ADM_2024_plastique

2024-01-11

Bonjour Monsieur. Voici mon TD sur mon article. Je n'ai pas pu utiliser ma VM j'ai donc tout fait depuis mon Rstudio en local, ce qui m'a vraiment permis de comprendre le code et les chemins d'accés etc. J'ai utiliser la pipeline de dada 2 classique du coup : https://benjjneb.github.io/dada2/tutorial.html. Cependant, pas mal de codes prenaient enormement de temps à run. Si j'avais eu l'idée de faire en local plus tôt, j'aurai pu, je pense, poursuivre sur phyloseq avec ma table d'ASV (bien que petite) et faire l'alpha et beta diversité. Je sais bien que c'est le plus interessant, je suis moi même un peu triste de ne pas avoir le temps de poursuivre pour voir les figures que j'aurai obtenues... Mais je vais faire tout ça pendant mon stage et je pense que ça va être très formateur.

La première étape est de charger tous les packages : dada2, ggplot2, BioManager etc

```
library(dada2)
```

```
## Le chargement a nécessité le package : Rcpp
```

On définit le chemin pour aller aux Fastas zippés.

```
path_to_fastqs <- "/stage/DADA2/dezippes"
```

On vérifie qu'on a tout en listant les fichiers.

```
list.files(path_to_fastqs)
```

```
## [1] "ERR4008926_1.fastq" "ERR4008926_2.fastq" "ERR4008927_1.fastq"
## [4] "ERR4008927_2.fastq" "ERR4008928_1.fastq" "ERR4008928_2.fastq"
## [7] "ERR4008929_1.fastq" "ERR4008929_2.fastq" "filtered"
```

On met dans la variable fnFs les forward et on verifie le chemin d'accés.

```
fnFs <- sort(list.files(path_to_fastqs,
                        pattern = "_1.fastq",
                        full.names = TRUE))
fnFs
```

```
## [1] "/stage/DADA2/dezippes/ERR4008926_1.fastq"
## [2] "/stage/DADA2/dezippes/ERR4008927_1.fastq"
## [3] "/stage/DADA2/dezippes/ERR4008928_1.fastq"
## [4] "/stage/DADA2/dezippes/ERR4008929_1.fastq"
```

On met dans la variable fnRs les reverse.

```
fnRs <- sort(list.files(path_to_fastqs,
                        pattern = "_2.fastq",
                        full.names = TRUE))
```

On divise la chaîne de carac selon _

```
sample.names <- basename(fnFs) |>
  strsplit(split = "_") |>
  sapply(head, 1) # appliquer une fonction à chaque éléments d'une liste
sample.names
```

```
## [1] "ERR4008926" "ERR4008927" "ERR4008928" "ERR4008929"
```

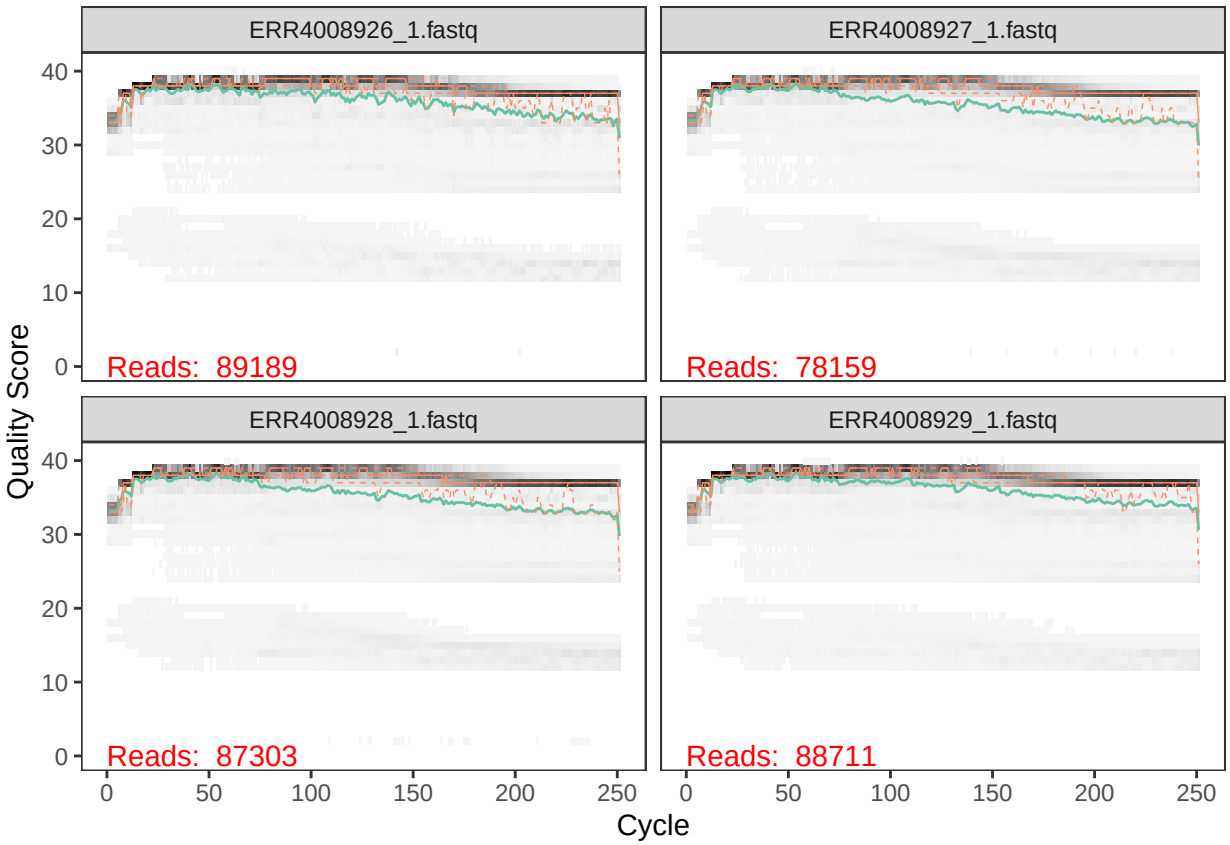
On sépare les noms des fichiers en deux vecteurs.

```
basename(fnFs) |>
  strsplit(split = "_") |>
  head()
```

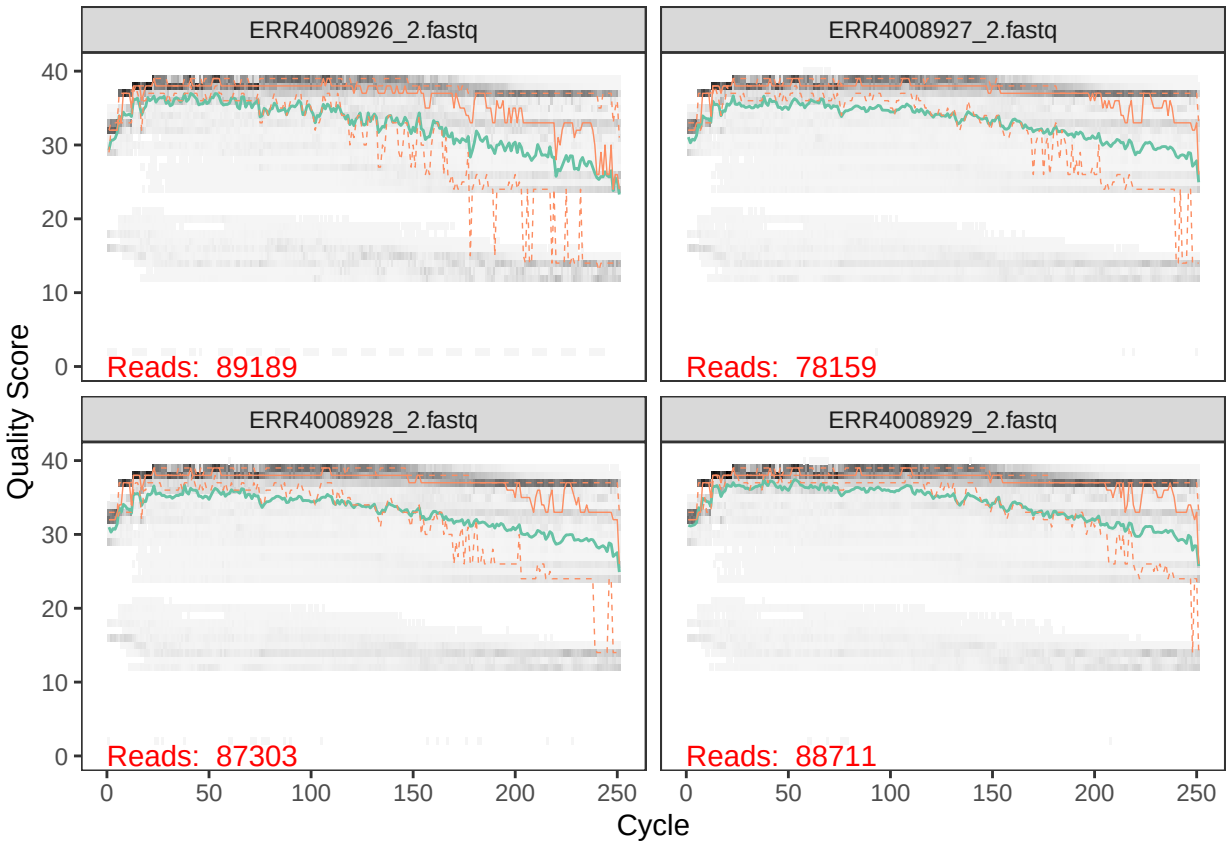
```
## [[1]]
## [1] "ERR4008926" "1.fastq"
##
## [[2]]
## [1] "ERR4008927" "1.fastq"
##
## [[3]]
## [1] "ERR4008928" "1.fastq"
##
## [[4]]
## [1] "ERR4008929" "1.fastq"
```

On montre les quality profils. On observe un QS très bon pour les forward, alors que pour les reverse on observe vers 200 nucléotides que le QS passe en dessous de 30 alors on coupera à ce niveau là. (QS 30 = 1 chance sur 1000 d'avoir la mauvaise base nucléotidique)

```
plotQualityProfile(fnFs)
```




```
plotQualityProfile(fnRs)
```




```
filtFs <- file.path(path_to_fastqs, "filtered", paste0(sample.names, "_1_filt.fastq"))
filtRs <- file.path(path_to_fastqs, "filtered", paste0(sample.names, "_2_filt.fastq"))
names(filtFs) <- sample.names
names(filtRs) <- sample.names
```

On coupe la fin des séquences après avoir vu que les séquences étaient d'assez bonne qualité. Les choix de ce qu'on garde sont de 240 pb pour les forward. Les reverses étaient de moins bonnes qualités vers la fin des amplifications alors on ne garde que 200 pb. Il faut faire attention à garder assez de pb pour les aligner ensuite.

```
out <- filterAndTrim(fnFs, filtFs, fnRs, filtRs, trimLeft = c(20,20), truncLen=c(240,200),
                     maxN=0, maxEE = c(2,2),truncQ=2, rm.phix=TRUE,
                     compress=FALSE, multithread=FALSE)
head(out)
```

```
##                       reads.in reads.out
## ERR4008926_1.fastq    89189     71891
## ERR4008927_1.fastq    78159     64568
## ERR4008928_1.fastq    87303     71185
## ERR4008929_1.fastq    88711     77349
```

On fait apprendre le modèle d'erreurs à partir de nos séquences forward.

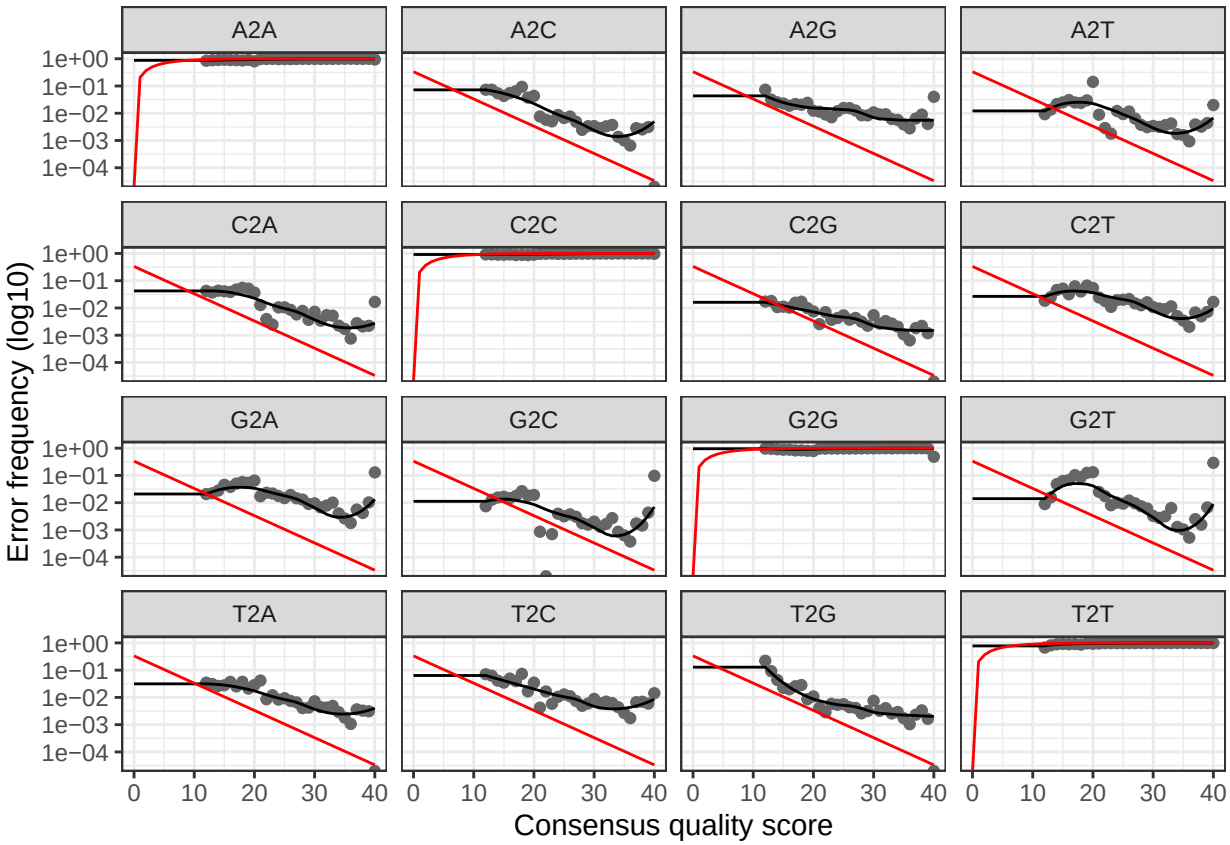
```
errF <- learnErrors(filtFs, multithread=FALSE)
```

```
## 62698460 total bases in 284993 reads from 4 samples will be used for learning the error rates.
```

On montre le graphique.

```
plotErrors(errF, nominalQ=TRUE)
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
## Transformation introduced infinite values in continuous y-axis
```



On apprend le modèle d'erreurs à partir de nos séquences reverse.

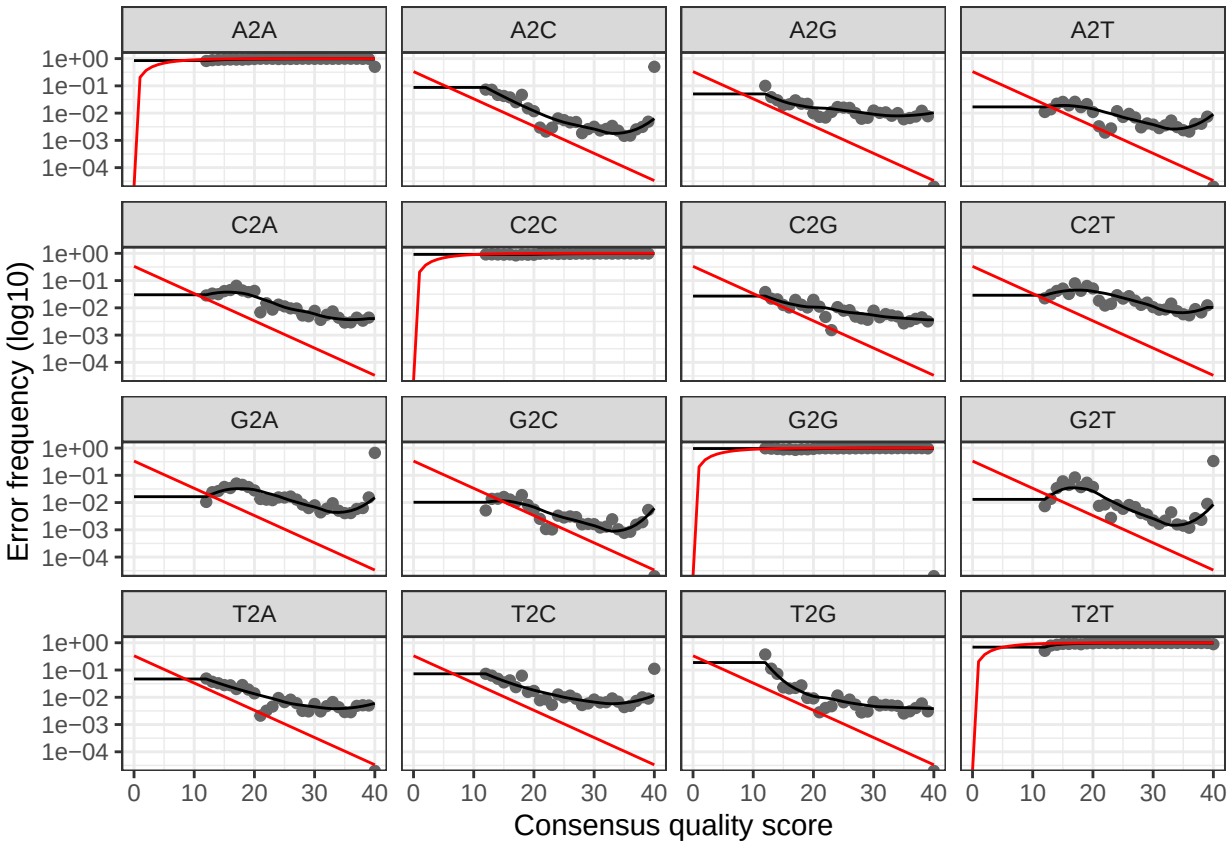
```
errR <- learnErrors(filtRs, multithread = FALSE)
```

```
## 51298740 total bases in 284993 reads from 4 samples will be used for learning the error rates.
```

On montre le graphique.

```
plotErrors(errR, nominalQ=TRUE)
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
## Transformation introduced infinite values in continuous y-axis
```




```
dadaFs <- dada(filtFs, err=errF, multithread=FALSE)
```

```
## Sample 1 - 71891 reads in 17907 unique sequences.
## Sample 2 - 64568 reads in 19958 unique sequences.
## Sample 3 - 71185 reads in 24658 unique sequences.
## Sample 4 - 77349 reads in 25055 unique sequences.
```

```
dadaRs <- dada(filtRs, err=errR, multithread=FALSE)
```

```
## Sample 1 - 71891 reads in 26562 unique sequences.
## Sample 2 - 64568 reads in 26023 unique sequences.
## Sample 3 - 71185 reads in 31072 unique sequences.
## Sample 4 - 77349 reads in 29626 unique sequences.
```

```
dadaFs[[1]]
```

```
## dada-class: object describing DADA2 denoising results
## 1510 sequence variants were inferred from 17907 input unique sequences.
## Key parameters: OMEGA_A = 1e-40, OMEGA_C = 1e-40, BAND_SIZE = 16
```

On merge les séquences qui sont trimées et filtrées.

```
mergers <- mergePairs(dadaFs, filtFs, dadaRs, filtRs, verbose=TRUE)
```

```
## 59912 paired-reads (in 1694 unique pairings) successfully merged out of 66738 (in 4800 pairings) inpu

## 46907 paired-reads (in 2601 unique pairings) successfully merged out of 57379 (in 9396 pairings) inpu

## 51042 paired-reads (in 3594 unique pairings) successfully merged out of 62098 (in 10807 pairings) inp

## 61430 paired-reads (in 4260 unique pairings) successfully merged out of 68584 (in 7913 pairings) inpu
```

```
head(mergers[[1]])
```

```
##
## 1 ACGGAGGGTGCGAGCGTTAATCGGAATTACTGGGCGTAAAGCGCACGCAGGCGGTTTGTTAAGCTAGATGTGAAAGCCCCGAGCTCAACTTGGGATGGT
## 2 TAACCTTGCGGCCGTACTCCCCAGGCGGTCTACTTATCGCGTTAGCTTCGCTACTCACGGCTTAAAGCCACAAACAGCTAGTAGACAGCGTTTACGGTC
## 3 ACGGAGGGTGCAAGCGTTAATCGGAATTACTGGGCGTAAAGCGCGCGTAGGCGGATTTGTCAGTCAGATGTGAAAGCCCTGGGCTCAACCTAGGAATTC
## 4 TAACCTTGCGGCCGTACTCCCCAGGCGGTCTACTTAGTGCGTTAGCTGCGCCACTAAAGAATCAAGTTCCCCAACGGCTAGTAGACATCGTTTACGGCG
## 5 ACGGAGGGTGCGAGCGTTAATCGGAATTACTGGGCGTAAAGCGCACGCAGGCGGTTTGTTAAGCTAGATGTGAAAGCCCCGAGCTCAACTTGGGATGGT
## 6 ACGAGGGGTGCAAGCGTTAATCGGAATTACTGGGCGTAAAGCGCGCGTAGGCGGTTATTTAAGCCAGATGTGAAATCCCCGGGCTCAACCTGGGAACTG
##   abundance forward reverse nmatch nmismatch nindel prefer accept
## 1      6258       1       1     28         0      0      1   TRUE
## 2      4767       2       2     28         0      0      1   TRUE
## 3      2576       3       4     28         0      0      1   TRUE
## 4      2572       4       3     28         0      0      1   TRUE
## 5      1552       1       5     28         0      0      1   TRUE
## 6      1529       6       9     28         0      0      1   TRUE
```

On crée un tableur avec ces séquences mergées et on donne sa dimensions pour voir combien on en a.

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

```
## [1]    4 9175
```

```
table(nchar(getSequences(seqtab)))
```

```
##
##  220  222  225  228  229  232  233  234  236  237  240  241  242  243  244  246
##   16    1    1    1    2    1    1    1    4    2    4    1    1    2    2    1
##  248  249  253  258  261  265  267  268  271  273  274  275  276  277  281  285
##   46    1    1    1    1    1    1    3    1    1    1    1    2    1    1    1
##  286  287  290  292  293  295  296  297  303  304  306  309  310  312  313  314
##    1    2    4    1    1    1    1    1    2    3    5    1    1    1    2    1
##  316  322  323  324  325  328  329  330  333  334  335  336  337  338  339  340
##    1    1    1    2    2    2    1    1    1    1    1    2    4    3    1    2
##  343  345  346  348  349  350  353  354  355  356  357  358  359  360  361  362
##    3    2    4    2    4    2    6    3    4    5    6    6    4    6   17    9
##  363  364  365  366  367  368  369  370  371  372  373  374  375  376  377  378
##    8   18   30   28  131  137  924  263 1495 1934 3034  603  131   99   43    8
##  379  380  381  382  383  385
##   14    9   14    2    6    2
```

On enlève les chimères du tableur, c'est à dire les séquences qu'on pense être faussement séquencées et faussement comptées comme des ASV.

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

```
## Identified 3052 bimeras out of 9175 input sequences.
```

```
dim(seqtab.nochim)
```

```
## [1]    4 6123
```

On ajoute les noms aux colonnes et aux lignes.

```
getN <- function(x) sum(getUniques(x))
track <- cbind(out, sapply(dadaFs, getN), sapply(dadaRs, getN), sapply(mergers, getN), rowSums(seqtab.n
colnames(track) <- c("input", "filtered", "denoisedF", "denoisedR", "merged", "nonchim")
rownames(track) <- sample.names
head(track)
```

```
##            input filtered denoisedF denoisedR merged nonchim
## ERR4008926 89189    71891     68737     68865  59912   54216
## ERR4008927 78159    64568     60415     60385  46907   43967
## ERR4008928 87303    71185     65840     66145  51042   48523
## ERR4008929 88711    77349     72266     72206  61430   58964
```

On assigne la taxonomie depuis la base de données Silva 138 qui a été téléchargée en local.

```
taxa <- assignTaxonomy(seqtab.nochim, "/stage/DADA2/silva_nr99_v138.1_train_set.fa.gz", multithread=FALS
```

On assigne la taxonomie au niveau de l'espèce qui ne se fait pas systématiquement et prend énormement de temps. J'ai trouvé l'explication dans une vidéo d'explication de la pipeline de dada2 mais de 2018 : https://www.youtube.com/watch?v=wV5_z7rR6yw . Les régions amplifiées du 16S sont les régions hypervariables. Parfois elles ne sont pas "hyper" hypervariables et donc cela ne permet pas d'obtenir une assignation a l'espèce.

```
taxa <- addSpecies(taxa, "/stage/DADA2/silva_species_assignment_v138.1.fa.gz")
```

On lie les noms des lignes qui étaient des chiffres aux noms des ASV trouvées dans le tableau d'ASV.

```
taxa.print <- taxa
rownames(taxa.print) <- NULL
head(taxa.print)
```

```
##      Kingdom    Phylum           Class                 Order
## [1,] "Bacteria" "Proteobacteria" "Gammaproteobacteria" "Enterobacterales"
## [2,] "Bacteria" NA               NA                    NA
## [3,] "Bacteria" "Bacteroidota"   "Bacteroidia"         "Flavobacteriales"
## [4,] "Bacteria" "Proteobacteria" "Gammaproteobacteria" "Pseudomonadales"
## [5,] "Bacteria" NA               NA                    NA
## [6,] "Bacteria" NA               NA                    NA
```

```
##      Family                Genus          Species
## [1,] "Alteromonadaceae"    "Alteromonas"  "genovensis"
## [2,] NA                    NA             NA
## [3,] "Cryomorphaceae"      "Luteibaculum" NA
## [4,] "Spongiibacteraceae"  "Zhongshania"  NA
## [5,] NA                    NA             NA
## [6,] NA                    NA             NA
```

On l'exporte.

```
export_folder <- ("export")

if (!dir.exists(export_folder)) dir.create(export_folder, recursive = TRUE)

saveRDS(object = seqtab.nochim,
        file = file.path(export_folder, "seqtab.nochim.rds"))

saveRDS(object = taxa.print,
        file = file.path(export_folder, "taxa.rds"))
```

On "l'écrit" en csv pour pouvoir l'utiliser dans phyloseq.

```
write.csv(taxa.print, "tableASVfinale.csv")
```