Unix Exercises

with Case Studies in Deep Sequencing Data Analysis

Exercises

1) Sample Files: For this exercise, please run the script /tmp/bootcamp/exercise1.sh to create a sample set of files in your home folder. This will create a directory containing a set of sample files for this question.

```
$ /tmp/bootcamp/exercise1.sh
```

You have 30 samples and you treated each of them in three different conditions: *untreated*, *overexpressed*, *underexpressed*. After your initial analysis, you collected all your files in one folder. You named the files as `condition.$i.bam` and `condition.$i.bed` where condition $\in \{\text{untreated}, \text{overexpressed}, \text{underexpressed}\}$ and $i \in \{1, 2, \ldots, 30\}$. Now you want to organize this folder according to condition name. You want to put all files that belong to the condition untreated in the folder `untreated`, all files that belong to the condition overexpressed in the folder `overexpressed` and so on. How can you do this without typing the move command for each file one-by-one?
Hint: Take a look at shell wild cards. For this case "*" will be useful for you.

Solution:
We assume that you are in the `exercise_1_files` directory. Then, first, create the subdirectories for the conditions.

```
$ mkdir untreated overexpressed underexpressed
```

Now move the files that belong to the untreated condition. The wildcard "*" stands for any number of any characters. To match all bam and bed files coming from the untreated condition, we can use `untreated*.*`. All file names in that condiotion start with `untreated` and have a ".". Hence `untreated*.*` matches `untreated2.bam`, `untreated5.bed` and etc. So

```
$ mv untreated*.* untreated/
```

will move the files that belong to the condition `untreated`. For the remaining libraries, using the same method, we can move the files using the commands

```
$ mv overexpressed*.* overexpressed/
```

and

```
$ mv underexpressed*.* underexpressed/
```

Pitfall: A command like

```
$ mv untreated* untreated/
```

doesn't work in this case. The pattern `untreated*` matches "untreated" as well. So the above command will try to move the directory `untreated*` on itself. This will cause an error.

2) How can you make a zipped tar archive file that contains the main directory in the previous exercise?

Solution:

Say we want to make the zipped tar file myfile.tar.gz in our home directory. Then

```
$  tar -cvzf ~/myfile.tar.gz ~/exercise_1_files
```

creates the desired file.

3) In the first exercise, delete all bed files, i.e., those files having the extension ".bed".

Solution:

`*.bed` matches all file names ending with ".bed". Thus, in each folder (untreated, overexpressed , underexpressed), the command

```
$ rm *.bed
```

will delete all bed files.

4) You have submitted a bunch of jobs to the cluster. Afterwards you accidentally submitted many tophat jobs. If the tophat jobs finish and produce output, you will run out of disk space and your previous jobs won't be able to finish successfully. So you want to cancel your tophat jobs. Each job submitted to the HPC system has a unique job identification number. This number is displayed on the first column when you type `bjobs`. Each job has a name (not necessarily unique) and it is displayed on the seventh column of this list. Luckily you named all your tophat jobs in the form `tophat_sample_number` where `sample_number` is an identifier of the RNASeq library being aligned. You can cancel a set of running or scheduled jobs using the `bkill` command followed by the job id numbers separated by white space, i.e.,

```
$ bkill JOB_ID_1 JOB_ID_2 ... JOB_ID_n
```

where JOB_ID_i is the identification number of the corresponding job to be cancelled. How can you cancel all of your tophat jobs without typing the job identification number of each individual job? For this solution, assume that

```
$ bkill JOB_ID_1 JOB_ID_2 ... JOB_ID_n
```

is the only way to cancel jobs though there are alternative and easier ways.
Simulation Script: There is a simulation script for this question. Run the script `/tmp/bootcamp/exercise4.sh` to start a set of tophat, bwa and fastqc jobs.

```
$ /tmp/bootcamp/exercise4.sh
```

These jobs will be alive for 10 minutes. If you are not done after ten minutes, run the script again. They will actually be dummy jobs but their names will be relevant to this exercise.
Hint 1: Use pipes "|" and `awk`.
Hint 2: Once you figure out how to get the job id's of tophat jobs, give the job id list to `bkill`.

Solution:
We can list all our tophat jobs using grep

```
$ bjobs | grep tophat
```

Note that we can make sure that we get the correct job ids by

```
$ bjobs | grep tophat | less
```

In the resulting list, we need the first column. We get this by

```
$ bjobs | grep tophat | awk '{print $1}'
```

Now, we are ready to cancel our tophat jobs.

```
$ bkill `bjobs | grep tophat | awk '{print $1}' `
```

Note that we need to put ` around our previous command to provide it as input to bkill.

5) In the previous exercise, suppose that you have some other jobs having the name `fastqc_i`, where i is the sample identifier. (You will see this if you run the simulation script mentioned in the first exercise.) How can you cancel all running or scheduled jobs except for the tophat jobs and the fastqc jobs?

Solution:
If you use `grep` with the `-v` option, you will get all non-matching entries. So

```
$ bkill `bjobs | grep -v tophat | grep -v fastqc`
```

will cancel all jobs that are neither tophat jobs nor fastqc jobs.

6) You have a bed file where the chromosome the entry is coming from, is given in the first column. The chromosomes are labeled as chr1, chr2, ..., chr22, chrX, chrY. How can you extract the entries that only come from chromosome 1?
Warning: A simple `grep chr1` statement will not work. Can you tell why?
Hint: Use `awk`.
Example File: /tmp/bootcamp/exercise6.bed
Note: Detailed information on bed file format can be found in
http://genome.ucsc.edu/FAQ/FAQformat.html#format1

Solution:
The command

```
$ grep chr1
```

will not work because it will also bring all entries from `chr11`. The reason is that the string `chr1` is a substring of `chr11`. To get the entries that are from `chr1` only, we use `awk` as follows. Compare the first column of the bed file to the string "chr1". Print only the rows for which the equality holds.

```
$ awk '{ if($1 == "chr1")print($0) }' exercise6.bed
```

Note that, in `awk`, by default, the n^{th} column is accessed by `$n`. The whole line is accessed by `$0`.

7) In a bed file, how can you get a list of all distinct chromosomes? For example, if your bed file consists of nine entries such as

chr2	74711	127472363	Pos1	0	+
chr3	74723	127473530	Pos2	0	+
chr1	73530	127474697	Pos3	0	+
chr2	17469	127475864	Pos4	0	+
chr3	12747	127477031	Neg1	0	-
chr5	17477	127478198	Neg2	0	-
chr7	74781	127479365	Neg3	0	-
chr7	74795	127480532	Pos5	0	+
chr1	12748	127481699	Neg4	0	-

then how can you obtain the list `chr1, chr2, chr3, chr5, chr7`?
Hint : Use the tools `awk`, `sort` and `uniq`. You need to preprocess the contents of the bed file before feeding them to `uniq`. Pipe these tools together to get the result.

Solution:
`uniq` lists repeated entries only once. But for this, we have to provide a sorted list to `uniq`. Let's say our bed file is `file.bed`. Then, we get the first column of this file by

```
$ awk '{print $1}' file.bed
```

Now we can sort the chromosomes with the `sort` command. Finally, we feed the sorted list to `uniq`.

```
$ awk '{print $1}' file.bed | sort | uniq
```

8) How can you sort a bed file, first by the first column, i.e., according to the chromosomes, and then by the second column, i.e., beginning of the entry position?

Solution:
You can specify the fields using the `-k field_start,field_end` parameter in `sort`. Here, `sort` takes the fields from field_start to field_end and sorts the lines accordingly. So if we want to sort by the first field only then we should provide `-k 1,1`. To break ties, we sort by the second column numerically. This is done by `-k 2,2g` where g tells `sort` to sort entries numerically. Combining all these we have

```
$ sort -k 1,1 -k 2,2g file.bed
```

9) Recall that each read in a fastq file spans four consecutive lines. The nucleotide sequence of the read is given in the second line (out of these four lines). How can you obtain the nucleotide sequence of the 22^{nd} **read** in a given fastq file?
Hint: You need `head` and `tail` programs with the `-n` option. A quick internet search will provide docuimentation and examples. Note that there are sample fastq file in the transcriptomics folder in `~/bootcamp` directory.

Solution:
Note that the nucleotide sequence of the 22^{nd} read is in the line $21 \times 4 + 2 = 86$. So, the problem is getting the 86^{th} line in a fastq file. Say, our fastq file is `file.fq`. We can get the first 86 lines by `head -n 86 file.fq`. We need the last line among these 86 lines. For this, `tail -n 1` does the job. So, we pipe the output of `head` to tail as follows.

```
$ head -n 86 file.fq | tail -n 1
```

10) Coming from your deep sequencing experiment, you have a fastq file consisting of the reads of one lane. You mixed several libraries in that lane and barcoded the molecules to determine the sample of each read. In the nucleotide sequence of each individual read, the first two nucleotides form the barcode of the read. The barcode of the sample you are interested in is `TT`. How can you get all nucleotide sequences that come from this sample? Once you find these nucleotide sequences, how can you remove the barcodes from the nucleotide sequences?
Hint: Use `awk`. There is a useful function of `awk` called `substr`. An internet search will provide you examples and documentation.
Note: In practice there are software tools, such as `cutadapt`, that are used to remove barcodes in fastq files.

Solution:
Say, our fastq file is `file.fq`. First we need all the nucleotide sequences in this file. This can be done by

```
$ awk '{ if( NR % 2 == 4 ) print($0) }' file.fq
```

We can get the first two nucleotides using the `substr` function. Namely, `substr($1,1,2)` gives us the two characters of \$1 starting at the first position. We take only the reads whose first two characters are `TT`. Hence

```
$ awk '{ if( NR % 2 == 4 ) print($0) }' file.fq | awk '{ if(substr($0,1,2) == "TT")print($0)}'
```

gives all entries that belong to the sample of interest. The function `substr` can be used to remove the barcodes. For this we take all nucleotides that come on and after the third position. This is done obtained by `substr($0,3)`. So, to get the reads with barcodes removed is done by

```
$ awk '{ if( NR % 2 == 4 ) print($0) }' file.fq | awk '{ if(substr($0,1,2) == "TT")print(substr($0,3))}'
```

11) You have a bed file containing reads from both + and - strands. You want to save all reads in this bed file that come from the negative strand and chr11. How can you create another bed file that contains only reads from the - strand and chr11?
Sample File: You can use the sample bed file that comes with a previous exercise. Hint: Use awk and the redirection operator ">".

Solution:
The strand information is found on the sixth column of a bed file. So, using awk, we can pick all reads coming from the negative strand.

```
$  awk '{if($6 == "-")print($0)}' exercise6.bed
```

To get the entries from chr11, we can use `grep`. Note that `grep` works as there is no chromosome name such as `chr110` or so.

```
$  awk '{if($6 == "-")print($0)}' exercise6.bed | grep chr11
```

We can save the output in a file named `chr11_negative.bed` by

```
$  awk '{if($6 == "-")print($0)}' exercise6.bed | grep chr11 > chr11_negative.bed
```