Systematic Microbiome Data Analysis in R

End to End Practical User Guides

Teresia Mrema-Buza, A Microbiome Data Science Enthusiast and Owner of the Complex Da

2022-05-21

Contents

Bioinformatics Analysis of Microbiome Data

Quick Glimpse

Microbiome data analysis is about asking questions to understand the microbial composition in a given sample. The Systematic Microbiome Data Analytics (**SMDA**) book series represent practical guides supporting the microbiome data analytics beyond the traditional analysis. This on-going project supports the transformation of complex microbiome data into optimal biological insights. For simplicity, the eBook is divided in four parts (Part 21 through Part 4). These series provide users with integrated solutions for gaining insight into microbial community biodiversity and investigating their role in disease, health and their impact in the environment.

License

Part I

BIOINFORMATICS ANALYSIS

Chapter 1

Preprocessing 16S rRNA Sequencing Data

1.1 Initial read quality scores

We will use fastqc software to explore the original read qualities before trimming and decontamination.

```
mkdir data/fastqc1
fastqc *.fastq.gz -o data/fastqc1
```

```
mkdir data/multiqc1
multiqc -f --data-dir data/fastqc1 -o data/multiqc1 --export
```

Plots from `multiqc` are exported to `multiqc_plots` folder.

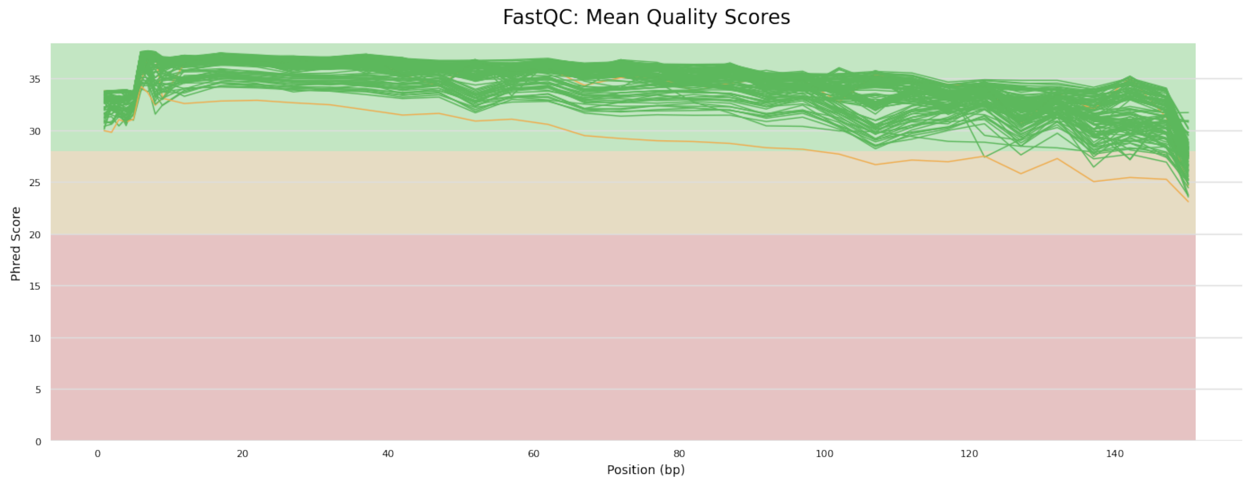


Figure 1.1: Original: Mean quality scores

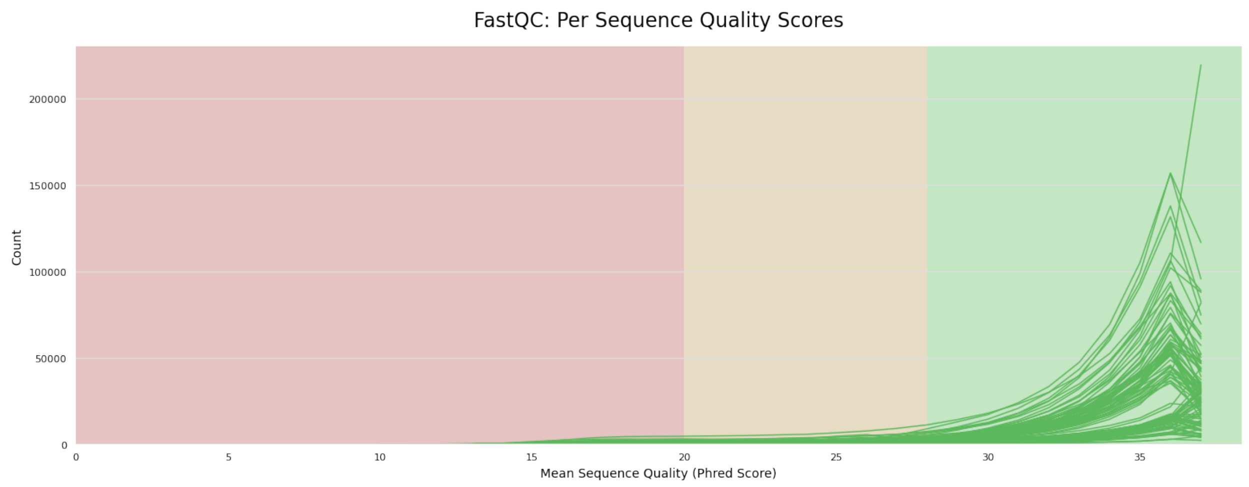


Figure 1.2: Original: Per sequence quality scores

1.2 Read trimming using `bbduk.sh` from bbmap platform

Hint! Make sure that the pattern in the script is in the file name. Some files may contain `R1_001.fastq.gz`. Here we use files downloaded from NCBI-SRA, they do not use R1_001 in the sample name.

```
for i in `ls -1 *_1.fastq.gz | sed 's/_1.fastq.gz//'`
  do
  bbduk.sh -Xmx4g in1=$i\_1.fastq.gz in2=$i\_2.fastq.gz out1=data/trimmed/$i\_1.fastq.gz
  done
```

```
mkdir -p data/stats2
seqkit stat data/trimmed/*.fastq.gz >data/stats2/seqkit_stats.txt

mkdir data/fastqc2
fastqc data/trimmed/*.fastq.gz -o data/fastqc2

mkdir data/multiqc2
multiqc -f --data-dir data/fastqc2 -o data/multiqc2 --export
```

1.3 Read decontamination

- Using `bbduk.sh` from bbmap platform
- This remove some contamination, e.g. PhiX Control reads.

```
for i in `ls -1 *_1.fastq.gz | sed 's/_1.fastq.gz//'`
do
bbduk.sh -Xmx4g in1=data/trimmed/$i\_1.fastq.gz in2=data/trimmed/$i\_2.fastq.gz out1=dat
done
```

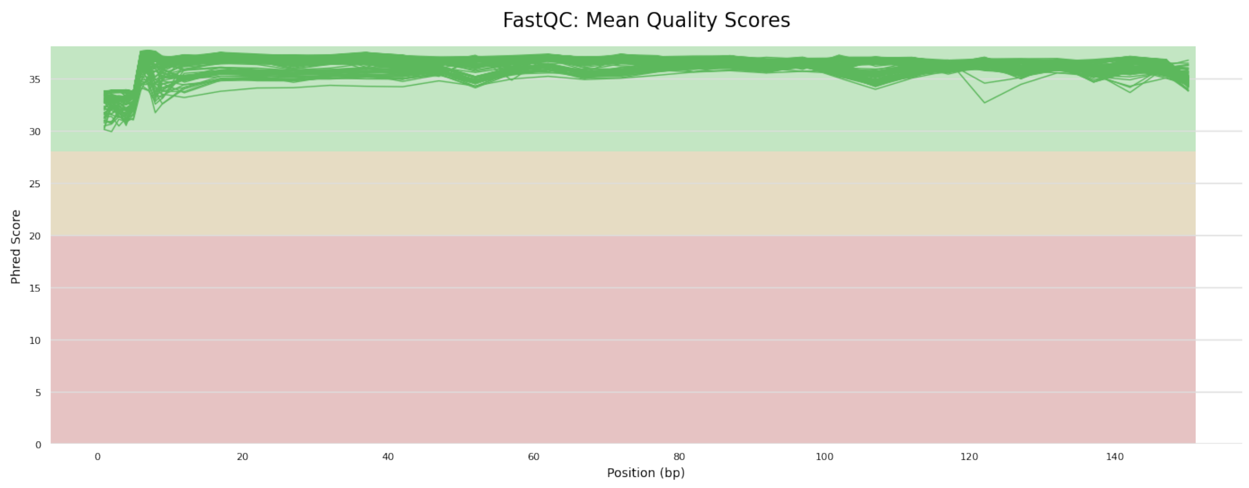


Figure 1.3: Trimmed: Mean quality scores

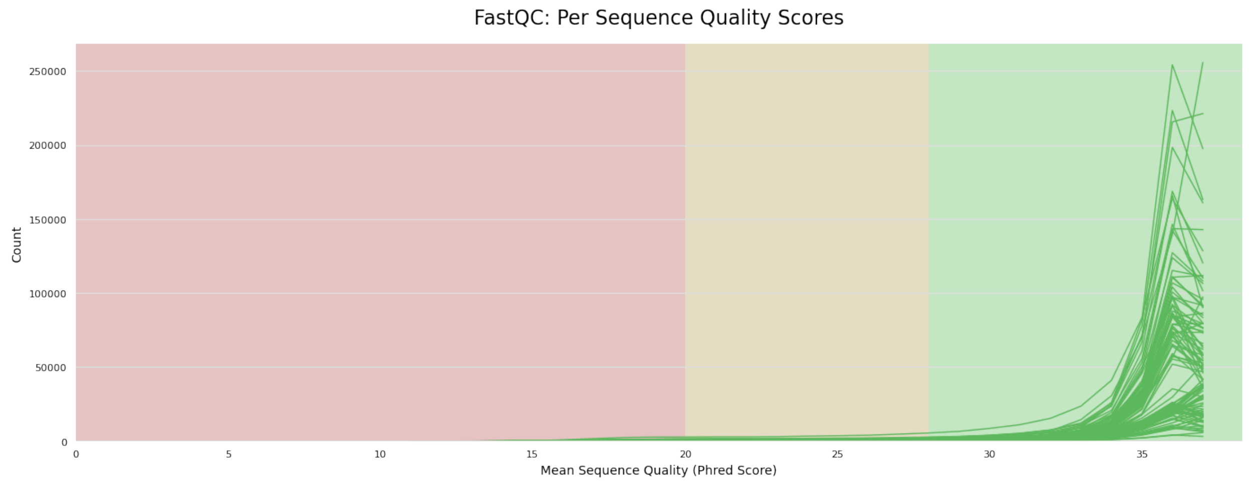


Figure 1.4: Trimmed: Per sequence quality scores

```
mkdir -p data/stats3
seqkit stat data/decontam/*.fastq.gz >data/stats3/seqkit_stats.txt

mkdir data/fastqc3
fastqc data/decontam/*.fastq.gz -o data/fastqc3

mkdir data/multiqc3
multiqc -f --data-dir data/fastqc3 -o data/multiqc3 --export
```

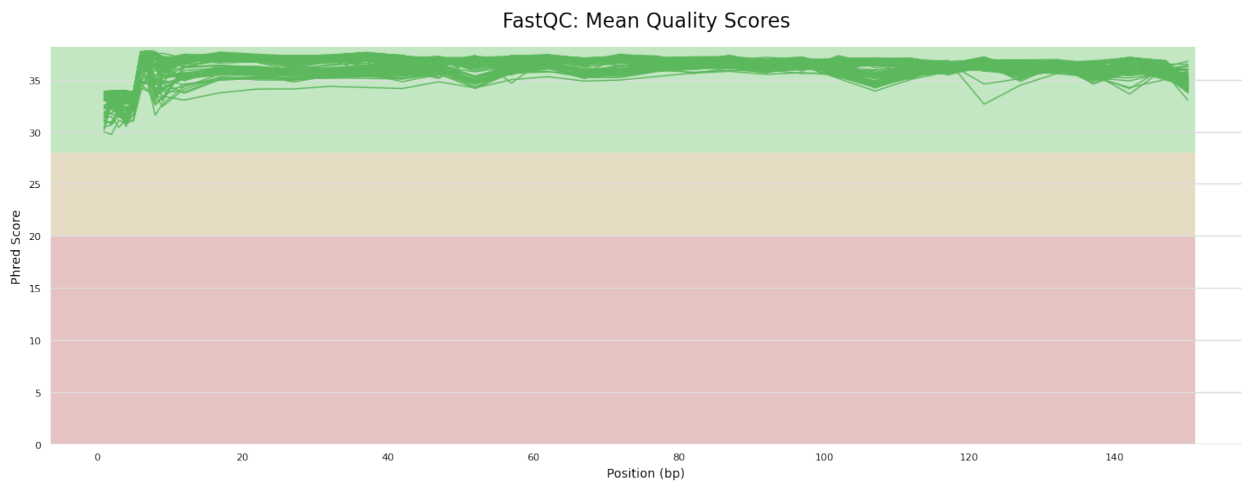


Figure 1.5: Decontaminated: Mean quality scores

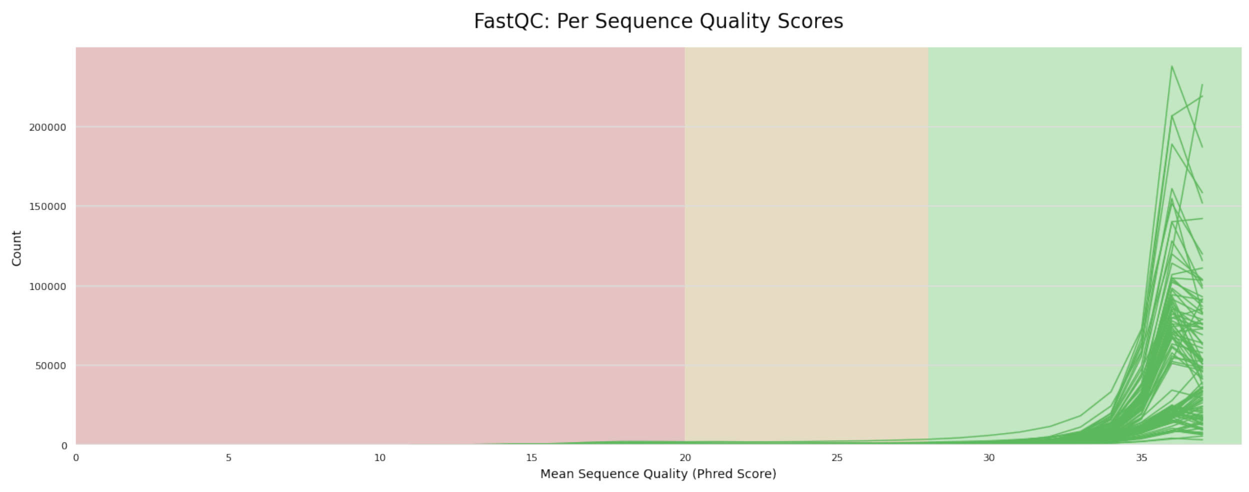


Figure 1.6: Decontaminated: Per sequence quality scores

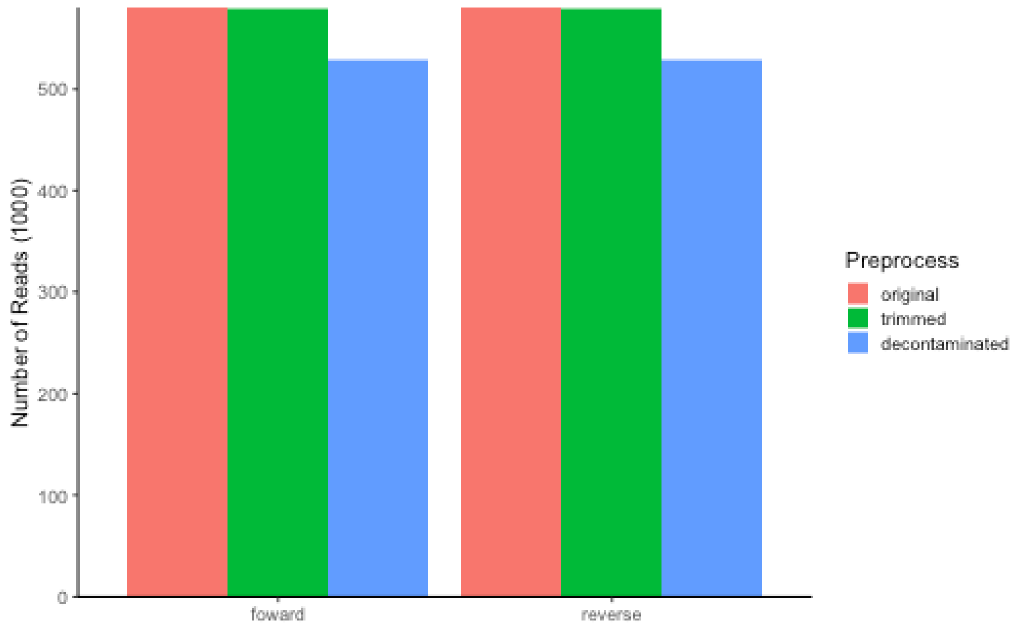
1.4 Compare preprocessed reads

```
library(tidyverse, suppressPackageStartupMessages())
library(ggtext)

stats1 <- read_table("data/stats1/seqkit_stats.txt") %>%
  mutate(file = str_replace_all(file, ".*/", "")) %>%
  select(file, original = num_seqs)
saveRDS(stats1, "RDataRDS/seqkit_stats.rds")

stats2 <- read_table("data/stats2/seqkit_stats.txt") %>%
  mutate(file = str_replace_all(file, ".*/", "")) %>%
  select(file, trimmed = num_seqs)

stats3 <- read_table("data/stats3/seqkit_stats.txt") %>%
  mutate(file = str_replace_all(file, ".*/", "")) %>%
  select(file, decontaminated = num_seqs)

read_status <- inner_join(stats1, stats2, by = "file") %>%
  inner_join(., stats3, by = "file") %>%
  mutate(strand = ifelse(str_detect(file, "_1"), "foward", "reverse"), .before=original)
  pivot_longer(cols = -c(file, strand), names_to = "variable", values_to = "num_seqs")

head(read_status)
```

```
# A tibble: 6 x 4
  file                 strand  variable       num_seqs
  <chr>                <chr>   <chr>             <dbl>
1 SRR7450706_1.fastq.gz foward  original         383666
2 SRR7450706_1.fastq.gz foward  trimmed          383062
3 SRR7450706_1.fastq.gz foward  decontaminated   358493
4 SRR7450706_2.fastq.gz reverse original         383666
5 SRR7450706_2.fastq.gz reverse trimmed          383062
6 SRR7450706_2.fastq.gz reverse decontaminated   358493
```

1.5 What is the distribution of the processed reads

```
read_status %>%
  mutate(variable = factor(variable),
         variable = fct_reorder(variable, num_seqs, .desc=TRUE)) %>%
  ggplot(aes(x = strand, y = num_seqs/1000, fill = variable)) +
  geom_col(position = "dodge") +
  labs(x = NULL, y = "Number of Reads (1000)", fill = "Preprocess") +
  theme_classic() +
  theme(axis.text.x = element_markdown(),
        legend.text = element_text(face = NULL),
        legend.key.size = unit(12, "pt")) + nowhitespace
```



Chapter 2

Preprocessing Metagenomics Sequencing Data

2.1 What are the read statistics

- Using `seqkit stat` function.
- Run the function on command line.

```
mkdir -p data
mkdir -p data/stats1
seqkit stat *.fastq.gz >data/stats1/seqkit_stats.txt
```

2.2 Initial read quality scores

The `kneaddata` from biobakery platform is exclusively used to preprocess the metagenomics data. - Here we set the `kneaddata` to run `fastqc` as the first step using `run-fastqc-start` function. - Then `fastqc` results are then summarized outside `kneaddata` pipeline using `multiqc` software.

```
for i in data/*.fastq.gz
do
time kneaddata --i $i \
-o kneaddata_fastqc1 \
--run-fastqc-start \
--bypass-trim \
--bypass-trf \
--sequencer-source "none" \
-t 4
done


mkdir -p kneaddata_fastqc1/multiqc1
```

```
for i in kneaddata_fastqc1/fastqc
    do
        multiqc -f --data-dir $i -o kneaddata_fastqc1/multiqc1 --export
    done
```

2.3 Read Trimming

- Using trimmomatic tool to trim the reads using the specified options.
- The fastqc function is run after trimming
- Then fastqc results are summarized using multiqc software.

```
for i in data/*.fastq.gz
do
  time kneaddata --i $i \
    -o kneaddata_FastQC2 \
    --trimmomatic /Users/tmbuza/opt/anaconda3/envs/biobakery3/bin/ \
    --trimmomatic-options \
        "ILLUMINACLIP:/Volumes/SeagateTMB/trimmomatic-0.36/adapters/NexteraPE-PE.fa:2:30
        LEADING:3 \
        TRAILING:3 \
        SLIDINGWINDOW:4:20 \
        MINLEN:60" \
    --sequencer-source "NexteraPE" \
    --run-fastqc-end \
    --bypass-trf \
    -t 4
done


seqkit stat kneaddata_fastqc2/*trimmed.fastq >QC/QC2_trimmed_stats.txt
mkdir -p kneaddata_fastqc2/multiqc12
for i in kneaddata_fastqc2/fastqc
    do
        multiqc -f --data-dir $i -o kneaddata_fastqc2/multiqc12 --export
    done
```

2.4 Read Decontamination

2.4.1 Option 1: Using Bowtie2 database

- Trimming is done using trimmomatic
- Then decontamination is done using the bowtie2 database.

- Then after decontamination fastqc is implemented to assess the read quality scores of the remaining reads.
- Then fastqc results are summarized using multiqc software.

```
for i in data/*.fastq.gz
do
time kneaddata --i $i \
  -o kneaddata_fastqc3 \
      --reference-db /Volumes/SeagateTMB/kneaddata_database/ \
      --trimmomatic /Users/tmbuza/opt/anaconda3/envs/biobakery3/bin/ \
      --trimmomatic-options \
          "ILLUMINACLIP:/Volumes/SeagateTMB/trimmomatic-0.36/adapters/NexteraPE-PE.fa:2:
          LEADING:3 \
          TRAILING:3 \
          SLIDINGWINDOW:4:20 \
          MINLEN:60" \
      --sequencer-source "NexteraPE" \
      --run-trf \
      --run-fastqc-end \
      -t 4
done

seqkit stat kneaddata_fastqc3/*kneaddata.fastq >QC/QC3_bowtie2decont_stats.txt
mkdir -p kneaddata_fastqc3/multiqc13
for i in kneaddata_fastqc3/fastqc
    do
        multiqc -f --data-dir $i -o kneaddata_fastqc3/multiqc13 --export
    done
```

2.4.2 Option 2: Using BMTagger database

- Trimming is done using trimmomatic
- Then decontamination is done using the BMTagger database.
- Then after decontamination fastqc is implemented to assess the read quality scores of the remaining reads.
- Then fastqc results are summarized using multiqc software.

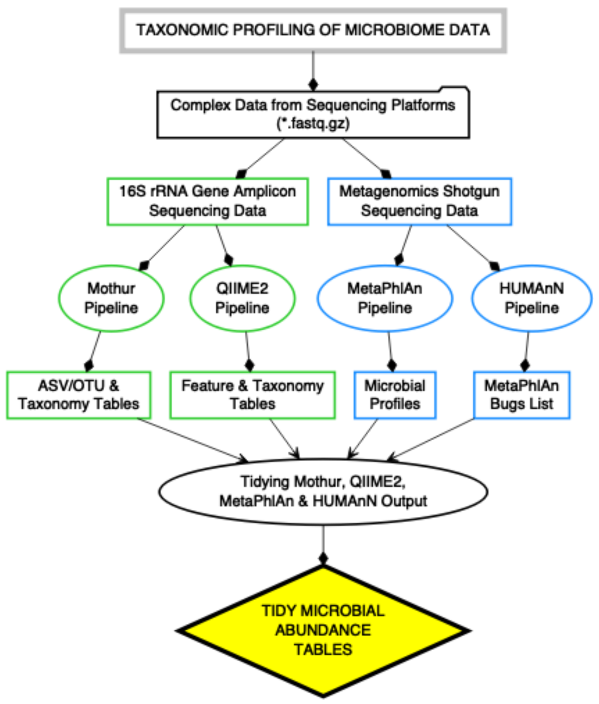
```
for i in data/*.fastq.gz
do
time kneaddata --i $i \
  -o kneaddata_fastqc4 \
      --reference-db /Volumes/SeagateTMB/Human_Assembly19_BMTagger_DB/ \
      --trimmomatic /Users/tmbuza/opt/anaconda3/envs/biobakery3/bin/ \
      --trimmomatic-options \
          "ILLUMINACLIP:/Volumes/SeagateTMB/trimmomatic-0.36/adapters/NexteraPE-PE.fa:2:
```

```
          LEADING:3 \
          TRAILING:3 \
          SLIDINGWINDOW:4:20 \
          MINLEN:60" \
      --sequencer-source "NexteraPE" \
      --run-trf \
      --run-bmtagger \
      --run-fastqc-end \
      -t 4
done

seqkit stat kneaddata_fastqc4/*kneaddata.fastq >QC/QC4_bmtaggerdecont_stats.txt
mkdir -p kneaddata_fastqc4/multiqc14
for i in kneaddata_fastqc4/fastqc
    do
        multiqc -f --data-dir $i -o kneaddata_fastqc4/multiqc14 --export
    done
```

Chapter 3

Taxonomic Profiling of Microbiome Data

3.1 Generalized bioinformatics workflow


TAXONOMIC PROFILING OF MICROBIOME DATA
Complex Data from Sequencing Platforms (*.fastq.gz)
16S rRNA Gene Amplicon Sequencing Data
Metagenomics Shotgun Sequencing Data
Mothur Pipeline
QIIME2 Pipeline
MetaPhlAn Pipeline
HUMAnN Pipeline
ASV/OTU & Taxonomy Tables
Feature & Taxonomy Tables
Microbial Profiles
MetaPhlAn Bugs List
Tidying Mothur, QIIME2, MetaPhlAn & HUMAnN Output
TIDY MICROBIAL ABUNDANCE TABLES

3.2 Format of input data

The default format used in this practical user guide is compressed fastq, `*.fastq.gz`.

3.2.1 Compressing fastq files using `gzip`

Run the command below only if the sequencing data files are not compressed!

```
gzip *.fastq
```

Chapter 4

Mcrobial profiling using mothur pipeline.

4.1 Download a Mothur trained classifer

- For demo purposes we will use Silva seed due to its smaller size.
- Other classifiers can be found here.

```
wget https://mothur.s3.us-east-2.amazonaws.com/wiki/silva.seed_v138_1.tgz
tar xvzf silva.seed_v138_1.tgz
rm *.tgz
```

4.2 Mothur basic wrapper

- Below is a minimal wrapper for sequence processing, classification and taxonomic assignment using `mothur` pipeline.
- For more information refer to `mothur` MiSeq SOP.
- We can save this wrapper as `mothur_code.batch` and place it in a folder named code.
- Then we will run the file on command line as shown below. See the commented lines.
- We assume that the input data is gunzipped i.e. **.fastq.gz**. If not run `gzip *.fastq` to compress the files.

1. Note that the `shebang line` remind us that we are running the analysis on `mothur` command line NOT `bash`. This takes the advantage of using the **current** option to refer to the last saved file.
2. If not generating the mapping file automatically you should make sure that it confer to accepted format as in `mothur_mapping_files.tsv` described in the previous section. In this demo we use `bush.files`. This means that all output files will be prefixed with the term bush to reflect the name of the project. Try to avoid loner names.

```
#!/mothur

# Usage: mothur code/mothur_code.batch # if mothur is in the path
# Usage: ./mothur code/mothur_code.batch # on mac/linux if the executable `mothur` is
# Usage: ./mothur.exe code/mothur_code.batch # on Windows machins if the executable `m

set.current(processors=1)
# set.logfile(name=make_files.logfile)
# make.file(inputdir=., type=fastq.gz, prefix=test)

set.logfile(name=seq_assembly.logfile)
make.contigs(file=bush.files, outputdir=./test, maxambig=0, maxlength=275);
unique.seqs(count=current);
summary.seqs(fasta=current, count=current)

set.logfile(name=seq_align_preclustering.logfile)
align.seqs(fasta=current, reference=silva.seed_v138_1.align);
screen.seqs(fasta=current, count=current, start=13862, end=23444, maxhomop=8);
filter.seqs(fasta=current, vertical=T, trump=.);
pre.cluster(fasta=current, count=current, diffs=2);
unique.seqs(fasta=current, count=current);

set.logfile(name=chimera_removal.logfile)
chimera.vsearch(fasta=current, count=current, dereplicate=t);

set.logfile(name=silva_seed_classification.logfile)
classify.seqs(fasta=current, count=current, reference=silva.seed_v138_1.align, taxonomy=
remove.lineage(fasta=current, count=current, taxonomy=current, taxon=Chloroplast-Mitocho

set.logfile(name=final_files.logfile)
rename.file(fasta=current, count=current, taxonomy=current, prefix=final)

set.logfile(name=otu_clustering.logfile)
dist.seqs(fasta=current, cutoff=0.03);
cluster(column=current, count=current, cutoff=0.03);
make.shared(list=current, count=current, label=0.03);
classify.otu(list=current, count=current, taxonomy=current, label=0.03);
make.lefse(shared=current, constaxonomy=current);
make.biom(shared=current, constaxonomy=current);

set.logfile(name=phylotype_clustering.logfile)
phylotype(taxonomy=current);
make.shared(list=current, count=current, label=1);
classify.otu(list=current, count=current, taxonomy=current, label=1);
make.lefse(shared=current, constaxonomy=current);
make.biom(shared=current, constaxonomy=current);
```

```
set.logfile(name=asv_clustering.logfile)
make.shared(count=current)
classify.otu(list=current, count=current, taxonomy=current, label=ASV)
make.lefse(shared=current, constaxonomy=current)
make.biom(shared=current, constaxonomy=current)

set.logfile(name=phylogenetic_clustering.logfile)
dist.seqs(fasta=current, output=lt)
clearcut(phylip=current)
```

Chapter 5

Mcrobial profiling using `qiime2` pipeline

5.0.1 Download a QIIME2 trained classifer

- You can use Naive Bayes (nb) classifiers trained on GreenGenes or SILVA database with 99% OTUs. You can train your own classifier using the q2-feature-classifier.

Here we will download the smallest classifier, the naive classifier trained on Greengenes 13_8 99% OTUs from 515F/806R region of sequences

```
wget \
  -O "gg-13-8-99-515-806-nb-classifier.qza" \
  "https://data.qiime2.org/2022.2/common/gg-13-8-99-515-806-nb-classifier.qza"
```

5.0.2 QIIME2 installation

- We assumes that you have already installed QIIME 2. If not please do so using the instructions described here!.

A simple demo for installing qiime2 on Mac OS

```
wget https://data.qiime2.org/distro/core/qiime2-2022.2-py38-osx-conda.yml
conda env create -n qiime2-2022.2 --file qiime2-2022.2-py38-osx-conda.yml
rm qiime2-2022.2-py39-osx-conda.yml

# Activate qiime environment
conda activate qiime2-2022.2

# Confirm installation
qiime info
```

5.0.3 Running QIIME2 commands

- We prefer to use the command line interface g2cli.

- Activate the qiime2 environment: We are using qiime2-2022.2.

```
conda activate qiime2-2022.2
```

5.0.4 Import paired-end fastq files

```
qiime tools import \
  --type 'SampleData[PairedEndSequencesWithQuality]' \
  --input-path pe-33-manifest.tsv \
  --output-path qiime2_bushmeat/paired-end-demux.qza \
  --input-format PairedEndFastqManifestPhred33V2
```

- Below is a minimal wrapper for
  - exploring the metadata,
  - importing the data,
  - sequence quality control,
  - sequence classification,
  - taxonomic assignment,
  - generating feature table.

For more information refer to `qiime2` tutorials.

```
#!/qiime

# Usage: qiime2 code/qiime2_code.bash
```

5.1 QIIME2 output tidying

```
taxlevels <- c("kingdom", "phylum", "class", "order", "family", "genus", "species")

read_delim("/Volumes/RedSeagate/MOTHUR_BUSHMEAT/SRR7450/qiime2_bushmeat/q2-transformed-t
  rename_all(tolower) %>%
  rename(otu = "feature id") %>%
  mutate(confidence = round(confidence, digits = 2)) %>%
  # filter(confidence == 1.00) %>%
  mutate(taxon = str_replace_all(taxon, "; s__$", ""),
         taxon = str_replace_all(taxon, "; g__$", ""),
         taxon = str_replace_all(taxon, "; f__$", ""),
         taxon = str_replace_all(taxon, "; o__$", ""),
         taxon = str_replace_all(taxon, "; c__$", ""),
         taxon = str_replace_all(taxon, "; p__$", ""),
         taxon = str_replace_all(taxon, "; k__$", ""),
         ) %>%
  separate(col = taxon, into = all_of(taxlevels), sep = "; ") %>%
  saveRDS("RDataRDS/q2_taxonomy.rds")
```

```
read_delim("/Volumes/RedSeagate/MOTHUR_BUSHMEAT/SRR7450/qiime2_bushmeat/q2-transformed-t
  rename(otu = "#OTU ID") %>%
  saveRDS("RDataRDS/q2_otutable.rds")
```

```
readRDS("RDataRDS/q2_taxonomy.rds") %>%
  inner_join(., readRDS("RDataRDS/q2_otutable.rds"), by="otu")
```

```
# A tibble: 494 x 19
   otu     kingdom phylum class order family genus species confidence SRR7450728
   <chr>   <chr>   <chr>  <chr> <chr> <chr>  <chr> <chr>        <dbl>      <dbl>
 1 04edaa~ k__Bac~ p__Fi~ c__C~ o__C~ f__Pe~ g__[~ s__sor~       0.96        267
 2 e27b9e~ k__Bac~ p__Fi~ c__B~ o__B~ f__St~ g__M~ s__cas~       0.91      15210
 3 d46e22~ k__Bac~ p__Pr~ c__G~ o__E~ f__En~ <NA>  <NA>          1          2374
 4 41db43~ k__Bac~ <NA>   <NA>  <NA>  <NA>   <NA>  <NA>          0.93        409
 5 9580aa~ k__Bac~ p__Fi~ c__C~ o__C~ f__Cl~ g__C~ s__per~       1           146
 6 6f70e5~ k__Bac~ p__Fi~ c__C~ o__C~ f__Cl~ g__C~ <NA>          0.8           0
 7 547357~ k__Bac~ <NA>   <NA>  <NA>  <NA>   <NA>  <NA>          0.97       7092
 8 bc3621~ k__Bac~ p__Pr~ c__G~ o__P~ f__Mo~ g__A~ <NA>          1           357
 9 9209e4~ k__Bac~ p__Fi~ c__C~ o__C~ f__Cl~ g__C~ <NA>          0.98        689
10 5970d8~ k__Bac~ p__Cy~ c__C~ o__C~ <NA>   <NA>  <NA>          1           111
# ... with 484 more rows, and 9 more variables: SRR7450732 <dbl>,
#   SRR7450746 <dbl>, SRR7450747 <dbl>, SRR7450752 <dbl>, SRR7450753 <dbl>,
#   SRR7450755 <dbl>, SRR7450757 <dbl>, SRR7450758 <dbl>, SRR7450759 <dbl>
```

Chapter 6

Taxonomic Profiling of Metagenomics Sequencing Data

6.1 Using MetaPhlAn pipeline.

- We loop through the fastq files using the following parameters.
- You can optionally customize these parameters to suit your needs.
- Note that most parameters are default setting except the *-t rel_ab_w_read_stats* which will add the read count to the output.

```
for i in data/*.fastq.gz
    do
        metaphlan $i \
        --input_type fastq \
        --force \
        --bowtie2db /Volumes/SeagateTMB/metaphlan_databases/ \
        --nproc 4 \
        --stat_q 0.02 \
        --min_mapq_val 5 \
        --output_file ${i%}_metaphlan3_profile.txt \
        --bowtie2out ${i%}_metaphlan3_bowtie2out.txt \
        --biom ${i%}_metaphlan3_abundance.biom \
    -t rel_ab_w_read_stats
    done
```

6.2 Output of taxonomic profiling

- OTU table in mothur and QIIME2
- MetaPhlAn pipeline output taxonomic profiles for each input fastq file.
- You can further use some integrated functions to manage the output. For example, the com-

mand below merge relative abundance into a single file similar to a transposed OTU table from Mothur and QIIME platforms.