Introduction to Bulk RNAseq data analysis

Quantification of Gene Expression with Salmon

Contents

Quasi-mapping and gene expression quantification **1**

1. Indexing the transcriptome for Salmon **2**

Indexing the transcriptome 2

Exercise 1 - Create Salmon index 3

2. Gene expression quantification **4**

Exercise 2 - Quantify with Salmon 4

Exercise 3 6

4 Make transcript to gene table **6**

References **7**

Quasi-mapping and gene expression quantification

So far we have QC'd our raw reads, and the next step is to quantify the gene expression. One way would be to map the reads to the genome and use information about where genes are located on the genome to count the number of reads coming from each gene. This method is shown in the extended materials using the `featureCounts` tool.

An alternative is to use faster methods known as *quasi-mapping* or *pseudo-alignment*, followed by inferential estimation of gene expression. This approach has been developed over the past few years and has the advantages of being faster and more lightweight than full alignment (they can be run on a laptop rather than needing a high performance cluster). In addition these methods give more accurate estimates of gene expression at the transcript level and incorporate corrections for GC-content bias and sequence bias. They are also able to quantify multimapping reads in a more pragmatic manner than the standard alignment/counting approaches giving more accurate counts, particularly for genes belonging to families of genes with very similar sequences.

There are various tools that use this approach such at `Kallisto`, `Sailfish` and `Salmon`; we will use `Salmon` (Patro 2017).

You can find the full manual here:

https://salmon.readthedocs.io/en/latest/salmon.html#using-salmon

This provides much more information about various options and the reasons for using them. For most purposes we can simply use the default settings with just a few extras.

1. Indexing the transcriptome for Salmon

Salmon encompasses both alignment and quantification in a single tool. Here we will be aligning directly to the transcriptome rather than the genome. The transcriptome reference is a FASTA file containing sequences for all transcripts. Before we can run `Salmon` we need to create an index, which allows the Salmon to process the sequences more efficiently. For common species is possible to download pre-generated Salmon indexes, however, it is worth knowing how to create your own. This process is more computationally intensive and will not run on standard laptop or desktop.

The details of the index creation are taken from:

https://combine-lab.github.io/alevin-tutorial/2019/selective-alignment/

As well as including the transcriptome, we also want to include the genomic sequences. These will act as a *decoys* so that non-transcriptomic reads will not be erroneously counted against transcripts.

The indexing takes too long with the full transcriptome for the purposes of this training session, so we will work with the transcriptome for just genes on mouse Chromosome 14.

The full transcriptome has been downloaded from Ensembl:

ftp://ftp.ensembl.org/pub/release-102/fasta/mus_musculus/cdna/Mus_musculus.GRCm38.cdna.all.fa.gz

Note: This release is a few years old now, in reality you would want to use the latest release. We will update the course materials in the future to use the latest release, but the overall process will be the same.

Indexing the transcriptome

We are going to be giving Salmon both genomic and transcriptomic sequences to index. These files will be in FASTA format. FASTA is a text-based format for representing either nucleotide sequences or peptide sequences, in which nucleotides or amino acids are represented using single-letter codes. Each sequence has a header line that starts with ">". The header includes the name of the sequence and possibly some information about the sequence. The sequence itself is represented by one or more lines containing a string of nucleotides or amino acids.

First we need to combine the transcriptome and genome files into a single FASTA file. It is important to have the transcriptome sequences first in the file.

The genomic sequences are included as *decoys*. The reason for including them is that there may be reads in our library that did not originate from transcripts in our references (either they are from transcripts that have not been annotated or they are from genomic DNA). All aligners will always try to find an aligment for a read if it is possible. If we only provided the transcript sequences, some of these reads might adequately match the sequence of some transcripts and be inappropriately aligned to these transcripts. However, these reads will always align better to their correct genomic sequences than to the transcript, and so including the genomic sequences as decoys means that these reads will be aligned to the genomic sequences instead.

In order that Salmon can distinguish between transcript sequences and the *decoys*, we need a text file that lists the names of the decoy sequences. In our case we are just using chromosome 14, so we just need to create a file containing this number. But normally this file would contain the names of all the decoy sequences. One way to create this decoy text file in an automated way is to extract the sequence names from our reference genome file using some command line utilities. Here is an example code for our full mouse genome:

```
cat references/Mus_musculus.GRCm38.dna_sm.primary_assembly.fa | grep ">" | sed 's/>//' > decoys.primary
```

- The `cat` command is used to print the content of the file. If your reference was compressed (with `.gz` extension) you could use the `zcat` command instead.
- The `grep` command is used to find lines that match the ">" character, i.e. our genome sequence names (usually chromosomes and possibly some unassembled scaffolds).

- The `sed` command is used to substitute the character ">" with nothing, i.e. we want to remove that character from our text string, so we retain only the actual sequence name.
- With `>`, we redirect the output of these commands to a text file, which you could use as the decoy file when you build the index file.

Finally, we will need to provide salmon with the name of a directory in which to create the index. We don't need to make the directory, salmon will do this itself.

Exercise 1 - Create Salmon index

1. Create combined trancriptome/genome FASTA file We can simply con*cat*enate the two files using the `cat` command.

```
cat references/Mus_musculus.GRCm38.cdna.chr14.fa.gz \
  references/Mus_musculus.GRCm38.dna_sm.chr14.fa.gz \
  > references/gentrome.chr14.fa.gz
```

   If you are not familar with the FASTA format, take a look at the first couple of lines of the two input fasta files. First, the transcriptome:

```
zcat references/Mus_musculus.GRCm38.cdna.chr14.fa.gz | head -n 2
```

```
%l%$ zcat references/Mus_musculus.GRCm38.cdna.chr14.fa.gz | head -n 2
>ENSMUST00000177564.1
ATCGGAGGGATACGAG
```

   and then the genome:

```
zcat references/Mus_musculus.GRCm38.dna_sm.chr14.fa.gz | head -n 2
```

```
%l%$ zcat references/Mus_musculus.GRCm38.dna_sm.chr14.fa.gz | head -n 2
>14
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
```

   In each case the first line we see is the name of the sequence - in the first case it is the transcript name and in the second the chromosome name - and the second line is the (beginning of the) sequence itself. The decoy list needs to contain the names of the genomic sequences.

2. Create decoy sequence list from the genomic fasta

```
zcat Mus_musculus.GRCm38.dna_sm.chr14.fa.gz | grep ">" | sed "s/>//" > references/decoys.txt
```

3. Use `salmon index` to create the index. You will need to provide three pieces of information:

   - the **Transcript fasta file** - *references/gentrome.chr14.fa.gz*
   - the **decoys** - *references/decoys.txt*
   - the **salmon index** - a directory to write the index to, use **references/salmon_index_chr14**

   Also add `-p 7` to the command to instruct salmon to parallelize the process over 7 cores, which is significantly faster than just doing the indexing in a single process on 1 core. To find the flags for the other three pieces of information use:

```
salmon index --help
```

   One thing to note here is that we have not specified the `-k` parameter. This parameter sets the kmer size that the index will be based on and relates to the minimum size of kmer used for quasi-mapping. The default is 31 bases, and this is fine for read sizes over 75 bases. If your read size is less that 75, you

would need to adjust this. You should set the kmer size to slightly less than half of the read length. Quasi-mapping looks for kmers that are perfect match to the reference sequence. If the kmer size is more than half the read length, then a read with a mismatch in the middle of the read will never be able to be mapped to the transcriptome, even if all other bases are a perfect match for the sequence at the location that it originated.

Answer

We are already given the commands to achieve tasks 1 and 2. After running those, we can build our `salmon index` command, following the instructions given and also using the help documentation with `salmon index --help`.

This is our command:

```
salmon index \
    -t references/gentrome.chr14.fa.gz \
    -d references/decoys.txt \
    -p 7 \
    -i references/salmon_index_chr14
```

- `-t` is the file we want to index. This is our "gentrome" file which includes the transcripts as well as the actual genome, which is used as a decoy to avoid false-positive matches to the transcripts.
- `-d` is used to tell Salmon which of the sequences in the FASTA file are "decoys".
- `-p` indicates we want to use 7 processors for parallel processing (our training computers have 8 CPUs).
- `-i` indicates where we want to save the index file.

2. Gene expression quantification

Now that we have an index we can quickly get gene expression estimates directly from our raw fastq files.

We can use the index for the full transcriptome/genome here, rather than just chromosome 14, as this step is relatively quick. The full index should already be in the references directory: `references/salmon_index`. For this exercise we'll just quantify one sample: **SRR7657883**. We've already run salmon on the complete data set. You can see the results in the `salmon` directory. We'll use this for the differential gene expression analysis in later sessions.

We are going to be asking Salmon to output the read alignments as a SAM file as well as the gene expression quantification results. This is optional but we would like this information for some QC. Unfortunately, this does cause Salmon to take longer and running Salmon like this on the full fastq takes about 20 minutes. For the purposes of this practical we will use a smaller fastq file with just 2 million reads - this way we can run Salmon in a reasonable time. The files we will use are called *SRR7657883.subset_2M.....*

Exercise 2 - Quantify with Salmon

1. There should already be a directory called `salmon_output` in the `Course_materials` directory. If not, create it.
2. Use `salmon quant` to quantify the gene expression from the raw fastq. To see all the options run `salmon quant --help-reads`. There are lot of possible parameters, we will need to provide the following:
    - **salmon index** - *references/salmon_index*
    - `-l A` - Salmon needs to use some information about the library preparation, we could explicitly give this, but it is easy enough for Salmon to **A**utomatically infer this from the data.
    - **File containing the #1 reads** - *fastq/SRR7657883.subset_2M.sra_1.fastq.gz*

- **File containing the #2 reads** - *fastq/SRR7657883.subset_2M.sra_2.fastq.gz*
- **Output quantification directory** - *salmon_output/SRR7657883*
- `--writeMappings=salmon_output/SRR7657883.salmon.sam` - Instructs Salmon to output the read alignments in SAM format to the file *salmon_output/SRR7657883.salmon.sam*.
- `--gcBias` - salmon can optionally correct for GC content bias, it is recommended to always use this
- **The number of threads to use** - *7*

Salmon creates a separate output directory for each sample analysed. This directory contains a number of files; the file that contains the quantification data is called `quant.sf`.

Answer

To create a directory we can use the standard `mkdir` command:

```
mkdir salmon_output
```

Then, using the instructions given and the help documentation (`salmon quant --help`), we can build our quantification command:

```
salmon quant \
    -i references/salmon_index \
    -l A \
    -1 fastq/SRR7657883.subset_2M.sra_1.fastq.gz \
    -2 fastq/SRR7657883.subset_2M.sra_2.fastq.gz \
    -o salmon_output/SRR7657883 \
    --writeMappings=salmon_output/SRR7657883/SRR7657883.salmon.sam \
    --gcBias \
    -p 7
```

- `-i` is the path to our transcriptome index, created in the previous exercise.
- `-l` indicates we want Salmon to **A**utomatically determine the strandness of our library.
- `-1` and `-2` are the paths to read 1 and read 2 of our sample (as we have paired-end sequencing in this case).
- `-o` is the path to the output transcript quantification file.
- `--writeMappings` is an optional argument specifying we want Salmon to also produce an alignment file, where our reads are aligned to the reference transcriptome (we will use this later for some QC).
- `--gcBias` is used to indicate we want Salmon to use the statistical model that adjusts for biases due to GC differences across transcripts.
- `-p` indicates we want to use 7 processors for parallel processing (our training computers have 8 CPUs).

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. The BAM file format is a binary (not human readable) file and is considerably smaller than the same data stored in SAM format. We will also sort the alignment entries by location (Contig/Chromosome name and the location on the contig), this further improves the compression of the SAM to BAM. We will use the `samtools sort` function.

The general command is:

```
samtools sort -o my_sample.sorted.bam my_sample.sam
```

Where the `-o` option is used to provide the output file name. Because we use the file extension `.bam` in our output file name, `samtools` will automatically convert the input file to the more compact BAM format. There are many other options, e.g. reads can be sorted by the read name instead of the position or we can specify the number of parallel threads to be used - to find out more use `samtools sort --help`.

Exercise 3

1. Sort and transform your aligned SAM file into a BAM file called `SRR7657883.salmon.sorted.bam`. Use the option `-@ 7` to use 7 cores, this vastly speeds up the compression.
2. Use, for example, `samtools view my_sample.sorted.bam` to check your BAM file.

Answer

The `samtools` command we use is the following:

```
samtools sort \
  -@ 7 \
  -o salmon_output/SRR7657883/SRR7657883.salmon.sorted.bam \
   salmon_output/SRR7657883/SRR7657883.salmon.sam
```

- `-@` we want to use 7 processors for parallel processing (our training computers have 8 CPUs).
- `-o` is the path for our output file. Note that we use the `.bam` extension, which will make `samtools` automatically save the output in BAM format.
- At the end of the command we include the SAM input file.

Once the command finishes running, we should have a BAM file in our output folder. We can view the content of this file using:

```
samtools view salmon_output/SRR7657883/SRR7657883.salmon.sorted.bam | less -S
```

The `less -S` command allows us to view the output in an interactive way, using the ↑ ↓ ← → arrow keys. You can type `Q` to **Q**uit and go back to your terminal.

4 Make transcript to gene table

Salmon quantifies gene expression at the transcript level. When we come to do our differential gene expression analysis in R, we will want to summarise this to the gene level. To do this we need a table that links transcript IDs to gene IDs. We have already created this for you, but, for reference, the code below was used to generate this table from the sequence headers in the transcriptome reference file.

You do not need to run this code, we have already done this for you.

```
echo -e "TxID\tGeneID" > salmon_outputs/tx2gene.tsv
zcat references/Mus_musculus.GRCm38.cdna.all.fa.gz |
  grep "^>" |
  head |
  cut -f 1,4 -d ' ' |
  sed -e 's/^>//' -e 's/gene://' -e 's/\.[0-9]*$//' |
  tr ' ' '\t' \
  >> salmon_outputs/tx2gene.tsv
```

1. `zcat references/Mus_musculus.GRCm38.cdna.all.fa.gz` - read the zipped fasta
2. `grep "^>"` - find the sequence headers, they all start with '>'
3. `cut -f 1,4 -d ' '` - extract the 1st and 4th entries on each line - transcript ID and gene ID

4. `sed -e 's/^>//' -e 's/gene://' -e 's/\.[0-9]*$//'` - remove the ">" from the beginning of the line, the "gene:" from the beginning of the gene ID, and the trailing ".x" number which indicates that version of the gene annotation
5. `tr ' ' '\t'` - replace spaces with tabs so that the table is tab delimited

---

References

Patro, Duggal, R. 2017. "Salmon Provides Fast and Bias-Aware Quantification of Transcript Expression." *Nature Methods* 14: 417–19. https://doi.org/10.1038/nmeth.4197.