11.2 RNA-sequencing alignment

11.2.1 How alignment works

Previously we discussed FASTQ files which contain RNA-seq data, and the genome FASTA and GTF files that make up the reference genome. At this point, the RNA-seq reads have to be aligned to the reference genome. Once they are aligned, the reads are much more informative and used for a variety of analyses. Quantities of transcripts that exist in a single experiment can inform which genes are being expressed (and to what degree). To align the reads, you must use alignment software that takes the following as input: RNA-seq files and the reference genome.

You can imagine that the alignment process is very time intensive, considering that each read needs to be compared to every section of the genome to find its best match. Considering that a genome is often composed of billions of base pairs, while a read can be 30 to 100 base pairs long, scanning each read through the entire genome sequence would not actually be feasible. Instead, a genome index is created which serves similarly to an index in a book. Depending on certain characteristics in a particular read, the index can find the region of the genome with the most likely match, and then the scan only has to occur in that area.

When aligning RNA-seq reads, the software performing the alignment considers the possibility that there may be errors in the sequence, or that certain reads may map to the reference genome multiple times, or that certain bases are of low quality (using FASTQ quality info). Using this information, the alignment tool outputs the most likely alignment possibilities. The resulting file is an extremely large document that records which reads have aligned to which positions of the genome and how well they have aligned. These files are called BAM or SAM files, and will be discussed later in this document.

11.2.2 Running the alignment

Aligning RNA-seq data takes a lot of time and computational resources, so we will not be running it in this course. However, we will be going over the various steps involved in preparing the files for alignment. We'll start by looking at how you would align a single FASTQ file of human RNA-seq reads to the human genome using the alignment tool "HISAT2". For a deep dive into exactly how to use this tool or to find out more about the commands we will be describing below, you can always check out the HISAT2 manual: http://daehwankimlab.github.io/hisat2/manual/ . The dataset you will be using is a subset of the entire dataset (which consists of 6 FASTQ files) which you will be analyzing in R next week.

HISAT2 is a popular piece of software used to align both DNA and RNA reads to either one or multiple reference genomes. For input, it requires (1) the sequencing reads, and (2) a reference genome. In our case, the sequencing reads are the full version of the FASTQ file that you looked at in section 11.1. If you were processing these data from scratch, you would run HISAT2 on each sample that you are working with.

The reference genome is a bit trickier, as it still needs to be indexed (indexing described in 11.2.1). To index the reference genome, you would run the command ` hisat2-build`. For example, here is how you would run it if the name of your reference genome FASTA file was "genome.fna" and you wanted the base name of your output to be "genome_index":

```
hisat2-build genome.fna genome_index
```

The output is the index for the reference genome, which is divided into 8 files that each have the .ht2 extension:

```
genome_index.1.ht2

genome_index.2.ht2

genome_index.3.ht2

genome_index.4.ht2

genome_index.5.ht2

genome_index.6.ht2

genome_index.7.ht2

genome_index.8.ht2
```

All of these files are part of a single index, making them a set of index files. These are all you need for the reference genome part of the HISAT2 alignment command. You don't need to understand how this particular file format works.

Now let's see what it would be like to run HISAT2 by looking at what the inputs and outputs would be if you were able to run it on your own machine. This is a completely new program for us, so it's good to get a feel of how it works before we try running it. Most programs have help pages that can be accessed on the command line as such:

```
hisat2 --help
```

What comes up now is the HISAT2 help page, with the main section looking like this (before a ton of other options!):

```
HISAT2 version 2.2.1 by Daehwan Kim (infphilo@gmail.com,
www.ccb.jhu.edu/people/infphilo)

Usage:

 hisat2 [options]* -x <ht2-idx> {-1 <m1> -2 <m2> | -U <r>} [-S <sam>]

  <ht2-idx>  Index filename prefix (minus trailing .X.ht2).

  <m1>       Files with #1 mates, paired with files in <m2>.

             Could be gzip'ed (extension: .gz) or bzip2'ed (extension:
.bz2).

  <m2>       Files with #2 mates, paired with files in <m1>.

             Could be gzip'ed (extension: .gz) or bzip2'ed (extension:
.bz2).

  <r>        Files with unpaired reads.

             Could be gzip'ed (extension: .gz) or bzip2'ed (extension:
.bz2).
```

```
  <sam>      File for SAM output (default: stdout)

  <m1>, <m2>, <r> can be comma-separated lists (no whitespace) and can
be specified many times.  E.g. '-U file1.fq,file2.fq -U file3.fq'.
```

This tells us exactly how we should be using HISAT2, and what arguments are required for the program to work. It also provides other useful information, like the version of the program (2.2.1). Another thing that we see is that this program gives us a SAM file as output, which is a specific type of alignment file.

Let's start building the command we're going to use. We start with `hisat2`. Right after this is where we would include any of the optional arguments listed in the printout above (as noted by [options]). Because these are not mandatory to include, we'll skip them for now. Then we see -x followed by <ht2-idx>, which suggests that we should be using the `-x` flag followed by the indexed reference genome. We can specify all of them by referring to the name of the file before the extension, like this:

```
-x genome_index
```

For the next step, we have to specify the reads to use. The formatting of this next chunk {-1 <m1> -2 <m2> | -U <r>} looks confusing. What do we put here? Well, take a look at the pipe and what it's separating within the curly brackets. If we look at the description of m1, m2, and r in the help text, we can see that these are specifying files with #1 or #2 mates, as well as unpaired reads. This is because RNA-seq reads can exist as either paired-end or single-end reads (described in 11.1.2), where paired-end reads are an individual read that has been sequenced from both ends. If you are using these reads, you have two sets of FASTQ files, one with the beginning sequences (#1 mates) and one with the end sequences (#2 mates). We only have one FASTQ file with unpaired reads, so we don't have to worry about this. The pipe indicates that we have another option, which is to use the -U flag and `r` is our FASTQ file placeholder. In the end, we can specify the reads like this:

```
-U 16T_reads.fastq
```

Finally we can specify the name of the SAM output file. We should name it something meaningful so that we recognize the file once it is created and placed in our directory. Let's call it ` 16T_hisat2.sam`. Now our command is:

```
hisat2 -x genome_index -U 16T_RNA.fastq -S 16T_hisat2.sam
```

After running this command, the alignment is complete, and you have created the file ` 16T_hisat2.sam`. What is this file? What does it look like? How do you read it? We'll go over all of this in the next section.

11.2.3 BAM/SAM files

SAM and BAM files are alignment file types. They specify where each read in your sample has aligned to the genome, and if there are any mismatches or gaps in the alignment, and many other helpful metrics. The main difference between SAM and BAM files, is that the SAM file is "human-readable" which means that you can interpret the text when you look into the file. BAM files, however, are a binary, compressed version of the SAM file; you can't read them, but they take up much less space on your computer. If you need to manipulate SAM or BAM files there is a command line tool called "Samtools" that is often used for converting files between SAM and BAM format, sorting these files, and other tasks involving these files. We won't go over how to use this tool in class, but it may be useful to you if you have to process genomic data outside of class. You can find more information about Samtools at http://www.htslib.org/ .

After running the alignment, you would have the file `16T_hisat2.sam` sitting in the directory in which you ran HISAT2. Often, if a program hasn't worked, the output will either be empty or contain an error. It is always good to check these files to see if this happens, because it could be incorrect and confusing if you continue your analysis with an incorrect or empty file. Even though you didn't actually run the command, the first 1000 lines of the file have already been placed into your directory for you to explore. Let's peek inside this file to see what it looks like.

```
less 16T_1000_lines.sam
```

The file starts off like this:

```
@HD     VN:1.0  SO:unsorted

@SQ     SN:chr1 LN:248956422

@SQ     SN:chr2 LN:242193529

@SQ     SN:chr3 LN:198295559
```

And eventually transitions into lines that look like this:

```
16T_RNA.8 4       *       0       0       *       *       0       0
GTTCGGCGAAGCTGAGAATTGCCCGTTGTTCGTTCAGC
CCCCCAAA=ACCCCCCCCCCCCCCCCCBBCCCBCBBA>  YT:Z:UU

16T_RNA.5 4       *       0       0       *       *       0       0
CACCCTCTACCATACTCTAGCTCGTCAGTTTTGAATGC
00/,/68677A@BBB?ABBB8ABBB6@B6@>>>>7?@6  YT:Z:UU

16T_RNA.18        4       *       0       0       *       *       0
0       GCCGGATAAAACTGCGTGGCGGGGGTGCGGCGGGGCTG
8878788723???<?#######################  YT:Z:UU

16T_RNA.3 16      chr17   58001449        60      38M     *       0
0       CAGACCCAAACAATCAAGAACCAGAAAATGGGTTTCTT
CCCCCBCCCCCCCCCCCCC@CCCCCCCCCCCCCCCCCC  AS:i:0  XN:i:0  XM:i:0  XO:i:0
XG:i:0  NM:i:0  MD:Z:38 YT:Z:UU NH:i:1
```

Yes, this is a SAM file! A bit confusing, isn't it? The first section that you see is an optional header section that begins with all of the @ symbols. The letters directly after the @ symbol are tags that indicate the type of information present on that line. The first line is HD and is only one line, and generally describes the information in the file. The other capital letters in the HD header are more tags that specify the rest of the information. VN is the version of the format (1.0) and SO indicates how the file is sorted (unsorted).

Following this, we have SQ lines. This is the "reference sequence dictionary" and defines the sorting order of the rest of the file. SN is the sequence name and corresponds to chromosomes in the human genome. (In a genome annotated without chromosomes, these would correspond to "contigs" or chunks of the genome sequence that are separate in the reference genome FASTA file). LN is the length of the reference sequence in base pairs.

There is one final type of line in the header that occurs before the alignment section of the file, PG. This specifies the programs that were used in the creation or processing of the file. It is not listed above, but you can find it in the SAM file.

Now we get to the alignment section, the most important and only mandatory section of the file. The lines are not proceeded by @ symbols, which is how you can distinguish them from the header. Generally, each line represents each cDNA sequence from the FASTQ file(s), and also includes information about how this sequence mapped to the genome. All of this information makes these files extremely large.

Now let's dissect how alignment information is stored in the SAM file by looking at a single line. Remember that when we look at a file with `less`, the text wraps and it can be difficult to tell when a new line begins. If this is confusing, you can always look at a file with `nano` (a text editor - to escape the editor, press `ctrl-x`) where this isn't a problem.

```
16T_RNA.6 0        chr1    109309976        60       38M      *        0
0        CGAGGATAAGGAGAGTATGACCAAAACCCTCCTCCCTA
CCBCCCC@BCBBBBB@B@BBBBBBBBBBBBBBBBBBB  AS:i:0  XN:i:0  XM:i:0  XO:i:0
XG:i:0  NM:i:0  MD:Z:38 YT:Z:UU NH:i:1
```

At first glance, this looks pretty confusing; but by looking up explanations of the SAM file format (https://samtools.github.io/hts-specs/SAMv1.pdf), we can begin to pick it apart.

- 16T_RNA.6 - "Query template name", or essentially a read identifier
- 0 - a flag indicating how the read has mapped (e.g. forward, reverse, or unmapped)
- chr1 - the reference sequence/chromosome the read has aligned to
- 109309976 - starting position of the read on the reference sequence; in this case, the read maps to the 109309976th base pair of chromosome 1 on the human genome
- 60 - measure of mapping quality
- 38M - "CIGAR", indicates mismatches in the alignment; in this case there is a mismatch after the 38th base pair
- * - "RNEXT", the reference sequence or contig of this particular read's "mate"; these reads do not have mates (paired-end data does) so this field is left blank, or as an asterisk
- 0 - "PNEXT", the starting position of the mate mentioned in "RNEXT"; there is no mate to this read, which is why the value here is 0
- 0 - "TLEN", the length of both mates from the leftmost position to the rightmost position; this is 0 because there is no mate
- CGAGGATAAGGAGAGTATGACCAAAACCCTCCTCCCTA - "SEQ", the query sequence or read that was in the sample
- CCBCCCC@BCBBBBB@B@BBBBBBBBBBBBBBBBBBB - "QUAL", the quality of each base pair in the sequence
- AS:i:0 XN:i:0 XM:i:0 XO:i:0 XG:i:0 NM:i:0 MD:Z:38 YT:Z:UU NH:i:1 - these tags contain various sources of information corresponding to the alignment of this particular read. We won't go through all of these tags, but more information can be found about each of them online. These tags become important when you are trying to filter or sort reads based on a particular characteristic that may be recorded by one of these tags.

This wraps up the second section of this week on RNA-sequencing alignment, where you have gone from raw reads to an alignment file. Next, we will be exploring how to turn this alignment file into a count matrix, which you can then analyze in R.