Computational practical 8: Metagenomics

Module Developers: Dr. Stanford Kwenda and Dr Ewan Harrison

Table Of Contents

Learning outcomes

Introduction

Metagenomics provides a culture-independent approach to investigate whole microbial communities (i.e. metagenomes) from a bulk sample. Metagenomics is often used to study a specific community of microorganisms, such as those residing on human skin, in the soil or in a water sample.

Two major approaches:
- Amplicon (e.g. 16S rRNA gene) sequencing
- Whole genome shotgun sequencing

Key differences between the two approaches

	16S rRNA sequencing	**Shotgun metagenomic sequencing**
Taxonomic resolution	Bacterial genus level (but can resolve down to species level using long reads)	Bacterial species level (can include strains)

Taxonomic coverage	Mainly bacteria and archaea	All microbial taxa including bacteria, viruses and fungi
Host contamination	Low	High
Bioinformatics expertise	Beginner to intermediate	Intermediate to advanced
Functional profiling	*No	Yes

*Predicted functional profiling might be possible

In this tutorial we will be focusing on shotgun metagenomic sequencing

Raw data quality and QC

This initial QC step can be performed using the same approaches introduced in practical 2. However, we will be using different tools to perform the raw read quality control, filtering and visualization. Before we proceed we should note the following:

QC of raw reads is often a standard first step used to assess the quality of reads and to identify potential problems or other quality related issues

1. Sequencing technologies can produce reads with varying quality
2. Some sample-specific issues such as contamination with adapter sequences
3. Base composition biases
4. Low base quality

Important to monitor key metrics at different steps of the data analysis workflow

Since we are performing read filtering and QC of metagenomic data, it might be beneficial to perform read deduplication during the initial QC step. This will be done in a single step as we perform our standard QC step.

We will perform read QC and filtering based on the steps below:

# First activate the appropriate conda environment

```
conda activate readQC

# create output directory for fastp output
clean_reads=/home/manager/course/cp8/clean_reads
mkdir -p $clean_reads

#  Provide path to the raw reads directory
raw_reads=/home/manager/course/cp8/raw_reads
# Execute the for loop to perform QC on all samples in the raw_reads directory

for fq in $(find $raw_reads -name "*R1.fq.gz"); do

    sampleid=$(basename -s "_R1.fq.gz" $fq)

    read1=$(find $raw_reads -name "${sampleid}*R1*f*q.gz")
    read2=$(find $raw_reads -name "${sampleid}*R2*f*q.gz")

    fastp -i "$read1" -I "$read2" \
        -q 20 -l 36 --cut_front -M 10 -W 4 \
        -R "$sampleid" -j $clean_reads/${sampleid}.fastp.json \
        -h $clean_reads/${sampleid}.fastp.html \
        --correction --dedup --overrepresentation_analysis --thread 4 \
        -o $clean_reads/${sampleid}.R1.fq.gz -O $clean_reads/${sampleid}.R2.fq.gz

Done >> $clean_reads/qc_step1.log
```

Key points:

Including a read deduplication step can potentially:

1. Increase the number of metagenome-assembled contigs (might be sample dependent)
2. Improve the length of the metagenome-assembled contigs (might be sample dependent)
3. Improve metagenomic binning yields (i.e. can contribute to the better recovery of MAGs from complex metagenomes)
4. Decrease the maximum memory requirement and time consumption during the computationally intensive meta-assembly step
5. Enhance the coverage abundance profiles of contigs

Read QC visualization

For QC visualization we will use a tool called multiqc.

```
qc_reports=/home/manager/course/cp8/multiqc

multiqc -f --no-data-dir $clean_reads --outdir $qc_reports
```

Group activity 1: Read QC and filtering (10min):

Navigate to the multiqc output folder and open the report in your browser. In groups of 2 or 3, discuss the following:

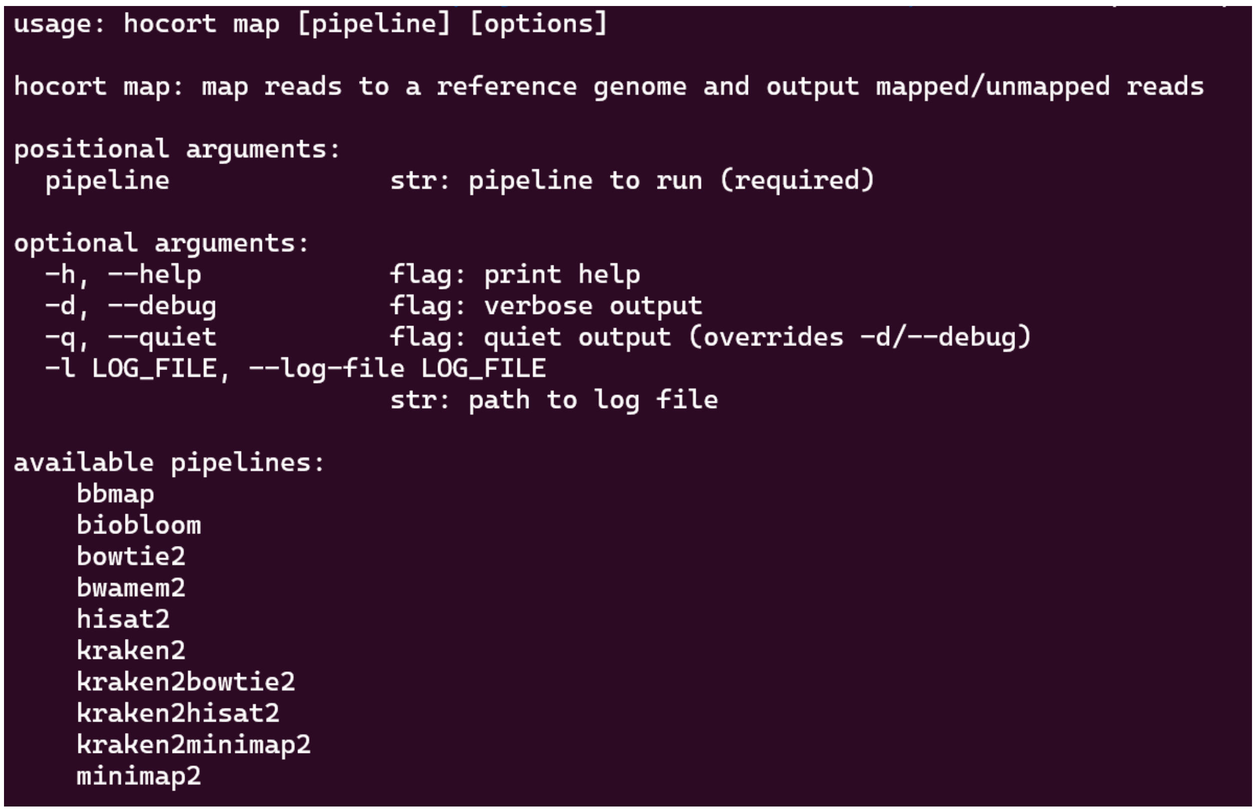
1. Number of reads in each sample?
2. Percentage of reads which passed filters in each sample?
3. What was the duplication rate before filtering?
4. The average quality after filtering?
5. What is the average length of the reads?

Host contamination removal

“A contaminated sequence is one that does not faithfully represent the genetic information from the biological source organism/organelle because it contains one or more sequence segments of foreign origin.” **NCBI VecScreen**

- Shotgun metagenome sequencing data obtained from a host environment will usually be contaminated with sequences from the host organism
- Host sequences should be removed before further analysis:
  - To avoid biases
  - Reduce downstream computational load
  - Data protection or unintended data sharing e.g. in the case of a human host
- Positive vs negative filtering

For the decontamination step, we will use a tool called hocort (**Ho**st **Co**ntamination **R**emoval **T**ool).



```
usage: hocort map [pipeline] [options]

hocort map: map reads to a reference genome and output mapped/unmapped reads

positional arguments:
  pipeline                str: pipeline to run (required)

optional arguments:
  -h, --help              flag: print help
  -d, --debug             flag: verbose output
  -q, --quiet             flag: quiet output (overrides -d/--debug)
  -l LOG_FILE, --log-file LOG_FILE
                          str: path to log file

available pipelines:
    bbmap
    biobloom
    bowtie2
    bwamem2
    hisat2
    kraken2
    kraken2bowtie2
    kraken2hisat2
    kraken2minimap2
    minimap2
```

```
# Activate the hocort environment
conda activate hocort

# Provide path to the bowtie2 index files
bwt=/home/manager/course/cp8/databases/hocort/human
```

If these are not available, or if you are working on a different host, then you will need to first index the host genome sequence, to prepare the bowtie2 index files, you can look at `hocort index --help` on how to perform this step.

```
# Create directory to save decontaminated reads
hocort=/home/manager/course/cp8/hocort
mkdir -p $hocort

for fq in $(find $clean_reads -name "*R1.fq.gz"); do

    sampleid=$(basename -s ".R1.fq.gz" $fq)

    read1=$(find $clean_reads -name "${sampleid}*R1*f*q.gz")
    read2=$(find $clean_reads -name "${sampleid}*R2*f*q.gz")

    hocort map bowtie2 --threads $threads --filter true \
    -x ${bwt}/grch38 -i $read1 $read2 \
    -o $hocort/${sampleid}.R1.fq $hocort/${sampleid}.R2.fq 2> $hocort/${sampleid}.err

    # compress reads
    gzip $hocort/${sampleid}.R1.fq
    gzip $hocort/${sampleid}.R2.fq

done
```

Taxonomic classification

Annotation of reads or contigs with taxonomic information using e.g. blast based methods against reference databases. Quality of taxonomic assignments depends on:

1. Choice of tools
2. Reference database

Prepare kraken2 database

First let's create a directory to store our databases.

```
mkdir -p /home/manager/course/cp8/databases/kraken2
```

Next we will decompress the kraken2 database into the path we created above

```
tar –xvzf /home/manager/k2_standard_16gb_20240112.tar.gz -C
/home/manager/course/cp8/databases/kraken2_8gb/
```

*This step takes a bit of time and should be done the day before (or overnight).

Perform taxonomic classification using kraken2

Now let's activate the environment with the tools that we will need to use for this section.

```
conda activate classify
```

```
# create directory for kraken2 output
mkdir  /home/manager/course/cp8/kraken2
```

```
krak=/home/manager/course/cp8/kraken2

# set threads
threads=4

# set path to kraken2 database and clean_reads directory

db=/home/manager/course/cp8/databases/kraken2
clean_reads=/home/manager/course/cp8/clean_reads
```

To speed things up a bit and avoid repeated ‘copy and paste’ for each sample, we can kick-off the classification step using a (for) loop. This will allow us to run the classification step once on all samples.

```
# execute kraken2

for fq in $(find $hocort -name "*R1.fq.gz")
   do
      sampleid=$(basename -s ".R1.fq.gz" $fq)

      read1=$(find $hocort -name "${sampleid}*R1*f*q.gz")
      read2=$(find $hocort -name "${sampleid}*R2*f*q.gz")


      kraken2 --db "$db" --threads $threads --quick --paired \
            --output $krak/${sampleid}.kraken \
            --report $krak/${sampleid}.kraken.report \
            --memory-mapping $read1 $read2 \
            --gzip-compressed \
            --unclassified-out $krak/${sampleid}#_unclassified.fq >> $krak/krak.log

done
```

Getting relative abundances

For this exercise we will be using a tool called bracken. Bracken computes the genus/species level abundance estimates based on DNA sequences from a metagenomic sample using taxonomic annotations assigned by kraken.

```
brak=/home/manager/course/cp8/bracken
mkdir -p $brak
```

```
for file in $(find $krak -name "*kraken.report"); do sampleid=$(basename -s ".kraken.report"
$file); bracken -d "$db" -i $file -o $brak/${sampleid}.bracken.report -w
${sampleid}.bracken_species.report; done
```

Visualize the taxonomic classification results

For easy visualization/ summarization of the bracken output, we will use 2 approaches:

1. multiqc
2. krona

```
krona=/home/manager/course/cp8/krona
mkdir -p $krona

ktImportTaxonomy -t 5 -m 3 -o $krona/grouped.krona.html $brak

conda deactivate
```

Now let's get a summary of the taxonomic classification output using multiqc

```
# conda activate readQC
multiqc -f --no-data-dir $krak --outdir $qc_reports -n krackenres
```

AMR profiling

We can determine the resistome of each metagenomic sample by either directly mapping/aligning cleaned reads to an AMR database, or using metagenomic assemblies. In this section, we will explore the read-based mapping option using kma and the resfinder database.

kma should already be available in your path, and can verify this by using any of the following command(s):

```
kma -v
which kma
```

You should either get the version of kma or the path to kma executable binary

```
# path to resfinder db
res_db=
kma_out=/home/manager/course/cp8/resistance
```

```
mkdir -p $kma_out


# Now let’s run kma

for fq in $(find $hocort -name "*R1.fq.gz"); do

        sampleid=$(basename -s ".R1.fq.gz" $fq)
        read1=$(find $hocort -name "${sampleid}*R1*f*q.gz")
        read2=$(find $hocort -name "${sampleid}*R2*f*q.gz")

        kma -mem_mode -ef -cge -nf -vcf -t $threads -ipe $read1 $read2 -t_db $res_db/all  -o
        $kma_out/amr -1t1

done
```

Output files:

For downstream analysis, the following output files can be used e.g. in R, as input for differential abundance analysis, generation of graphs and other analyses.

1. amr.mapstat
2. Amr.res
3. amr.vcf.gz