CC2_ADM

Chloé Eudier

2026-01-12

Introduction

Dans le cadre d'un programme de préservation des populations tortues vertes (Chelonia mydas) peuplant la mer de Chine du Sud, il a été envisagé d'introduire des tortues élevées en captivité au sein du milieu sauvage. Le microbiote des tortues étant toutefois énormément influencé par les conditions d'évolution de celles-ci, des chercheurs ont analysé et comparé le microbiote de tortues sauvages et issues d'environnement artificiel. Cela avait pour objectif de vérifier que ces tortues ne risquent pas de représenter un danger pour les tortues sauvages dans le cas d'une introduction dans la nature.

Matériel et Méthodes

Des échantillons provenant de 14 tortues élevées en captivité et de 6 tortues sauvages ont été récoltés avant extraction de l'ADN puis l'amplification de la région variable V3-V4 de l'ARNr 16S des bactéries s'y trouvant et enfin le séquençage des amplicons obtenus après purification.

Les données de séquençage doivent ensuite être traitées afin de pouvoir être utilisées pour effectuer des analyses statistiques et répondre à la problématique.

Traitement des données et résultats/discussion

Chargement des données

Chargement de la librarie dada2 pour utiliser les commandes nécessaires à l'analyse de données de séquençage.

```
library(dada2)
```

```
## Loading required package: Rcpp
```

```
packageVersion("dada2")
```

```
## [1] '1.28.0'
```

Chargement des données et création de la variable path pour indiquer le chemin d'accès aux lectures utilisées.

```
path<-"~/ADM-CC2/DATA"
list.files(path)
```

```
##  [1] "filtered"               "SRR28441889_1.fastq.gz" "SRR28441889_2.fastq.gz"
##  [4] "SRR28441890_1.fastq.gz" "SRR28441890_2.fastq.gz" "SRR28441891_1.fastq.gz"
##  [7] "SRR28441891_2.fastq.gz" "SRR28441892_1.fastq.gz" "SRR28441892_2.fastq.gz"
## [10] "SRR28441893_1.fastq.gz" "SRR28441893_2.fastq.gz" "SRR28441894_1.fastq.gz"
## [13] "SRR28441894_2.fastq.gz" "SRR28441895_1.fastq.gz" "SRR28441895_2.fastq.gz"
## [16] "SRR28441896_1.fastq.gz" "SRR28441896_2.fastq.gz" "SRR28441897_1.fastq.gz"
## [19] "SRR28441897_2.fastq.gz" "SRR28441898_1.fastq.gz" "SRR28441898_2.fastq.gz"
## [22] "SRR28441899_1.fastq.gz" "SRR28441899_2.fastq.gz" "SRR28441900_1.fastq.gz"
## [25] "SRR28441900_2.fastq.gz" "SRR28441901_1.fastq.gz" "SRR28441901_2.fastq.gz"
## [28] "SRR28441902_1.fastq.gz" "SRR28441902_2.fastq.gz" "SRR28441903_1.fastq.gz"
## [31] "SRR28441903_2.fastq.gz" "SRR28441904_1.fastq.gz" "SRR28441904_2.fastq.gz"
## [34] "SRR28441905_1.fastq.gz" "SRR28441905_2.fastq.gz" "SRR28441906_1.fastq.gz"
## [37] "SRR28441906_2.fastq.gz" "SRR28441907_1.fastq.gz" "SRR28441907_2.fastq.gz"
## [40] "SRR28441908_1.fastq.gz" "SRR28441908_2.fastq.gz"
```

Lecture du nom des fichiers et création d'une liste des lectures forward (fnFS) et reverse (fnRs) par la manipulation de leur chaîne de caractères variables.

```
fnFs <- sort(list.files(path, pattern="_1.fastq", full.names = TRUE))
fnRs <- sort(list.files(path, pattern="_2.fastq", full.names = TRUE))
```

Création d'une variable afin d'extraire le nom des échantillons auxquels appartiennent les couples de lectures.

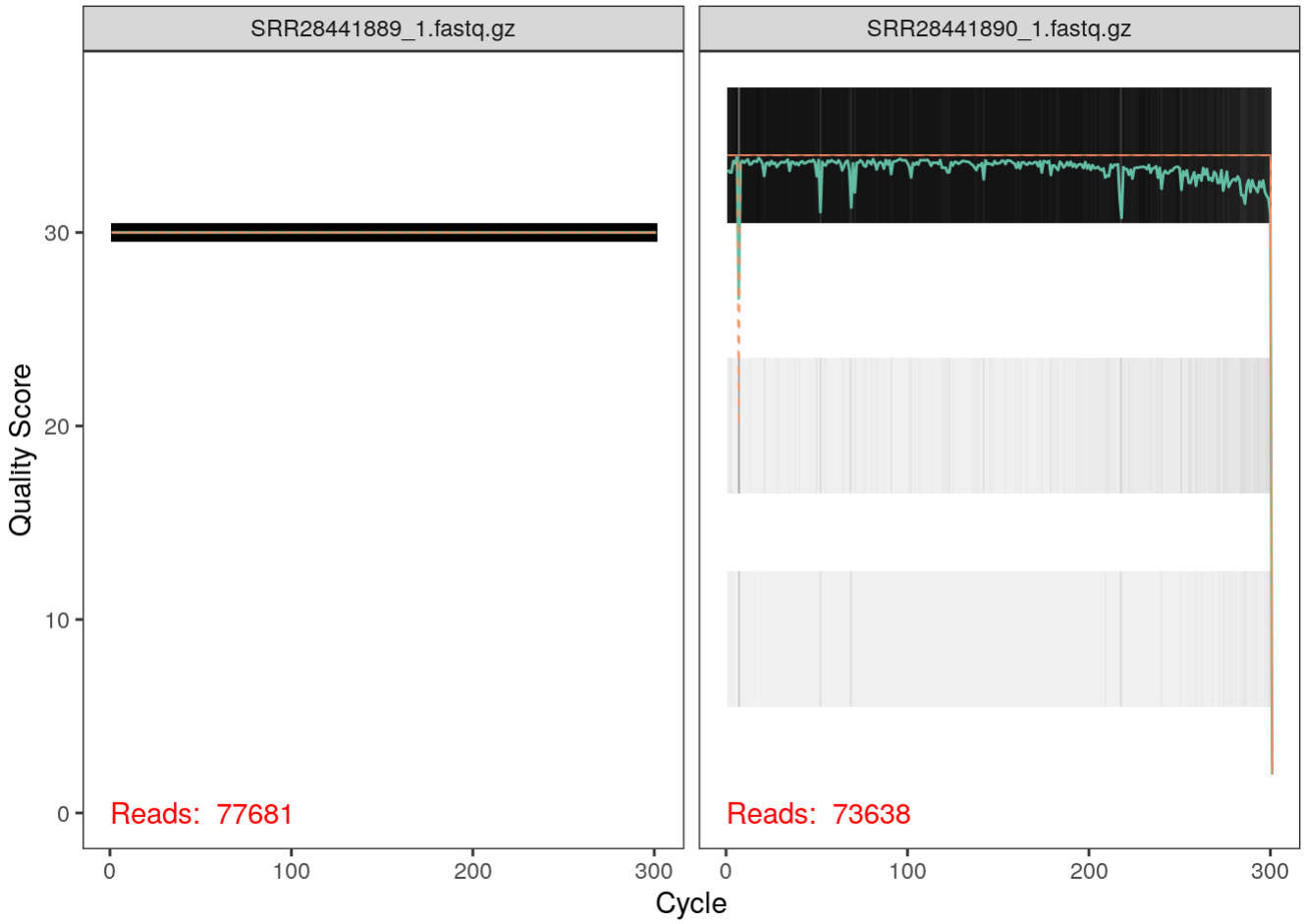
```
sample.names <- sapply(strsplit(basename(fnFs), "_"), `[`, 1)
```

Etude des profils de qualité

Visualisation de la qualité de chaque séquence forward (fwd) grâce au score qualité de chaque nucléotide la composant.

```
plotQualityProfile(fnFs[1:2])
```

```
## Warning: Removed 5 rows containing missing values or values outside the scale range
## (`geom_tile()`).
```



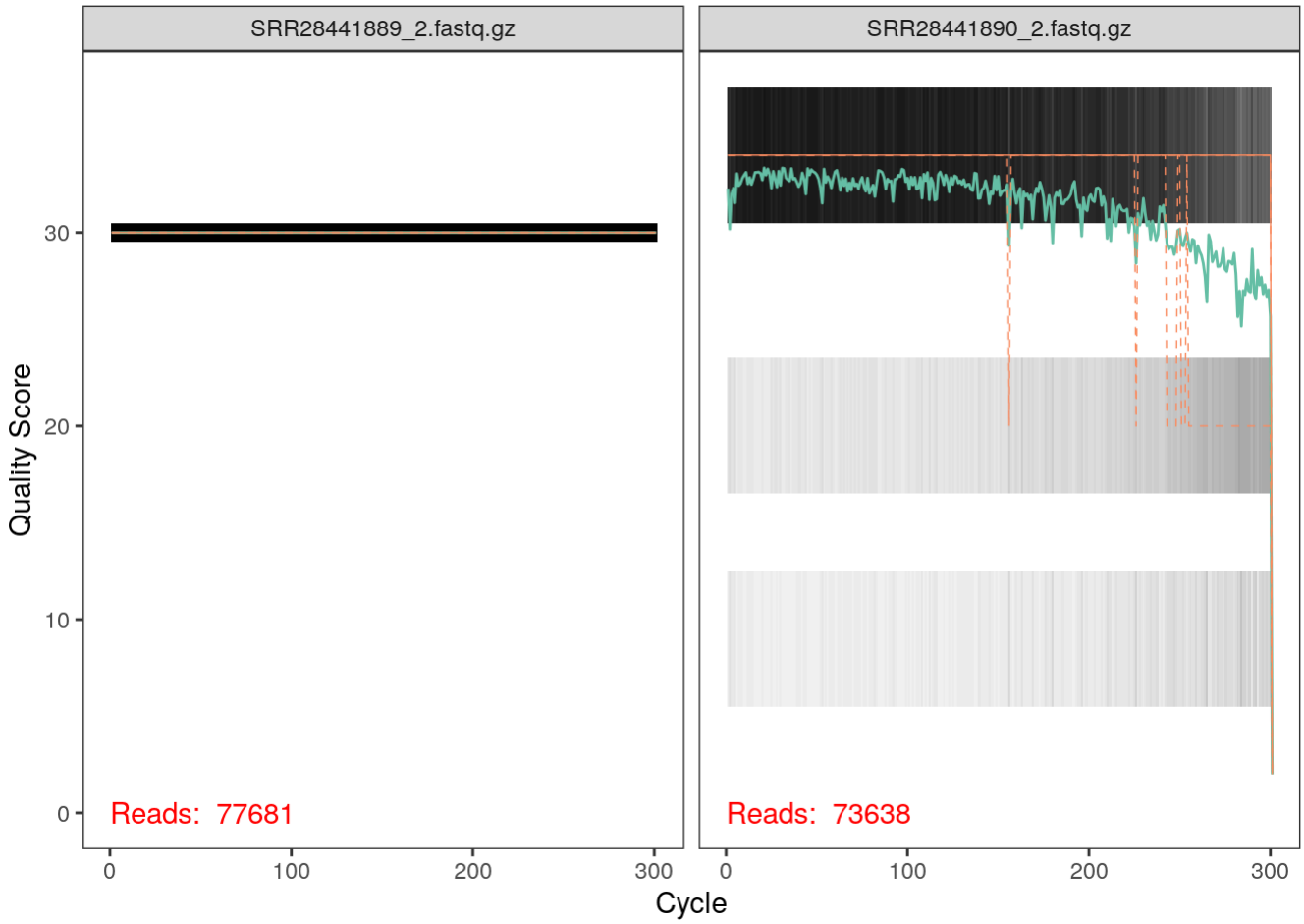
On obtient une heat map nous indiquant la fréquence de chaque score de qualité pour chaque position. Sur les graphiques on observe le score de qualité moyen par la ligne verte, les quartiles de la distributions des scores par les lignes oranges et la proportion des lectures allant jusqu'à au moins cette position.

On peut observer que les séquences fwd sont de bonnes qualité en regardant la ligne verte et les lignes oranges qui se maintiennent à un score entre 30 et 40. La qualité diminue cependant sur les dernières bases des séquences donc il faudra couper les 30 derniers nucléotides des séquences (à partir de la position 270) pour augmenter le score et garder un maximum d'échantillons lors du filtrage.

Visualisation de la qualité de chaque séquence reverse (rs) grâce au score qualité de chaque nucléotide la composant.

```
plotQualityProfile(fnRs[1:2])
```

```
## Warning: Removed 58 rows containing missing values or values outside the scale range
## (`geom_tile()`).
```



En revanche, on observe que les séquences rs sont de moins bonne qualité. Il faudra les couper à partir de la position 220 qui est celle à partir de laquelle la qualité des séquences chute.

Cela permet de ne pas sse retrouver avec des échantillons n'ayant aucune reads une fois les prochaines étapes effectuées.

Tronquage et Filtrage

Création d'objets (filtFs et filtRs) servant à stocker les séquences filtrées et dont les noms sont rattachés à ceux des échantillons.

```
filtFs <- file.path(path, "filtered", paste0(sample.names, "_F_filt.fastq.gz"))
filtRs <- file.path(path, "filtered", paste0(sample.names, "_R_filt.fastq.gz"))
names(filtFs) <- sample.names
names(filtRs) <- sample.names
```

Filtrage des données en jouant sur différents paramètres de sorte à obtenir un meilleur score de qualité de séquence.

```
out <- filterAndTrim(fnFs, filtFs, fnRs, filtRs, truncLen = c(270,220),
              trimLeft = c(0,0), maxN=0, maxEE=c(2,5), truncQ=2, rm.phix=TRUE,
              compress=TRUE, multithread=FALSE)
head(out)
```

```
##                         reads.in reads.out
## SRR28441889_1.fastq.gz    77681     70597
## SRR28441890_1.fastq.gz    73638     64319
## SRR28441891_1.fastq.gz    73528     64660
## SRR28441892_1.fastq.gz    76363     66635
## SRR28441893_1.fastq.gz    65368     56724
## SRR28441894_1.fastq.gz    73730     67078
```

Grâce à cet outil, une petite partie des séquences a été retirée pour augmenter le score de qualité de celles-ci. Donc comme expliqué précédemment, les séquences fwd ont été coupées sur les 30 derniers nucléotides et les rs sur les 80 derniers car le score de qualité des bases a chuté bien avant. On obtient ainsi la liste des lectures avant et après filtrage.

Aprentissage des taux d'erreurs

Estimation du nombre d'erreurs de séquençage dans les échantillons pour différencier les séquences mutées de celles erronées. Le modèle d'erreurs est calculé en alternant l'estimation des taux d'erreur et l'inférence de la composition de l'échantillon jusqu'à ce qu'ils convergent vers une solution cohérente.

Nombre de bases utilisées par le modèle pour les séquences fwd

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

```
## 105303510 total bases in 390013 reads from 6 samples will be used for learning the error r
ates.
```

Nombre de bases utilisées par le modèle pour les séquences rs

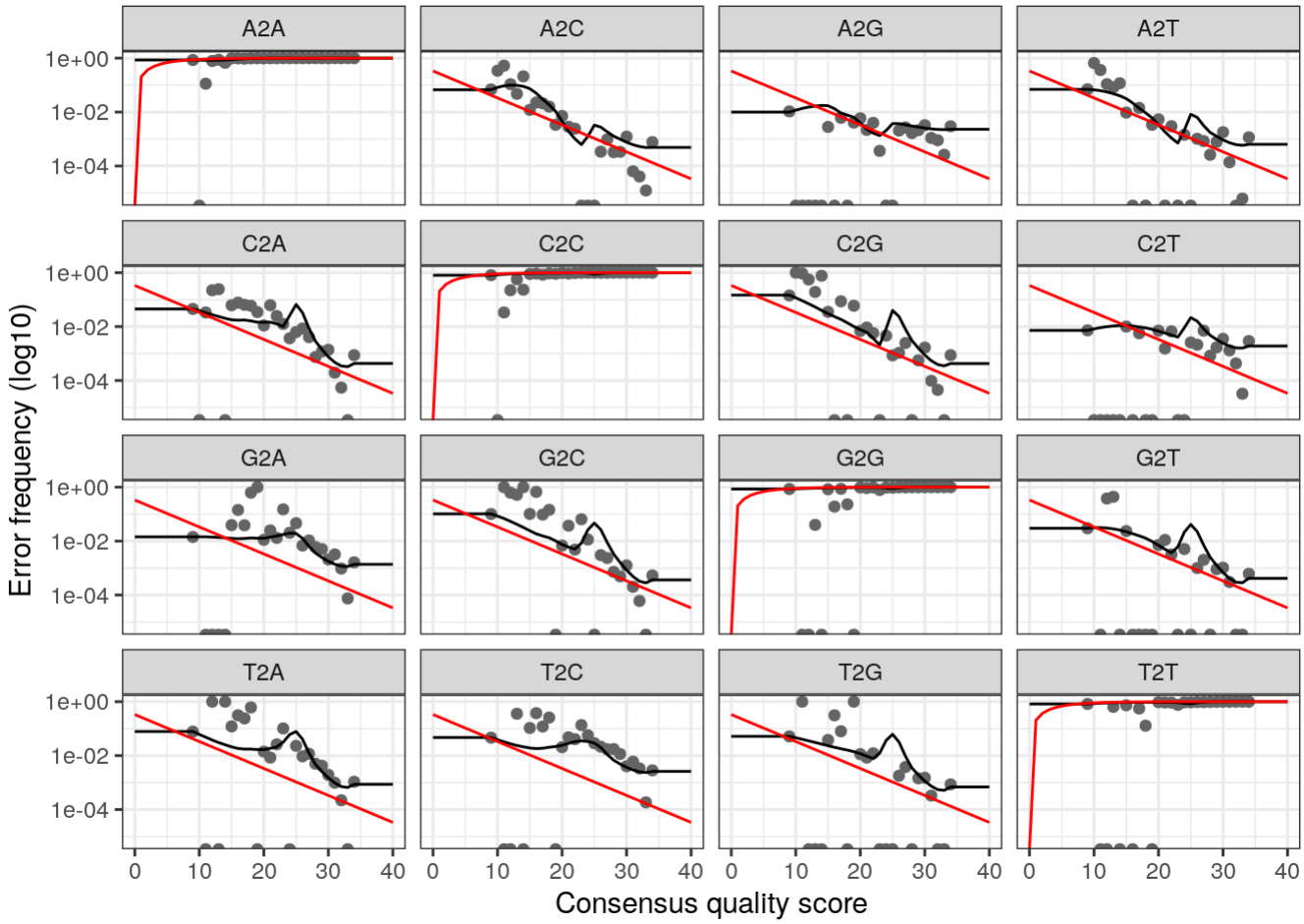
```
errR <- learnErrors(filtRs, multithread=TRUE)
```

```
## 114366780 total bases in 519849 reads from 8 samples will be used for learning the error r
ates.
```

Visualisation des erreurs estimées (pour chaque transition de base possible).

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

```
## Warning in scale_y_log10(): log-10 transformation introduced infinite values.
## log-10 transformation introduced infinite values.
```



Les taux d'erreurs observés pour chaque score de qualité sont représentés par un point noir. La ligne noire indique les taux d'erreur estimés après convergence de l'algorithme d'apprentissage automatique. La ligne rouge indique les taux d'erreur attendus selon la définition nominale du score de qualité.

Comme il est possible de le voir sur les graphiques, certains taux d'erreurs estimés correspondent aux taux observés car les points noirs suivent la ligne rouge pour quelques transitions. Cependant pour d'autres transitions il y'a beaucoup moins de correspondance puisqu'on observe que les points sont dispersés et que la ligne noire ne suit pas tout à fait à la rouge. Mais dans l'ensemble on peut dire que les taux d'erreurs diminuent avec l'augmentation de la qualité.

Inférence des échantillons

Détection des inférences dans les séquences filtrées afin de différencier les vraies séquences biologiques (variants biologiques) des erreurs de séquençage en utilisant un modèle d'erreurs paramétrique.

Inférences dans les séquences fwd filtrées

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

```
## Sample 1 - 70597 reads in 19440 unique sequences.
## Sample 2 - 64319 reads in 12041 unique sequences.
## Sample 3 - 64660 reads in 13320 unique sequences.
## Sample 4 - 66635 reads in 14441 unique sequences.
## Sample 5 - 56724 reads in 18769 unique sequences.
## Sample 6 - 67078 reads in 22668 unique sequences.
## Sample 7 - 63797 reads in 19753 unique sequences.
## Sample 8 - 66039 reads in 18528 unique sequences.
## Sample 9 - 51229 reads in 10243 unique sequences.
## Sample 10 - 49062 reads in 8419 unique sequences.
## Sample 11 - 59465 reads in 13567 unique sequences.
## Sample 12 - 58351 reads in 12351 unique sequences.
## Sample 13 - 62737 reads in 16075 unique sequences.
## Sample 14 - 51328 reads in 10118 unique sequences.
## Sample 15 - 65703 reads in 16720 unique sequences.
## Sample 16 - 64341 reads in 19138 unique sequences.
## Sample 17 - 55725 reads in 14435 unique sequences.
## Sample 18 - 67026 reads in 17661 unique sequences.
## Sample 19 - 63320 reads in 19060 unique sequences.
## Sample 20 - 66390 reads in 20287 unique sequences.
```

Inférences dans les séquences rs filtrées

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

```
## Sample 1 - 70597 reads in 29895 unique sequences.
## Sample 2 - 64319 reads in 23200 unique sequences.
## Sample 3 - 64660 reads in 24083 unique sequences.
## Sample 4 - 66635 reads in 25484 unique sequences.
## Sample 5 - 56724 reads in 27166 unique sequences.
## Sample 6 - 67078 reads in 35490 unique sequences.
## Sample 7 - 63797 reads in 29811 unique sequences.
## Sample 8 - 66039 reads in 27627 unique sequences.
## Sample 9 - 51229 reads in 19468 unique sequences.
## Sample 10 - 49062 reads in 16889 unique sequences.
## Sample 11 - 59465 reads in 24394 unique sequences.
## Sample 12 - 58351 reads in 23134 unique sequences.
## Sample 13 - 62737 reads in 27296 unique sequences.
## Sample 14 - 51328 reads in 19376 unique sequences.
## Sample 15 - 65703 reads in 28647 unique sequences.
## Sample 16 - 64341 reads in 29402 unique sequences.
## Sample 17 - 55725 reads in 22709 unique sequences.
## Sample 18 - 67026 reads in 30031 unique sequences.
## Sample 19 - 63320 reads in 27184 unique sequences.
## Sample 20 - 66390 reads in 30527 unique sequences.
```

Evaluation du nombre d’ASVs dans l’échantillon n°1

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

```
## dada-class: object describing DADA2 denoising results
## 887 sequence variants were inferred from 19440 input unique sequences.
## Key parameters: OMEGA_A = 1e-40, OMEGA_C = 1e-40, BAND_SIZE = 16
```

L’algorithme a trouvé 887 variants de séquence réels à partir des 19440 séquences du premier échantillon.

Fusion des séquences paires

Fusion des séquences R1 et R2 pour obtenir les amplicons en entier et observer leur fiabilité. Pour que la fusion se fasse, il faut que les séquences se chevauchent sur au moins 12 nucléotides identiques entre les 2 séquences.

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

```
## 48548 paired-reads (in 4018 unique pairings) successfully merged out of 65941 (in 8590 pai
rings) input.
```

```
## 56517 paired-reads (in 2932 unique pairings) successfully merged out of 62203 (in 4955 pai
rings) input.
```

```
## 53371 paired-reads (in 3195 unique pairings) successfully merged out of 61911 (in 6010 pai
rings) input.
```

```
## 51887 paired-reads (in 3545 unique pairings) successfully merged out of 63844 (in 7185 pai
rings) input.
```

```
## 41060 paired-reads (in 2970 unique pairings) successfully merged out of 52547 (in 7555 pai
rings) input.
```

```
## 41509 paired-reads (in 2921 unique pairings) successfully merged out of 60997 (in 9082 pai
rings) input.
```

```
## 46593 paired-reads (in 4002 unique pairings) successfully merged out of 58890 (in 9395 pai
rings) input.
```

```
## 50302 paired-reads (in 4956 unique pairings) successfully merged out of 61576 (in 9241 pai
rings) input.
```

```
## 4458 paired-reads (in 415 unique pairings) successfully merged out of 49934 (in 5280 pairi
ngs) input.
```

```
## 1936 paired-reads (in 191 unique pairings) successfully merged out of 48059 (in 3723 pairi
ngs) input.
```

```
## 3839 paired-reads (in 390 unique pairings) successfully merged out of 57428 (in 6350 pairi
ngs) input.
```

```
## 31043 paired-reads (in 2050 unique pairings) successfully merged out of 56169 (in 6025 pai
rings) input.
```

```
## 48328 paired-reads (in 5414 unique pairings) successfully merged out of 59660 (in 9031 pai
rings) input.
```

```
## 2797 paired-reads (in 221 unique pairings) successfully merged out of 49937 (in 3356 pairi
ngs) input.
```

```
## 15731 paired-reads (in 1387 unique pairings) successfully merged out of 62792 (in 7567 pai
rings) input.
```

```
## 23850 paired-reads (in 2463 unique pairings) successfully merged out of 60425 (in 10041 pa
irings) input.
```

```
## 44055 paired-reads (in 2946 unique pairings) successfully merged out of 53169 (in 5786 pai
rings) input.
```

```
## 27564 paired-reads (in 1945 unique pairings) successfully merged out of 63481 (in 8246 pai
rings) input.
```

```
## 45245 paired-reads (in 2854 unique pairings) successfully merged out of 59410 (in 7363 pai
rings) input.
```

```
## 48629 paired-reads (in 3391 unique pairings) successfully merged out of 61786 (in 8460 pai
rings) input.
```

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

```
##
sequence
## 1 TCGAAGACTCCTACGGGAGGCAGCAGTGGGGAATCTTAGACAATGGGGGCAACCCTGATCTAGCCATGCCGCGTGAGTGATGAAGGCC
TTAGGGTTGTAAAGCTCTTTCAGCTGGGAAGATAATGACGGTACCAGCAGAAGAAGCCCCGGCTAACTCCGTGCCAGCAGCCGCGGTAATACG
GAGGGGGCTAGCGTTGTTCGGAATTACTGGGCGTAAAGCGCACGTAGGCGGACCAGCAAGTTAGGGGTGAAATCCCGGGGCTCAACCTCGGAA
CTGCCTTTAAAACTGTTGGTCTGGAGTTCGAGAGAGGTGAGTGGAATACCGAGTGTAGAGGTGAAATTCGTAGATATTCGGTGGAACACCAGT
GGCGAAGGCGGCTCACTGGCTCGATACTGACGCTGAGGTGCGAAAGCGTGGGGAGCAAACAGGATTAGAAACCCCAGTAGTCCTCCACT
## 2 TCGAAGACTCCTACGGGAGGCAGCAGTGGGGAATCTTAGACAATGGGGGCAACCCTGATCTAGCCATGCCGCGTGAGTGATGAAGGCC
TTAGGGTTGTAAAGCTCTTTCAGCTGGGAAGATAATGACGGTACCAGCAGAAGAAGCCCCGGCTAACTCCGTGCCAGCAGCCGCGGTAATACG
GAGGGGGCTAGCGTTGTTCGGAATTACTGGGCGTAAAGCGCACGTAGGCGGACCAGCAAGTTAGGGGTGAAATCCCGGGGCTCAACCTCGGAA
CTGCCTTTAAAACTGTTGGTCTGGAGTTCGAGAGAGGTGAGTGGAATACCGAGTGTAGAGGTGAAATTCGTAGATATTCGGTGGAACACCAGT
GGCGAAGGCGGCTCACTGGCTCGATACTGACGCTGAGGTGCGAAAGCGTGGGGAGCAAACAGGATTAGAAACCCCTGTAGTCCTCCACT
## 3 TCGAAGACTCCTACGGGAGGCAGCAGTGGGGAATCTTAGACAATGGGGGCAACCCTGATCTAGCCATGCCGCGTGAGTGATGAAGGCC
TTAGGGTTGTAAAGCTCTTTCAGCTGGGAAGATAATGACGGTACCAGCAGAAGAAGCCCCGGCTAACTCCGTGCCAGCAGCCGCGGTAATACG
GAGGGGGCTAGCGTTGTTCGGAATTACTGGGCGTAAAGCGCACGTAGGCGGACCAGCAAGTTAGGGGTGAAATCCCGGGGCTCAACCTCGGAA
CTGCCTTTAAAACTGTTGGTCTGGAGTTCGAGAGAGGTGAGTGGAATACCGAGTGTAGAGGTGAAATTCGTAGATATTCGGTGGAACACCAGT
GGCGAAGGCGGCTCACTGGCTCGATACTGACGCTGAGGTGCGAAAGCGTGGGGAGCAAACAGGATTAGAAACCCTTGTAGTCCTCCACT
## 4 TCGAAGACTCCTACGGGAGGCAGCAGTGGGGAATCTTAGACAATGGGGGCAACCCTGATCTAGCCATGCCGCGTGAGTGATGAAGGCC
TTAGGGTTGTAAAGCTCTTTCAGCTGGGAAGATAATGACGGTACCAGCAGAAGAAGCCCCGGCTAACTCCGTGCCAGCAGCCGCGGTAATACG
GAGGGGGCTAGCGTTGTTCGGAATTACTGGGCGTAAAGCGCACGTAGGCGGACCAGCAAGTTAGGGGTGAAATCCCGGGGCTCAACCTCGGAA
CTGCCTTTAAAACTGTTGGTCTGGAGTTCGAGAGAGGTGAGTGGAATACCGAGTGTAGAGGTGAAATTCGTAGATATTCGGTGGAACACCAGT
GGCGAAGGCGGCTCACTGGCTCGATACTGACGCTGAGGTGCGAAAGCGTGGGGAGCAAACAGGATTAGAAACCCCGGTAGTCCTCCACT
## 5 TCGAAGACTCCTACGGGAGGCAGCAGTGGGGAATCTTAGACAATGGGGGCAACCCTGATCTAGCCATGCCGCGTGAGTGATGAAGGCC
TTAGGGTTGTAAAGCTCTTTCAGCTGGGAAGATAATGACGGTACCAGCAGAAGAAGCCCCGGCTAACTCCGTGCCAGCAGCCGCGGTAATACG
GAGGGGGCTAGCGTTGTTCGGAATTACTGGGCGTAAAGCGCACGTAGGCGGACCAGCAAGTTAGGGGTGAAATCCCGGGGCTCAACCTCGGAA
CTGCCTTTAAAACTGTTGGTCTGGAGTTCGAGAGAGGTGAGTGGAATACCGAGTGTAGAGGTGAAATTCGTAGATATTCGGTGGAACACCAGT
GGCGAAGGCGGCTCACTGGCTCGATACTGACGCTGAGGTGCGAAAGCGTGGGGAGCAAACAGGATTAGATACCCCAGTAGTCCTCCACT
## 6 TCGAAGACTCCTACGGGAGGCAGCAGTGGGGAATCTTAGACAATGGGGGCAACCCTGATCTAGCCATGCCGCGTGAGTGATGAAGGCC
TTAGGGTTGTAAAGCTCTTTCAGCTGGGAAGATAATGACGGTACCAGCAGAAGAAGCCCCGGCTAACTCCGTGCCAGCAGCCGCGGTAATACG
GAGGGGGCTAGCGTTGTTCGGAATTACTGGGCGTAAAGCGCACGTAGGCGGACCAGCAAGTTAGGGGTGAAATCCCGGGGCTCAACCTCGGAA
CTGCCTTTAAAACTGTTGGTCTGGAGTTCGAGAGAGGTGAGTGGAATACCGAGTGTAGAGGTGAAATTCGTAGATATTCGGTGGAACACCAGT
GGCGAAGGCGGCTCACTGGCTCGATACTGACGCTGAGGTGCGAAAGCGTGGGGAGCAAACAGGATTAGAAACCCTAGTAGTCCTCCACT
##   abundance forward reverse nmatch nmismatch nindel prefer accept
## 1      1649       1       1     34         0      0      1   TRUE
## 2      1324       1       2     34         0      0      1   TRUE
## 3      1124       1       3     34         0      0      1   TRUE
## 4      1022       1       5     34         0      0      1   TRUE
## 5      1005       1       4     34         0      0      1   TRUE
## 6       955       1       7     34         0      0      1   TRUE
```

On obtient ainsi la liste de données pour chaque échantillon qui comprend : la séquence entière, son abondance et les indices de variants de séquence (ASV) fwd et rs qui ont été fusionnées. De plus, les lectures appariées qui ne se chevauchaient pas exactement ont été supprimées pour réduire le risque de résultats erronés.

Création d’une table d’ASV

Construction d’une table d’ASV pour observer l’abondance des séquences dans les échantillons

Evaluation du nombre d’ASV parmi l’ensemble des échantillons

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

```
## [1]    20 52206
```

Etude de la distribution de la longueur des séquences

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

```
##
##   270   276   287   314   369   374   388   389   394   398   413   414   416
##     1     1     1     1     2     1     1     1     1     1     1     3     1
##   427   430   432   435   436   444   445   450   453   454   455   456   457
##     2     1     1     1     1    10     1     2     4    25  2619 26632   849
##   458   459   460   461   462   463   464   465   466   467   468   469   470
##  1335  2403  1301   503   150    23    37     3     2     2     2     9     1
##   472   473   474   475   476   477   478
##     3    45    60  1000 14702   447    14
```

Comme on peut le voir dans le tableau, on obtient 52206 ASVs de différentes longueurs parmi les 20 échantillons. Le nombre de nucléotides dans les séquences est indiqué en haut et le nombre de ses séquences en bas. Par exemple, on trouve une séquence avec 270 nucléotides.

Eliminitaion des ASVs chimériques

Suppression des ASVs chimériques (séquences non biologiques). Les séquences chimériques sont identifiées si elles peuvent être reconstruites exactement en combinant un segment gauche et un segment droit provenant de deux séquences « parentales » plus abondantes.

Identification du nombre d'ASVs chimériques

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

```
## Identified 44021 bimeras out of 52206 input sequences.
```

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

```
## [1]   20 8185
```

Ici,il y'a 44021 ASVs qui sont en réalité des chimères sur les 52206 trouvées précédemment.

Evaluation de la proportion des chimères parmi l'ensemble ASVs

```
sum(seqtab.nochim)/sum(seqtab)
```

```
## [1] 0.5301515
```

Les chimères représentent donc environ 87 % des variants de séquences fusionnées, mais si l'abondance de ces variants est prise en compte, les chimères représentent environ 43 % des lectures de séquence fusionnées puisque 53 % des séquences ont été conservées.

Exportation de la table ASV pour création de l'objet phyloseq

```
asv_table <- t(seqtab.nochim)
write.table(asv_table,
            file="table_ASV.tsv",
            sep="\t",
            quote=FALSE,
            col.names=NA)
```

Tableau de suivi

Observation du nombre de séquences éliminées à chaque étape pour voir si un maximum de lectures présentent au départ ont été conservées

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

```
##             input filtered denoisedF denoisedR merged nonchim
## SRR28441889 77681    70597     67948     68352  48548   26208
## SRR28441890 73638    64319     63031     63374  56517   31330
## SRR28441891 73528    64660     62946     63488  53371   37595
## SRR28441892 76363    66635     64932     65417  51887   29361
## SRR28441893 65368    56724     54430     54576  41060   21968
## SRR28441894 73730    67078     64013     63590  41509   23634
```

Une grande majorité des séquences a été conservée ce qui indique qu'il n'y a pas de problèmes.

Assigniation taxonomique des séquences

Assigniation de taxonomie pour chaque séquences afin d'obtenir des informations sur les différents rangs (Domaine jusqu'à l'espèce si c'est possible) par la comparaison avec des séquences de référence connues

```
taxa <- assignTaxonomy(seqtab.nochim,"~/CC2-ADM/silva_nr99_v138.2_toSpecies_trainset.fa.gz?do
wnload=1", multithread=TRUE)
```

Affichage des résultats de l'assignation

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

```
##      Kingdom    Phylum           Class                Order
## [1,] "Bacteria" "Fusobacteriota" "Fusobacteriia"      "Fusobacteriales"
## [2,] "Bacteria" "Pseudomonadota" "Alphaproteobacteria" "Rhodobacterales"
## [3,] "Bacteria" "Pseudomonadota" "Alphaproteobacteria" "Rhodobacterales"
## [4,] "Bacteria" "Pseudomonadota" "Alphaproteobacteria" "Rhodobacterales"
## [5,] "Bacteria" "Fusobacteriota" "Fusobacteriia"      "Fusobacteriales"
## [6,] "Bacteria" "Pseudomonadota" "Alphaproteobacteria" "Rhodobacterales"
##      Family             Genus           Species
## [1,] "Fusobacteriaceae" "Cetobacterium" NA
## [2,] "Paracoccaceae"    "Paracoccus"    "stylophorae"
## [3,] "Paracoccaceae"    "Paracoccus"    NA
## [4,] "Paracoccaceae"    "Paracoccus"    "stylophorae"
## [5,] "Fusobacteriaceae" "Cetobacterium" NA
## [6,] "Paracoccaceae"    "Paracoccus"    NA
```

La majorité des séquences des différents échantillons ont été assignées au phylum des Pseudomonadota et des Fusobecteriota. Dans le cas où l'outil ne peut pas assigner précisement des séquences à une espèce spécifique, il va seulement l'assigner aux rangs supérieurs tels que le genre ou la famille.

Exportation de la table de taxonomie pour création de l'objet phyloseq

```
write.table(taxa.print,
            "Table_taxo.tsv",
            sep = "\t",
            quote = FALSE,
            col.names = NA)
```

Installation du package phyloseq

```
source("https://raw.githubusercontent.com/joey711/phyloseq/master/inst/scripts/installer.R",
local = TRUE)
```

```
## Installing the release version from BioC
```

```
## 'getOption("repos")' replaces Bioconductor standard repositories, see
## 'help("repositories", package = "BiocManager")' for details.
## Replacement repositories:
##     CRAN: https://packagemanager.posit.co/cran/__linux__/jammy/latest
```

```
## Bioconductor version 3.17 (BiocManager 1.30.27), R 4.3.1 (2023-06-16)
```

```
## Warning: package(s) not installed when version(s) same as or greater than current; use
##   `force = TRUE` to re-install: 'phyloseq'
```

```
## Installation paths not writeable, unable to update packages
##   path: /usr/local/lib/R/library
##   packages:
##     boot, class, cluster, codetools, foreign, KernSmooth, lattice, nlme, nnet,
##     rpart, spatial, survival
```

```
## Old packages: 'forecast', 'FSA', 'gert', 'Hmisc', 'later', 'lifecycle',
##   'modEvA', 'parallelly', 'purrr', 'RcmdrMisc', 'Rcpp', 'reformulas', 'rlang',
##   'testthat', 'tibble', 'tidytree', 'tseries'
```

Chargement de la librairie phyloseq

```
library(phyloseq)
```

Lecture des différentes tables et définition des différents paramètres phyloseq

Table ASV

```
ASV <- read.table("table_ASV.tsv",
                  sep="\t",
                  header=TRUE,
                  row.names=1,
                  check.names=FALSE)
asv <- otu_table(as.matrix(ASV), taxa_are_rows = TRUE)
```

Table de taxonomie

```
tax <- read.table("Table_taxo.tsv",
                  sep="\t",
                  header=TRUE,
                  row.names=1)

tax <- tax_table(as.matrix(tax))
```

Synchonisation des données : on donne les séquences ADN comme noms de lignes à la taxonomie en vérifiant que les tables ont bien le même nombre de lignes

```
if (nrow(asv) == nrow(tax)) {
    rownames(tax) <- rownames(asv)
    print("Succès : Les noms de la taxonomie ont été synchronisés avec les séquences.")
} else {
    stop("Erreur : Le nombre de lignes ne correspond pas entre ASV et Taxonomie.")
}
```

```
## [1] "Succès : Les noms de la taxonomie ont été synchronisés avec les séquences."
```

Table de métadonnées

```
meta <- read.csv("~/ADM-CC2/Métadonnées.csv",
                 sep=";",
                 header=TRUE)

# Correspondance entre la colonne Run et les noms d'échantillons

row.names(meta) <- meta$Run

sample <- sample_data(meta)
```

Construction de l'objet phyloseq

```
ps <- phyloseq(asv, tax, sample)

ps
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 8185 taxa and 20 samples ]
## sample_data() Sample Data:       [ 20 samples by 25 sample variables ]
## tax_table()   Taxonomy Table:    [ 8185 taxa by 7 taxonomic ranks ]
```

Nettoyage : Retirer Mitochondries et Chloroplastes (souvent présents chez les tortues)

```
ps <- subset_taxa(ps, Family != "Mitochondria" & Order != "Chloroplast")
```

Analyses de données

Regroupement des échantillons en 2 groupes : Wild Type (WT) et Artificial (AC)

```
metadata <- data.frame(sample_data(ps))
metadata$Group[grepl("Wild", metadata$Sample.Name, ignore.case = TRUE)] <- "WT"
metadata$Group[grepl("Artificial", metadata$Sample.Name, ignore.case = TRUE)]<-"AC"
meta$Group <- ifelse(grepl("Wild", metadata$Sample.Name, ignore.case = TRUE), "WT", "AC")
sample_data(ps) <- sample_data(metadata)
table(sample_data(ps)$Group)
```

```
##
## AC WT
## 14  6
```

Raréfaction des données nécessaire pour effectuer les analyses

```
set.seed(123)
ps_rar <- rarefy_even_depth(
  ps,
  sample.size = min(sample_sums(ps)),
  rngseed = 123,
  replace = FALSE,
  verbose = FALSE
)
```

Grâce à cette étape, les échantillons ont été classés en 2 groupes de tortues en fonction de leur appelation dans la table de métadonnées (Wild ou Artificial) correspondant aux tortues sur lesquelles ils ont été prélévés. Cela permet de comparer la richesse et la diversité des microbiotes des tortues sauvages et élevées en milieu artificiel.

Indices d'alpha diversité

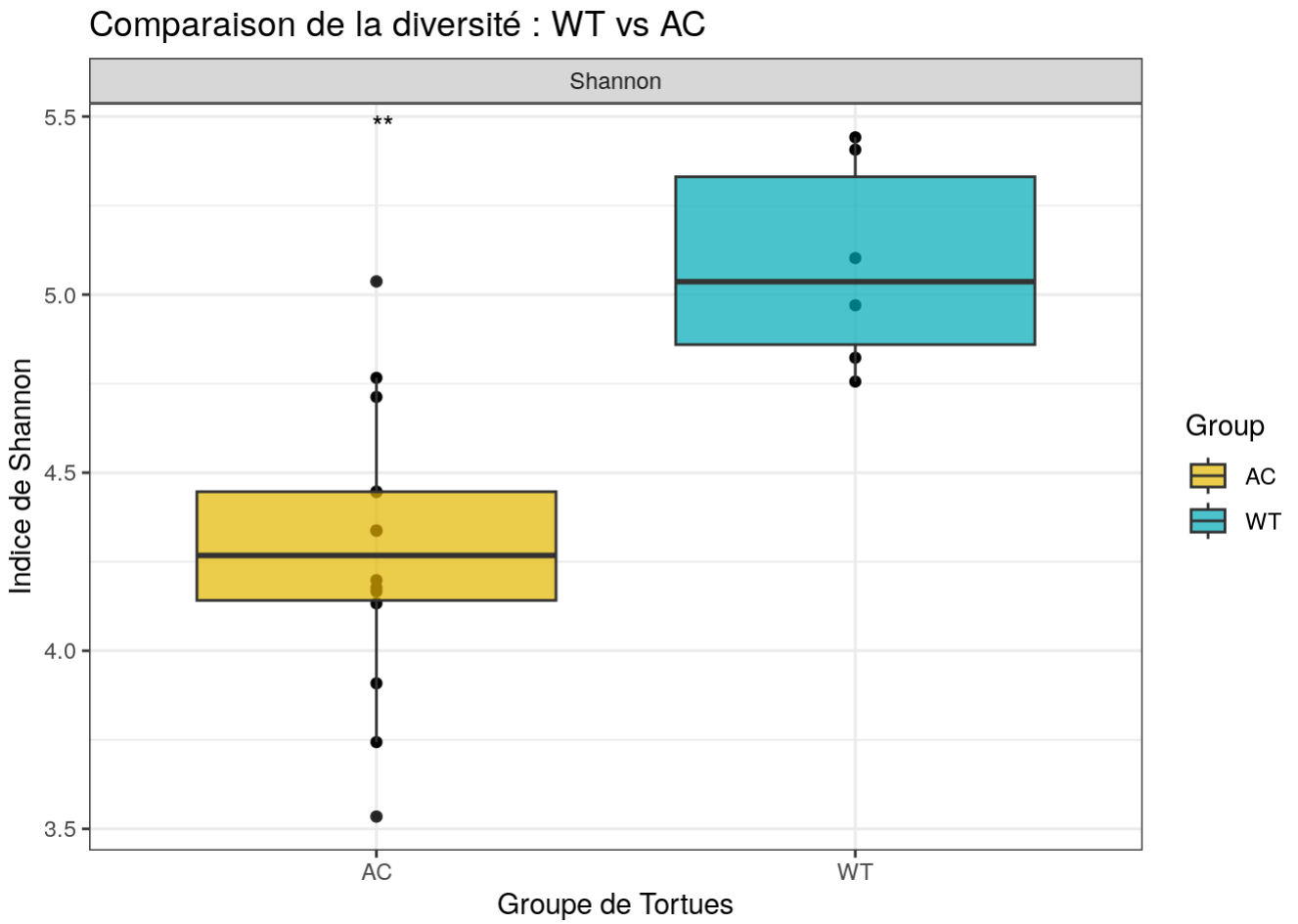
Chargement des librairies nécessaire à la construction de figures

```
library(ggplot2)
library(ggpubr)
```

Indice de Shannon

```
plot_richness(ps_rar, x="Group", measures=c("Shannon")) + # Test de Shannon à parrir des donn
ées raréfiées
  geom_boxplot(aes(fill=Group), alpha=0.7) + # Données sous forme de boite à moustaches
  scale_fill_manual(values = c("WT" = "#00AFBB", "AC" = "#E7B800")) + # Couleurs de chaque gr
oupe
  theme_bw() +
  stat_compare_means(method = "wilcox.test", label = "p.signif") + # Test statistique de l'ar
ticle
  labs(title = "Comparaison de la diversité : WT vs AC", x = "Groupe de Tortues", y = "Indice
de Shannon") # Titre données aux axes
```

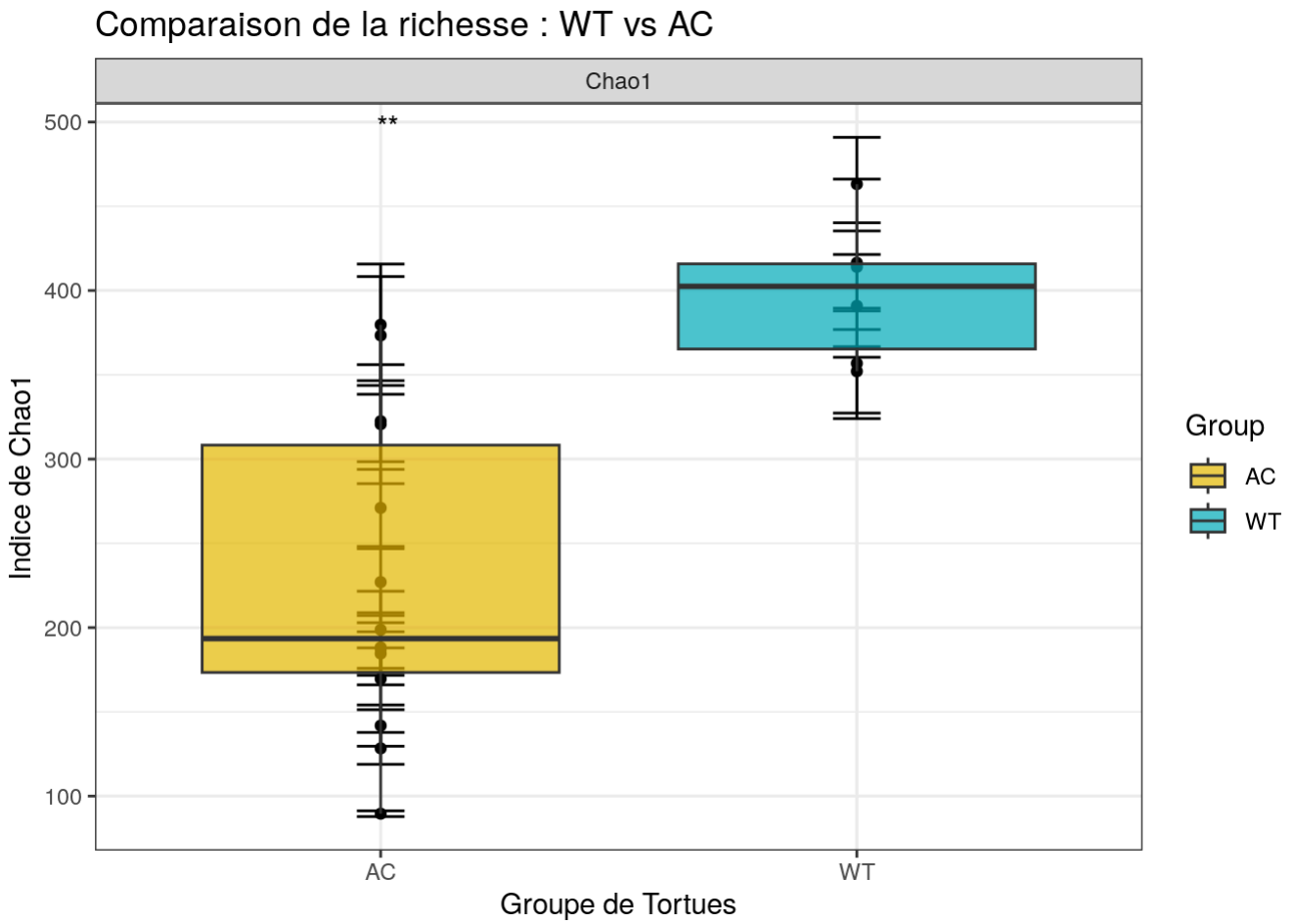
```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## ℹ The deprecated feature was likely used in the phyloseq package.
##   Please report the issue at <https://github.com/joey711/phyloseq/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



Comme on peut le voir sur le graphique ci-dessus, les échantillons du microbiote des tortues élevées en captivité (AC) ont un indice de Shannon plus faible que ceux des tortues sauvages (WT) traduisant une plus faible diversité bactérienne chez celles-ci.

Indice de Chao1

```
plot_richness(ps_rar, x="Group", measures=c("Chao1")) + # Test de Chao1 à parrir des données raréfiées
  geom_boxplot(aes(fill=Group), alpha=0.7) +
  scale_fill_manual(values = c("WT" = "#00AFBB", "AC" = "#E7B800")) +
  theme_bw() +
  stat_compare_means(method = "wilcox.test", label = "p.signif") +
  labs(title = "Comparaison de la richesse : WT vs AC", x = "Groupe de Tortues", y = "Indice de Chao1")
```



On observe la même chose que précédemment, les AC ont un indice de Chao1 plus faible que les WT, ce qui signifie que leur microbiote est moins riche.

Les 2 indices d'alpha diversité permettent donc d'affirmer que les WT ont un microbiote bien plus riche et diversifié et donc plus stable que les AC dont le microbiote semble appauvri et déséquilibré.

Beta diversité

Transformation des données en abondance relative

```
ps_gen <- tax_glom(ps, "Genus")
ps_rel <- transform_sample_counts(ps_gen, function(x) x/sum(x))
ps_rel
```

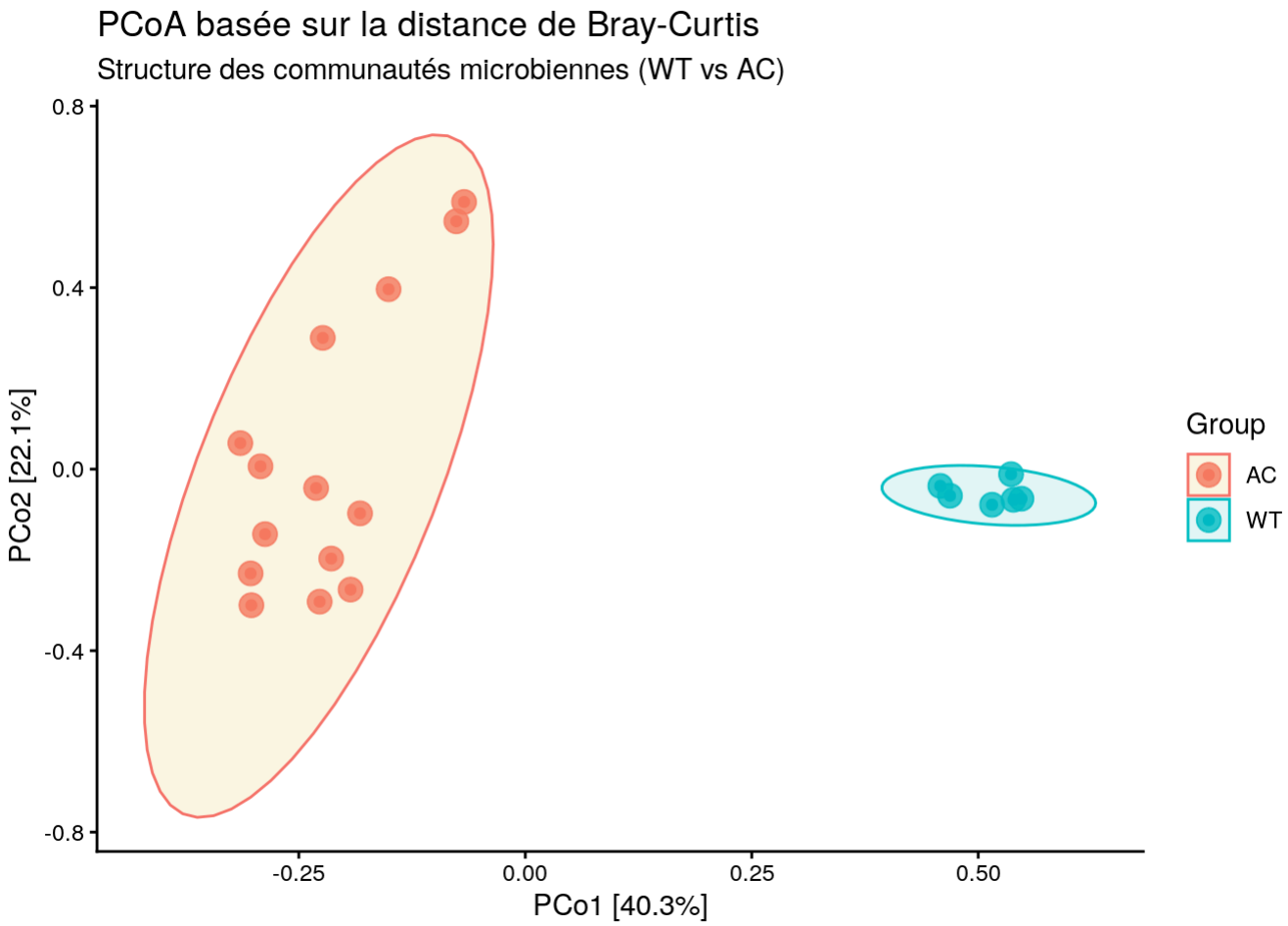
```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 325 taxa and 20 samples ]
## sample_data() Sample Data:       [ 20 samples by 26 sample variables ]
## tax_table()   Taxonomy Table:    [ 325 taxa by 7 taxonomic ranks ]
```

Calcul de la matrice de distance de Bray-Curtis et de l'ordination (PCoA)

```
dist_matrix <- distance(ps_rel, method = "bray")
pcoa_res <- ordinate(ps_rel, method = "PCoA", distance = dist_matrix)
```

Tracé du graphique

```
plot_ordination(ps_rel, pcoa_res, color = "Group") +
  geom_point(size = 4, alpha = 0.8) + # Paramètres des points sur le graphique (taille, opacité)
  stat_ellipse(aes(fill = Group), geom = "polygon", alpha = 0.1, level = 0.95) + # Ajout des ellipses de groupe en calculant la zone de confiance autour de chaque point
  scale_fill_manual(values = c("WT" = "#00AFBB", "AC" = "#E7B800")) +
  theme_classic() +
  labs(title = "PCoA basée sur la distance de Bray-Curtis",
    subtitle = "Structure des communautés microbiennes (WT vs AC)",
    x = paste0("PCo1 [", round(pcoa_res$values$Relative_eig[1] * 100, 1), "%]"),
    y = paste0("PCo2 [", round(pcoa_res$values$Relative_eig[2] * 100, 1), "%]")
  ) # Attribution du score de variance des axes
```



Ici on remarque dans un premier temps que les points de chaque groupes de tortues sont proches donc que les tortues sauvages ont des communautés microbiennes similaires entre elles et idem pour les tortues élevées en captivité. Cepedant, comme cela est visible sur la PCoA, les communautés des 2 types de tortues forment des groupes bien distincts, ce qui signifie que le microbiote des tortues est complètement différent en fonction de l'environnement dans lequel elle se trouvent.

Test Permanova

```
library(vegan)
```

```
## Loading required package: permute
```

```
# Extraire les métadonnées de l'objet phyloseq
metadata <- data.frame(sample_data(ps_rel))

# Test de PERMANOVA
permanova <- adonis2(dist_matrix ~ Group, data = metadata, permutations = 999)

# Afficher le résultat (cherchez la valeur de Pr(>F))
print(permanova)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = dist_matrix ~ Group, data = metadata, permutations = 999)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1   2.2650 0.39198 11.604  0.001 ***
## Residual 18   3.5133 0.60802
## Total    19   5.7783 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Grâce à ce test, il est possible d'affirmer que la composition du microbiote des WT et AC est significativement différente puisqu'on obtient un p-value inférieur à 0,05.

Taxonomie

Abondance relative des phyla

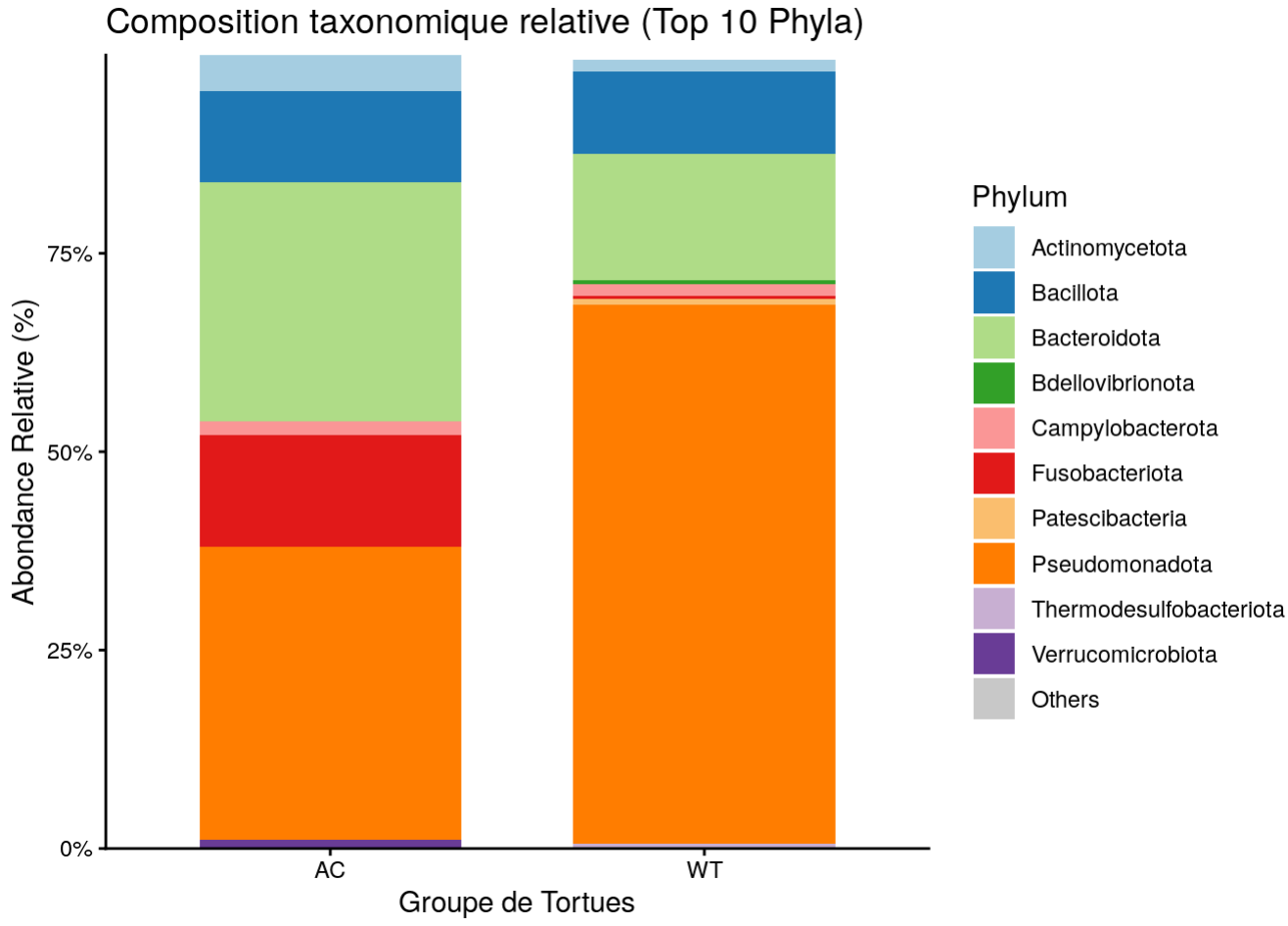
```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.6
## ✓ forcats   1.0.1     ✓ stringr   1.6.0
## ✓ lubridate 1.9.4     ✓ tibble    3.3.0
## ✓ purrr     1.2.0     ✓ tidyr     1.3.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to be
come errors
```

```
ps_phylum <- tax_glom(ps_rel, taxrank = "Phylum")
df_phylum <- psmelt(ps_phylum)
top10_phyla <- df_phylum %>%
  group_by(Phylum) %>%
  summarize(mean_abund = sum(Abundance)) %>%
  arrange(desc(mean_abund)) %>%
  slice(1:10) %>% pull(Phylum)
df_phylum <- df_phylum %>%
  mutate(Phylum = fct_other(Phylum, keep = top10_phyla, other_level = "Others")) # rajoute un
e catégorie autres pour regrouper le reste des phyla
df_grouped <- df_phylum %>%
  group_by(Group, Phylum) %>%
  summarize(Abundance = mean(Abundance))
```

```
## `summarise()` has grouped output by 'Group'. You can override using the
## `.groups` argument.
```

```
ggplot(df_grouped, aes(x = Group, y = Abundance, fill = Phylum)) +
  geom_bar(stat = "identity", position = "stack", width = 0.7) +
  scale_fill_manual(values = c(RColorBrewer::brewer.pal(10, "Paired"), "grey80")) +
  scale_y_continuous(labels = scales::percent_format(), expand = c(0,0)) +
  theme_classic() +
  labs(title = "Composition taxonomique relative (Top 10 Phyla)",
       y = "Abondance Relative (%)",
       x = "Groupe de Tortues")
```



On observe une forte abondance des phyla Pseudomonadota et Bacteroidata chez les 2 groupes mais aussi une abondance de Fusobacteriota, un phylum abritant des bactéries potentiellement

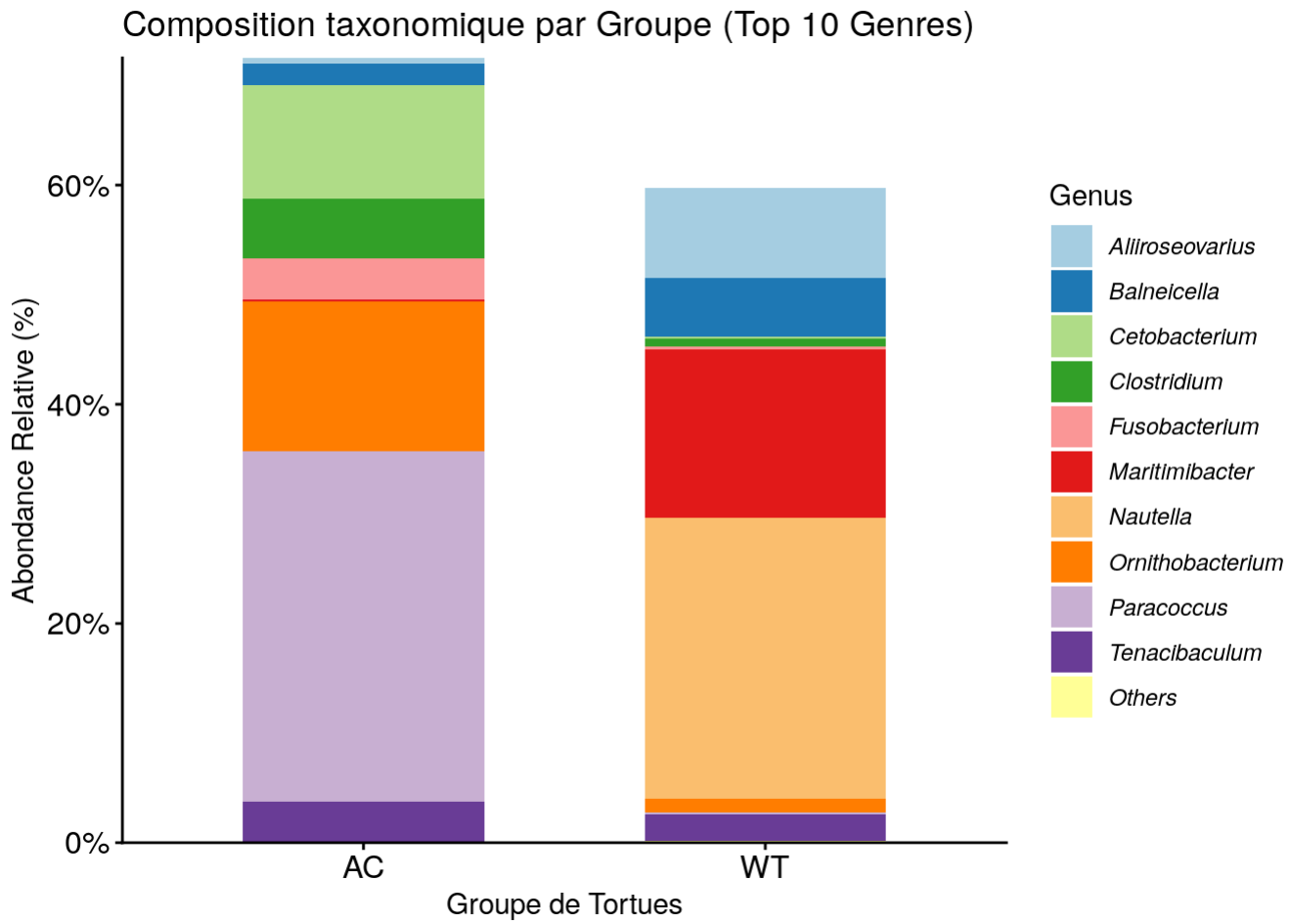
pathogènes, chez les AC.

Abondance relative des genres

```
ps_genus <- tax_glom(ps_rel, taxrank = "Genus")
df_genus <- psmelt(ps_genus)
top10_genera <- df_genus %>%
  group_by(Genus) %>%
  summarize(mean_abund = sum(Abundance)) %>%
  arrange(desc(mean_abund)) %>%
  slice(1:10) %>%
  pull(Genus)
df_genus <- df_genus %>%
  mutate(Genus = fct_other(Genus, keep = top10_genera, other_level = "Others")) # rajoute une
catégorie autres pour regrouper le reste des genres
df_grouped_gen <- df_genus %>%
  group_by(Group, Genus) %>%
  summarize(Abundance = mean(Abundance)) %>%
  ungroup()
```

```
## `summarise()` has grouped output by 'Group'. You can override using the
## `.groups` argument.
```

```
ggplot(df_grouped_gen, aes(x = Group, y = Abundance, fill = Genus)) +
  geom_bar(stat = "identity", position = "stack", width = 0.6) +
  scale_fill_manual(values = c(RColorBrewer::brewer.pal(12, "Paired"),
                               RColorBrewer::brewer.pal(4, "Set3"))) +
  scale_y_continuous(labels = scales::percent_format(), expand = c(0,0)) +
  theme_classic() +
  labs(title = "Composition taxonomique par Groupe (Top 10 Genres)",
       y = "Abondance Relative (%)",
       x = "Groupe de Tortues") +
  theme(legend.text = element_text(face = "italic"),
        axis.text = element_text(size = 12, color = "black")) # met les genres en italique et
règle les paramètres de la légende
```



On retrouve des espèces typiques de environnements marins chez WT comme Nautella et Aliroseovarius qui sont connus pour avoir un effet bénéfique sur la santé des tortues tandis que chez les AC on retrouve des bactéries du genre Paracoccus ou encore Fusobacterium qui sont des pathogènes responsables de diverses inflammations. Ce graphique permet de confirmer que les 2 types de tortues abritent des genres bactériens très différents et met en évidence un potentiel danger des AC pour les WT.

Conclusion

Les analyses effectuées ont permis de montrer que les tortues sauvages et élevées en environnement artificiel ont des microbiotes de composition significativement différentes. En effet, tandis les tortues sauvages ont un microbiote riche, diversifié et équilibré abritant des bactéries typiques des environnements marins et bénéfiques pour leur santé. Au contraire, les tortues élevées ont un microbiote déséquilibré et peuplé de bactéries considérées comme des agents pathogènes. Ainsi, relacher ces tortues parmi les populations sauvages pourrait leur transmettre des bactéries susceptibles de perturber leur microbiote et donc de les mettre en danger en les exposant aux infections par d'autres pathogènes. Pour que ce plan de réintroduction soit envisageable, il serait judicieux de

commencer par étudier les paramètres de environnements sauvages pour essayer d’adapter les conditions artificielles et réguler le microbiote des tortues élevées.