OTU Clustering and Denoising I

Luke E. Holman

2019-06-24

Analysing metabarcoding data using metabarTOAD

There are a great number of pipelines and methods for the analysis of metabarcoding data. No single method is appropriate or inappropriate in all situations. Methods and ideas are constantly being developed so keep an eye on the horizon and focus on fundamental and transferable skills. This tutorial is for beginners and demonstrates the workflow.

Requirements

Library design

This pipeline is designed to work with paired-end Illumina amplicon data generated from a 2-step PCR method as Bista et al.2017 (https://www.nature.com/articles/ncomms14087). We are expecting sufficient (>30bp) overlap of the paired end reads.

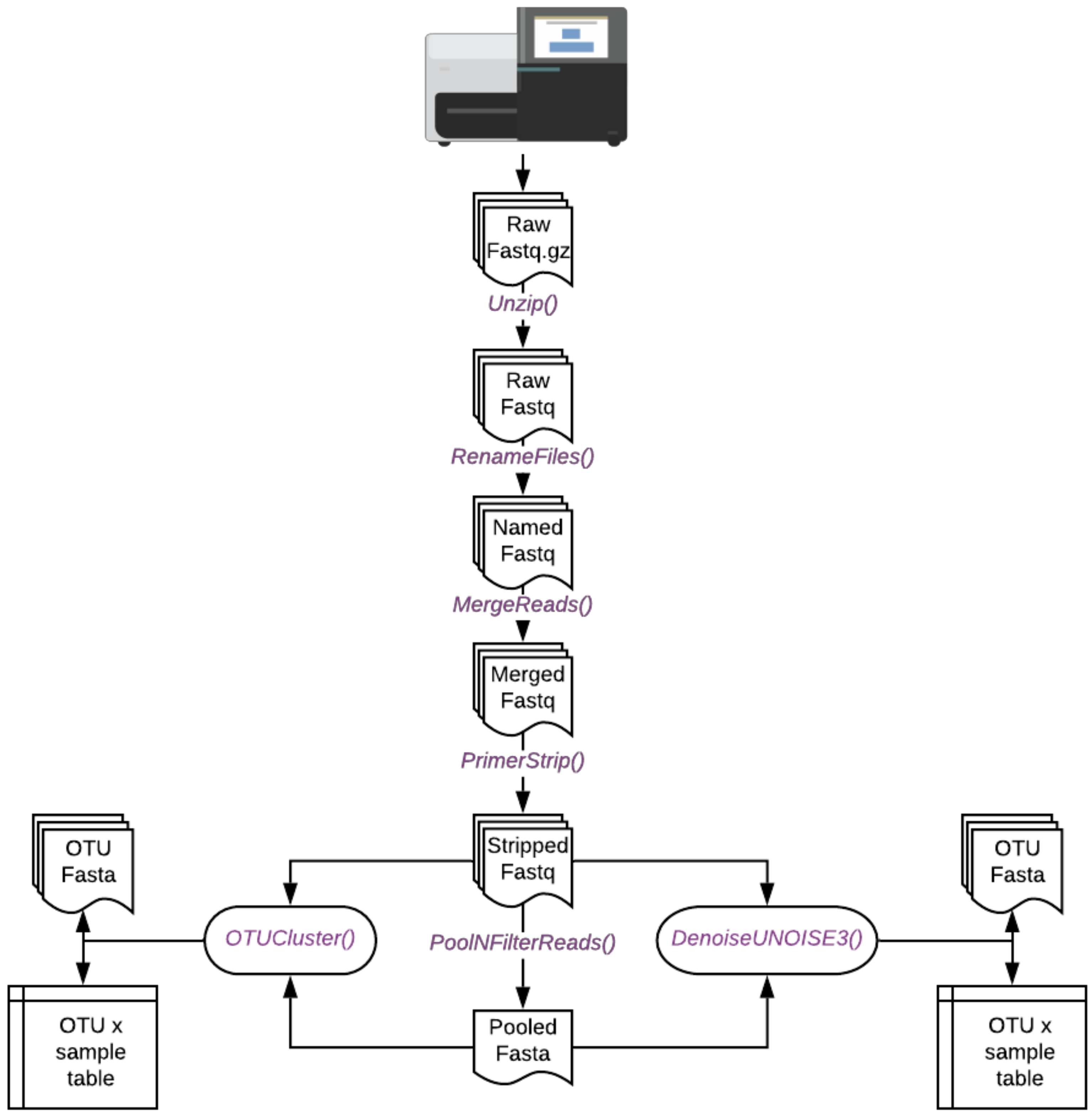
Software

For this tutorial you will need the following

- Cutadapt (version 2.0+) link (https://cutadapt.readthedocs.io/en/stable/installation.html)
- VSEARCH (version 2.10.4+) link (https://github.com/torognes/vsearch)
- USEARCH (version 11+) link (https://drive5.com/usearch/download.html)

These should be installed and working on your system. Keep a note of the directory path to the binary or have them in your PATH. Not sure about the PATH? - go here (http://www.linfo.org/path_env_var.html)

Workflow Overview



Setting up

Make sure you start off with a clean folder in a new working directory. Check there is nothing in your directory with `dir()`. Let's start by loading up the required package(s)

```
require(metabarTOAD)
```

Now we use the first command to set up the required folder structure.

```
Folders()
```

Great, lets check the folders. We can use `dir()` as before. We should see 8 folders with different names prefixed with a number.

Now we have a folder structure lets move our raw Illumina data into the `1.rawreads`. You might wish to copy your raw files into the `0.backup` folder as well in case you want to start the analysis again from scratch.

The pipeline can be run for multiple primer sets by providing sample and primer data in the required format. Download the examples here (). This tutorial assumes we're only working on a single set of primers but provide primer and sample data.

Unzipping and Counting

First lets check our files to make sure everything is as we expect.

```
list.files("1.rawreads/")
```

We should have pairs of compressed sample files ready for analysis.

Now everything is in place we can start to analyse the data. First lets unzip all the data, and then check to make sure everything has been unzipped.

```
Unzip()
list.files("1.rawreads/")
```

We can rename our files at this stage if we wish. First we pull the names from our metadata file.

```
index <- read.csv("metadata.csv")
```

We now put the desired and current names into the function `RenameFiles` to change the names of our samples.

```
RenameFiles(SeqIDs = as.character(index$RunID),DesiredIDs = as.character(index$Rea
lID))
```

If we prefer we can perform the same operation using the metadata sheet that we have copied into the base of the working directory with the below function. If you ran the above function then you have already renamed your files so no need to run the line below!

```
RenameFiles(usemetadatafile = TRUE)
```

Now we have everything named we can count the number of reads per sample. First we create a list of files to be counted.

```
files <- list.files("1.rawreads",pattern=".fastq",full.names = TRUE)
```

Then we use `FastqCount` and `sapply` to count the reads for each sample.

```
rawreadcount <- sapply(files,FastqCount)
```

We can calculate the mean and standard deviation of reads across the raw samples.

```
mean(rawreadcount)
sd(rawreadcount)
```

Merging and Filtering

Next we merge the forward and reverse reads as below.

```
MergeReads(usearchdest = "/path/to/usearch")
```

We can count the number of merged reads like this.

```
sapply(list.files("2.mergedreads",pattern=".fastq",full.names = TRUE),FastqCount)
```

Our sequences are now combined into a single fastq per read pair. Next we need to strip the primers from each end of the read. We do this as below.

```
PrimerStrip(PrimerF = "NNNNNNGGWACWGGWTGAACWGTWTAYCCYCC",
            PrimerR = "TAIACYTCIGGRTGICCRAARAAYCA",
            MinLen = 303,
            MaxLen = 323,
            cutadaptdest = "cutadapt",
            ncores=1)
```

Alternativly we can provide much of the above information in a primer file and process many primers at the same time as below.

```
PrimerStrip(UsePrimerFile = TRUE,cutadaptdest = "/path/to/cutadapt", ncores=7)
```

Next we pool all our sequences together and discard singleton sequences (sequences only found once in the data).

```
PoolNFilterReads(vsearchdest="/path/to/vsearch")
```

We can also do this with multuple primers at once as shown with the primer stripping step above. To do this use the argument `UsePrimerFile = TRUE` .

Clustering and Denoising

Now everything is prepped and ready to go we can generate 97% OTUs using the below command. As previously we can use `UsePrimerFile=TRUE` if we are using multiple primer sets.

```
OTUCluster(usearchdest = "/path/to/usearch")
```

We can produce denoised OTUs (sometimes called MOTUS or ASVs or ESVs) using the UNOISE3 algorithm (paper here (https://www.biorxiv.org/node/22505.abstract)). Again use `UsePrimerFile=TRUE` in the case of multiple primers.

```
DenoiseUNOISE3(usearchdest = "/path/to/usearch")
```

When using the above two functions the OTUs will be output to the `5.OTUs` folder and the OTU by sample table to `6.mappings/OTUtabs` . These files can be used in downstream ecological analyses.

Post Clustering Curation with LULU

Very often in metabarcoding datasets we see a large number of OTUs with close percentage identity. These may correspond with intraspecific diversity, unfiltered PCR errors or other methodological foibles. One way to deal with these is to use the LULU algorithm (paper here (https://www.nature.com/articles/s41467-017-01312-x)). A great tutorial is provided on the GitHub page (https://github.com/tobiasgf/lulu), but you can just use the below wrapper script if you prefer.

```r
library("lulu")

ApplyLulu(seqs="/path/to/OTUs.csv",
          table="/path/to/OTU/by/sample/table.csv",
          output="/path/todesired/output/table/lulu.csv",
          vsearchdest="/path/to/vsearch")
```