# mothur pipeline

*Kim Dill-McFarland*

*version April 12, 2019*

## Contents



### Data description

Sequence data is derived from Saanich Inlet Cruise 72. Information about the sequencing technique can be found in Hawley et al. 2017. Sequences were amplified using the 515F and 808R primers. Sequences were generated on a MiSeq with Phred33 quality scores.

### mothur

Linux version
Using ReadLine
Running 64Bit Version
mothur v.1.39.0

### Import data

Create `.files` listing the full path and sample name of all the fastqs containing sequences for this project.

```
make.file(inputdir=[filePath]/Saanich, prefix=Saanich)
```

### Clean-up

Combine paired end reads into contigs using their overlapping regions.

```
make.contigs(file=Saanich.files, processors=10)
```

Summarize the resulting contigs to see potential error, *e.g.* homopolymers, ambiguous base pairs (N), too long, too short, etc.

```
summary.seqs(fasta=Saanich.trim.contigs.fasta)
```

Remove low quaility sequences.

```
screen.seqs(fasta=Saanich.trim.contigs.fasta, group=Saanich.contigs.groups,
            maxambig=0, maxhomop=8, minlength=200, maxlength=600)
```

Determine unique sequences to reduce computational needs. Then, mothur can analysis a single sequence and apply the output to all other sequences in the data set that are identical.

```
unique.seqs(fasta=Saanich.trim.contigs.good.fasta)
```

Summarize again.

```
summary.seqs(fasta=Saanich.trim.contigs.good.unique.fasta,
            count=Saanich.trim.contigs.good.count_table)
```

Align sequences to the SILVA database.

```
align.seqs(fasta=Saanich.trim.contigs.good.unique.fasta,
          reference=/home/GLBRCORG/dillmcfarlan/mothur_files/silva.nr_v128.align,
          flip=T, processors=10)
```

Summarize to determine the start and end sites of the data.

```
summary.seqs(fasta=Saanich.trim.contigs.good.unique.align,
            count=Saanich.trim.contigs.good.count_table)
```

Cut the sequences to consistent start and end sites.

```
screen.seqs(fasta=Saanich.trim.contigs.good.unique.align,
           count=Saanich.trim.contigs.good.count_table,
           summary=Saanich.trim.contigs.good.unique.summary,
           start=10368, end=25434, processors=10)
```

Remove useless alignment data like columns that are all blank or all "."

```
filter.seqs(fasta=Saanich.trim.contigs.good.unique.good.align, vertical=T, trump=.)
```

Determine unique sequences again as alignment will result in more similarity between very similar sequences.

```
unique.seqs(fasta=Saanich.trim.contigs.good.unique.good.filter.fasta,
            count=Saanich.trim.contigs.good.good.count_table)
```

Reduce sequencing error by combining (clustering) sequences that are 3 or fewer bp different based on the general rule of 1 bp error per 100 bp sequence.

```
pre.cluster(fasta=Saanich.trim.contigs.good.unique.good.filter.unique.fasta,
           count=Saanich.trim.contigs.good.unique.good.filter.count_table,
           diffs=3)
```

Summarize to few data.

```
summary.seqs(fasta=Saanich.trim.contigs.good.unique.good.filter.unique.precluster.fasta,
             count=Saanich.trim.contigs.good.unique.good.filter.unique.precluster.count_table)
```

Identify and remove chimeric sequences.

```
chimera.uchime(fasta=Saanich.trim.contigs.good.unique.good.filter.unique.precluster.fasta,
              count=Saanich.trim.contigs.good.unique.good.filter.unique.precluster.count_table,
              dereplicate=t)
```

```
remove.seqs(fasta=Saanich.trim.contigs.good.unique.good.filter.unique.precluster.fasta,
           count=Saanich.trim.contigs.good.unique.good.filter.unique.precluster.count_table,
           accnos=Saanich.trim.contigs.good.unique.good.filter.unique.precluster.denovo.uchime.accnos)
```

Remove singletons, sequences that occur only once across the entire data set, as they are likely error.

```
split.abund(fasta=Saanich.trim.contigs.good.unique.good.filter.unique.precluster.pick.fasta,
           count=Saanich.trim.contigs.good.unique.good.filter.unique.precluster.pick.count_table,
           cutoff=1)
```

Cope and rename the sequence (fasta) and sample data (count_table) to "final" files.

```
system(cp Saanich.trim.contigs.good.unique.good.filter.unique.precluster.pick.abund.fasta
      Saanich.final.fasta)

system(cp Saanich.trim.contigs.good.unique.good.filter.unique.precluster.pick.abund.count_table
      Saanich.final.count_table)

count.groups(count=Saanich.final.count_table)
```

## Determine OTUs

Calculate distances between sequences.

```
dist.seqs(fasta=Saanich.final.fasta, processors=15)
```

De novo cluster the sequences.

```
cluster.split(column=Saanich.final.dist, count=Saanich.final.count_table,
             method=opti, processors=10, large=T)
```

Select OTUs at 97% similarity (3% difference, *e.g.* species level).

```
make.shared(list=Saanich.final.opti_mcc.unique_list.list, count=Saanich.final.count_table,
            label=0.03)
```

## Classify OTUs

Classify all sequences using the SILVA database.

```
classify.seqs(fasta=Saanich.final.fasta, count=Saanich.final.count_table,
              template=/home/GLBRCORG/dillmcfarlan/mothur_files/silva.nr_v128.align,
              taxonomy=/home/GLBRCORG/dillmcfarlan/mothur_files/silva.nr_v128.tax,
              cutoff=80, processors=10)
```

Condense these taxonomies for each OTU.

```
classify.otu(list=Saanich.final.opti_mcc.unique_list.list,
            taxonomy=Saanich.final.nr_v128.wang.taxonomy,
            count=Saanich.final.count_table,
            label=0.03, cutoff=80, basis=otu, probs=F)
```

## Representative sequences

Obtain a representative sequence using the most abundance unique sequence for each OTU.

```
get.oturep(list=Saanich.final.opti_mcc.unique_list.list, fasta=Saanich.final.fasta,
          count=Saanich.final.count_table, label=0.03, method=abundance)
```

Rename sequences with OTU numbers (python, courtesy Anthony Neumann).

```
import sys
import re

#an empty list to hold the lines, as strings, of the input fasta file
info = []

#iterate through the input file and file up the list
with open(sys.argv[1], 'r') as fasta_in:
    for line in fasta_in:
```

```
        info.append(line.replace('\r', '').rstrip('\n'))

#an empty list to hold the OTU fasta seqs and their IDs as tuples
seqs = []

#iterate through the info list, replace the complex seq IDs with just the OTU ID,
#and fill up the seqs list
for index, item in enumerate(info):
    if index == 0:
        ID = re.search('Otu[0-9]*',item[1:]).group(0)
    else:
        if index % 2 == 0:
            ID = re.search('Otu[0-9]*',item[1:]).group(0)
        else:
            seqs.append((ID, item))

#write the contents of the seqs list to a new file called "clean_repFasta.fasta"
with open("clean_repFasta.fasta", 'w') as fasta_out:
    for item in seqs:
        fasta_out.write('>' + item[0] + '\n' + item[1] + '\n')
```

```
python clean_repFasta_FAST.py Saanich.final.opti_mcc.unique_list.0.03.rep.fasta
```