RNA-seq analysis in R

Read alignment with HISAT2

Mapping reads to a reference genome

Once we are happy with the data quality, we can start to analyse the data. Usually, the first step into the analysis requires mapping the RNA-seq reads to the genome. There are numerous tools we could use to perform short read alignment and the choice should be made carefully according to the analysis goals and requirements. For RNAseq gene expression analysis HISAT2 is a very fast tool that has been shown to have a good performance on published benchmarks.

We are going to be aligning the reads from the fastq files against the GRCm38 reference genome from Ensembl.

1 Indexing the reference genome for Hisat2

Typically our genome reference will be in FASTA format. Before we can start mapping RNA-seq reads to the genome, we need to create an index to the genome. This index allows HISAT2 to quickly search the genome for possible mapping positions for each read. It is analogous to an index in a book: if you want to find out where in the book a particular word occurs it is quicker to look at the index than to search through the book word by word, then you can jump straight to the correct page to find the exact line and position of the word.

The command to index the genome for HISAT2 is `hisat2-build`.

Access the help page to find the basic usage and other options:

```
hisat2-build --help
```

The **Usage** provides an outline of the way to use the command:

```
hisat2-build [options]* <reference_in> <ht2_index_base>
    reference_in            comma-separated list of files with ref sequences
    hisat2_index_base       write ht2 data to files with this dir/basename
```

This means that we have to use the command `hisat2-build` to run the programme.

The `[options]*` means that there are various optional arguments we can provide after the main command. These are listed in detail under the basic usage. We will only be using one option - `-p`.

Then we need to replace `<reference_in>` with the name of our reference genome files and replace `<hisat2_index_base>` with a 'basename' for the index files that the command will generate.

HISAT2 is going to generate a number of files for the index. The 'basename' is what all the name of these files will start with. If the 'basename' we give is `references/my_index` then we will get a number of files called `myindex.1.ht2`, `myindex.2.ht2`, `myindex.3.ht2` ...etc. inside the directory `references` (this directory must already exist).

For example, if our reference fasta file is called `my_reference.fa` and we want to write the index to `references/my_index`, then we use the command:

```
hisat2-build -p 7 my_reference.fa references/my_index
```

Note that we are not typing the `[`, `]`, `<`, and `>` symbols. In the usage the square brackets denote *optional* arguments and the `<>` denote *required* arguments.

Exercise 1

1. Check your current working directory and if necessary navigate to the `Course_Materials/` directory using the command `cd` (change directory).
2. Use `ls` to list the contents of the directory.
   There should be a `references` directory. This will contain various reference materials that we will need throughout the analysis, such as the mouse genome and gene annotations.
3. Use `ls references` to list the contents of the `references` directory.
   There should be a file called `Mus_musculus.GRCm38.dna_sm.chr14.fa`. This is the reference genome sequence for chromosome 14 in FASTA format. We are justgoing to work with chromosome 14 for this exercise as indexing the entire genome would take too long. You'll also notice a directory called `hisat2_index`, this is the index for the entire genome, which has already been generated. We'll be using this later.
4. We need a directory for hisat2 to write the index files in. Make a directory (`mkdir`) inside the `references` directory called `hisat2_index_chr14`:

```
mkdir references/hisat2_index_chr14
```

5. To create the hisat2 index run the following command:

```
hisat2-build -p 7 \
    references/Mus_musculus.GRCm38.dna_sm.chr14.fa \
    references/hisat2_index_chr14/mmu.GRCm38
```

Note: The `\` at the end of each line tells the terminal that when you press `Enter`, you have not yet finished typing the command. You can if you wish, type the whole command on a single line, omitting the `\`. The command is written across multiple lines here just to make it easier to read.

With this command we are:

- providing the fasta file *references/Mus_musculus.GRCm38.chr14.fa* for the `<reference_in>`
- setting the `<hisat2_index_base>` to *references/hisat2_index_chr14/mmu.GRCm38*, so all the files will be created in the directory *references/hisat2_index_chr14* and their names will start with `mmu.GRCm38`.

Questions:
a) Why do we use `-p 7`? Take a look at the `hisat2-build` help page.
b) How many files are created?

2 Align with HISAT2

To map the reads to the genome we need to run `hisat2`.

We will need to provide the command with three pieces of information:

- The path to the index files - we do this by just supplying the basename for the index as in the previous command
- A fastq file containing our unaligned reads
- A name for the output file

We should also instruct HISAT2 how many threads (processors) it should use (these machines have 8 processors, so we should let HISAT2 use 7 of them, keeping 1 free).

Take a quick look to HISAT2's description:

```
hisat2 --help
```

The usage is:

```
Usage:
  hisat2 [options]* -x <ht2-idx> {-1 <m1> -2 <m2> | -U <r> | --sra-acc <SRA accession number>} [-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).
  <SRA accession number>        Comma-separated list of SRA accession numbers, e.g. --sra-acc SRR353653
  <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'.
```

`[options]*` refers the many optional parameters listed in the subsequent `Options` section of the help. We do not need to worry about these for now.

After the options there are three arguments.

The first required argument is the name of the hisat file index, which should be preceded by the `-x` flag:

```
`-x <ht2-idx>`
```

The usage tells us that the *"filename prefix"* should be provided, that is the filename without the `.X.ht2` suffix:

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

The chromosome 1 index that we created in Exercise 1 had 8 files:

```
references/hisat2_index_chr14/mmu.GRCm38.1.ht2
references/hisat2_index_chr14/mmu.GRCm38.2.ht2
...
references/hisat2_index_chr14/mmu.GRCm38.8.ht2
```

So the *"index filename prefix"* would be `references/hisat2_index_chr14/mmu.GRCm38`.

The next argument is the raw read data to be aligned. The curly brackets (`{...|...}`) indicate that we must provide this in one of three ways:

- with the flags `-1` and `-2` to give two fastq files when we have paired end reads (as is the case today)

```
  <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).
```

- with the `-U` flag to just give one fastq if our reads are single end

```
  <r>        Files with unpaired reads.
             Could be gzip'ed (extension: .gz) or bzip2'ed (extension: .bz2).
```

- with the `-r` flag to provide an SRA accession number to download and process data directly from the NCBI's Sequence Read Archive (SRA).

```
<SRA accession number>        Comma-separated list of SRA accession numbers, e.g. --sra-acc SRR353653
```

The third argument is in square brackets (`[...]`), which indicates that argument `-S` is entirely optional. This argument is used to provide a file to write the results to.

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

If we do not provide this, the default is to write to `stdout`, which refers to the "standard output" and means the results would just be printed on the screen.

So we need to provide:

- the prefix of the index file after the `-x` flag
- the path to our fastq files after the `-1` and `-2` flags
- the name for the output SAM file after the `-S` flag. The output file format is SAM, so the output filename should end `.sam`.

There many additional options that allow us to tweak the parameters used in the alignment. As a general rule, unless you really know what you are doing you should stick to the defaults.

Exercise 2

Use HISAT2 to align the fastq file.

We wil be writing the output to the `bam` directory. You will notice there is already a file in there called `SRR7657883.chr14.sorted.bam`. This contains reads aligned to chromosome 14. We will be using this file in some exercises later in the course, try not to write over it or delete it.

Use the following parameters:

- Index (the full genome this time) - `references/hisat2_index/mmu.GRCm38`
- Fastq file #1 mate (read 1) - `fastq/SRR7657883.sra_1.fastq.gz`
- Fastq file #2 mate (read 2) - `fastq/SRR7657883.sra_2.fastq.gz`
- Output file - `bam/SRR7657883.sam`
- Set the number of threads (number of processors to use) to `7` - check the help page to find the appropriate flag
- Add the `-t` flag so that HISAT2 will print a time log to the console

Note: The alignment may take 5-10 minutes - go get a nice cup of tea.

3 Convert the SAM output to BAM

The output of HISAT2 is a SAM file. This is the standardized format for presenting aligned sequence data. You can read details of the specifications here:

https://samtools.github.io/hts-specs/SAMv1.pdf

Unfortunately, this is a plain text file and they tend to be very large. BAM files are the binary (compressed) version of SAM files. They are much smaller and can be indexed making them quicker to access and process.

3.1 SAM to BAM with samtools

We can transform from SAM to BAM using `samtools`. `samtools` is a toolkit that provides a number of useful tools for working with SAM/BAM files. In this case we will use the `view` command to transform the SAM file into a BAM file.

The general command is:

```
samtools view -b my_sample.sam > my_sample.bam
```

`view` is the tool for viewing a sam or bam file.

Normally it outputs in SAM format so that the results can be read by a human, in this case the `-b` flag tells it to output in BAM format.

By default `samtools view` outputs it's results directly to the console so that we can view them. The `>` redirects the output to the file `my_sample.bam`.

3.2 Sorting a bam file

After we have transformed the SAM file to a BAM file we will want to index it, but first we should sort the file so that the reads are in order with regard to chromosome number and position.

To sort the file the general command is:

```
samtools sort my_sample.bam > my_sample.sorted.bam
```

3.3 Indexing a bam file

Finally, we should index the bam with the `index` command. This makes accessing accessing the data quicker for downstream tools.

To index the file the general command is:

```
samtools index my_sample.sorted.bam
```

This would create a new file called `my_sample.sorted.bam.bai` (**ba**m **i**ndex).

Exercise 3

1. Transform your aligned SAM file in to a BAM file called `SRR7657883.bam`. Use the option `-@ 7` to use 7 cores, this vastly speeds up the compression.
2. Sort the BAM file to a create a bam files called `SRR7657883.sorted.bam`. Again use the `-@ 7` options to use 7 cores.
3. Index the sorted BAM file
4. Now that you have a bam file have a look at the header of the file with the command:
   `samtools view -H <my_sample.sorted.bam>`
   Replacing "`<my_sample.sorted.bam>`" with your bam file.
5. View the first few aligned sequences with the command: `samtools view my_sample.sorted.bam | head`
   The `|` symbol is known as the "pipe", it "pipes" the output of the first command into the next command. Take a look at the SAM format specifications here. Jump to section 1.4 " The alignment section: mandatory fields" and see if you can begin to interpret the the first 11 fields of the alignment information in your bam file. Don't forget the website Explain SAM flags which helps with decoding the flag field of the alignment.
