SYSB 3036 W10: Read mapping

Version 4

Frank Aylward[1]

[1]Virginia Tech

dx.doi.org/10.17504/protocols.io.v4pe8vn

Frank Aylward
Virginia Tech

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 STATUS

Working
We use this protocol in our group and it is working

Download some data from GitHub and enter the new folder

1 This tutorial will be combined with next week's since read mapping and analysis is rather complex. For this reason we will download a file that I posted on GitHub, but we won't need it until next week. It will be easiest to do everything from this week and next in the same folder, so that is why we will initialize everything now at the start.

So to get started let's clone the read_mapping_tutorial folder from GitHub, and then enter that folder:

```
git clone https://github.com/faylward/read_mapping_tutorial
cd read_mapping_tutorial
```

Now let's see what is there:

```
ls -lh
```

You should see a file called D29.bed. Don't worry about this for now. Just ensure that this file stays there and is not altered until we use it. Now let's continue with the next steps...

Download the reference genome and index a database with bowtie2

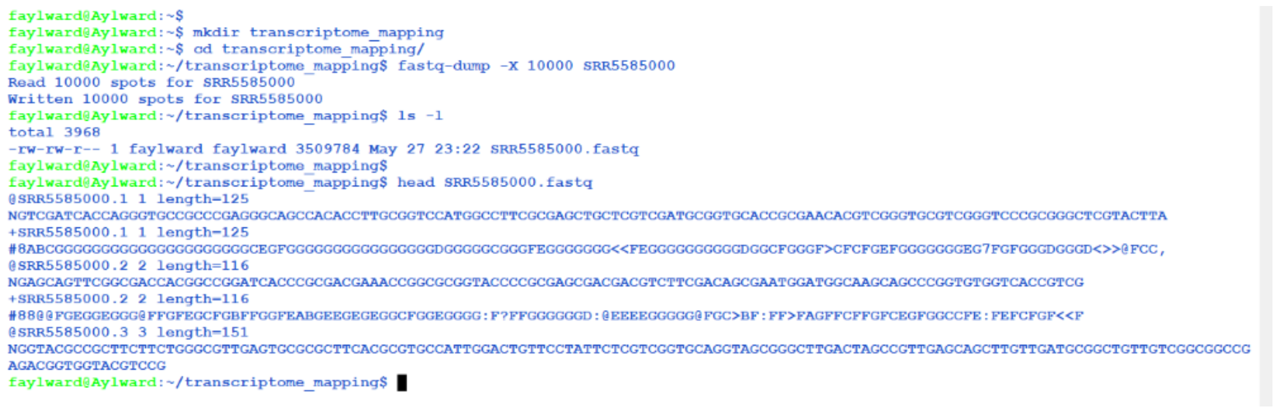
2 The datasets we will be using here are part of the Dedrick et al BMC Microbiology paper (more details in the Description for this tutorial), which describes some very cool transcriptomic experiments of Mycobacterium smegmatis while is infected with various bacteriophage.

There are several short-read Illumina transcriptomes that are part of this study, but this week we 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).

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
#8ABCGGGGGGGGGGGGGGGGGGGGCEGFGGGGGGGGGGGGGGGDGGGGGCGGGFEGGGGGGG<<FEGGGGGGGGGGDGGCFGGGF>CFCFGEFGGGGGGGEG7FGFGGGDGGGD<>>@FCC,
@SRR5585000.2 2 length=116
NGAGCAGTTCGGCGACCACGGCCGGATCACCCGCGACGAAACCGGCGCGGTACCCCGCGAGCGACGACGTCTTCGACAGCGAATGGATGGCAAGCAGCCCGGTGTGGTCACCGTCG
+SRR5585000.2 2 length=116
#88@@FGEGGEGGG@FFGFEGCFGBFFGGFEABGEEGEGEGGCFGGEGGGG:F?FFGGGGGGD:@EEEEGGGGG@FGC>BF:FF>FAGFFCFFGFCEGFGGCCFE:FEFCFGF<<F
@SRR5585000.3 3 length=151
NGGTACGCCGCTTCTTCTGGGCGTTGAGTGCGCGCTTCACGCGTGCCATTGGACTGTTCCTATTCTCGTCGGTGCAGGTAGCGGGCTTGACTAGCCGTTGAGCAGCTTGTTGATGCGGCTGTTGTCGGCGGCCG
AGACGGTGGTACGTCCG
faylward@Aylward:~/transcriptome_mapping$ █
```

Download some data from GitHub and enter the new folder

3 FASTQ files are similar to FASTA file, except that they in addition to sequence information they also encode quality values for each base of the sequence. This is because there is an error rate associated with DNA sequencing, and sometimes with raw sequencing data there can be quite a bit of low-quality bases that we want to be cautious when using (sometimes we may wish to remove these low-quality regions altogether!).

Let's take a look at the FASTQ file we just downloaded.

head SRR5585000.fastq

Note that the header line starts with a "@" rather than a ">" that is typical of FASTA files. Also note that for every sequence there are 4 lines. The first contains the "@" symbol and the non-redundant sequence identifier, the second contains the sequence, the third starts with a "+" and contains option descriptions, and the fourth contains the quality scores for the sequence (encoded symbolically).

We can also parse through FASTQ files using seqkit:

seqkit fx2tab -n -l -i SRR5585000.fastq | head

Make sure to pipe the results to "head" since there are many sequenced here.
Using seqkit again we can pipe the results to datamash to get the mean sequence length:

seqkit fx2tab -n -l -i SRR5585000.fastq | datamash mean 4

Download a reference genome

4 Now we need to get a reference genome to map reads against. The raw reads themselves are not that interesting unless we have a good quality reference genome that we can map the reads against.

The focus of this particular 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

Index the reference genome

5 We are going to map reads using bowtie2, which requires that the FASTA genome file is indexed first. Most (if not all) read mappers and homology search tools require that some kind of index or database is constructed first before the sequence comparisons are performed. Bowtie is no different.

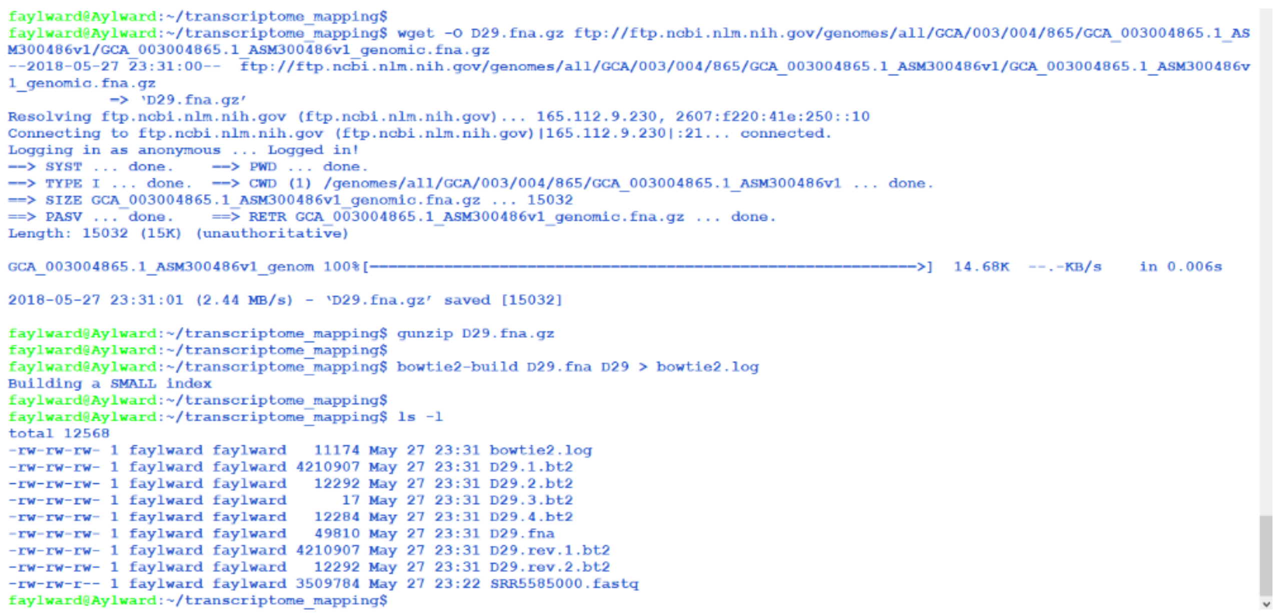
To do this we can use the bowtie2-build command. First just try typing "bowtie2-build" just to get a handle on the general usage and options of this tool. For purposes here we can run the following command:

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. It is also sometimes annoying to have a lot of text output directly to the command line.

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

6 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 T1.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. For purposes here let's call the outfile T1 since it's our first time-point.
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).

Take a look at the SAM output

7 Now let's take a look at the SAM output. SAM stands for "Sequence Alignment Map". SAM files are typically very large and very difficult to interpret. Most people would probably analyze SAM files with programs like Samtools (which we will go into later) and examine summary statistics rather than look in any detail at an actual raw SAM file. However it is useful to acquaint ourselves with the format a bit.

```
head T1.sam
```

and

```
tail T1.sam
```

You will probably notice a few things. The SAM file starts with a few description lines that start with the "@" symbol. These lines contain information about the program used for mapping, the reference sequence name, and some other things (all in very specific format).
After this there is one line per sequence that was mapped, together with some columns that encode specific information. The first column after the name encoded a flag that specified whether or not the read was mapped. Note that even reads that were not mapped (specified with a "4" in this first column). You may see a few that are mapped (code "0") and that some of the other columns after this have information about the sequence they mapped to, and the coordinates.

Overall SAM files are a fairly standard way of storing sequence and mapping information together in one file, but they are quite large. Check out the files with "ls":

```
ls -lh
```

Notice that the SAM file is about a large as the FASTQ file. This becomes problematic when we need to store lots and lots of mapped read data.

Convert to SAM to BAM

8 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 T1.sam > T1.bam
```

the -F 4 flag specifies that we remove all reads with the "4" flag, which corresponds to those that did not map. The -bS flag is a combination of -b and -S (combined for convenience here). -b specifies that the output should be BAM format, and the -S specifies that the input is SAM.

Check out the output file with ls:

ls -la

Note that the BAM file is much smaller than the SAM file. This is because the BAM file only contains reads that mapped, and it is in a more compact binary format.

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 T1.bam T1.sort
```

Finally we need to index the sorted read bam file:

```
samtools index T1.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.

To check that everything worked as we expected, we can get some summary statistics using the samtools sub-command "idxstats":

```
samtools idxstats T1.sort.bam
```

You should see that 13670 reads mapped, just like we were told when we ran bowtie the first time. This tells us that the sorted BAM file contains the same core information as the original SAM file (only now in much more compact and sorted format).