FastQC – Hands-on

Klebsiella dataset:

```
wget https://zenodo.org/api/records/6323000/files-archive
```

make a directory - **mkdir** kleb_data

move a directory - **mv** data to folder kleb_data

Ref genome – NC_012731.1 (send via slack or gdrive)

Install fastqc through conda/mamba:

```
conda install bioconda::fastqc
conda install bioconda/label/broken::fastqc
conda install bioconda/label/cf201901::fastqc
```

Install fastqc through source:

```
wget https://github.com/s-
andrews/FastQC/archive/refs/tags/v0.12.1.zip
```

check installation:

```
fastqc -v / --version
```

Usage:

Different ways:

1. GUI (graphical interface, load fastQ files)
2. CLI (non graphical interface) – we will do this one (:
3. Galaxy – some extra reading

Help:

```
fastqc -help
```

Run FastQC on GUI:

1. On the CLI type **`fastqc`**

Run FastQC on CLI through conda:

1. **`conda create env fastqc fastqc`** (you can use preferred env i.e. mamba)
2. **`conda activate fastqc`**
3. **`fastqc -o [output - specify path] -f [fastq|bam|sam – if necessary] [path to dataset] *.fastq.gz`**
4. open .html link to view

side note: only follow step 3 & 4 if you created FastQC through the CLI

MultiQC

Install multiQC through source:

```
pip install multiqc
```

Usage:

When in an existing folder and you want to apply on all files simply run (make sure you are in the working directory)

```
multiqc .
```

Or navigate within directory:

```
multiqc [path to fastqc files] data/*_fastqc.zip
```

Other parameters / options:

`-o/--outdir` (desired folder name)

`-p/--export` (export .pdf .jpg .png files)

Adapter / trimming: Trimmomatic

```
Conda create env trimmomatic
```

Side note: you could just type trimmomatic twice

1. *It would create env name*
2. *It would install packages and software of tool*

Install trimmomatic through conda/mamba:

```
conda install bioconda::trimmomatic
conda install bioconda/label/broken::trimmomatic
conda install bioconda/label/cf201901::trimmomatic
```

Usage:

```
trimmomatic PE -phred33 [path_to_file input_forward.fastq]
[path_to_file input_reverse.fastq] output_forward_paired.fastq
output_forward_unpaired.fastq output_reverse_paired.fastq
output_reverse_unpaired.fastq ILLUMINACLIP:[adapter_TruSeq3-
PE.fa]:2:30:10 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:36
```

SIDE NOTE:

- `PE`: Indicates paired-end mode.
- `-phred33`: Specifies the quality encoding format.
- `ILLUMINACLIP`: Removes adapter sequences.
- `LEADING`: Removes low-quality bases from the beginning.
- `TRAILING`: Removes low-quality bases from the end.
- `SLIDINGWINDOW`: Performs sliding window trimming.
- `MINLEN`: Discards reads shorter than the specified length.

Adapter / trimming: Fastp

```
conda create env fastp
```

Side note you could just type fastp twice

1. *It would create env name*
2. *It would install packages and software of tool*

Install fastp through conda/mamba:

```
conda install -c bioconda fastp
```

Usage:

```
fastp -i in.R1.fq.gz -I in.R2.fq.gz -o out.R1.fq.gz -O
out.R2.fq.gz
```

side note:

- **-5, --cut_front** move a sliding window from front (5') to tail, drop the bases in the window if its mean quality is below cut_mean_quality, stop otherwise. Default is disabled. The leading N bases are also trimmed.
  Use **cut_front_window_size** to set the widnow size,
  and **cut_front_mean_quality** to set the mean quality threshold. If the window size is 1, this is similar as the Trimmomatic **LEADING** method.
- **-3, --cut_tail** move a sliding window from tail (3') to front, drop the bases in the window if its mean quality is below **cut_mean_quality,** stop otherwise. Default is disabled. The trailing N bases are also trimmed.
  Use **cut_tail_window_size** to set the widnow size,
  and **cut_tail_mean_quality** to set the mean quality threshold. If the window size is 1, this is similar as the Trimmomatic **TRAILING** method.
- **-r, --cut_right** move a sliding window from front to tail, if meet one window with mean quality < threshold, drop the bases in the window and the right part, and then stop. Use **cut_right_window_size** to set the widnow size,
  and **cut_right_mean_quality** to set the mean quality threshold. This is similar as the Trimmomatic **SLIDINGWINDOW** method.

NB!

If **--cut_right** is enabled, then there is no need to enable **--cut_tail**, since the former is more aggressive. If **--cut_right** is enabled together with **--cut_front, --cut_front** will be performed first before **--cut_right** to avoid dropping whole reads due to the low quality starting bases.

Please be noted that **--cut_front** will interfere deduplication for both PE/SE data, and **--cut_tail** will interfere deduplication for SE data, since the deduplication algorithms rely on the exact matchment of coordination regions of the grouped reads/pairs.

If you don't set window size and mean quality threshold for these function respectively, fastp will use the values from **`-W, --cut_window_size`** and **`-M, --cut_mean_quality`**

Run FastQC again and compare the difference in quality of data

Now that we ran each tool individually lets create a bash script to run all the datasets together (:

1. create a plain text file using a text editor on the CLI – use nano name_file.sh
2. Specifying the interpreter – which is bash therefore we use "shebang" - **`#!/bin/bash`**
3. Implement commands – for the following tools we will create something called a ''loop'', this would ideally execute all genomes simultaneously through a single script.

   For the following script we will run Fastp:

   Name file: **nano kleb_fastp.sh**

```
#!/bin/bash


##these are comments to help you navigate with your script
writing
# Loop through all pairs of FASTQ files
for file1 in
/home/user/klebsiella_workshop_2024/klebsiella_data/kleb_fastq
_isolates/*_1.fastq.gz;
do
    file2=$(echo ${file1} | sed 's/_1/_2/')
    # remove double.gz filename
    base_filename1="${file1%.fastq.gz}"
    base_filename2="${file2%.fastq.gz}"
    # Run fastp
    ./fastp -i "$file1" -I "$file2" -o
"${base_filename1}_trimmed_R1.fastq.gz" -O
"${base_filename2}_trimmed_R2.fastq.gz" --cut_front --
cut_right
done
```

We now have a good quality dataset

Alignment / mapping: Snippy

```
conda create env snippy
```

Install snippy through conda/mamba:

```
conda install -c conda-forge -c bioconda -c defaults snippy
```

Usage: snippy on 1 isolate

```
snippy --cpus 8 --outdir mysnps --ref klebsiella.gbk --R1
SRR_R1.fastq.gz --R2 SRR_R2.fastq.gz
```

The above is a VCF output for 1 file. How would we execute this on a dataset?

Usage: snippy on multiple isolates (same ref)

```
snippy-multi
```

First create a tab delimited file showing the following:

```
# input_kleb.txt = ID R1 [R2]
Isolate1  /path/to/R1.fq.gz   /path/to/R2.fq.gz
Isolate1b /path/to/R1.fastq.gz     /path/to/R2.fastq.gz
Isolate1c /path/to/R1.fa      /path/to/R2.fa
```

Then run snippy-multi that automates a bash script "runme.sh"

```
snippy-multi input_kleb.txt --ref
/home/user/klebsiella_workshop_2024/klebsiella_data/ncbi_datas
et_ref/NC_012731.1.fasta > runme.sh
```

check that all the components are in your runme.sh file

cat runme.sh

nano runme.sh

now run the bash script

```
bash runme.sh
```

How to view contents of your VCF file:

```
cat snps.vcf
```

Side note: you can use head and less

Vcf outputs can be viewed visually through tools such as IVG

Snippy also provides a **`snippy-core`** output that allows you to build trees