```
######--------- preprocessing outside mothur

## Unzip the .gz files

gunzip *.gz


## 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


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 (module load biokit has it)
for i in $(ls *.fastq)

do

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

done


# Remove the old files

rm -rf *.fastq


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

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



# remove reverse primer
for i in $(ls *.fastq)
do
cutadapt -b GGACTACCAGGGTATCTAAT -o ${i%}.noprimer ${i%}
```

```
done

rm -rf *.fastq
for file in *.noprimer; do
mv -- "$file" "${file%%.noprimer}"
done



## verify
cat saliva_S7_L001_R1_001.fastq | grep CCTACGGGAGGCAGCAG | wc -l
cat saliva_S7_L001_R2_001.fastq | grep GGACTACCAGGGTATCTAAT | wc
-l



##Trim bad quality sequence. Note minlen - we need at least 250
long fragments (previously used min 200 bp)

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:250
done


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

#Remove the '_paired.fastq' extension from the file names

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


########################################################## mothur
part ####################################################

######--------- create stability file
mothur > make.file(inputdir=mother_child_raw, type=fastq,
prefix=stability)
# modified in txt file to add sample ids
```

```
######--------- Reducing sequencing and PCR errors
##--merge F and R sequences
mothur > make.contigs(file=/wrk/mishra/mother_child/
mother_child_raw/stability.paired.files, processors=8)



## check summary
mothur > summary.seqs(fasta=stability.paired.trim.contigs.fasta)


######### filter out badly merged reads

mothur > screen.seqs(fasta=stability.paired.trim.contigs.fasta,
group=stability.paired.contigs.groups, maxambig=0,
maxlength=480,minlength=400)

## check summary
mothur > summary.seqs()



########### get unique sequences ###########

mothur >
unique.seqs(fasta=stability.paired.trim.contigs.good.fasta)

# 176042 (all sequences) 36663 (unique sequences)
Output File Names:
stability.paired.trim.contigs.good.names
stability.paired.trim.contigs.good.unique.fasta


############## make count table ###########

mothur >
count.seqs(name=stability.paired.trim.contigs.good.names,
group=stability.paired.contigs.good.groups)

Output File Names:
stability.paired.trim.contigs.good.count_table

mothur >
summary.seqs(count=stability.paired.trim.contigs.good.count_tabl
e)
```

```
############ align to reference sequences  #######


mothur > align.seqs(candidate=ecoli.fasta,
template=silva.bacteria.fasta)

Output File Names:
ecoli.align
ecoli.align.report


## now check the start and end positions
mothur > summary.seqs(fasta=ecoli.align)


### Now customize silva using above start and end position

mothur > pcr.seqs(fasta=silva.gold.align.fasta, start=6388,
end=43116, keepdots=F, processors=8)

Output File Names:
silva.gold.align.pcr.fasta (it is the customized silva align
data)


### check summary

mothur > summary.seqs(fasta=silva.gold.align.pcr.fasta)



mothur > rename.file(input=silva.gold.align.pcr.fasta,
new=silva.v3v4.fasta)


## alignment (

mothur >
align.seqs(fasta=stability.paired.trim.contigs.good.unique.fasta
, reference=silva.v3v4.fasta)

#!/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



mothur >
summary.seqs(fasta=stability.paired.trim.contigs.good.unique.ali
gn, count=stability.paired.trim.contigs.good.count_table)

## filter out deviants
mothur >
screen.seqs(fasta=stability.paired.trim.contigs.good.unique.alig
n, count=stability.paired.trim.contigs.good.count_table,
summary=stability.paired.trim.contigs.good.unique.summary,
start=40, end=17056, maxhomop=8)


mothur > summary.seqs(fasta=current, count=current)


## filter over hang of the alignments
mothur >
filter.seqs(fasta=stability.paired.trim.contigs.good.unique.good
.align, vertical=T, trump=.)

Length of filtered alignment: 683
Number of columns removed: 36045
Length of the original alignment: 36728
Number of sequences used to construct filter: 37899


mothur >
unique.seqs(fasta=stability.paired.trim.contigs.good.unique.good
.filter.fasta,
count=stability.paired.trim.contigs.good.good.count_table)


# precluster
mothur >
pre.cluster(fasta=stability.paired.trim.contigs.good.unique.good
.filter.unique.fasta,
count=stability.paired.trim.contigs.good.unique.good.filter.coun
t_table, diffs=2)

# rm chimera
```

```
# from count table
mothur >
chimera.vsearch(fasta=stability.paired.trim.contigs.good.unique.
good.filter.unique.precluster.fasta,
count=stability.paired.trim.contigs.good.unique.good.filter.uniq
ue.precluster.count_table, dereplicate=t)


# from fasta file
mothur >
remove.seqs(fasta=stability.paired.trim.contigs.good.unique.good
.filter.unique.precluster.fasta,
accnos=stability.paired.trim.contigs.good.unique.good.filter.uni
que.precluster.denovo.vsearch.accnos)



mothur > summary.seqs(fasta=current, count=current)


# rm Archaea, chloroplasts, and mitochondria. Download
'Trainset16_022016.pds' version 16 from https://mothur.org/wiki/
RDP_reference_files

mothur >
classify.seqs(fasta=stability.paired.trim.contigs.good.unique.go
od.filter.unique.precluster.pick.fasta,
count=stability.paired.trim.contigs.good.unique.good.filter.uniq
ue.precluster.denovo.vsearch.pick.count_table,
reference=trainset16_022016.pds.fasta,
taxonomy=trainset16_022016.pds.tax, cutoff=80)

# rm undesirables
mothur >
remove.lineage(fasta=stability.paired.trim.contigs.good.unique.g
ood.filter.unique.precluster.pick.fasta,
count=stability.paired.trim.contigs.good.unique.good.filter.uniq
ue.precluster.denovo.vsearch.pick.count_table,
taxonomy=stability.paired.trim.contigs.good.unique.good.filter.u
nique.precluster.pick.pds.wang.taxonomy, taxon=Chloroplast-
Mitochondria-unknown-Archaea-Eukaryota)

## remove all contaminants in file 'contaminants_filter2.txt'.
Rename the outputs to make it easier as there will be many
repetations due to many contaminants

# repeat this for each bacteria in the contaminants file
```

```
mothur > remove.lineage(fasta=data.fasta,
count=data.count_table, taxonomy=data.taxonomy,
taxon='Bacteria;Firmicutes;Erysipelotrichia;Erysipelotrichales;E
rysipelotrichaceae;Sharpea;')


##################################################################
##################################################
############################################## analysis part
##################################################
##################################################################
##################################################

## OTU
mothur > dist.seqs(fasta=data.fasta, cutoff=0.03) # OTUs formed
with distance no larger than 3% (i.e, 97% similarity)

Output File Names:
data.dist


mothur > cluster(column=data.dist, count=data.count_table)

Output File Names:
data.opti_mcc.list
data.opti_mcc.steps
data.opti_mcc.sensspec


# number of seqs in each OTUs
mothur > make.shared(list=data.opti_mcc.list,
count=data.count_table, label=0.03)

Output File Names:
data.opti_mcc.shared


# taxonomy for each otu
mothur > classify.otu(list=data.opti_mcc.list,
count=data.count_table, taxonomy=data.taxonomy, label=0.03)

Output File Names:
data.opti_mcc.0.03.cons.taxonomy
data.opti_mcc.0.03.cons.tax.summary
```

```
# number of sequences in each sample
mothur > count.groups(shared=data.opti_mcc.shared)


mothur > sub.sample(shared=data.opti_mcc.shared, size=3157)


##################################################################
################################
######################################## beta diversity
########################################
##################################################################
################################


# calculate similarity
mothur > dist.shared(shared=data.opti_mcc.shared, calc=thetayc-
jclass, subsample=3157)


## dendrogram to describe the similarity of the samples to each
other. We will generate a dendrogram using the jclass and
thetayc calculators within the tree.shared command

mothur > tree.shared(phylip=data.opti_mcc.thetayc.
0.03.lt.ave.dist)


# Using the parsimony command let's look at the pairwise
comparisons. 'family.design' below.
mothur > parsimony(tree=data.opti_mcc.thetayc.0.03.lt.ave.tre,
group=family.design,  groups=all)


mothur > amova(phylip=data.opti_mcc.thetayc.0.03.lt.ave.dist,
design=family.design)




mothur > homova(phylip=data.opti_mcc.thetayc.0.03.lt.ave.dist,
design=family.design)


# plot abundance data
```

```
abun.dat <- read.table("data.opti_mcc.shared",header=T)
rownames(abun.dat) <- abun.dat[,2]
abun.dat <- abun.dat[,-c(1:3)]
abun.dat <- t(abun.dat)
dim(abun.dat) # [1] 312  25


# remove low abundant OTUs and consider only major abundant OTUs
ind <- as.numeric(which(rowSums(abun.dat)<50))
abun.dat <- abun.dat[-ind,]
abun.dat <- abun.dat[,-c(5,6)]
sample.labels <-
c(1,1,1,1,1,1,2,2,2,2,1,1,1,1,1,1,1,1,2,2,1,1)
design <- model.matrix(~factor(sample.labels))
# voom transformation
y <- voom(abun.dat,design,plot=TRUE) # plot looked okay
plotMDS(y,xlim=c(-2.5,2.5),col=sample.labels)


####### grouping only core families

#core_A <- c("3a","3c","4a","4c","7a","7c","8c")
#core_B <- c("1a","1c","2a","4a","4c","5a","6b","6c")

sample.labels.core <-
c(0,0,0,0,0,0,0,0,2,2,2,1,1,1,1,2,2,2,1,1,1,0,0) # THE ONES NOT
IN CORE ARE MARKED 0.
plotMDS(y,xlim=c(-2.5,2.5),col=sample.labels.core)



######### Venn diagram showing the unique and shared OTUs
between parents and children in core families  ##########
## use 'data.opti_mcc.shared' file generated by make.shared
command.

Caption: OTUs defined at 97% sequence similarity. (A) Venn
diagram for mother and children; (B) Venn diagram for father and
children;

data.shared <- read.table("data.opti_mcc.shared",header=TRUE)
data.shared$Group <- as.character(data.shared$Group)



## Nuclear family A (1,2,(4,5,6)) and B (3,4,(7,8)) # ones in
bracket are children.
```

```
data.shared$Group[c(11,12)] <-
c("CoreA_Father_rep1","CoreA_Father_rep2")
data.shared$Group[13] <- c("CoreA_Mother")
data.shared$Group[c(16,17,18,19,20)] <-
c("CoreA_Daughter1_rep1","CoreA_Daughter1_rep2","CoreA_Daughter2
","CoreA_Daughter3_rep1","CoreA_Daughter3_rep2")


# make duplicate of data.shared as sample '4' is present in both
core families
data.shared.cp <- read.table("data.opti_mcc.shared",header=TRUE)
data.shared.cp$Group <- as.character(data.shared.cp$Group)

data.shared.cp$Group[c(14,15)] <-
c("CoreB_Father_rep1","CoreB_Father_rep2")
data.shared.cp$Group[c(16,17)] <-
c("CoreB_Mother_rep1","CoreB_Mother_rep2")
data.shared.cp$Group[c(21,22,23)] <-
c("CoreB_Daughter_rep1","CoreB_Daughter_rep2","CoreB_Son")



# mothur > venn(shared=data.shared, groups=CoreA_Father_rep1-
CoreA_Father_rep2-CoreA_Mother-CoreA_Daughter1_rep1-
CoreA_Daughter1_rep2-CoreA_Daughter2-CoreA_Daughter3_rep1-
CoreA_Daughter3_rep2) ## this will generate plots for all
possible combination as mothur can have only 4 groups at a time
for venn. Take average of counts to avoid too many combinatorics
problem. Moreover, different overlap with different replicates
was observed. So, averaging the replicates can be a better idea.

# take average of replicates
abun.shared <- data.shared[,c(4:315)]
coreA_father_avg <-
colMeans(rbind(abun.shared[11,],abun.shared[12,]))
coreA_father_avg <- c(0.03,"coreA_father_avg",
312,coreA_father_avg)
data.shared <- rbind(data.shared,coreA_father_avg) # adding row
of averaged data

coreA_Daughter1_avg <-
colMeans(rbind(abun.shared[16,],abun.shared[17,]))
coreA_Daughter1_avg <- c(0.03,"coreA_Daughter1_avg",
312,coreA_Daughter1_avg)
data.shared <- rbind(data.shared,coreA_Daughter1_avg)
```

```
coreA_Daughter3_avg <-
colMeans(rbind(abun.shared[19,],abun.shared[20,]))
coreA_Daughter3_avg <- c(0.03,"coreA_Daughter3_avg",
312,coreA_Daughter3_avg)
data.shared <- rbind(data.shared,coreA_Daughter3_avg)

# now make sure data types are intact
data.shared$label <- as.numeric(data.shared$label)
data.shared$Group <- factor(data.shared$Group)
for(i in 3:315){
    data.shared[,i] <- as.numeric(data.shared[,i])
}


for(i in 4:315){
    data.shared[,i] <- round(data.shared[,i],digits=0) # make
interegs
}


write.table(data.shared,file="data.shared",quote=FALSE,row.names
=FALSE,sep="\t")

mothur > get.group(shared=data.shared)
mothur > venn(shared=data.shared, groups=coreA_father_avg-
CoreA_Mother-coreA_Daughter1_avg-coreA_Daughter3_avg-
CoreA_Daughter2)

Output File Names:
data.0.03.sharedsobs.CoreA_Mother-CoreA_Daughter2-
coreA_father_avg-coreA_Daughter1_avg.svg
data.0.03.sharedsobs.CoreA_Mother-CoreA_Daughter2-
coreA_father_avg-coreA_Daughter3_avg.svg
data.0.03.sharedsobs.CoreA_Mother-coreA_father_avg-
coreA_Daughter1_avg-coreA_Daughter3_avg.svg

### Pvalues (https://stackoverflow.com/questions/18340123/
calculate-venn-diagram-hypergeometric-p-value-using-r)

require(gmp)
enrich_pvalue <- function(N, A, B, k)
{
    # N: total number of OTUs
    # A: unique number of OTUs in setA
    # B: unique number of OTUs in setB
    # k: overlap between sets A and B
    m <- A + k
```

```
    n <- B + k
    i <- k:min(m,n)

    as.numeric( sum(chooseZ(m,i)*chooseZ(N-m,n-i))/
chooseZ(N,n) )
}

#---data.0.03.sharedsobs.CoreA_Mother-CoreA_Daughter2-
coreA_father_avg-coreA_Daughter1_avg.svg
#--mother:daughter1
enrich_pvalue(137, 31, 21, 54) # 0.006811284

#--mother:daughter2
enrich_pvalue(137, 46, 24, 39) # 0.5825842

#--father:daughter1
enrich_pvalue(137, 21, 23, 52) # 3.138095e-05

#--father:daughter2
enrich_pvalue(137, 36, 26, 37) # 0.156985



#---data.0.03.sharedsobs.CoreA_Mother-CoreA_Daughter2-
coreA_father_avg-coreA_Daughter3_avg.svg
#--mother:daughter1
enrich_pvalue(125, 31, 21, 54) #  0.1638773

#--mother:daughter3
enrich_pvalue(125, 33, 13, 52) # 0.002430472

#--father:daughter1
enrich_pvalue(125, 21, 23, 52) #  0.002156828

#--father:daughter3
enrich_pvalue(125, 27, 19, 46) # 0.002984995


#--data.0.03.sharedsobs.CoreA_Mother-coreA_father_avg-
coreA_Daughter1_avg-coreA_Daughter3_avg.svg

#--mother:daughter2
enrich_pvalue(136, 46, 24, 39) #  0.6223158

#--mother:daughter3
enrich_pvalue(136, 33, 13, 52) # 4.593393e-05
```

```
#--father:daughter2
enrich_pvalue(136, 36, 26, 37) # 0.1773834

#--father:daughter3
enrich_pvalue(136, 27, 19, 46) # 0.0001144497




#### core B ##
abun.shared.copy <- data.shared.cp[,c(4:315)]
coreB_father_avg <-
colMeans(rbind(abun.shared.copy[14,],abun.shared.copy[15,]))
coreB_father_avg <- c(0.03,"coreB_father_avg",
312,coreB_father_avg)
data.shared.cp <- rbind(data.shared.cp,coreB_father_avg)

coreB_mother_avg <-
colMeans(rbind(abun.shared.copy[16,],abun.shared.copy[17,]))
coreB_mother_avg <- c(0.03,"coreB_mother_avg",
312,coreB_mother_avg)
data.shared.cp <- rbind(data.shared.cp,coreB_mother_avg)

coreB_daughter_avg <-
colMeans(rbind(abun.shared.copy[21,],abun.shared.copy[22,]))
coreB_daughter_avg <- c(0.03,"coreB_daughter_avg",
312,coreB_daughter_avg)
data.shared.cp <- rbind(data.shared.cp,coreB_daughter_avg)

# now make sure data types are intact
data.shared.cp$label <- as.numeric(data.shared.cp$label)
data.shared.cp$Group <- factor(data.shared.cp$Group)
for(i in 3:315){
    data.shared.cp[,i] <- as.numeric(data.shared.cp[,i])
}


for(i in 4:315){
    data.shared.cp[,i] <- round(data.shared.cp[,i],digits=0) #
make interegs
}

write.table(data.shared.cp,file="data.shared.copy",quote=FALSE,r
ow.names=FALSE,sep="\t")
```

```
mothur > get.group(shared=data.shared.copy)
mothur > venn(shared=data.shared.copy, groups=coreB_father_avg-
coreB_mother_avg-coreB_daughter_avg-CoreB_Son)

Output File Names:
data.shared.0.03.sharedsobs.CoreB_Son-coreB_father_avg-
coreB_mother_avg-coreB_daughter_avg.svg
data.shared.0.03.sharedsobs.CoreB_Son-coreB_father_avg-
coreB_mother_avg-coreB_daughter_avg.sharedotus


#### Pvalue calculation

#--father:son
require(gmp)
#enrich_pvalue(total_OTUs, unique_OTUs_in_father,
Unique_OTUs_in_son, overlap_between_father_son)
enrich_pvalue(140, 6, 68, 42) # 0.04714968

#--father:daughter
#enrich_pvalue(total_OTUs, unique_OTUs_in_father,
Unique_OTUs_in_daughter, overlap_between_father_daughter)
enrich_pvalue(140, 9, 48, 39) #  0.0005486819


#--mother:son
#enrich_pvalue(total_OTUs, unique_OTUs_in_mother,
Unique_OTUs_in_son, overlap_between_mother_son)
enrich_pvalue(140, 13, 48, 62) #  0.144163

#--mother:daughter
#enrich_pvalue(total_OTUs, unique_OTUs_in_mother,
Unique_OTUs_in_daughter, overlap_between_mother_daughter)
enrich_pvalue(140, 15, 27, 60) #  2.649257e-06



## sub function
enrich_pvalue <- function(N, A, B, k)
{
    m <- A + k
    n <- B + k
    i <- k:min(m,n)

    as.numeric( sum(chooseZ(m,i)*chooseZ(N-m,n-i))/
chooseZ(N,n) )
}
```