Introduction to read mapping for transcriptomics Version 3

Frank Aylward

Abstract

This is an example of a simple short read mapping analysis that could be used as part of a transcriptomics or RNA-seq workflow.

Code is intended for use on an Ubuntu 16.04 LTS OS, but it may work on other Unix or Unix-like systems.

This tutorial uses data from the following very nice paper from the Hatfull group:

Dedrick, Mavrich, Ng, and Hatfull, Expression and evolutionary patterns of mycobacteriophage D29 and its temperate close relatives. BMC Microbiology, 2017. https://doi.org/10.1186/s12866-017-1131-2

The tools that are used include:

SRA toolkit: https://www.ncbi.nlm.nih.gov/sra/docs/toolkitsoft/

bowtie2: http://bowtie-bio.sourceforge.net/bowtie2/index.shtml

samtools: http://samtools.sourceforge.net/

BEDOPS: https://bedops.readthedocs.io/en/latest/content/reference/file-management/conversion/bam2bed.html

BEDtools: https://bedtools.readthedocs.io/en/latest/

Make sure that these tools are installed before starting the tutorial. On a Ubuntu OS you should be able to install most with "sudo apt install", but you may wish to use a package manager such as Anaconda or Miniconda as well.

Throughout the tutorial some notes may be made about particular versions of tools. For example, note that the version of SRA toolkit that is installed with apt install often leads to downstream errors, it may be desirable to install this tool with Miniconda instead.

Miniconda bash installers can be found here: https://conda.io/miniconda.html

Protocol

Download the reference genome and index a database with bowtie2

Step 1.

The datasets we will be using here are part of the Dedrick et al BMC Microbiology paper mentioned in the description, which describes some very cool transcriptomic experiments of Mycobacterium smegmatis while is infected with various bacteriophage. There are several short-read Illumina transcriptomes that are part of this study, but here will only consider that associated with the accession SRR5585000, which corresponds to a transcriptomic study of M. smegmatis during infection with mycobacteriophage D29. The RNA from this accession was extracted 15 minutes after infection, and other timepoints are also available (see all of the datasets on the Gene Expression Omnibus: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE99182).

First let's create a new folder and navigate into it, just so we can keep all of the subsequent files organized.

mkdir transcript_mapping

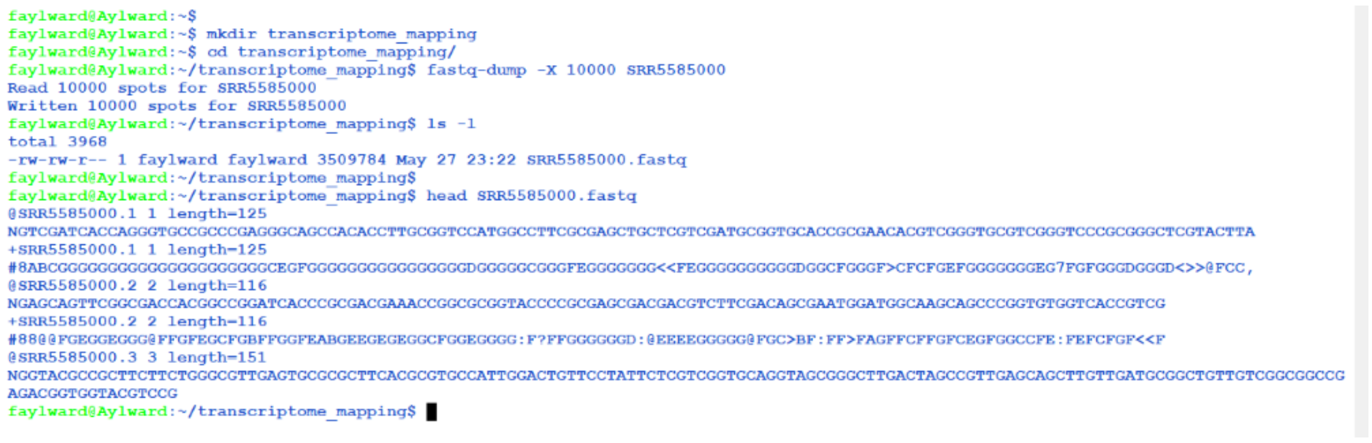
cd transcript mapping

and then let's download the raw Illumina reads using the fastq-dump utility in the SRA toolkit:

fastq-dump -X 10000 SRR5585000

The -X flag here indicates that instead of downloading the entire dataset, we only wish to download the first 10,000 reads. This is just for simplicity and to keep the datasets small (10,000 may seem large, but its a small number of reads compared to most transcriptome studies).

After this we can us the 'ls' command to ensure that the file was downloaded, and then take a quick look inside with the 'head' command to ensure it looks like typical FASTQ format. You should see something like this:



```
faylward@Aylward:~$
faylward@Aylward:~$ mkdir transcriptome_mapping
faylward@Aylward:~$ cd transcriptome_mapping/
faylward@Aylward:~/transcriptome_mapping$ fastq-dump -X 10000 SRR5585000
Read 10000 spots for SRR5585000
Written 10000 spots for SRR5585000
faylward@Aylward:~/transcriptome_mapping$ ls -l
total 3968
-rw-rw-r-- 1 faylward faylward 3509784 May 27 23:22 SRR5585000.fastq
faylward@Aylward:~/transcriptome_mapping$
faylward@Aylward:~/transcriptome_mapping$ head SRR5585000.fastq
@SRR5585000.1 1 length=125
NGTCGATCACCAGGGTGCCGCCCGAGGGCAGCCACACCTTGCGGTCCATGGCCTTCGCGAGCTGCTCGTCGATGCGGTGCACCGCGAACACGTCGGGTGCGTCGGGTCCCGCGGGCTCGTACTTA
+SRR5585000.1 1 length=125
#8ABCGGGGGGGGGGGGGGGGGGGCEGFGGGGGGGGGGGGGGDGGGGGCGGGFEGGGGGGG<<FEGGGGGGGGGGDGGCFGGGF>CFCFGEFGGGGGGGEG7FGFGGGDGGGD<>>@FCC,
@SRR5585000.2 2 length=116
NGAGCAGTTCGGCGACCACGGCCGGATCACCCGCGACGAAACCGGCGCGGTACCCCGCGAGCGACGACGTCTTCGACAGCGAATGGATGGCAAGCAGCCCGGTGTGGTCACCGTCG
+SRR5585000.2 2 length=116
#88@@FGEGGEGGG@FFGFEGCFGBFFGGFEABGEEGEGEGGCFGGEGGGG:F?FFGGGGGD:@EEEEGGGGG@FGC>BF:FF>FAGFFCFFGFCEGFGGCCFE:FEFCFGF<<F
@SRR5585000.3 3 length=151
NGGTACGCCGCTTCTTCTGGGCGTTGAGTGCGCGCTTCACGCGTGCCATTGGACTGTTCCTATTCTCGTCGGTGCAGGTAGCGGGCTTGACTAGCCGTTGAGCAGCTTGTTGATGCGGCTGTTGTCGGCGGCCG
AGACGGTGGTACGTCCG
faylward@Aylward:~/transcriptome_mapping$
```

Download a reference genome

Step 2.

Now we need to get a reference genome to map reads against. The focus of this experiment was mycobacteriophage D29, so let's download the FASTA file of that genome using the Unix wget command:

wget -O D29.fna.gz ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/003/004/865/GCA_003004865.1_ASM300486v1/GCA_003004865.1_ASM300486v1_genomic.fna.gz

and, because the file is compressed, let's unpack it:

gunzip D29.fna.gz

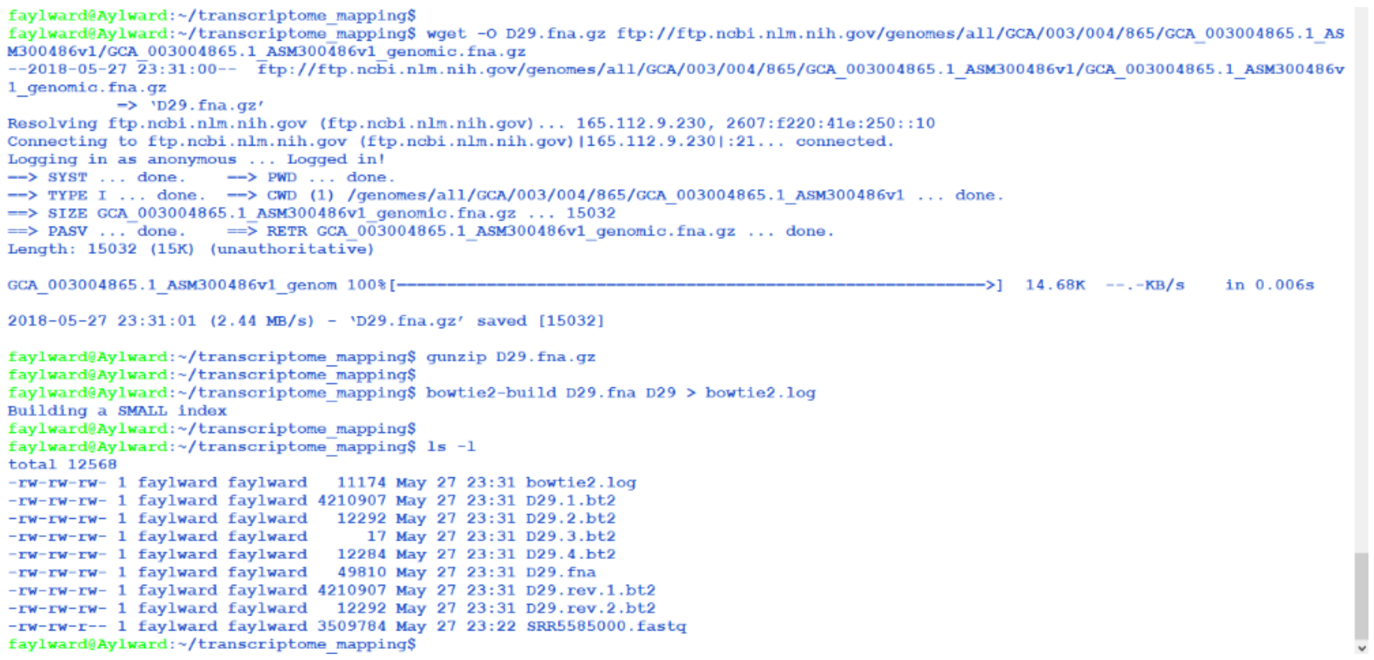
We are going to map reads using bowtie2, which requires that the FASTA genome file is indexed first. To do this we can use the bowtie2-build command. First just try typing "bowtie2-build" just to get a handle on the general usage and options of this tool. For purposes here we can run the following command:

bowtie2-build D29.fna D29 > bowtie2.log

The reason I put a "> bowtie2.log" at the end was because this command outputs a large amount of information to the command line, and I wanted to save it in a log file. It's usually not necessary to look at this, but if something weird happens when mapping reads later we can always go back and check to make sure the genome was indexed properly.

The important thing that bowtie2-build does is create a number of bt2 files, which you should be able to see after running the command.

Overall this step should look something like this:



```
faylward@Aylward:~/transcriptome_mapping$
faylward@Aylward:~/transcriptome_mapping$ wget -O D29.fna.gz ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/003/004/865/GCA_003004865.1_AS
M300486v1/GCA_003004865.1_ASM300486v1_genomic.fna.gz
--2018-05-27 23:31:00--  ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/003/004/865/GCA_003004865.1_ASM300486v1/GCA_003004865.1_ASM300486v
1_genomic.fna.gz
           => 'D29.fna.gz'
Resolving ftp.ncbi.nlm.nih.gov (ftp.ncbi.nlm.nih.gov)... 165.112.9.230, 2607:f220:41e:250::10
Connecting to ftp.ncbi.nlm.nih.gov (ftp.ncbi.nlm.nih.gov)|165.112.9.230|:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD (1) /genomes/all/GCA/003/004/865/GCA_003004865.1_ASM300486v1 ... done.
==> SIZE GCA_003004865.1_ASM300486v1_genomic.fna.gz ... 15032
==> PASV ... done.    ==> RETR GCA_003004865.1_ASM300486v1_genomic.fna.gz ... done.
Length: 15032 (15K) (unauthoritative)

GCA_003004865.1_ASM300486v1_genom 100%[===========================================================>]  14.68K  --.-KB/s    in 0.006s

2018-05-27 23:31:01 (2.44 MB/s) - 'D29.fna.gz' saved [15032]

faylward@Aylward:~/transcriptome_mapping$ gunzip D29.fna.gz
faylward@Aylward:~/transcriptome_mapping$
faylward@Aylward:~/transcriptome_mapping$ bowtie2-build D29.fna D29 > bowtie2.log
Building a SMALL index
faylward@Aylward:~/transcriptome_mapping$
faylward@Aylward:~/transcriptome_mapping$ ls -l
total 12568
-rw-rw-rw- 1 faylward faylward   11174 May 27 23:31 bowtie2.log
-rw-rw-rw- 1 faylward faylward 4210907 May 27 23:31 D29.1.bt2
-rw-rw-rw- 1 faylward faylward   12292 May 27 23:31 D29.2.bt2
-rw-rw-rw- 1 faylward faylward      17 May 27 23:31 D29.3.bt2
-rw-rw-rw- 1 faylward faylward   12284 May 27 23:31 D29.4.bt2
-rw-rw-rw- 1 faylward faylward   49810 May 27 23:31 D29.fna
-rw-rw-rw- 1 faylward faylward 4210907 May 27 23:31 D29.rev.1.bt2
-rw-rw-rw- 1 faylward faylward   12292 May 27 23:31 D29.rev.2.bt2
-rw-rw-r-- 1 faylward faylward 3509784 May 27 23:22 SRR5585000.fastq
faylward@Aylward:~/transcriptome_mapping$
```

Perform the read mapping

Step 3.

Generally speaking one should do a bit of quality-checking on the reads before jumping straight into the mapping step. Also, it is typical to remove adapter sequences and trim off low-quality bases before mapping.

Since this is intended to be a conceptual tutorial, and since it doesn't change the results too much, we will skip these steps here. Keep them in mind if you are interested in performing analyses for a formal study, though. If you are interested in adapter clipping and quality trimming, I would recommend cutadapt for the former and sickle for the latter (https://cutadapt.readthedocs.io/en/stable/guide.html and https://github.com/najoshi/sickle)

Before we get started, just type '**bowtie2**' and take a look at the general usage and options for this tool. Note that bowtie2 is a versatile tool that can be used in many ways. We will be using some rather simple and straightforward code here, but there are many possible options you may wish to play around with.

Let's try a simple command:

```
bowtie2 -U SRR5585000.fastq -x D29 -S mapping_out.sam
```

The -U specifies that we are using unpaired reads. Keep in mind that Illumina reads are often paired, so for other data you may want to use the -1 and -2 flags to specify input data. We specify the database prefix with the -x flag, and the output SAM file with the -S flag. Bowtie2 outputs the data in 'sequence alignment map' format, which is pretty typical. SAM format is rather complex and compact and contains information about which reads map, and to what locations; you can see details here: https://en.wikipedia.org/wiki/SAM_(file_format).

Process the SAM file with samtools

Step 4.

Rather than spending too much time looking at the SAM file I usually just start processing it with samtools. Often you will see SAM files converted into BAM files (binary alignment map). BAM files are smaller, since they do not require encoding to make them human-readable, and they can be processed more quickly. Since SAM and BAM files contain the same information, you will often see SAM files deleted and BAM files used for storage.

The following commands are a bit tedious, but they are all required to convert SAM->BAM, sort, and index the mapped reads. Before we begin take a quick look at all of the utilities available with samtools by typing "**samtools**" in your command line.

First we need to convert SAM->BAM and remove reads that did not map:

```
samtools view -bS -F 4 mapping_out.sam > mapping_out.bam
```

Then we need to sort the reads [note: the following command works with samtools v 0.1.19-96b5f2294a, but the syntax is different with different versions. If you get an error message try using the -o flag and specifying mapping_out.sort.bam as the output file]:

```
samtools sort mapping_out.bam mapping_out.sort
```

Finally we need to index the sorted read bam file:

```
samtools index mapping_out.sort.bam
```

All of this may seem a bit tedious, but at the end we have a nice sorted bam file that we can use for multiple applications downstream. Sorted bam files are also useful for storing/sharing data, since they are typically smaller than fastq or sam files.

Download the GFF file for the genome

Step 5.

Now we have the genome file and a sorted bam file with information about where certain transcriptome reads mapped, but we still need information about where genes and other features are encoded in the mycobacteriophage D29 genome. After all, knowing that a large number of reads map between coordinates 10377 and 11421 in the phage genome is not necessarily useful unless we also have information about what genes are encoded in that region.

To get the genome annotation we will download the Gene Feature Format table (GFF) for the D29 genome. GFF files are typically used for storing genome annotation information, and one such file was already generated for D29 as part of the genome sequencing project for this bacteriophage. We can do this in the same way that we downloaded the genome above:

```
wget -O D29.gff.gz
ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/003/004/865/GCA_003004865.1_ASM300486v1/
GCA_003004865.1_ASM300486v1_genomic.gff.gz
gunzip D29.gff.gz
```

As it turns out, there are many different formats that are used to encode genome annotation information. In the next step we will be using a tool called BEDtools which will require a Browser Extensible Data (BED) file. So we need to convert the GFF file to a BED file. For this we will use a utility called gff2bed in the BEDOPS tool:

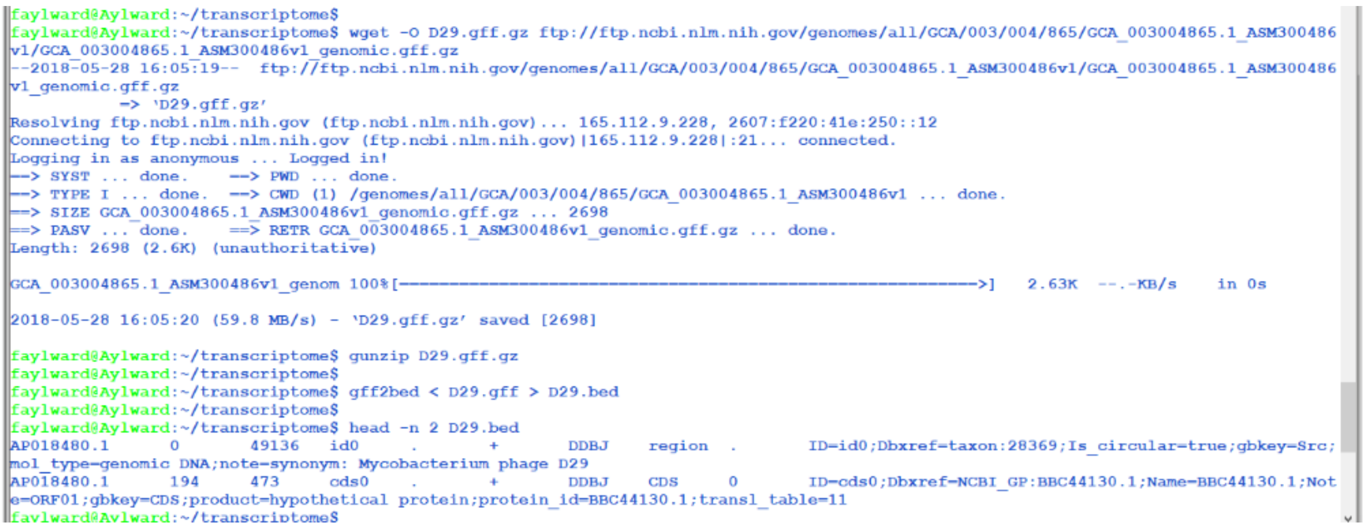
```
gff2bed < D29.gff > D29.bed
```

BED files are tab-delimited tables with information about the different features of a genome, their coordinates, which strand they are encoded in, and some annotation information. The tables are a bit large and somewhat unweildy to look at in the command line, but just to get an idea of a format let's look at the first two lines:

```
head -n 2 D29.bed
```

This should give you the general idea. Note that the 10th column is super long and contains lots of details about the gene name, protein name, etc. For details on BED format see here: https://genome.ucsc.edu/FAQ/FAQformat.html#format1

The code from this step should look something like this:



```
faylward@Aylward:~/transcriptome$
faylward@Aylward:~/transcriptome$ wget -O D29.gff.gz ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/003/004/865/GCA_003004865.1_ASM300486
v1/GCA_003004865.1_ASM300486v1_genomic.gff.gz
--2018-05-28 16:05:19--  ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCA/003/004/865/GCA_003004865.1_ASM300486v1/GCA_003004865.1_ASM300486
v1_genomic.gff.gz
           => 'D29.gff.gz'
Resolving ftp.ncbi.nlm.nih.gov (ftp.ncbi.nlm.nih.gov)... 165.112.9.228, 2607:f220:41e:250::12
Connecting to ftp.ncbi.nlm.nih.gov (ftp.ncbi.nlm.nih.gov)|165.112.9.228|:21... connected.
Logging in as anonymous ... Logged in!
==> SYST ... done.    ==> PWD ... done.
==> TYPE I ... done.  ==> CWD (1) /genomes/all/GCA/003/004/865/GCA_003004865.1_ASM300486v1 ... done.
==> SIZE GCA_003004865.1_ASM300486v1_genomic.gff.gz ... 2698
==> PASV ... done.    ==> RETR GCA_003004865.1_ASM300486v1_genomic.gff.gz ... done.
Length: 2698 (2.6K) (unauthoritative)

GCA_003004865.1_ASM300486v1_genom 100%[=========================================================>]   2.63K  --.-KB/s    in 0s

2018-05-28 16:05:20 (59.8 MB/s) - 'D29.gff.gz' saved [2698]

faylward@Aylward:~/transcriptome$ gunzip D29.gff.gz
faylward@Aylward:~/transcriptome$
faylward@Aylward:~/transcriptome$ gff2bed < D29.gff > D29.bed
faylward@Aylward:~/transcriptome$
faylward@Aylward:~/transcriptome$ head -n 2 D29.bed
AP018480.1      0       49136   id0     .       +       DDBJ    region  .       ID=id0;Dbxref=taxon:28369;Is_circular=true;gbkey=Src;
mol_type=genomic DNA;note=synonym: Mycobacterium phage D29
AP018480.1      194     473     cds0    .       +       DDBJ    CDS     0       ID=cds0;Dbxref=NCBI_GP:BBC44130.1;Name=BBC44130.1;Not
e=ORF01;gbkey=CDS;product=hypothetical protein;protein_id=BBC44130.1;transl_table=11
faylward@Aylward:~/transcriptome$
```

Identify highly expressed genes with BEDtools

Step 6.

Now we have all of the necessary files to identify which genes were highly expressed in our transcriptome.

First let's take a look at the tool BEDtools and what utilities it has by typing 'bedtools'. You will see a large number of options- this is a very comprehensive tool!

Here we will be using the 'intersect' utility, which will tell us which reads in our sorted bam file intersect with regions in the BED file.

bedtools intersect -a D29.bed -c -bed -f 0.2 -b mapping_out.sort.bam > intersect.txt

A quick note on the flags used here.

-a and -b denote the BED file and sorted bam file, respectively.

-c indicates that we want to count the number of reads that overlap with BED regions. The default output is to provide each read individually with the feature it mapped to.

-f 0.2 denotes that the read has to overlap by at least 20% of it's length to be considered mapping to a feature. One can imagine a long polycistronic mRNA with many genes encoded in it; some reads may overlap just slightly with a specific gene. Here we require at least 20% overlap, which will help ensure reads are mapping properly.

The output will be a BED file, the same as we used for the input, but with an additional column appended to the end. This column will have the number of reads tha map to each feature.

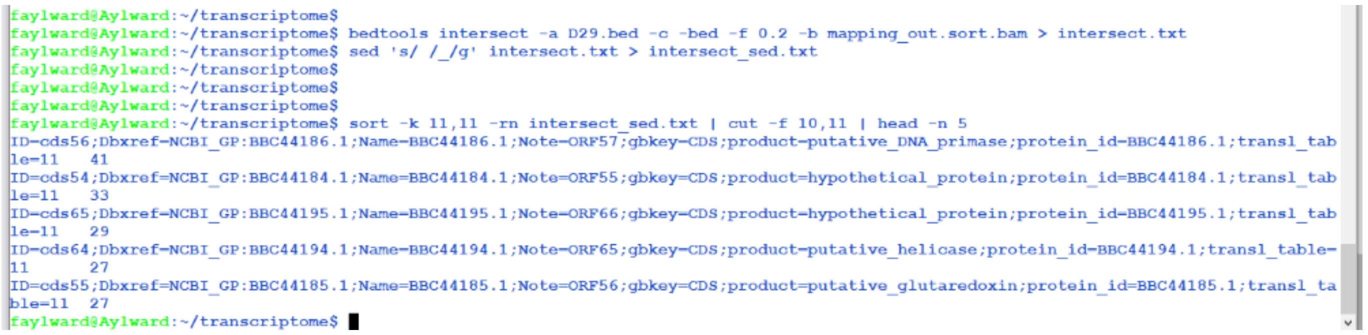
Now a rather basic question that it would be nice to answer is: What are the highest expressed genes in our transcriptome?

To answer this we should be able to sort the intersect file by the last column with the Unix sort command. Sometimes this can be complicated by spaces that are present in the intersect file, so let's do a quick text replacement first:

sed 's/ /_/g' intersect.txt > intersect_sed.txt

And then sort by the 11th column, which is where our mapping counts are:

sort -k 11,11 -rn intersect_sed.txt | cut -f 10,11 | head -n 5



```
faylward@Aylward:~/transcriptome$
faylward@Aylward:~/transcriptome$ bedtools intersect -a D29.bed -c -bed -f 0.2 -b mapping_out.sort.bam > intersect.txt
faylward@Aylward:~/transcriptome$ sed 's/ /_/g' intersect.txt > intersect_sed.txt
faylward@Aylward:~/transcriptome$
faylward@Aylward:~/transcriptome$
faylward@Aylward:~/transcriptome$
faylward@Aylward:~/transcriptome$ sort -k 11,11 -rn intersect_sed.txt | cut -f 10,11 | head -n 5
ID=cds56;Dbxref=NCBI_GP:BBC44186.1;Name=BBC44186.1;Note=ORF57;gbkey=CDS;product=putative_DNA_primase;protein_id=BBC44186.1;transl_tab
le=11   41
ID=cds54;Dbxref=NCBI_GP:BBC44184.1;Name=BBC44184.1;Note=ORF55;gbkey=CDS;product=hypothetical_protein;protein_id=BBC44184.1;transl_tab
le=11   33
ID=cds65;Dbxref=NCBI_GP:BBC44195.1;Name=BBC44195.1;Note=ORF66;gbkey=CDS;product=hypothetical_protein;protein_id=BBC44195.1;transl_tab
le=11   29
ID=cds64;Dbxref=NCBI_GP:BBC44194.1;Name=BBC44194.1;Note=ORF65;gbkey=CDS;product=putative_helicase;protein_id=BBC44194.1;transl_table=
11      27
ID=cds55;Dbxref=NCBI_GP:BBC44185.1;Name=BBC44185.1;Note=ORF56;gbkey=CDS;product=putative_glutaredoxin;protein_id=BBC44185.1;transl_ta
ble=11  27
faylward@Aylward:~/transcriptome$
```

So of the top 5 most highly expressed genes in the mycobacteriophage D29 genome, we have a DNA primase, a helicase, two hypothetical proteins, and one glutaredoxin. The first two are noteworthy since this particular transcriptome sample was taken after only 15 minutes of infection, and at this early stage we would expect to find genes associated with viral DNA replication ("early genes"). The hypothetical proteins are also not surprising, since many genes in bacteriophage genomes are uncharacterized and have no known function.