variable which stores the number of lines for each file. The command `echo $file` outputs the name of the file being processed at the current time. The command `echo $number_of_lines` outputs the number of lines in the fastq file. Since we are to find the number of reads in the fastq files, and we know that the format of a fastq has 4 lines per read, we simply have to divide the number of lines by 4. The command to do so is `echo $(( $number_of_lines / 4 ))` where the format for the mathematical expression is `$(( ))` in which the the expression inside the parentheses must be one space away from both the beginning and ending of the inner parenthesis. The `done` command tells that we have finished the for loop. This output is formatted in a logical structure in which it shows the name of the file being processed, followed by the number of lines in the file, and finally the number of reads in the file. We show the results in the table below.

File Name	Number of Lines	Number of Reads
ERR458878.fastq.gz	5870276	1467569
ERR458879.fastq.gz	5800048	1450012
ERR458880.fastq.gz	5766276	1441569
ERR458881.fastq.gz	5286992	1321748
ERR458882.fastq.gz	4527136	1131784
ERR458883.fastq.gz	4562752	1140688
ERR458884.fastq.gz	5846584	1461646