Tutorial on next generation sequencing (NGS) data analysis with emphasis on RNA sequencing (RNA-seq) technique

Prepared by:

Kennedy Mwangi …, MSc student UoN

Mtakai Ngara, PhD

April 6, 2022

Scope

- Bioinformatics background and resources
- Next-generation sequencing (NGS) concepts and applications
- Gene expression profiling using RNA-sequencing technique
- Basic command line for unix/linux users
- Conda environment installation and basics
- NGS datasets
- Read trimming
- Quality control (QC) analysis
- Reference mapping
- Mapping quality evaluation
- Gene expression quantification
- Differential expression analysis
- Basic visualization

Read mapping

Here we will use CLI to map our reads from the downloaded FASTQ files to the reference genome.

After studying this section, you should be able to:

1. Explain the process of sequence read mapping.
2. Use bioinformatics tools to map sequencing reads to a reference genome.
3. Filter mapped reads based on quality.

Create a directory for read mapping:

```
$ cd ~/analysis
# Create a mapping result directory
$ mkdir mappings
$ ls -lF
```

Mapping sequence reads to a reference genome

Use the quality trimmed forward and backward DNA sequences and a program called HISAT2 (14) to map the reads towards a reference human transcriptome. HISAT2 is a short read aligner, that can take a reference genome and map single- or paired-end sequence data to it 15). It requires an indexing step in which one supplies the reference genome and HISAT2 will create an index that in the subsequent steps will be used for aligning the reads to the reference genome. While this step can take some time, the good thing is the index can be reused over and over. The general command structure of the HISAT2 tools we are going to use are shown below:

```
# Create the conda environment and Install HISAT2
$ conda create --yes -n mapping samtools hisat2 qualimap r-base
$ conda activate mapping
```

---

1. MultiQC: https://multiqc.info/
2. FastQC: http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
3. HISAT2: http://daehwankimlab.github.io/hisat2/
4. Kim D, et al. Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. Nature Biotechnology. 2019.

```
# Download the human chromosome 2 genome in compressed fasta format
$ cd ~/analysis
$mkdir genomes_and_annotations
$ cd genomes_and_annotations
$ rsync -avzP
rsync://hgdownload.cse.ucsc.edu/goldenPath/hg38/chromosomes/chr2.fa.gz .
$ gunzip -c chr2.fa.gz > hg38_chr2_genome.fa
$ cd ~/analysis
# make the index directory
$ mkdir index
```

 # Build HISAT2 index

$ hisat2-build genomes_and_annotations/hg38_chr2_genome.fa index/hg38_chr2_genome
In case this takes longer, you may use the already built one.
 # Align the samples to reference using HISAT2

Mapping reads in a paired-end manner
Now that we have created our index, it is time to map the trimmed sequencing reads to the reference genome.

$ hisat2 -q -x index/hg38_chr2_genome -1 trimmed/GSM1275862_R1.fastq.gz -2 trimmed/GSM1275862_R2.fastq.gz -S mappings/GSM1275862.sam

To do: Use the correct HISAT2 command structure from above and map the reads of the two evolved line to the reference genome.

The sam mapping file-format
HISAT2, like most mappers, will produce a mapping file in sam-format. Have a look into the sam-file that was created by either program. A quick overview of the sam-format can be found here (16) and even more information can be found here (17). Briefly, first there are a lot of header lines. Then, for each read, that mapped to the reference, there is one line.

The columns of such a line in the mapping file are described in Table 1 below.

5. http://bio-bwa.sourceforge.net/bwa.shtml#4
6. http://samtools.github.io/hts-specs/SAMv1.pdf

Table 1. Sam format line column description

Col	Field	Description
1	QNAME	Query (pair) NAME
2	FLAG	bitwise FLAG
3	RNAME	Reference sequence NAME
4	POS	1-based leftmost POSition/coordinate of clipped sequence
5	MAPQ	MAPping Quality (Phred-scaled)
6	CIAGR	extended CIGAR string
7	MRNM	Mate Reference sequence NaMe ('=' if same as RNAME)
8	MPOS	1-based Mate POSition
9	ISIZE	Inferred insert SIZE
10	SEQ	query SEQuence on the same strand as the reference
11	QUAL	query QUALity (ASCII-33 gives the Phred base quality)
12	OPT	variable OPTional fields in the format TAG:VTYPE:VALUE

One line of a mapped read can be seen here below:

```
SRR1039508.7    133     chr2    106698547       0       *       =       106698547       0       CGTAGATTCGGGCAAGTCCACCACTACTGGCCATCTGATCTATAAATGCGGTGGCATCGACAA HJHIJJJJJJJJJJJJJJJIJJJJJJJJIJJJJJJJJJIIJJJJJJJJJJJEHHFFFFDEDDD YT:Z:UP
```

It defines the read and the position within the reference genome, where the read mapped and a quality of the mapping.

Mapping post-processing using SAMtools.
SAMtools (18) is a collection of tools for manipulating and analyzing SAM and BAM alignment files.

Fix mates and compress
Because aligners can sometimes leave unusual SAM flag (19) information on SAM records, it is helpful when working with many tools to first clean up read pairing information and flags with SAM tools (18).

A very useful tool to explain flags can be found here:
http://broadinstitute.github.io/picard/explain-flags.html

---

7. http://samtools.sourceforge.net/
8. http://bio-bwa.sourceforge.net/bwa.shtml#4

We are going to produce also compressed bam output for efficient storing of and access to the mapped reads. Note, samtools fixmate expects name-sorted input files, which we can achieve with samtools sort -n.

 # Sort the alignment
$ samtools sort -n -O sam mappings/GSM1275862.sam | samtools fixmate -m -O bam - mappings/GSM1275862.fixmate.bam

- -m: Add ms (mate score) tags. These are used by markdup (below) to select the best reads to keep
- -O bam: specifies that we want compressed bam output from fixmate

Attention: The step of sam to bam -file conversion might take a few minutes to finish, depending on how big your mapping file is.

Once we have bam-file, we can also delete the original sam-file as it requires too much space and we can always recreate it from the bam-file.

 # remove the sam file

$ rm mappings/GSM1275862.sam

Sorting

 # Convert to bam file and sort
$ samtools sort -O bam -o mappings/GSM1275862.sorted.bam mappings/GSM1275862.fixmate.bam
 # Once it successfully finished, delete the fixmate file to save space
$ rm mappings/GSM1275862.fixmate.bam

Remove duplicates

In this step we remove duplicate reads. The main purpose of removing duplicates is to mitigate the effects of PCR amplification bias introduced during library construction.

$ samtools markdup -r -S mappings/GSM1275862.sorted.bam mappings/GSM1275862.sorted.dedup.bam
 # If it worked, delete the original file
$ rm mappings/GSM1275862.sorted.bam

Mapping statistics using SAMtools

$ samtools flagstat mappings/GSM1275862.sorted.dedup.bam
 # For indepth statistics run
$ samtools index mappings/GSM1275862.sorted.dedup.bam
$ samtools idxstats mappings/GSM1275862.sorted.dedup.bam

To do: Look at the mapping statistics and understand their meaning. Discuss your results. Explain why we may find mapped reads that have their mate mapped to a different chromosome/contig? Can they be used for something?

For the sorted bam-file we can get read depth for at all positions of the reference genome, e.g. how many reads are overlapping the genomic position.

$ samtools depth mappings/GSM1275862.sorted.dedup.bam | gzip > mappings/GSM1275862.depth.txt.gz

To do: Extract the depth values for chromosome 2 and load the data into R, calculate some statistics of our scaffold.

Now we can quickly use some R or python script to make a coverage plot for chromosome 2.

To do: Look at the created plot. Explain why it makes sense that you find relatively bad coverage at the beginning and the end of the contig.

Statistics with QualiMap

For a more in-depth analysis of the mappings, one can use QualiMap (20).
QualiMap examines sequencing alignment data in SAM/BAM files according to the features of the mapped reads and provides an overall view of the data that helps to the detect biases in the sequencing and/or mapping of the data and eases decision-making for further analysis.

```
# Run QualiMap
$ qualimap bamqc -bam mappings/GSM1275862.sorted.dedup.bam
# Once finished open result page using your web browser.
# You may also install firefox in your conda env and use it to visualize the results
$ firefox mappings/GSM1275862.sorted.dedup_stats/qualimapReport.html
```

To do: Investigate the mapping of the sample. Write down your observations.

---

9. http://qualimap.bioinfo.cipf.es/

Sub-selecting reads
The mapping commands we used above, without additional parameters to sub-select specific alignments (e.g. for Bowtie2 (21) there are options like --no-mixed, which suppresses unpaired alignments for paired reads or --no-discordant, which suppresses discordant alignments for paired reads, etc.), are going to output all reads, including unmapped reads, multi-mapping reads, unpaired reads, discordant read pairs, etc. in one file. We can sub-select from the output reads we want to analyse further using SAMtools.

To do: Explain what concordant and discordant read pairs are?

We can select read pair that have been mapped in a correct manner (same chromosome/contig, correct orientation to each other, distance between reads are sensible).

```
# Selecting concordantly mapped reads
$ samtools view -h -b -f 3 mappings/GSM1275862.sorted.dedup.bam >
mappings/GSM1275862.sorted.dedup.concordant.bam
```

-b: Output will be bam-format
-f 3: Only extract correctly paired reads

-f extracts alignments with the specified SAM flag set

Quality-based sub-selection

In this section we want to sub-select reads based on the quality of the mapping. It seems a reasonable idea to only keep good mapping reads. As the SAM-format contains at column 5 the $MAPQ$ value, which we established earlier is the "MAPping Quality" in Phred-scaled, this seems easily achieved.
The formula to calculate the $MAPQ$ value is:
$MAPQ = -10 * log10(p)$
Where p is the probability that the read is mapped wrongly.

However, while the MAPQ information would be very helpful indeed, the way that various tools implement this value differs. A good overview can be found here (21). It is important to be aware that different tools use this value in different ways and it is good to know the information that is encoded in the value. Once you dig deeper into the mechanics of the $MAPQ$ implementation it becomes clear that this is not an easy topic. If you want to know more about the $MAPQ$ topic, please follow the link above.

---

10. https://sequencing.qcfail.com/articles/mapq-values-are-really-useful-but-their-implementation-is-a-mess/

Unmapped reads

We can use Kraken2 (22) to classify all un- mapped sequence reads and identify the species they are coming from and test for contamination.

```
#Get unmapped reads
$ samtools view -b -f 4 mappings/GSM1275862.sorted.dedup.bam >
mappings/GSM1275862.sorted.unmapped.bam
# Delete the original to save space, however, in reality you might want to save it to
investigate later
$ rm mappings/GSM1275862.sorted.dedup.bam
# count the unmapped reads
$ samtools view -c mappings/GSM1275862.sorted.unmapped.bam
```

- -b: indicates that the output is BAM.
- -f INT: only include reads with this SAM flag (23) set.
- -c: count the reads.

$ samtools fastq -1 mappings/GSM1275862.sorted.unmapped.R1.fastq.gz -2 mappings/GSM1275862.sorted.unmapped.R2.fastq.gz mappings/GSM1275862.sorted.unmapped.bam
 # delete files not needed
$ rm mappings/GSM1275862.sorted.unmapped.bam

Contamination investigation
We want to investigate if there are sequences of other species in our collection of sequenced DNA pieces. This might be a way of quality control, e.g. have the samples been contaminated?

We will use the tool Kraken2 (22) to assign taxonomic classifications to our sequence reads. This tool uses k-mers to assign a taxonomic labels in form of NCBI Taxonomy (24) to the sequence (if possible). The taxonomic label is assigned based on similar k-mer content of the sequence in question to the k-mer content of reference genome sequence. The result is a classification of the sequence in question to the most likely taxonomic label. If the k-mer content is not similar to any genomic sequence in the database used, it will not assign any taxonomic label.

 # Installing kraken
$ conda create --yes -n kraken kraken2
$ conda activate kraken
 # make sure you are in your analysis root folder
$ cd ~/analysis

---

11. https://www.ccb.jhu.edu/software/kraken2/
12. http://bio-bwa.sourceforge.net/bwa.shtml#4
13. https://www.ncbi.nlm.nih.gov/taxonomy

 # create dir
$ mkdir kraken
$ cd kraken

Create or download a Kraken2 (## ref) database that can be used to assign the taxonomic labels to sequences.

$ curl -O ftp://ftp.ccb.jhu.edu/pub/data/kraken2_dbs//minikraken_8GB_202003.tgz
 # alternatively we can use wget
$ wget ftp://ftp.ccb.jhu.edu/pub/data/kraken2_dbs//minikraken_8GB_202003.tgz
 # once the download is finished, we need to extract the archive content:
$ tar -xvzf minikraken_8GB_202003.tgz

Note: The “minikraken2” database was created from bacteria, viral and archaea sequences.

Running Kraken2

Now that we have installed Kraken2 (25) and downloaded and extracted the minikraken2 database, we can attempt to investigate the sequences we got back from the sequencing provider for other species as the one it should contain. We call the Kraken2 tool and specify the database and fasta-file with the sequences it should use. The general command structure looks like this:

```
# Running kraken
$ kraken2 --use-names --threads 4 --db PATH_TO_DB_DIR --report example.report.txt example.fa >example.kraken
```

---

14. https://www.ccb.jhu.edu/software/kraken2/
15. https://www.ccb.jhu.edu/software/kraken2/index.shtml?t=manual

However, we may have fastq-files, so we need to use --fastq-input which tells Kraken2 that it is dealing with fastq-formated files. The --gzip-compressed flag specifies tat te input-files are compressed. In addition, we are dealing with paired-end data, which we can tell Kraken2 with the switch --paired. Here, we are investigating one of the unmapped paired-end read files of the GSM1275862 sample.

```
# Run kraken
$ kraken2 --use-names --threads 4 --db minikraken2_v2_8GB_201904_UPDATE --fastq-input --report GSM1275862 --gzip-compressed --paired
../mappings/GSM1275862.sorted.unmapped.R1.fastq.gz
../mappings/GSM1275862.sorted.unmapped.R2.fastq.gz > GSM1275862.kraken
```

Each sequence classified by Kraken2 results in a single line of output. Output lines contain five tab- delimited fields; from left to right, they are:

1. C/U: one letter code indicating that the sequence was either classified or unclassified.
2. The sequence ID, obtained from the FASTA/FASTQ header.

3. The taxonomy ID Kraken2 used to label the sequence; this is 0 if the sequence is unclassified and otherwise should be the NCBI Taxonomy identifier.
4. The length of the sequence in bp.
5. A space-delimited list indicating the lowest common ancestor (in the taxonomic tree) mapping of each k-mer in the sequence. For example, 562:13 561:4 A:31 0:1 562:3 would indicate that:
    - the first 13 k-mers mapped to taxonomy ID #562
    - the next 4 k-mers mapped to taxonomy ID #561
    - the next 31 k-mers contained an ambiguous nucleotide
    - the next k-mer was not in the database
    - the last 3 k-mers mapped to taxonomy ID #562

Investigate taxa

We can use the webpage NCBI TaxIdentifier (26) to quickly get the names to the taxonomy identifier. However, this is impractical as we are dealing potentially with many sequences. Kraken2 has some scripts that help us understand our results better.
Because we used the Kraken2 switch --report FILE, we have got also a sample-wide report of all taxa found. This is much better to get an overview what was found.

The output of kraken-report is tab-delimited, with one line per taxon. The fields of the output, from left-to-right, are as follows:

1. Percentage of reads covered by the clade rooted at this taxon
2. Number of reads covered by the clade rooted at this taxon
3. Number of reads assigned directly to this taxon
4. A rank code, indicating (U)nclassified, (D)omain, (K)ingdom, (P)hylum, (C)lass, (O)rder, (F)amily, (G)enus, or (S)pecies. All other ranks are simply "-".
5. NCBI Taxonomy ID
6. The indented scientific name

---

16. https://www.ncbi.nlm.nih.gov/Taxonomy/TaxIdentifier/tax_identifier.cgi

Expression quantification and normalization

Here we will use HTSeq (28) to count the number of reads aligned to the human genes.

```
$ cd ~/analysis
$ conda create --yes -n htseq_count htseq samtools
$ conda activate htseq_count
$ mkdir expression
$ samtools index mappings/GSM1275862.sorted.dedup.concordant.bam
$ htseq-count -f bam -s no -t exon -i gene_id --additional-attr=gene_name
mappings/GSM1275862.sorted.dedup.concordant.bam
genomes_and_annotations/hg38_chr2_annotations.gtf >
expression/GSM1275862_gene_counts.txt
```

IGV visualization: https://anaconda.org/bioconda/igv

17. https://htseq.readthedocs.io/en/master/