# Trimming reads before using FIGARO in case of inconsistent read length (written by: Timothy Lim, version 15/03/2024)

Sometimes sequencing data may have inconsistent length. In this case trimming of sequencing data to constant length is required. We will be using *fastp* module in this case (see <https://github.com/OpenGene/fastp#input-and-output> for more information) prior to the use of FIGARO.

#launch fastp module

module load fastp/0.20.0 (note that there will not be any response after successful loading)

#Make a copy of the raw data reads into a new folder (e.g., Raw\_Data\_backup) in order to trim for FIGARO input. Create a bash script (e.g., FASTP.sh) and place it within the Raw\_Data\_backup folder (you can also put file path/directory in front of the file name so that you do not have to copy paste the .sh file into the folder you’re working on), with the following script (see <https://github.com/OpenGene/fastp/issues/106> for more information):

#!/bin/bash

#Before using, please move this bash script to the folder which you want to initiate fastp trimming on (make sure you have backup)

#Change directory, and then initiate this bash script using "source FAST.sh"

for f1 in \*\_R1\_001.fastq.gz

do

f2=${f1%%\_R1\_001.fastq.gz}"\_R2\_001.fastq.gz"

echo "f2 is $f2"

fastp -i $f1 -I $f2 -o "trimmed-$f1" -O "trimmed-$f2" -A -Q -l 300 -b 300 -B 300

done

echo "Done."

#initiate fastp trimming by first changing directory to the Raw\_Data\_Backup, then initiate the bash script

cd ~data/Raw\_Data\_Backup

source FASTP.sh

#After completing fastp trimming, move the trimmed files (with prefix “trimmed”) into a new folder (e.g., Raw\_Data\_Trimmed). Create a bash script (e.g., RemoveTrimmed.sh) and place it within the Raw\_Data\_Trimmed folder (you can also put file path/directory in front of the file name so that you do not have to copy paste the .sh file into the folder you’re working on), with the following script to remove the prefix “trimmed-”

#!/bin/bash

for i in \*001.fastq.gz

do

newName=$(echo "$i" | cut -c8-)

newName=${i:8}

mv "$i" "$newName"

done

echo "Done."

#Initiate RemoveTrimmed.sh

cd ~data/Raw\_Data\_Trimmed

source RemoveTrimmed.sh

#If you are planning to use the trimmed reads for demultiplex in QIIME2, make sure to remove the RemoveTrimmed.sh file (if you put the file in the folder you are working on) from the directory before proceeding.

#These files are now ready to be used by FIGARO.

#If FIGARO says DIVISIONZEROERROR, then there could be some reads that are too short to be trimmed. In WINSCP, sort the files according to size, and remove the ones with size 1 kb.

#If FIGARO says something like “Could not connect to display 172.22.48.1:0”, resolve by running *export QT\_QPA\_PLATFORM='offscreen'*

# Miniconda installation for FIGARO

The purpose of FIGARO is to have an automated or more objective way of selecting the truncation and trimming lengths for DADA2. See <https://www.biorxiv.org/content/10.1101/610394v1> for more details.

**Update on this v1 setup: Decided not to install miniconda using m3’s website, but instead download from the official miniconda website.**

#Install miniconda by first download the linux version from this installation link: <https://docs.conda.io/projects/miniconda/en/latest/miniconda-install.html> (I downloaded Miniconda3 Linux 64-bit). Then, move the downloaded *sh* file into WinSCP. I then executed the following command:

sha256sum ~data/Miniconda3-latest-Linux-x86\_64.sh

bash ~data/Miniconda3-latest-Linux-x86\_64.sh (during installation I was asked to whether initiate conda automatically, I answered “no”)

source ~data/miniconda3/bin/activate

#Test by create and activate my python conda environment using the following command prompts (environments can be found in ~data/miniconda3/envs):

source ~data/miniconda3/bin/activate

conda create --name mypythonenv python=3.6

conda activate mypythonenv

conda deactivate (you will first be navigated back to “base”, use conda deactivate again to completely unload conda)

#Create FIGARO environment

wget http://john-quensen.com/wp-content/uploads/2020/03/figaro.yml

conda env create -n figaro -f figaro.yml

*conda activate figaro* (to activate the figaro environment)

#Install FIGARO (as per <https://john-quensen.com/tutorials/figaro/>)

wget https://github.com/Zymo-Research/figaro/archive/master.zip

unzip master.zip

rm master.zip (remove zip file)

cd figaro-master/figaro (change directory)

chmod 755 \*.py (change file permission within figaro-master/figaro)

#Run FIGARO

* First, change directory to where you installed FIGARO

~data/miniconda3/envs/figaro/figaro-master/figaro

* Then, run FIGARO, see <https://john-quensen.com/tutorials/figaro/> or <https://github.com/Zymo-Research/figaro/> for more information. Remember to leave at least 15 to 20 base pairs (BP) minimum overlap, and the amplicon length (a) does not include primers and minimum overlap. Basically if I were to specify primer lengths of 20 for both forward and reverse, and minimum overlap length of 20, I would be expecting total truncation length (forward + reverse) of 550.

python figaro.py -i ~data/Raw\_Data\_FIGARO/Raw\_Data\_Trimmed -o ~data/FIGARO\_output -f 20 -r 20 -a 490 -F illumina -m 20

#Output

In the output folder, three files were received:

1. trimParameters.json (recommended truncation parameters)
2. forwardExpectedError.png
3. reverseExpectedError.png

To view the trimParameters.json, initiate the following command:

cd ~data/FIGARO\_output (basically your output folder for figaro)

less trimParameters.json

#To run FIGARO in the future. After attaching to a MASSIVE session, just initiate the command source ~data/miniconda3/bin/activate, then conda activate figaro, and then follow instructions as per #Run FIGARO