Part I: DADA2

I- Packages/Libraries

a/ Package Installation via Bioconductor
https://www.bioconductor.org/

#**Bioconductor** :
 # Bioconductor Install & bioconductor packages

```
if (!requireNamespace("BiocManager", quietly = TRUE)) BiocManager::install()
if (!requireNamespace("dada2", quietly = TRUE))       BiocManager::install("dada2", version = "3.9")
if (!requireNamespace("phangorn", quietly = TRUE))    BiocManager::install("phangorn", version = "3.9")
if (!requireNamespace("DECIPHER", quietly = TRUE))    BiocManager::install("DECIPHER", version = "3.9")
if (!requireNamespace("phyloseq", quietly = TRUE))    BiocManager::install("phyloseq", version = "3.9")
if (!requireNamespace("Biobase", quietly = TRUE))     BiocManager::install("Biobase", version = "3.9")
if (!requireNamespace("DESeq2", quietly = TRUE))      BiocManager::install("Deseq2", version = "3.9")
if (!requireNamespace("microbiome", quietly = TRUE))  BiocManager::install("microbiome", version = "3.9")

####### Install packages from CRAN & load libraries (both)
install.packages("pacman")
pacman::p_load("vegan","scales","gplots","ggplot2","permute","dplyr","tibble","ape","RcolorBrewer","reshape2
","FSA","gridExtra")
#######
###### Load libraries from Bioconductor packages
#library(ShortRead)
library(phyloseq)
library(dada2)
library(DECIPHER)
library(phangorn)
library(Biobase)
library(microbiome)
library(DESeq2)
####################
#Attached script to my session
source("./ggrare.R")
source("./Fonctions_dada2.R")
####################
```

II- Path- set working directory

Rstudio Working Directory: Location of where you are going to work.

Move to the directory which contains your data files, follow:
Menu ->Session -> Set working directory -> Choose Directory (which is Formation2020)
Your sequencing data is in the directory data so :

```
#Put the path directory of your data files in a « variable » named path
path="./data"
```

III- Read the sequencing files

a/ list of sequencing files

The « forward » & « reverse » files are in fastq format with the label:
SAMPLENAME_R1.fastq for the Forward files and **SAMPLENAME_R2.fastq** for the reverses files.
R1 is Forward, R2 is Reverse

Function : list.files
https://stat.ethz.ch/R-manual/R-devel/library/base/html/list.files.html

```
fnFs  <- sort(list.files(path, pattern="_R1.fastq", full.names=TRUE))
fnRs  <- sort(list.files(path, pattern="_R2.fastq", full.names=TRUE))
```

To understand: What FnFs & FnRs files contain??

```
fnFs
```

```
 [1] "./data/S11B_R1.fastq" "./data/S1B_R1.fastq"
"./data/S2B_R1.fastq"
"./data/S2S_R1.fastq"
"./data/S3B_R1.fastq"
"./data/S3S_R1.fastq"
"./data/S4B_R1.fastq"
etc
```

b/ Extract the names of the samples

```
sample.names <- sapply(strsplit(basename(fnFs), "_"), `[`, 1)
```

To understand:
strsplit : split character chain according a defined pattern
https://www.rdocumentation.org/packages/base/versions/3.6.0/topics/strsplit
basename: simplify the « PATH »
https://www.rdocumentation.org/packages/base/versions/3.6.0/topics/basename

```
strsplit(basename(fnFs), "_")
```

"strsplit" function?

```
sapply(strsplit(basename(fnFs), "_"), `[`, 2)
sapply(strsplit(basename(fnFs), "_"), `[`, 3)
```
« Sapply » function?

IV- Sequence Quality Check

Function: plotQualityProfile :
https://www.rdocumentation.org/packages/dada2/versions/1.0.3/topics/plotQualityProfile
We use a function implemented in the Fonctions_dada2.R script, which takes the list of R1 files and R2 files and put all quality plot result in one pdf file.
qualityprofile(fnFs,fnRs,'qualityplot.pdf')

look the qualityplot.pdf

V- Sequence Trimming

a/ Prepare the directory for the Trimming process

Function : file.path :
https://www.rdocumentation.org/packages/R.utils/versions/2.8.0/topics/filePath

On crée un dossier nommé « Filtered », et à l'intérieur on y met les fichiers nommés « nom du sample" pour R1 et R2. Ces fichiers sont vides… c'est une préparation pour l'étape de filtrage.

```
filtFs <- file.path(path, "Filtered", basename(fnFs))
filtRs <- file.path(path, "Filtered", basename(fnRs))
names(filtFs) <- sample.names
names(filtRs) <- sample.names
```

b/ Trimming process (important !!!!)

Function filterAndTrim: https://rdrr.io/bioc/dada2/man/filterAndTrim.html

```
out <- filterAndTrim(fnFs, filtFs, fnRs, filtRs, trimLeft=20, trimRight=20,
                     minLen=150,  maxN=0, maxEE=c(3,3), truncQ=2, compress=FALSE,
multithread=TRUE)
```

To see your results

```
out
```

Details:

fnFs : input, your row Forward list (path) of sequencing files.
filtFs : output, the **empty files created** at the previous stage that you are now going to fill with this command.
fnRs, filRs : Same as above, but for the Reverse data

TruncLen : Truncate reads after truncLen bases. Reads shorter than this are discarded.
Exple : TruncLen=c(200,150), means forwards R1 are cut at 200 pb & the reverse R2 at 150 pb.
TrimLeft : The number of nucleotides to remove from the start of each read, from the left

Trimright : The number of nucleotides to remove from the start of each read, from the right

maxN : Max number of ambiguous bases accepted

maxEE : Quality system to remove low quality read. Standard **maxEE=c(2,2).**
If you want to be more flexible (accept more low quality), increase the value, maxEE=c(2,5). 2 is for the forward, 5 for le reverse.

TruncQ=2. Truncate reads at the first instance of a quality score less than or equal to `truncQ`.

VI- Learn « Errors »

Function: **learnErrors** https://rdrr.io/bioc/dada2/man/learnErrors.html
Parametric model to distinguish sequencing artefacts from true biological variations

```
errF <- learnErrors(filtFs, multithread=TRUE)
errR <- learnErrors(filtRs, multithread=TRUE)
plotErrors(errF, nominalQ=TRUE)
```

VII- Dereplication (identical sequences)

→ "remove" the redundancy …

Function: derepFastq

https://www.rdocumentation.org/packages/dada2/versions/1.0.3/topics/derepFastq

```
derepFs <- derepFastq(filtFs, verbose=TRUE)
derepRs <- derepFastq(filtRs, verbose=TRUE)

names(derepFs) <- sample.names
names(derepRs) <- sample.names
```

VIII-Amplicon Sequence Variant =ASV

Error Corrections.

```
dadaFs <- dada(derepFs, err=errF, multithread=TRUE)
dadaRs <- dada(derepRs, err=errR, multithread=TRUE)
```

IX- Assembly

Function: mergePairs :

https://www.rdocumentation.org/packages/dada2/versions/1.0.3/topics/mergePairs

→ Merge the forwards & reverses together

→ maxMismatch : how many mismatch do you accept in the overlapping region?

```
mergers <- mergePairs(dadaFs, derepFs, dadaRs, derepRs, maxMismatch=1, verbose=TRUE)
```

X- Table of Amplicon sequence Variants

Function: makeSequenceTable :

https://www.rdocumentation.org/packages/dada2/versions/1.0.3/topics/makeSequenceTable

```
seqtab <- makeSequenceTable(mergers)
```

XI- remove Chimera

Function: removeBimeraDenovo

https://www.rdocumentation.org/packages/dada2/versions/1.0.3/topics/removeBimeraDenovo

```
seqtab.nochim <- removeBimeraDenovo(seqtab, method="consensus", multithread=TRUE, verbose=TRUE)
```

XI – Summarize the pre-processing

```
getN <- function(x) sum(getUniques(x))

summarytab <- data.frame(Samples=sample.names,imput=out[,1],filtered=out[,2],
denoised=sapply(dadaFs, getN),merged=sapply(mergers,
getN),nonchimeric=rowSums(seqtab.nochim),Final_retained=rowSums(seqtab.nochim)/out[
,1]*100)

summarytab
```

XII- Taxonomic Assignment using Dada2

Function "assignTaxonomy": Assignment from Phylum to Genus level
Function "addSpecies": Assignment to the species level if possible (100% identity).

```
taxa <- assignTaxonomy(seqtab.nochim, "silva_nr_v132_train_set.fa", taxLevels=
c("Kingdom", "Phylum","Class", "Order", "Family","Genus", "Species"), multithread=TRUE,
minBoot=60)
taxa <- addSpecies(taxa, "silva_species_assignment_v132.fa.gz",allowMultiple=FALSE)
```

Caution: when using silva_nr_v132_train_set.fa.gz file, you MUST be located in the directory which contains this file (set working directory).

XIII- Phylogenetic Tree

NB: This step can be skipped if you have some problems to perfom it or to many ASVs.

#Get the ASV sequences (equivalent to representative OTUs)

```
seqs<-getSequences(seqtab.nochim)
```

#Alignment by DECIPHER

```
aln <- AlignSeqs(DNAStringSet(seqs), anchor=NA)

#See alignment
BrowseSeqs(aln, highlight=0)
```

#Transformed by Phydat

```
phang.align <- phyDat(as(aln, "matrix"), type="DNA")
```

#Distance Matrix

```
dm <- dist.ml(phang.align)
treeNJ <- NJ(dm)
```

#Change label (leaves) of the tree (important)

```
treeNJ$tip.label<-rownames(taxa)
```

XIV- Build the Phyloseq Object including the phylogenetic tree

#Mapfile
Must contain sample names identically as you defined it at the beginning of the TP
#Check it, if necessary:
names(filtFs)
You need to have correspondence between Mapfile SampleID and name(filtFs). Adapt your Mapfile file if it's not the case.

#Import the mapfile in Rstudio

```
MAP= "mapfileFA.txt"
MAPFILE <-import_qiime_sample_data(MAP)
```

 # create the phyloseq object (OUT_table, Tax_table, tree)

```
Final<- phyloseq(otu_table(seqtab.nochim, taxa_are_rows=FALSE)
                 ,sample_data(MAPFILE),tax_table(taxa),treeNJ)
```

XV- Add the ASV sequences to your Phyloseq object

```
dna      <- DNAStringSet(taxa_names(Final))
names(dna) <- taxa_names(Final)
```

#Add the Class sequence (refseq) to the Object Final1

```
Final1    <- merge_phyloseq(Final, dna)
```

 # see

```
Final1
```

#Change name in ASV
#see

```
taxa_names(Final1)
```

#The name of ASV is the sequence!!! You need to change this to have ASV1, ASV2 etc

```
taxa_names(Final1) <- paste0("ASV", seq(ntaxa(Final1)))
```

#see

```
taxa_names(Final1)
```

#See your final object for the next analyses

Final1