working dir: /wrk/mishra/identify1

1. Unzip the .gz files

```
gunzip raw_data/*.gz
```

2. Check quality of all the files

```
chmod 0777 ./FastQC/fastqc
./FastQC/fastqc raw_data/*.fastq

# results are here - /wrk/mishra/identify1/raw_data/
fastQC_results

zip -r fastQC_results.zip fastQC_results

copy all the files to /wrk/mishra/identify1/mydata.
```

3. Trim primers

# check forward primer (found)

```
cat saliva_S7_L001_R1_001.fastq | grep CCTACGGGAGGCAGCAG | wc -l

cat saliva_S7_L001_R2_001.fastq | grep CCTACGGGAGGCAGCAG | wc -l
```

# reverse primer is GGACTACHVGGGTWTCTAAT. Nucleotides corresponding to H, V, and W.

```
cat saliva_S7_L001_R1_001.fastq | grep GGACTACCAGGGTATCTAAT | wc -l

cat saliva_S7_L001_R2_001.fastq | grep GGACTACCAGGGTATCTAAT | wc -l
```

# remove forward primer

```
for i in $(ls /wrk/mishra/identify1/mydata/*.fastq)

do

python cutadapt -b CCTACGGGAGGCAGCAG -o ${i%}.noprimer ${i%}

done
```

```
# Remove the old files

rm -rf mydata/*.fastq


# The output files will have no primer at the end. Remove those!

for file in mydata/*.noprimer; do
   mv -- "$file" "${file%%.noprimer}"
done




# remove reverse primer

for i in $(ls /wrk/mishra/identify1/mydata/*.fastq)

do

python cutadapt -b GGACTACCAGGGTATCTAAT -o ${i%}.noprimer ${i%}

done


# Remove the old files

rm -rf mydata/*.fastq

for file in mydata/*.noprimer; do
   mv -- "$file" "${file%%.noprimer}"
done


## verify
cat saliva_S7_L001_R1_001.fastq l grep CCTACGGGAGGCAGCAG l wc -l

cat saliva_S7_L001_R2_001.fastq l grep GGACTACCAGGGTATCTAAT l wc -l
```

4. Trim bad quality sequence. Note minlen - we need at least 200 long fragments (run from /wrk/mishra/identify1/mydata)

```
for f1 in *_R1_001.fastq
do
    f2="${f1/R1/R2}"
    trimmomatic PE -threads 8 -phred33 $f1 $f2 ${f1}
_paired.fastq ${f1}_unpaired.fastq ${f2}_paired.fastq ${f2}
_unpaired.fastq ILLUMINACLIP:/appl/bio/trimmomatic/adapters/
TruSeq3-PE.fa:2:30:10 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:200
done


# remove unpaired ones and old ones
rm -rf *_unpaired.fastq
```

5. Quality check of trimmed samples

```
./FastQC/fastqc mydata/*.fastq

# move fastQC results to another folder
mkdir fastQC_afterTrim
mv *.html fastQC_afterTrim/



## Move all paired.fastq files to mydata_trimmed

mkdir mydata_trimmed

cp mydata/*_paired.fastq mydata_trimmed/
```

6. Remove the '_paired.fastq' extension from the file names in my data

```
for file in *_paired.fastq; do
    mv -- "$file" "${file%%_paired.fastq}"
done
```

7. Merge paired ends in /wrk/mishra/identify1/mydata_trimmed

```
for i in $(ls *.fastq | rev | cut -c 13- | rev | uniq)
do
join_paired_ends.py -f ${i}R1_001.fastq -r ${i}R2_001.fastq -o
merged${i}
done


# remove unmerged files (separate file haru)
rm -rf *.fastq
```

8. Rename the funny merged file names by qiime (repeat manually for all)

```
cd mergedsaliva_S61_L001_
mv fastqjoin.join.fastq saliva_S61_L001.fastq
cd ..
cp mergedsaliva_S61_L001_/saliva_S61_L001.fastq
saliva_S61_L001.fastq
rm -rf mergedsaliva_S61_L001_
```

9. convert all the files into fasta from fastq

```
for i in $(ls *fastq | rev | cut -c 7- | rev | uniq)
do
paste - - - - < ${i}.fastq | sed 's/^@/>/g'| cut -f1-2 | tr '\t' '\n' > ${i}.fasta
done

# get rid of fast files
rm -rf *fastq
```

10. number of seqs in files (useful for normalization)

```
count_seqs.py -i "*.fasta"
```

11. Merge all the samples to one file

```
add_qiime_labels.py -i mydata_trimmed/ -m map.txt -c InputFileName -n 1 -o
combined_fasta
```

12. Chimera detection. we use usearch61 method as we want to do it before taxonomy assignment and without reference

```
#!/bin/bash -l
#SBATCH -J chimera
#SBATCH -o chimera.stdout
#SBATCH -e chimera.stderr
#SBATCH -n 1
#SBATCH -t 12:00:00
#SBATCH --nodes=1
#SBATCH --mem-per-cpu=23000

identify_chimeric_seqs.py -i combined_fasta/combined_seqs.fna  -m usearch61 -o
chimeric_seqs_blast --suppress_usearch61_ref

------------------------------------------------

#  remove chimeric sequences

filter_fasta.py -f combined_fasta/combined_seqs.fna  -o seqs_chimeras_filtered.fna -s
chimeric_seqs_blast/chimeras.txt -n
```

13. remove contaminants acrchaea and eukarya using mothur

a. Classify the sequences based on training dataset downloaded from mothur references

```
mothur > classify.seqs(fasta=seqs_chimeras_filtered.fna,
reference=trainset14_032015.pds.fasta,
taxonomy=trainset14_032015.pds.tax, cutoff=80)

#!/bin/bash -l
#SBATCH -J mothur
#SBATCH -o motur.stdout
#SBATCH -e motur.stderr
#SBATCH -n 5
```

```
#SBATCH -t 12:00:00
#SBATCH --nodes=1
#SBATCH --mem-per-cpu=23000
module load biokit
mothur mothur_task.txt
```

Output File Names:
seqs_chimeras_filtered.pds.wang.taxonomy
seqs_chimeras_filtered.pds.wang.tax.summary
seqs_chimeras_filtered.pds.wang.flip.accnos

b. Remove mitochondrial, chroroplast, archea, eukarya and unknowns

```
mothur > remove.lineage(fasta=seqs_chimeras_filtered.fna,
taxonomy=seqs_chimeras_filtered.pds.wang.taxonomy,taxon=Chloroplast-
Mitochondria-unknown-Archaea-Eukarya)
```

```
Output File Names:
seqs_chimeras_filtered.pds.wang.pick.taxonomy
seqs_chimeras_filtered.pick.fna
```

14. Filter out contaminants identified as contaminants_filter2.fasta (based one the second filtering approach. Reference : identify_filter2.rtf in 'identification' folder).

```
usearch -usearch_global seqs_chimeras_filtered.pick.fna -db
contaminants_filter2.fasta -id 0.99 -strand plus --notmatched cleaned_seqs.fasta
```

usearch.sh

```
#!/bin/bash -l
#SBATCH -J usearch
#SBATCH -o usearch.stdout
#SBATCH -e usearch.stderr
#SBATCH -n 1
#SBATCH -t 2:00:00
#SBATCH --nodes=1
#SBATCH --mem-per-cpu=23000
usearch -usearch_global seqs_chimeras_filtered.pick.fna -db
contaminants_filter2.fasta -id 0.99 -strand plus --notmatched cleaned_seqs.fasta
```

15. OTU picking

```
#!/bin/bash -l
#SBATCH -J otu
#SBATCH -o otu.stdout
#SBATCH -e otu.stderr
#SBATCH -n 1
#SBATCH -t 12:00:00
#SBATCH --nodes=1
#SBATCH --mem-per-cpu=23000

module load qiime

pick_otus.py -i cleaned_seqs.fasta -o otu
```

16. Pick representative sequences

```
pick_rep_set.py -i otu/cleaned_seqs_otus.txt -f cleaned_seqs.fasta -o representative_seqs.fna
```

17. Align representative sequences (latest Silva from - https://www.arb-silva.de/download/archive/qiime/)

http://qiime.org/1.6.0/tutorials/processing_18S_data.html. - to confirm the reference fast used is the right one
The other files in silva123 folder (e.g., 97_otu.fasta) are used to pick reference based OTUs.

```
align_seqs.py -i representative_seqs.fna  -t  core_aligned_SILVA123.fasta -o aligned/
```

18. Taxonomy assignment

```
assign_taxonomy.py -i representative_seqs.fna -r 97_otus_16S.fasta -t taxonomy_all_levels.txt  -c 0.60 -o assigned_taxonomy
```

Or with home database,

```
assign_taxonomy.py -i representative_seqs.fna -r /wrk/mishra/sav/HOMD_16S_rRNA_RefSeq_V14.51.aligned.fasta -t /wrk/mishra/sav/HOMD_16S_rRNA_RefSeq_V14.5.qiime.taxonomy -c 0.60 -o assigned_taxonomy_homd
```

19. Filter alignment

# The suggested alignment filtering settings for filter_alignment.py (1.9.0 QIIME and later) are recommended to be -e 0.10 -g 0.80. See "Alignment Filtering" section for older versions of QIIME.

```
filter_alignment.py -i aligned/representative_seqs_aligned.fasta -e 0.10 -g 0.80 -o 
filtered_alignment/
```

20. Building tree

```
make_phylogeny.py -i filtered_alignment/representative_seqs_aligned_pfiltered.fasta -o 
rep_phylo.tre
```

21. Make OTU table

```
make_otu_table.py -i otu/cleaned_seqs_otus.txt -t assigned_taxonomy/
representative_seqs_tax_assignments.txt -o otu_table.biom
```

Or, based on homd database,

```
make_otu_table.py -i otu/cleaned_seqs_otus.txt -t assigned_taxonomy_homd/
representative_seqs_tax_assignments.txt -o otu_table_homd.biom
```

##### remove singletons ######

21.1 Remove singletons

```
filter_otus_from_otu_table.py -i otu_table.biom  -o otu_table_no_singletons.biom -n 2
```

Or, based on homd database

```
filter_otus_from_otu_table.py -i otu_table_homd.biom  -o 
otu_table_no_singletons_homd.biom -n 2
```

22. Normalize OTU table

```
module load qiime/1.9.1
```

```
normalize_table.py -i otu_table_no_singletons_homd.biom -a CSS -o
CSS_normalized_otu_table_homd.biom
```

```
# check summary
 biom summarize-table -i CSS_normalized_otu_table.biom
```

23. There are many metrics but we use unweigted unifrac (most popular)

```
 beta_diversity.py -i CSS_normalized_otu_table.biom -m
unweighted_unifrac -t rep_phylo.tre -o beta_div
```

Or, using homd database,

```
beta_diversity.py -i CSS_normalized_otu_table_homd.biom -m
unweighted_unifrac -t rep_phylo.tre -o beta_div_homd
```

24.Adonis

```
compare_categories.py --method adonis -i ./beta_div/
unweighted_unifrac_CSS_normalized_otu_table.txt -m map.txt -c
Treatment -o adonis_out
```

Or, with homd database,

```
compare_categories.py --method adonis -i ./beta_div_homd/
unweighted_unifrac_CSS_normalized_otu_table_homd.txt -m map.txt
-c Treatment -o adonis_homd_out

# results same as above!!
```

25. PERMDISP

```
compare_categories.py --method permdisp -i ./beta_div/
```

```
unweighted_unifrac_CSS_normalized_otu_table.txt -m map.txt -c
Treatment -o permdisp_out
```

Or, with homd database,

```
compare_categories.py --method permdisp -i ./beta_div_homd/
unweighted_unifrac_CSS_normalized_otu_table_homd.txt -m map.txt
-c Treatment -o permdisp_homd_out
```

26. PCoA plot (yaha chu)

```
principal_coordinates.py -i beta_div/unweighted_unifrac_CSS_normalized_otu_table.txt
-o ./beta_div_coords.txt
```

```
make_2d_plots.py -i beta_div_coords.txt -m map.txt -o PCoA
```

27. MDS plot

```
summarize_taxa.py -i otu_table_no_singletons.biom -a -o taxonomy_summaries/
```

Or with homd database,

```
summarize_taxa.py -i otu_table_no_singletons_homd.biom -a -o
taxonomy_summaries_homd/
```

# normalize

```
normalize_table.py -i taxonomy_summaries/otu_table_no_singletons_L6.biom -a CSS -
o CSS_normalized_otu_table_summarized.biom
```

See R file in MDS folder.

28. Relative abundance plot

## summarize for relative abundance comparison

```
normalize_table.py -i otu_table.biom -a CSS -o CSS_normalized_otu_table.biom #
summarize_taxa.py -i CSS_normalized_otu_table.biom   -o
```

```
taxonomy_summaries_relative/

## convert the boom file to R format (http://biom-format.org/documentation/biom_conversion.html)
biom convert -I taxonomy_summaries_relative/CSS_normalized_otu_table_L6.biom -o CSS_normalized_otu_table_summarized.biom.txt - -to-tsv
```

29. Differential abundance analysis

```
## (https://github.com/alexcritschristoph/Qiime16sTutorial) Note: use summarized otu table to avoid duplicates

differential_abundance.py -i  taxonomy_summaries/otu_table_no_singletons_L6.biom -o diff_otus.txt -m map.txt -a DESeq2_nbinom -c Treatment -x family -y family1 -d
```

Or with homd,

```
differential_abundance.py -i  taxonomy_summaries_homd/otu_table_no_singletons_homd_L6.biom -o diff_otus_homd.txt -m map.txt -a DESeq2_nbinom -c Treatment -x family -y family1 -d
```