DADA2

https://www.bioconductor.org/packages/release/bioc/vignettes/dada2/inst/doc/dada2-intro.html#overview-of-the-dada2-pipeline

```
library(dada2); packageVersion("dada2")

fnF1 <- system.file("extdata", "sam1F.fastq.gz", package="dada2")
fnR1 <- system.file("extdata", "sam1R.fastq.gz", package="dada2")
filtF1 <- tempfile(fileext=".fastq.gz")
filtR1 <- tempfile(fileext=".fastq.gz")
```

1 Filter and Trim

check quality:

```
plotQualityProfile(fnF1) # Forward
plotQualityProfile(fnR1) # Reverse
```

```
filterAndTrim(fwd=fnF1, filt=filtF1, rev=fnR1, filt.rev=filtR1,
                  trimLeft=10, truncLen=c(240, 200),
                  maxN=0, maxEE=2,
                  compress=TRUE, verbose=TRUE)
```

you don't need to the forward and the reverse reads at the same position, resulting in reads of different length.

The FilterAndTrim() function filter forward and reverse reads jointly

- *trimLeft=10* unable to remove the first 10 nucleotides of each reads(as there usually is a drop for the first nucleotides)
- *truncLen=c(240, 200)* trunk the forward at nucleotide 240 and the backward at nucleotide 200
- *maxN=0* we filter out all the reads with more than 0 ambiguous nucleotides

If using a pair-end sequencing data, must have at the very least 20 nucleotides that overlap after the trimming.

2 Dereplicate

```
derepF1 <- derepFastq(filtF1, verbose=TRUE)
derepR1 <- derepFastq(filtR1, verbose=TRUE)
```

Condense the data by collapsing together all reads that encode the same sequence, which significantly reduces later computation times derepFastq maintain a summary of the quality information for each dereplicated sequence in $quals

3 Learn the errors rates

```
errF <- learnErrors(derepF1, multithread=FALSE) # multithreading is available on
many functions
errR <- learnErrors(derepR1, multithread=FALSE)
```

4 Infer sample composition

```
dadaF1 <- dada(derepF1, err=errF, multithread=FALSE)
dadaR1 <- dada(derepR1, err=errR, multithread=FALSE)
print(dadaF1)
```

5 Merge forward/reverse reads

We've inferred the sample sequences in the forward and reverse reads independently. Now it's time to merge those inferred sequences together, throwing out those pairs of reads that don't match.

It will return a data frame corresponding to each successfully merged sequences.

```
merger1 <- mergePairs(dadaF1, derepF1, dadaR1, derepR1, verbose=TRUE)
```

6 Remove chimeras

when sequence incompletely amplified, a chimera is formed and will result in the next generation in half of this amplicon and half of an other. Hence we have to remove those sequences.

```
merger1.nochim <- removeBimeraDenovo(merger1, multithread=FALSE, verbose=TRUE)
```

7 A second sample

repeat the same steps for an other sample, pretty straight forward.

8 Create a sequence table

If we want to combine all the inferred samples into one unified table: give a matrix.

Each row is a processed sample and each column is a non-chimeric inferred sample sequence.

```
seqtab <- makeSequenceTable(list(merger1, merger2))
seqtab.nochim <- removeBimeraDenovo(seqtab, verbose=TRUE)
dim(seqtab.nochim)
```

9 further analysis

They recommend the "*phyloseq*" package: a tool to import, store, analyze, and graphically display complex phylogenetic sequencing data that has already been clustered into Operational Taxonomic Units.

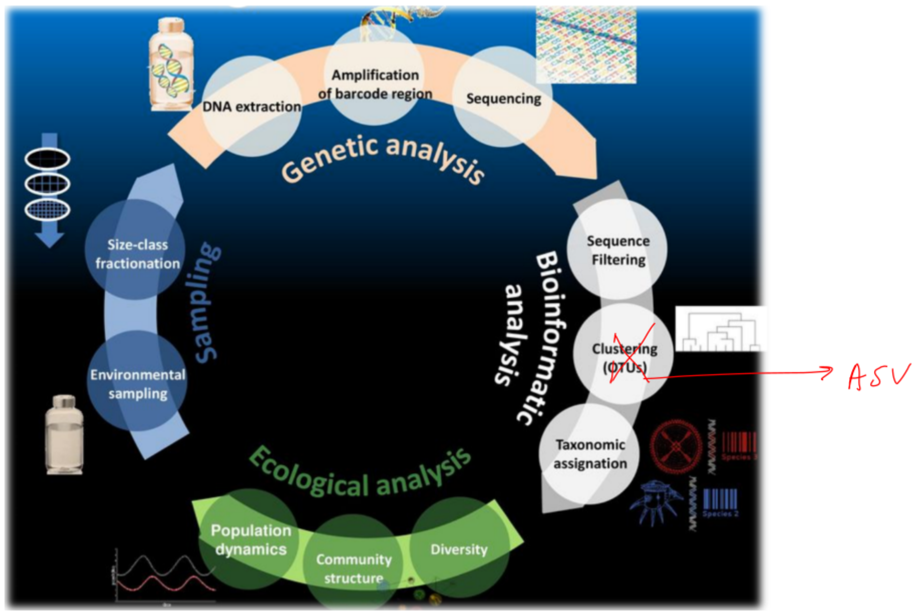
I'm not sure if it is really useful in our case.

Notes from Mar

Goal of metabarcoding

Characterization of the taxonomic diversity inhabiting various ecosystems using direct environmental DNA

We can use the full genome, hence, we have to use 16S region (~200-400bp): a marker for bacterial and Archeal identification. But there is PCR and sequencing (Illumina) errors in the sequences.



Pair-end sequencing is usually better, as max read of illumina is 300bp, we obtain 2 x 300bp.

We pool all the sample together(multiplexing and add an index/barcode to the sequence so that we can then separate the samples: demultiplexing). It cost less, is faster and we can avoid as much batch effect as possible.

Then we demultiplex using the barcodes.

Filter: remove bad quality sequencing reads

Cluster(OTUs): one cluster = sequence identity > 97%. And there is a centroid sequence taht is the sequence that minimize the sum of the distances to the other sequences of the cluster (commonly the most abundant one).

Taxonomic assignation

Setting the environment

Store the paths

```
fnFs <- sort(list.files(path, pattern="_R1.fastq", full.names=TRUE))
# List the files names located at path containing the pattern and gives the full
name (full path)
# and all these names are stored in fnFs for R1 and fnFr for R2
```

Retrieve sample names

```
sample.names <- sapply(strsplit(basename(fnFs), "_"), `[`, 1)
# basename retrieve the name before the last separator "/"
# strsplit split by removing the "_"
```

Create a folder "filtered" with the file names

```
filtFs <- file.path(path, "Filtered", basename(fnFs))
# will create a new folder "Filtered" withe all the file names inside (the files are
empty).
```

Set working directory

Phyloseq objects

Abundance table

```
otu_table()
# OTU abundance in the different samples
```

Metadata

```
sample_data()
# contains group, treatments, localisations, sample names, env data
```

Taxonomy classification

```
tax_table()
# devide each taxa in taxonomic ranks: Kingdom, Phylum, Class, Order, Family (and
the sample name of course)
```

Phylogenic tree

```
phy_tree()
# number of nodes, number of tips
```

DNA sequence of ASV/OTU

```
ref_seq
# gives the reference sequences I think
```

Table with abundance and taxonomic assignement

> we have to fuse two tables: otu_table() and tax_table()

```
tableau=cbind(t(otu_table(Myselection2)),tax_table(Myselection2))
# t() is to transpose the table
# but you don't always need to transpose when you are working on microbes data
apparently
```