---
title: "R Notebook"
output: github_document
---

1. Preparing the R session

1.1 Load some libraries
```{r}
library(phyloseq)
library(ggplot2)
library(dplyr)
devtools::load_all(path="course-material-main")
```

1.2 Load the data in R
On choisi notre set working directory pour que ça fontionne correctement
```{r}
output_beta <- here::here("outputs", "beta_diversity")
if (!dir.exists(output_beta)) dir.create(output_beta, recursive = TRUE)
```

On met les données et on inspecte l'objet phyloseq
```{r}
file_path <- here::here("data", "asv_table", "phyloseq_object_alpha_beta_div.rds")
```

```{r}
file.exists(file_path)
```

```{r}
here::here("data", "asv_table", "phyloseq_object_alpha_beta_div.rds")
```

```{r}
physeq <- readRDS(here::here("data","asv_table","phyloseq_object_alpha_beta_div.rds"))
```

```{r}
physeq
```

2. Preparation f the data

2.1 Normalisation

2.1.1 Rarefaction
On sous échantillonne les reads pour chaque échantilon, sans remise et à une profondeur constante

Combien de lecture nous avons :
```{r}
#Normalisation des tables de données
#Ici on le fait par raréfaction : nous sous-échantillons les reads de chaque échantillon sans remise à une "profondeur constante"
rowSums(physeq@otu_table@.Data)
```

```{r}

readsumsdf <- data.frame(nreads = sort(taxa_sums(physeq), decreasing = TRUE),
                         sorted = 1:ntaxa(physeq),
                         type = "OTUs")
#On range les données en mettants les plus grandes en haut, donc par ordre décroissant

#La première colonne nreads est remplie avec les sommes des abondances (reads) des taxons dans l'objet physeq triées dans l'ordre décroissant à l'aide de la fonction sort(taxa_sums(physeq), decreasing = TRUE). signifie que les valeurs les plus élevées seront en haut

#La deuxième colonne sorted est remplie avec les numéros de rang de 1 à ntaxa(physeq), où ntaxa(physeq) donne le nombre de taxons dans l'objet physeq

#La troisième colonne type est remplie avec la chaîne de caractères "OTUs"

tmp <- data.frame(nreads = sort(sample_sums(physeq), decreasing = TRUE),
                  sorted = 1:nsamples(physeq),
                  type = "Samples")

#La première colonne nreads est remplie avec les sommes des abondances (reads) des échantillons (samples) dans l'objet physeq triées dans l'ordre décroissant à l'aide de la fonction sort(sample_sums(physeq), decreasing = TRUE). Encore une fois, place les valeurs les plus élevées en haut
```

#La deuxième colonne sorted est remplie avec les numéros de rang de 1 à nsamples(physeq), où nsamples(physeq) donne le nombre d'échantillons dans l'objet physeq.

#La troisième colonne type est remplie avec la chaîne de caractères "Samples"

readsumsdf <- rbind(readsumsdf, tmp)

#fusionne les data frames readsumsdf et tmp en les ajoutant ensemble par lignes à l'aide de la fonction rbind(). signifie que les données de tmp sont ajoutées en dessous de celles de readsumsdf

head(readsumsdf)
#vérifier les premières lignes du data frame résultant, la fonction head() est utilisée. affiche les premières lignes de readsumsdf à la console
```

```{r}
ggplot(readsumsdf, aes(x = sorted, y = nreads)) +
  geom_bar(stat = "identity") +
  ggtitle("Total number of reads") +
  scale_y_log10() +
  facet_wrap(~type, nrow = 1, scales = "free")

#ggplot(readsumsdf, aes(x = sorted, y = nreads)): crée un objet ggplot en spécifiant les données à utiliser (readsumsdf) et les esthétiques du graphique. associe la colonne sorted aux valeurs sur l'axe des x et la colonne nreads aux valeurs sur l'axe des y

#geom_bar(stat = "identity"): ajoute une couche de barres au graphique. La spécification stat = "identity" signifie que les hauteurs des barres sont directement déterminées par les valeurs de la colonne nreads dans le data frame, plutôt que d'utiliser une agrégation ou un comptage

#ggtitle("Total number of reads"): ajoute un titre au graphique, indiquant "Total number of reads"

#scale_y_log10(): applique une échelle logarithmique base 10 à l'axe des ordonnées (y). Cela permet de mieux visualiser les données lorsque les valeurs couvrent plusieurs ordres de grandeur

#facet_wrap(~type, nrow = 1, scales = "free"): divise le graphique en facettes (sous-graphiques) en fonction de la colonne type dans le data frame. Les facettes sont placées en une seule ligne (nrow = 1) et les échelles des axes y sont indépendantes pour chaque facette (scales = "free"). signifie que vous obtiendrez un sous-graphique pour "OTUs" et un autre pour "Samples"
```

```{r}
# set the seed for random sampling

# it allows reproductibility
set.seed(10000)

#définit la graine (seed) pour le générateur de nombres aléatoires. La fonction set.seed() est utilisée pour initialiser la séquence de nombres aléatoires de manière à ce qu'elle produise les mêmes résultats chaque fois que le code est exécuté
#fixe à 10000 --> reproductibilité des résultats

# minimum reads in a sample

min(rowSums(physeq@otu_table@.Data))
#calcule le nombre minimum de lectures (reads) dans un échantillon en examinant la table des abondances des OTUs (taxons) stockée dans l'objet physeq
```

```{r}
physeq_rar <- rarefy_even_depth(physeq, sample.size = 800)

#utilise la fonction rarefy_even_depth pour raréfier (sous-échantillonner) les données dans l'objet physeq de manière à ce que chaque échantillon possède exactement 800 reads
#utilisé pour normaliser les données de seq qd les echantillons ont des tailles différentes, comme ça on peut les comparer de manière équitable

rowSums(physeq_rar@otu_table@.Data)
#calcule la somme des reads par échantillon après avoir effectué la raréfaction.utilise la fonction rowSums() pour additionner les valeurs de chaque échantillon dans la table des abondances des OTUs (taxons) dans l'objet physeq_rar. résultat =  nb total de reads par échantillon après la raréfaction
```

```{r}
physeq
```

```{r}
physeq_rar
```

2.1.2 Centered log-ratio (CLR) transformation
```{r}
 # we first replace the zeros using
 # the Count Zero Multiplicative approach
tmp <- zCompositions::cmultRepl(physeq@otu_table,
                        method = "CZM",
                        label = 0,
                        z.warning = 1)

#utilise la fonction cmultRepl du package zCompositions pour remplacer les zéros dans la table des abondances des OTUs et utilise l'approche de remplacement multiplicatif de zéros (Count Zero Multiplicative, CZM) avec les paramètres :

##physeq@otu_table : extrait la table des abondances des OTUs de l'objet physeq
##method = "CZM" : spécifie l'utilisation de la méthode CZM pour le remplacement des zéros
##label = 0 : indique que les zéros dans la table doivent être remplacés par la valeur 0
##z.warning = 1 : active les avertissements en cas de zéros présents dans les données

# generate the centered log-ratio transformed. ASVs are in rows!!!!!
physeq_clr_asv <- apply(tmp, 1, function(x) log(x) - mean(log(x)))

#calcule la transformation du logarithme des rapports centrés (Centered Log-Ratio, CLR) pour chaque OTU (ASV) dans la table tmp avec comme paramètre :

##tmp : matrice qui est le résultats du remplacement des zéros.
##1 : argument 1 dans la fonction apply = que les calculs sont effectués par ligne (c'est-à-dire, pour chaque OTU).
##function(x) log(x) - mean(log(x)) : appliquée à chaque ligne de la matrice tmp. calcule le logarithme des valeurs dans chaque ligne, puis soustrait la moyenne du logarithme des valeurs de cette ligne. produit la transformation CLR pour chaque OTU.
```

```{r}
#create a new phyloseq object with CLR tranformed counts
physeq_clr <- physeq
otu_table(physeq_clr) <- otu_table(t(physeq_clr_asv),
                                   taxa_are_rows = FALSE)
#remplace la table des abondances des OTUs (taxons) dans l'objet physeq_clr par la matrice des valeurs CLR transformées stockées dans physeq_clr_asv

data.frame(physeq_clr@otu_table@.Data[1:5, 1:10])
#extrait les premières 5 lignes et les 10 premières colonnes de la table des abondances transformées en CLR dans l'objet physeq_clr, et les affiche dans un data frame -> examine un échantillon des données transformées
```

3. Visualisation of the meta-community composition

```{r}
physeq_phylum <- physeq_rar %>%
  tax_glom(taxrank = "Family") %>%
# agglomerate at the Family level
#agréger les données au niveau de la famille (taxrank = "Family") / signifie qu'il rassemble les données des mêmes familles de m-o

```
  transform_sample_counts(function(x) {x/sum(x)} ) %>%
 # Transform to rel. abundance
  #applique une transformation aux comptages d'échantillons / la fonction transform_sample_counts est utilisée pour appliquer la fonction anonyme qui divise chaque comptage par la somme des comptages de l'échantillon -> permet de convertir les comptages en abondances relatives

  psmelt() %>%
 # Melt to long format
  # fondre la structure de données en format long, chaque ligne représente un echantllon-taxa -> facilite l'ananylse et la visualisation des données

  filter(Abundance > 0.02) %>%
 # Filter out low abundance taxa quand abondance relative est sup à 0,02%

  arrange(Family)
# Sort data frame alphabetically by phylum
  # trie les data frame par ordre alphébatique en fct de la colonne family

head(physeq_phylum)
# -> afficher les premieres lignes

# %>% permet de faire une chaîne dopération
```

3.1 Treemaps

3.1.2 Using the package treemap

```{r}
#pdf(file="treemap.pdf", wi = 7, he = 7)

treemap::treemap(physeq_phylum, index=c("Class", "Family"), vSize="Abundance", type="index",
      fontsize.labels=c(15,12),
# size of labels. Give the size per level of aggregation: size for group, size for subgroup, sub-subgroups...
      fontcolor.labels=c("white","black"),
# Color of labels
      fontface.labels=c(2,1),
# Font of labels: 1,2,3,4 for normal, bold, italic, bold-italic...
      align.labels=list(
        c("center", "center"),
        c("left", "bottom")),
# Where to place labels in the rectangle?
      overlap.labels=0.5,
```

# number between 0 and 1 that determines the tolerance of the overlap between labels. 0 means that labels of lower levels are not printed if higher level labels overlap, 1 means that labels are always printed. In-between values, for instance the default value .5, means that lower level labels are printed if other labels do not overlap with more than .5 times their area size.
inflate.labels=F, # If true, labels are bigger when rectangle is bigger.
border.col=c("black","white"),
#Color of the boders separating the taxonomic levels
border.lwds=c(4,2),
#palette = "Set3",
# Select your color palette from the RColorBrewer presets or make your own.
fontsize.title=12
)

crée un treemap basé sur les données de physeq_phylum, en utilisant les colonnes "Class" et "Family" pour définir les niveaux de hiérarchie, avec la taille des rectangles basée sur la colonne "Abundance" + personnalise aussi l'apparence du treemap en modifiant plusieurs paramètres, comme la taille des étiquettes et la couleur des bordures. Une fois le treemap généré, il est affiché graphiquement pour visualiser la distribution des données hiérarchiques
```

```{r}
#dev.off()
```

```{r}
tmp <- transform_sample_counts(physeq,function(x) {x/sum(x)} ) %>%
 psmelt() %>% #fondre la structure des données
 group_by(Family, Class) %>% #groupe les données par les colonnes family et class -> les données sont agrégées en fct des 2 zones
 summarise(abundance = sum(Abundance)) %>% # calcul la sommedes abondances realtives pour chaque combinaison unique de familyet class + creer une nvlle colonne abundance qui possèdent la somme des abondances relatives
 na.omit() #supprime les lignes où il n'y a pas de valeur

ggplot(tmp,aes(area=abundance,label=Family,fill=Class,subgroup=Class))+
 treemapify::geom_treemap()+
 treemapify::geom_treemap_subgroup_border() +
 treemapify::geom_treemap_subgroup_text(place = "centre",
                                        grow = T,
                                        alpha = 0.5,
                                        colour = "black",
                                        fontface = "italic",
                                        min.size = 0) +
 treemapify::geom_treemap_text(colour = "white",
                               place = "topleft",
                               reflow = TRUE)+
```
```

```
  theme(legend.position="none")

#trnasformation des abondances relatives, rassemblement par family et class, somme les abondances relatives et supprime les données maquantes
```

```{r}
ggsave(here::here(output_beta,"treemap_treemapify.pdf"))
#enregistre le treemap en fichier pdf
```

3.2 Stacked barplots
```{r}

ggplot(physeq_phylum, aes(x = Sample, y = Abundance, fill = Family)) +
  geom_bar(stat = "identity") +
  # facet_wrap(~Treatment, nrow=1, scales = "free_x") +
  ylab("Relative Abundance (Family > 2%)") +
  scale_y_continuous(expand = c(0,0)) + #remove the space below the 0 of the y axis in the graph
  ggtitle("Community composition") +
  theme_bw() +
  theme(axis.title.x = element_blank(),
        axis.text.x = element_text(angle = 45, size = 10,
                                   hjust = 0.5, vjust = 0.8),
        axis.ticks.x = element_blank(),
        panel.background = element_blank(),
        panel.grid.major = element_blank(),  #remove major-grid labels
        panel.grid.minor = element_blank())  #remove minor-grid labels

#crée un graphique à barres des abondances relatives des familles en fonction des échantillons, avec des couleurs différentes pour chaque famille
```

```{r}
ggsave(here::here(output_beta, "asv_composition.pdf"))
```

4. Distance or dissimilarity matrices

4.1 Calculation

4.4.1 Compositional taxonomic

```{r}
```

#calcul de la matrice de distance de jaccard : présence/absence des taxons
physeq_rar_jaccard <- phyloseq::distance(physeq_rar,
                                         method = "jaccard",
                                         binary = TRUE)

# trick to avoid negative egein values in PCoA
# it recreates what ade4::dist.binary() does
physeq_rar_jaccard <- sqrt(physeq_rar_jaccard)

#transformation sur la matrice de distances Jaccard en prenant la racine carrée de chaque élément de la matrice
#tilisée pour éviter les valeurs négatives dans les analyses suivantes, car la distance de Jaccard originale peut avoir des valeurs négatives
```

```{r}
ape::is.rooted(physeq_rar@phy_tree)
#utilise la fonction is.rooted de la bibliothèque ape pour vérifier si l'arbre phylogénétique (stocké dans l'objet physeq_rar) est enraciné (rooted). Un arbre phylogénétique enraciné a un nœud racine, alors qu'un arbre non enraciné n'a pas de nœud racine
```

```{r}
unifracs <- GUniFrac::GUniFrac(physeq_rar@otu_table@.Data, physeq_rar@phy_tree, alpha=c(0, 0.5, 1))$unifracs
#calcule les distances UniFrac en utilisant la bibliothèque GUniFrac
#argument physeq_rar@otu_table@.Data représente les données de la table des opérations taxonomiques unitaires (OTU) dans l'objet physeq_rar, physeq_rar@phy_tree est l'arbre phylogénétique enraciné, et alpha=c(0, 0.5, 1) spécifie les valeurs d'alpha pour calculer les distances UniFrac
```

```{r}
physeq_rar_du <- unifracs[, , "d_UW"]

# Unweighted UniFrac
# extrait les distances UniFrac non pondérées (Unweighted UniFrac) de la matrice unifracs et les stocke dans l'objet physeq_rar_du
#distances UniFrac mesurent la similarité ou la dissimilarité entre les échantillons en fonction de la structure phylogénétique des taxons
#distances UniFrac non pondérées ne tiennent pas compte de l'abondance des taxons, se basant uniquement sur leur présence/absence
```

4.1.3 Structural taxonomic (Bray-Curtis)
```

```{r}
# physeq_rar_bray <- vegan::vegdist(physeq_rar@otu_table@.Data, method = "bray")
#Bray-Curtis est couramment utilisée pour mesurer la dissimilarité entre les échantillons en fonction de la composition des taxons, en tenant compte de l'abondance relative

tmp <- transform_sample_counts(physeq,function(x) {x/sum(x)} )
#transformer les comptages d'échantillons dans l'objet physeq en abondances relatives

physeq_rar_bray <- phyloseq::distance(tmp, method = "bray")
#calcule la distance de Bray-Curtis à partir de l'objet tmp en utilisant la fonction distance de la bibliothèque phyloseq. L'argument method = "bray" spécifie que la distance de Bray-Curtis doit être utilisée
```

```{r}
physeq_rar_dw <- unifracs[, , "d_1"]   # Weighted UniFrac
#extraire les distances UniFrac pondérées spécifiques associées à l'index "d_1" dans la matrice unifracs
#mesurent la dissimilarité entre les échantillons en fonction de la structure phylogénétique des taxons, en prenant en compte l'abondance des taxons
#différentes valeurs d'index dans unifracs correspondent généralement à différentes mesures de distances ou d'indices
```

```{r}
#Select the distances of interest

#attribue un numéro de position séquentiel à chaque méthode de distance et stocke ces informations dans un tableau de données avec deux colonnes : "position" et "dist_methods"

dist_methods <- unlist(distanceMethodList)
 #unlist pour extraire les éléments de la liste distanceMethodList et les stocker dans un vecteur appelé dist_methods, liste qui doit contenir les noms des méthodes de calcul de distances

data.frame(position = seq_along(dist_methods),
        dist_methods)
#crée un tableau de données avec deux colonnes :
##"position": première colonne contient les numéros de position séquentiels des méthodes de distance, la fonction seq_along(dist_methods) génère une séquence de numéros de 1 à la longueur de dist_methods
##"dist_methods": deuxième colonne contient les noms des méthodes de distance provenant du vecteur dist_methods
```

```{r}
#Select the distances of interest
```

```
dist_methods <- dist_methods[c(1, 2, 10, 8)]
#nouveau vecteur dist_methods en extrayant les éléments des positions 1, 2, 10 et 8 du vecteur original

dist_methods
```

```{r}
#Loop through each distance method, save each plot to a list, called plist.
plist <- vector("list")

for(i in dist_methods){
  # Calculate distance matrix
  iDist <- phyloseq::distance(physeq_rar, method = i)
  # Calculate PCoA ordination
  iMDS <- ordinate(physeq_rar, "MDS", distance = iDist)
  ## Make plot. Don't carry over previous plot (if error, p will be blank)
  p <- NULL
  # Create plot, store as temp variable, p
  p <- plot_ordination(physeq_rar, iMDS, color= "Geo")
  # Add title to each plot
  p <- p + ggtitle(paste("MDS using distance method ", i, sep=""))
  # Save the graphic to list
  plist[[i]] = p
}

#boucle à travers différentes méthodes de calcul de distances, calcule les matrices de distances, effectue des ordinations PCoA, crée des graphiques d'ordination avec des titres correspondant aux méthodes de distance, et stocke ces graphiques dans une liste plist
#permet de générer des graphiques pour différentes méthodes de distances et de les stocker
```

```{r}
df <- plyr::ldply(plist, function(x) x$data)
head(df)

#extrait les données de chaque graphique dans la liste plist et les combine en un tableau de données unique df
#regrouper les données de tous les graphiques en un seul endroit pour une analyse ou pour les regarder plus tard
```

```{r}
names(df)[1] <- "distance"

ggplot(df, aes(Axis.1, Axis.2, color = Geo)) +
  geom_point(size=3, alpha=0.5) +
```

```
  theme_bw() +
  facet_wrap(~distance, scales="free") +
  ggtitle("PCoA (MDS) on various distance metrics")

#graphique PCoA qui affiche les échantillons dans un espace à deux dimensions en fonction des axes principaux (Axis.1 et Axis.2) pour différentes méthodes de calcul de distances

#points sont colorés en fonction de la variable "Geo", et chaque méthode de distance est représentée dans une facette distincte avec des échelles d'axes ajustées
```

5 Hierarchical clustering

5.1 Hierarchical ascendant classification (HAC)

```{r}
#distance matrix calculation

physeq_clr_dist <- phyloseq::distance(physeq_clr, method = "euclidean")

#matrice de distances Euclidiennes entre les échantillons de physeq_clr
#matrice peut être utilisée pour évaluer la similarité ou la dissimilarité entre les échantillons en fonction de leurs compositions ou de leurs caractéristiques
```

```{r}
#Simple aggregation criterion
spe_single <- hclust(physeq_clr_dist, method = "single")
#effectue l'agglomération hiérarchique en utilisant la méthode "single linkage" (agglomération simple) -> lien entre deux clusters est déterminé par la paire d'échantillons les plus proches, c'est-à-dire que deux clusters sont fusionnés si leur distance la plus courte est inférieure à un seuil -> résultat est stocké dans l'objet spe_single

#Complete aggregation criterion
spe_complete <- hclust(physeq_clr_dist, method = "complete")
#effectue l'agglomération hiérarchique en utilisant la méthode "complete linkage" (agglomération complète) -> lien entre deux clusters est déterminé par la paire d'échantillons les plus éloignés, c'est-à-dire que deux clusters sont fusionnés si leur distance la plus longue est inférieure à un seuil -> résultat est stocké dans l'objet spe_complete

#Unweighted pair group method with arithmetic mean
spe_upgma <- hclust(physeq_clr_dist, method = "average")
#ligne effectue l'agglomération hiérarchique en utilisant la méthode "average linkage" (agglomération par la moyenne) -> calcule la distance moyenne entre tous les échantillons des deux clusters à fusionner -> résultat est stocké dans l'objet spe_upgma
```

#Ward criterion
spe_ward <- hclust(physeq_clr_dist, method = "ward.D")
#effectue l'agglomération hiérarchique en utilisant la méthode "ward.D" (criterion de Ward) #méthode de Ward minimise la variance des distances intra-clusters après chaque fusion. Le résultat est stocké dans l'objet spe_ward

par(mfrow = c(2, 2))
#configure la disposition de la fenêtre graphique pour afficher les quatre dendrogrammes dans une grille de 2x2
plot(spe_single, main = "single")
plot(spe_complete, main = "complete")
plot(spe_upgma, main = "UPGMA")
plot(spe_ward, main = "ward")
#affichent les dendrogrammes résultants des quatre méthodes d'agglomération dans la grille #chaque dendrogramme représente le regroupement hiérarchique des échantillons, avec les titres indiquant la méthode d'agglomération correspondante

#réalise l'agglomération hiérarchique de données d'échantillons en utilisant différentes méthodes d'agglomération et affiche les dendrogrammes résultants pour chaque méthode, permettant de visualiser comment les échantillons sont regroupés en clusters en fonction des critères d'agglomération utilisés
```

```{r}
#Cophenetic correlation
spe_single_coph <- cophenetic(spe_single)
#calcule la matrice de distances cophénétique pour l'agglomération hiérarchique réalisée avec la méthode "single linkage" (spe_single) -> matrice cophénétique représente la distance entre les paires d'échantillons dans le regroupement hiérarchique obtenu

cor(physeq_clr_dist, spe_single_coph)
#alcule la corrélation cophénétique entre la matrice de distances d'origine physeq_clr_dist et la matrice cophénétique spe_single_coph -> corrélation cophénétique mesure à quel point la structure hiérarchique des échantillons est préservée dans la matrice cophénétique

# répétiton pour les trois autres méthodes d'agglomération (complete, upgma, et ward) afin de calculer la corrélation cophénétique pour chaque méthode
spe_complete_coph <- cophenetic(spe_complete)
cor(physeq_clr_dist, spe_complete_coph)
spe_upgma_coph <- cophenetic(spe_upgma)
cor(physeq_clr_dist, spe_upgma_coph)
spe_ward_coph <- cophenetic(spe_ward)
cor(physeq_clr_dist, spe_ward_coph)

# permet d'évaluer à quel point la structure hiérarchique des échantillons, obtenue par agglomération hiérarchique avec différentes méthodes, est préservée dans les matrices de distances cophénétiques
#corrélation cophénétique est calculée pour chaque méthode d'agglomération et est un indicateur de la qualité de l'agglomération
#corrélation élevée indique une meilleure préservation de la structure hiérarchique
```

```{r}
plot_coph_cor <- function(cophenetic_distance, hclust_type){

  # first calculate the correlation between
  # the cophenetic distance and the observed distance
  cor_res <- round(cor(physeq_clr_dist, cophenetic_distance),3)

  # generate a scatter plot to visualise
  # the relationship
  plot(x = physeq_clr_dist,
     y = cophenetic_distance,
     xlab = "Aitchison distance",
     ylab = "Cophenetic distance",
     xlim = c(10, 35), ylim = c(10, 35),
     main = c(hclust_type, paste("Cophenetic correlation ", cor_res)))
  abline(0, 1)
}

par(mfrow=c(2,2))

plot_coph_cor(cophenetic_distance = spe_complete_coph,
          hclust_type = "Single linkage")

plot_coph_cor(cophenetic_distance = spe_complete_coph,
          hclust_type = "Complete linkage")

plot_coph_cor(cophenetic_distance = spe_upgma_coph,
          hclust_type = "Average linkage")

plot_coph_cor(cophenetic_distance = spe_ward_coph,
          hclust_type = "Ward linkage")
```

##5.3 Looking for Interpretable Clusters
```{r}
#Fusion level plot
par(mfrow = c(1, 1))

#configure la disposition de la fenêtre graphique pour afficher le graphique dans une seule cellule (1x1)

```
plot(x = spe_upgma$height,
    y = phyloseq::nsamples(physeq_clr):2,
    type = "S",
    main = "Fusion levels - Aitchison - Average",
    ylab = "k (number of cluster)",
    xlab = "h (node height)")

text(x = spe_upgma$height,
    y = phyloseq::nsamples(physeq_clr):2,
    labels = phyloseq::nsamples(physeq_clr):2,
    col = "red",
    cex = 0.8)
```

#graphique "Fusion levels" affiche les niveaux de fusion lors de l'agglomération hiérarchique, montrant comment le nombre de clusters diminue à mesure que les clusters sont fusionnés à différentes hauteurs -> étiquettes numériques indiquent le nombre de clusters à chaque niveau de fusion
```

```{r}
install.packages("NbClust", lib = ".")
library("NbClust", lib.loc = ".")
nclust <- nb_clust_all(data = t(physeq_clr_asv), seed = 1000)
###installation du package "NbClust", le chargent, puis utilisent une fonction de ce package pour calculer le nombre optimal de clusters sur un ensemble de données spécifié (physeq_clr_asv).
```

```{r}

#Cut the dendrogram in order to obtain K groups and compare their compositionC
k <- 2 # nbr de groupes donnés par le graphique de level de fusion

#couper le dendogramme
spe_upgma_clust <- cutree(tree = spe_upgma, k = k)
table(spe_upgma_clust)

#découpent un dendrogramme UPGMA en un nombre spécifié de clusters (k) et génèrent une table récapitulative montrant combien d'éléments appartiennent à chaque cluster
```

```{r}
spe_upgma_clust2 <- data.frame(UPGMA_clusters = spe_upgma_clust)
```
```

```

```{r}
# faire un graphique avec des labels de groupe
plot(spe_upgma,
    hang = -1,
    ylab = "Height",
    main="Aitchison distance - UPGMA")

rect.hclust(spe_upgma,
          k = k,
          border = 2:6,
          cluster = spe_upgma_clust)

legend("topright",
      paste("Cluster", 1:k),
      pch = 22,
      col = 2:(k + 1),
      bty = "n")
```

```{r}
#il y a plusieurs façons de mesurer la robustesse d'un algorithme de clustering :
# - Dunn index : calculer comme un ratio des plus petites distances inter-cluster et des plus grandes -> plus le DI est grand, mieux le clustering est, parce que les observations dans chaque cluster est plus proche, pendant que les clusters eux-mêmes sont plus loin les uns des autres.
#on va utiliser la fonction cluster.stats() dans le package fpc pour calculer le Dunn Index ce qui peut être utiliser pour valider les cluster
# - Davis-Bouldinindex
# - Silhouette index


cs <- fpc::cluster.stats(d = physeq_clr_dist,
                         clustering = spe_upgma_clust)

cs$dunn
```

```{r}
#le résultat du DI est haut ce qui montre un bon clustering des échantillons
#maintenant qu'on a ID les groupes basé sur leur composition en communauté microbienne, on veut regarder quel clade microbien ou ASV sont dans chaque groupes
```

```{r}

# Les heat-map du Z-score sont normalisées (centrées autour de la moyenne(par ligne)) et réduites (écart-type (standar deviation) = SD)
#c'est la comparaison entre une valeur observée d'un échantillon et la moyenne de la population
# donc ça rep à la question d'a quel point on est loin de la réalité
#on va sélectionner les 30 ASV les plus représentées :

#Transformation les comptes Row/normalisés en % : transform_sample_conts
pourcentS <- phyloseq::transform_sample_counts(physeq_rar, function(x) x/sum(x) * 100)
#Selection des 30 taxas
mytop30 <- names(sort(phyloseq::taxa_sums(pourcentS), TRUE)[1:30])
#extraction des taxa depuis l'objet %
selection30 <- phyloseq::prune_taxa(mytop30, pourcentS)
#Voir les nouveaux objets avec seulements les 30 meilleurs ASV
selection30
```

```{r}
#Récuperer l'abondance des ASV(otu_table) comme un tableau et mettre dans la variable
selection30_asv <- phyloseq::otu_table(selection30)
selection30_sample <- phyloseq::sample_data(selection30)

#changer les noms bruts (row names)
rownames(selection30_asv)
```

```{r}
#Change... Why?

# rownames(data.prop)<-
c("S11B_South5B","S1B_North1B","S2B_North2B","S2S_North2S","S3B_North3B","S3S_North3S","S4B_North4B","S4S_North4S","S5B_North5B","S5S_North5S","S6B_South1B","S6S_South1S","S7B_South2B","S7S_South2S","S8B_South3B","S8S_South3S","S9B_South4B","S9S_South4S")

sample_new_names <- paste(selection30_sample$SampName,
                          selection30_sample$Description,
                          sep = "_")

#Z-score transformation (with scale)
heat <- t(base::scale(selection30_asv))
#See
head(data.frame(heat))
```

```{r}
ComplexHeatmap::Heatmap(

```
  heat,
  row_names_gp = grid::gpar(fontsize = 6),
  cluster_columns = FALSE,
  heatmap_legend_param = list(direction = "vertical",
                              title = "Z-scores",
                              grid_width = unit(0.5, "cm"),
                              legend_height = unit(3, "cm"))
)
```

```{r}
# récupérer le tableau taxonomique
taxon <- phyloseq::tax_table(selection30) |>
  as.data.frame()

#concatène ASV avec les phylums et les noms des familles
myname <- paste(rownames(taxon), taxon$Phylum, taxon$Family, sep="_")

#appliquer le tout
colnames(selection30_asv) <- myname
```

```{r}
#re-run Z-score to take into account the colnames change
#refaire tourner l'algorithme du Z-score pour prendre en compte les noms changés des colonnes
heat <- t(scale(selection30_asv))

my_top_annotation <- ComplexHeatmap::anno_block(gp = grid::gpar(fill =c(3,4)),
                                  labels = c(1, 2),
                                  labels_gp = grid::gpar(col = "white",
                                                         fontsize = 10))

ComplexHeatmap::Heatmap(
  heat,
  row_names_gp = grid::gpar(fontsize = 6),
  cluster_columns =TRUE,
  heatmap_legend_param = list(direction = "vertical",
    title ="Z-scores",
    grid_width = unit(0.5, "cm"),
    legend_height = unit(4, "cm")),
  top_annotation = ComplexHeatmap::HeatmapAnnotation(foo = my_top_annotation),
  column_km = 2,
  column_names_gp= grid::gpar(fontsize = 6)
  )
```

```{r}
#ajouter un boxplot de distribution d'abondance d'ASV
boxplot <- ComplexHeatmap::anno_boxplot(t(selection30_asv),
                                        which = "row",
                                        gp = grid::gpar(fill = "turquoise3"))

my_boxplot_left_anno <- ComplexHeatmap::HeatmapAnnotation(Abund = boxplot,
                                                  which = "row",
                                                  width = unit(3, "cm"))

my_top_anno <- ComplexHeatmap::anno_block(gp = grid::gpar(fill = c(3, 6)),
                                          labels = c("South", "North"),
                                          labels_gp = grid::gpar(col = "white",
                                                                 fontsize = 10))

my_top_anno <- ComplexHeatmap::HeatmapAnnotation(foo = my_top_anno)

ComplexHeatmap::Heatmap(
  heat,
  row_names_gp = grid::gpar(fontsize = 7),
  left_annotation = my_boxplot_left_anno,
  heatmap_legend_param = list(direction = "vertical",
                              title ="Z-scores",
                              grid_width = unit(0.5, "cm"),
                              legend_height = unit(3, "cm")),
  top_annotation = my_top_anno,
  column_km = 2,
  cluster_columns = TRUE,
  column_dend_side = "bottom",
  column_names_gp = grid::gpar(fontsize = 7)
  )

# on peut observer (un jour j'espère j'aimerai que ça fonctionne) que les communautés microbiennes dans les échantillons du sud diffèrent de celles du nord. L'effet significatif du traitement doit être testé statistiquement. la diff entre la composition des commu est due à l'abondance differentielle apparente de pleins des top ASV du dataset. L'identification des biomarqueurs significatifs des échantillons du nord et du sud seront recouverts(???) plus tard
```
#partie 6 : indirect gradient analysis
```{r}
#pendant que l'analyse des cluster cherche pour une discontinuité dans le dataset, l'ordination extrait la plus obvious tendance en forme d'axes continus.
#c'est bien adapté d'analyser les data depuis des communautés écologiques naturelles qui sont généralement structurées en gradient
#c'est pourquoi ces types d'analyses sont appelées des analyses gradientes. Le but de la méthode d'ordination c'est de représenter la data le long d'un nombre réduit d'axe orthogonaux, construits dans un sens où ils représentent dans un ordre décroissant la
```

tendance la plus représent&e des data. On va voir 4 types d'analyses vue en écologie : PCA, PCoA, NMDS : toutes ces méthodes sont descriptives : aucun test stat est fourni pour évaluer la signification des structures détectées. C'est le rôle des ordinations sous contraintes ou des analyses de test d'hypothèses qui sont présentées par la suite.

```

#partie 6.1 : PCA
```{r}
#PCA : principal component analysis : méthode pour résumé, dans un espace avec peu de dimension, la variance dans une dispersion mutivariée de points. Ca donne une vue sur les relations linéaires entre nos objets et nos variables. Ca peut souvent servir de bon point de départ dans l'analyse de data multivariées en nous permettant de faire une tendance, des groupes, des variables clées et des potentiels valeurs aberrantes. Ici on va utiliser la distance d'Aitchinson. Attention : c'est le tableau ASV CLR transformé  qu'on utilise directement, pas la matrice de distance Aitchinson. Cette fonction va calculer une distance euclidienne sur le tableau CLR transformé pour avoir la matrice Aitchinson. Il y a beaucoup de packages qui permettent l'analyse PCA. On va utilisé le PCAtools qui est récent et qui fourni les fonctions pour l'exploration des data par une PCA, et qui permet à l'utilisateur de générer des figures prêtes pour la publication

#nombre de PC à retenir :
#prepare the ASV table to add taxonomy
tax_CLR <-  as.data.frame(tax_table(physeq_clr)) # get taxnomic table
#concatene ASV with Family & Genus names
ASVname <- paste(rownames(tax_CLR), tax_CLR$Family, tax_CLR$Genus,sep="_")
#apply
rownames(physeq_clr_asv) <- ASVname
p <- PCAtools::pca(physeq_clr_asv,
                   metadata = data.frame(sample_data(physeq_clr)))
PCAtools::screeplot(p, axisLabSize = 18, titleLabSize = 22)
```

```{r}
#on voit que chaque PC ressortent avec 31% de variance expliquée et après on a une diminution graduelle pour les composants qui restent. Un diagramme d'éboulis(??) tout seul montre juste la proportion cumulée d'une variation expliquée, mais on veut déterminer le nombre optimum de PC à retenir.

#Horn's parallel analysis (Horn 1965) (Buja and Eyuboglu 1992)
horn <- PCAtools::parallelPCA(physeq_clr_asv)
horn$n
```

```{r}
#elbow method
elbow <- PCAtools::findElbowPoint(p$variance)
```
```

elbow

#les deux méthodes indiques qu'on doit retenir les 2 ou 3 premières PC. La raison de cette divergence c'est parce que trouver le nombre correct de PC c'est difficile et ça demande de trouver le bon nombre de clusters dans le dataset - y'a pas de bonne rep. la plupart des études prennent en compte que les deux premières PC
```

```{r}
# mettre en plot l'ordination :
PCAtools::biplot(
  p,
  lab = p$metadata$SampName,
  colby = "Geo",
  pointSize = 5,
  hline = 0, vline = 0,
  legendPosition = "right"
)
# chaque point est un échantillon, et les échantillons qui apparaissent proches sont plus probables d'être similaires que ceux qui sont loins (bienvu sherlock) -> en colorant les points par un traitement on peut voir que les microbes du nord est plus souvent, mais pas toujours, vachement différent des échantillons du sud
```

```{r}
#déterminer les variables qui dirigent la variation dans chaque PC
# un des bénéfices à ne pas utiliser une matrice de distance, on peut faire un plot des "taxa loadings" sur nos axes PCA. PCAtools permet de ploter le nombre de vecteurs loading taxa avec lesquels on veut commencer qui ont le plus gros poids sur chaque PC. la taille relative de chaque vecteur loading idique sa contribution sur chaque axe de PCA montré, et permet d'estimer à peu près chaque echantillon qui contient plus que ce dit taxon

PCAtools::biplot(
  p,
  # loadings parameters
  showLoadings = TRUE,
  lengthLoadingsArrowsFactor = 1.5,
  sizeLoadingsNames = 3,
  colLoadingsNames = 'red4',
  ntopLoadings = 3,
  # other parameters
  lab = p$metadata$X.SampleID,
  colby = "Geo",
  hline = 0, vline = 0,
  legendPosition = "right"
)

#ASV 7,11,12 ont une grosse contribution à PC1 tandis qui ASV 38, 40 et 47 ont une grosse contribution à PC2, ces ASV appartiennt à 2 familles. Les échantillons du sud contiennent une plus grande abondance de ASV11 et 12. Les deux échantillons aberrants du nord en haut du plot sont caractérises par une plus haute abondance de ASV38,40 et 47
```

```{r}
#correler le principal composant à nos data envtale

PCAtools::eigencorplot(
  p,
  components = PCAtools::getComponents(p, 1:horn$n),
  metavars = c('SiOH4','NO2','NO3','NH4','PO4',
               'NT','PT','Chla',"T", "S", "Sigma_t"),
  col = c('white', 'cornsilk1', 'gold',
          'forestgreen', 'darkgreen'),
  cexCorval = 1.2,
  fontCorval = 2,
  posLab = "all",
  rotLabX = 45,
  scale = TRUE,
  main = bquote(PC ~ Spearman ~ r^2 ~ environmental ~ correlates),
  plotRsquared = TRUE,
  corFUN = "spearman",
  corUSE = "pairwise.complete.obs",
  corMultipleTestCorrection = 'BH',
  signifSymbols = c("****", "***", "**", "*", ""),
  signifCutpoints = c(0, 0.0001, 0.001, 0.01, 0.05, 1)
)

#la seule correlation significative trouvé entre PC1 expliqué par la séparation entre les échantillons du nord et du sud et la salinité. C'est interessant mais la correlation entre deux variables ne veut pas tout de suite dire que le changement dans une variable est la cause de changements dans l'autre. On verra plus tard si c'est une relation de cause à effet entre la salinité et la difference entre les deux lieux échantillonés.


```

#partie 6.2 : PCoA : principal component analysis
```{r}
#PCoA veut representer la distance entre les échantillons dans un espace euclidien avec peu de dimension, en particulier, ça maximise la correlation linéaire entre les ditances dans une matrice de distance, et les distances dans un espace avec peu de dimensions(tipiquement 2 ou 3 axes sont sélectionnés). Comme toujours le choix de mesure de (dis)similarité est critique et doit coller au data en question. Ici on utilise la distance de Bray-curtis. Quand la

distance métrique est euclidienne, PCoA est équivalente à la PCA -> l'interpretation des résultats est la meme.

```
#BPCoA on Bray-Curtis dissimilarity
pcoa_asv <- ape::pcoa(physeq_rar_bray)
pcoa_coord <- pcoa_asv$vectors[, 1:2]

#Data frame for hull
hull <- data.frame("Axis.1" = pcoa_coord[, 1],
                   "Axis.2" = pcoa_coord[, 2],
                   "sample" = as.data.frame(sample_data(physeq_rar@sam_data)))


# North <- hull[hull$sample.Geo  == "North", ][chull(hull[hull$sample.Geo ==  "North",
c("Axis.1", "Axis.2")]), ]  # hull values for North
# South <- hull[hull$sample.Geo == "South", ][chull(hull[hull$sample.Geo ==
#                                             "South", c("Axis.1", "Axis.2")]), ]  # hull values for Jellyfishes

# hull_data <- rbind(North, South)

#Vector of color for hulls
# color <- rep("#a65628", length(hull_data$sample.Geo))
# color[hull_data$sample.Geo == "North"] <- "#1919ff"
# hull_data <- cbind(hull_data, color)

hull_col <- c("#a65628","#1919ff")
names(hull_col) <- c("North","South")

hull_data <- hull %>%
  dplyr::group_by(sample.Geo) %>%
  dplyr::slice(chull(Axis.1,Axis.2)) %>%
  dplyr::mutate(color = hull_col[sample.Geo])

head(hull_data)
```

```{r}
ggplot(data = hull, aes(x = Axis.1, y = Axis.2)) +
  geom_hline(yintercept = 0, colour = "lightgrey", linetype = 2) +
  geom_vline(xintercept = 0, colour = "lightgrey", linetype = 2) +
  geom_polygon(data = hull_data,
               aes(group = sample.Geo,
                   fill = sample.Geo),
               alpha = 0.3) + # add the convex hulls)
  scale_fill_manual(values = c("Darkgrey", "#1919ff")) +
  geom_point(data = hull,
             aes(color = sample.Geo,
```

```
            size = sample.S),
        alpha = 0.7) +
 scale_color_manual(values = c("Darkgrey", "#1919ff")) +
 xlab(paste("PCo1 (", round(pcoa_asv$values$Relative_eig[1]*100, 1), "%)")) +
 ylab(paste("PCo2 (", round(pcoa_asv$values$Relative_eig[2]*100, 1), "%)")) +
 theme_bw() +
 coord_equal() +
 theme(axis.title.x = element_text(size = 14), # remove x-axis labels
     axis.title.y = element_text(size = 14), # remove y-axis labels
     panel.background = element_blank(),
     panel.grid.major = element_blank(),  #remove major-grid labels
     panel.grid.minor = element_blank(),  #remove minor-grid labels
     plot.background = element_blank())
```

#l'odination des échantillons dans PCoA est très similaire de celle observée dans PCA avec une ségrégation claire entre le nord et le sud. Cela vient de la salinité qui augmente du nord au sud mais ça doit encore etre testé

#y'a pas d'espèces dans cette ordination, c'est parce qu'on a utilisé une matrice de dissimilarité pour que la fonction PCoA se fasse -> donc on peut pas calculer de score de spéciation. par contre on peut travailler sur ce problème avec la fonction biplot.pcoa() du package ape

#PCoA souffre d'un nombre de défauts, surtout sur l'effet d'arc. ces défauts viennent du fait que PCoA maximise la correlation linéaire. Le NMDS rectifie ça par maximiser la correlation de l'ordre de classement.
```

#partie 6.3 : NMDS : non metric multidimensional scaling
```{r}
#ça veut représenter la dissimilarité par paire entre les objets dans un espace à peu de dimensions. chaque coeficient de dissimilarité ou mesure de ditance peut etre utilisé pour constuire une matrice de distance comme entrée. C'est une approche basée sur le classement.Cela veut dire que la data de distance originelle est substituée par des classements. Pendant que l'info sur la magnitude des distances est perdue, les méthodes basées sur le classement est généralement plus robuste sur les data qui n'ont pas de distribution identifiable.

# NMDS est un algo iteratif, il commence par un placement aléatoire des objets dans un espace d'ordination. L'algo commence par affiner ce placement par un process iteratif, pour essayer de trouver une ordination qui ordonne la distance des objets aussi proche que possible de leur vrai distance de dissimilarité (vue dans l'originale matrice de distance). La valeur de stress montre à quel point l'ordination résume la distance observée entre les échantillons.
#ce n'est pas une analyse propre, ça a trois conséquences :
#il n'y a pas de résultat d'ordination unique
#les axes d'ordination ne sont pas ordonés à la variance qu'ils expliquent
```
```

le nbr de dimensions de l'espace à peu de dimension doit être spécifié avant de faire l'analyse
les axes ne sont pas ordonés avec le NMDS, vegan::metaMDS() rotationne automatiquement le résultat final en utilisant PCA pour faire correspondre l'axe 1 à la plus grande variance des points d'échantillons du NMDS

#NMDS plot on Aitchison distance
physeq_clr_nmds <- vegan::metaMDS(physeq_clr_dist, k=2, trymax=100) #Aitchison distance

```

```{r}
#Un moyen utile d'évaluer la pertinence d'un résultat NMDS est de comparer dans un shepard diagram les distances entre les objets dans un plot d'orination avec les distances originelles
vegan::stressplot(physeq_clr_nmds)
```

```{r}
# il existe un bon ajustement non métrique entre les dissimilarités observées (dans notre matrice de distance) et les distances dans l'espace d'ordination. Aussi le stress de notre résultat final est bon.

#la valeur stress peut être utilisée comme un indicateur de bon ajustement. Quand la valeur de stress est >0.2 c'est que c'est pauvre et souvent non interprétable, alors que les valeurs <0.1 sont bonnes et les <0.05 sont excellentes, ce qui laisse un mini danger de mauvais interpretation
#on peut plot le résultat :
nmds_coord <- data.frame(physeq_clr_nmds$points)

#Data frame for hull
hull <- data.frame("Axis.1" = nmds_coord[,1],
                   "Axis.2" = nmds_coord[,2],
                   "sample" = as.data.frame(sample_data(physeq_clr@sam_data)))

# North <- hull[hull$sample.Geo  == "North", ][chull(hull[hull$sample.Geo ==
#                                              "North", c("Axis.1", "Axis.2")]), ]  # hull values for North
# South <- hull[hull$sample.Geo == "South", ][chull(hull[hull$sample.Geo ==
#                                             "South", c("Axis.1", "Axis.2")]), ]  # hull values for Jellyfishes

# hull_data <- rbind(North, South)

# #Vector of color for hulls
# color <- rep("#a65628", length(hull_data$sample.Geo))
# color[hull_data$sample.Geo == "North"] <- "#1919ff"
# hull_data <- cbind(hull_data, color)

```
hull_col <- c("#a65628","#1919ff")
names(hull_col) <- c("North","South")

hull_data <- hull %>%
  dplyr::group_by(sample.Geo) %>%
  dplyr::slice(chull(Axis.1,Axis.2)) %>%
  dplyr::mutate(color = hull_col[sample.Geo])

#pdf(file="NMDS_Aitchison.pdf", wi = 7, he = 7)
ggplot(hull,aes(x = Axis.1, y = Axis.2)) +
  geom_hline(yintercept = 0, colour = "lightgrey", linetype = 2) +
  geom_vline(xintercept = 0, colour = "lightgrey", linetype = 2) +
  geom_polygon(data = hull_data,
               aes(group = sample.Geo,
                   fill = sample.Geo),
               alpha = 0.3) + # add the convex hulls)
  scale_fill_manual(values = c("Darkgrey", "#1919ff")) +
  geom_point(data = hull,
             aes(color = sample.Geo,
                 size = sample.S),
             alpha = 0.7) +
  scale_color_manual(values = c("Darkgrey", "#1919ff")) +
  geom_text(data = hull_data,
            x = -0, y = -9,
            label = paste("Stress =", round(physeq_clr_nmds$stress, 2)),
            colour = "Black",
            size = 5)  +
  xlab(paste("MDS1")) +
  ylab(paste("MDS2")) +
  theme_bw() +
  coord_equal() +
  theme(axis.title.x = element_text(size=14), # remove x-axis labels
        axis.title.y = element_text(size=14), # remove y-axis labels
        panel.background = element_blank(),
        panel.grid.major = element_blank(),  #remove major-grid labels
        panel.grid.minor = element_blank(),  #remove minor-grid labels
        plot.background = element_blank())

#on observe le même pattern d'ordination des échantillons comme avec PCA et PCoA. Il n'y a pas de score d'espèce (le même problème avec PCoA) on peut régler ce problème en utilisant la fonction wascores qui donne metaMDS la matrice de commu originelle comme entrée et qui spécifie une mesure de distance
```

```{r}
```

#quelles variable envtale drive les differences observées dans la compo des sp ? comme on a fait avec PCA, on peut correler les variables envtales avec nos axes d'ordination

# Correlation with environmental data
data.frame(names(hull))
```

```{r}
env <- hull[, 13:23]

# The function envfit will add the environmental variables as vectors to the ordination plot
ef <- vegan::envfit(physeq_clr_nmds, env, permu = 1000)
ef
```

```{r}
# The two last columns are of interest: the squared correlation coefficient and the associated p-value
# Plot the vectors of the significant correlations and interpret the plot
plot(physeq_clr_nmds, type = "t", display = "sites")
plot(ef, p.max = 0.05)

#ici on peut voir que la salinité est très correlée avec le premier axe qui sépare les échantillons du nord et du sud
#dans une moindre mesure, les nouvelles variables envtale related avec les conditions trophiques des habitats (NH4 et PT) sont correlés avec le deuxième axe de NMDS. La détection de ces nouvelles relations entre les commu microbiennes et l'envt peut etre related au fait que NMDS est le mieux adapté pour détecter la réponse non linéaire des microbes aux gradients envtaux.

#plusieurs types diff d'analyse de gradient peuvent se faire -> graphe
```

#partie 7 : analyses pour tester nos hypothèses
```{r}
#on va tester si les cluster d'échantillons se regroupent au delà de ce qui est attendu en utilisant des méthodes de test d'hypo comme PERMANOVA (multivariate analysis of variane with permutation) et l'analyse des groupes de similarité (ANOSIM) multiresponse permutation procedures (MRPP) et le test de Mantel (MANTEL) et plus récemment Dirichlet-multinomial models.

```

```{r}
# PERMANOVA : permet d'appliquer ANOVA à un dataset écologique multivarié. C'est le plus utilisé comme méthode non-paramétrique pour fiter les modèles multivariés à une data de microbes. C'est une analyse multivariée de variance basé sur les matrices de distances et la

permutation. Ca se fait en utilisant le concept de centroides. plusieurs permutations de la data (random shuffling) est utilisé pour générer une distribution nulle. On va pouvoir évaluer si la groupe du nord ou du sud à un effet significatif sur la compo globale de la commu bactérienne.

```
#PERMANOVA
metadata <- data.frame(sample_data(physeq_clr))
results_permanova <- vegan::adonis2(physeq_clr_dist ~ Geo,
                                    data = metadata,
                                    perm = 1000)

results_permanova

#on voit que les regroupements nord et sud expliquent significativement (p<0.001) 20% de la variance dans la matrice ASV Aitchison. du coup, les deux groupes diffèrent significativement dans leur compo bac.
```

```{r}
#Le test d'ADONIS peut être perturbé par des diff de dispersion (ou de propagation), nous voulons donc vérifier ça.

# Testing the assumption of similar multivariate spread among the groups (ie. analogous to variance homogeneity)
anova(vegan::betadisper(physeq_clr_dist, metadata$Geo))

# ici les groupes ont des diff de dispersion significatives et le résultat de la permanova peut être impacté par ça, bien que permanova est très robuste pour différencier les dispersion des groupes
```

```{r}
#onpeut regarder quel taxa contribue le plus à la diff de commu en utilisant l'ancienne fonction adonis() et la table des ASV des comptes transformés CLR.

#Show coefficients for the top taxa separating the groups

permanova <- vegan::adonis(t(physeq_clr_asv) ~ Geo,
                           data = metadata,
                           permutations = 1000,
                           method = "euclidean")

coef <- coefficients(permanova)["Geo1",]

top.coef <- coef[rev(order(abs(coef)))[1:10]]

par(mar = c(3, 14, 2, 1))
```

```
barplot(sort(top.coef),
      horiz = TRUE,
      las = 1,
      main = "Top taxa",
      cex.names = 0.7)

```


```{r}
#adonis() et adonis2() nous permet d'explorer les effets des catégories ou des variables continues

# une diff importante entre ces deux fonctions : les termes avec adonis sont testés séquentiellement et c'est la seule option. Cela veut dire que l'ordre dans lequel on entre nos variables est important (si le design est non balancés). Le prochain est ajouté pour voir si ça explique de manière significative plus de variation non expliquées par les variables d'avant. c'est l'équivalent d'utiliser by="terms" dans adonis2(). Si tu ne veux pas que l'ordre compte on peut utiliser adonis2 avec by="margin", ou alors si on veut voir si le modèle en entier est signif on peut utiliser by=NULL. L'ordre on s'en fiche quand by="margin" parce que la signification est testé contre un modèle qui inclue toutes les autres variables pas juste celles qui précèdent dans la formule.

#Permanova on continuous variables
permanova_S <- vegan::adonis2(physeq_clr_dist ~ S,
                      data = metadata,
                      perm = 1000)
permanova_S
```

```{r}
permanova_NH4 <- vegan::adonis2(physeq_clr_dist ~ NH4,
                       data = metadata,
                       perm = 1000)
permanova_NH4
```

```{r}
permanova_PT <- vegan::adonis2(physeq_clr_dist ~ PT,
                       data = metadata,
                       perm = 1000)
permanova_PT
```

```{r}
```

#le résultat confirme que la salinité et à moindre mesure le NH4 et le PT sont d'importants facteurs qui caractérisent les commu microbiennes mais qu'en est-il des autres variables ? on va construire un modèle avec toutes les co-variables

#Inspecting co-variables
permanova_all <- vegan::adonis2(physeq_clr_dist ~ SiOH4 + NO2 + NO3 + NH4 + PO4 + NT + PT + Chla + T + S + Sigma_t,
                        by="margin",
                        data=metadata,
                        perm=1000)

permanova_all

#pourquoi aucune des variables n'a plus d'effets signifs mtn ?
#on a du mal à ne pas construire un modèle pour prendre tout en compte. Bcp de temps et d'argent sont dépensés là dedans. on espère aussi identifié toutes les variables signifs pour encore mieux caractériser les relations avec une pertinence biologique. Une correlation excessive entre les variables explicatives peut compliquer ou prévenir l'identification d'un set optimal de variables explicatives pour un modèle stat.
```

```{r}
#on regarde quelles varaibles explicatives sont correlées :

# inpecting autocorrélation
# compute the correlation matrix
cor_metadadata <- cor(metadata[, 11:21], method = "spearman")

cor_mtest <- function(mat, ...) {
  mat <- as.matrix(mat)
  n <- ncol(mat)
  p_mat <- matrix(NA, n, n)
  diag(p_mat) <- 0
  for (i in 1:(n - 1)) {
    for (j in (i + 1):n) {
      tmp <- cor.test(mat[, i], mat[, j], method = "spearman", ...)
      p_mat[i, j] <- p_mat[j, i] <- tmp$p.value
    }
  }
  colnames(p_mat) <- rownames(p_mat) <- colnames(mat)
  p_mat
}

# matrix of the p-value of the correlation
p_mat <- cor_mtest(metadata[, 11:21])
```
```

```
# Leave blank on no significant coefficient
corrplot::corrplot(cor_metadadata,
                   type = "upper",
                   order = "hclust",
                   p.mat = p_mat,
                   sig.level = 0.05,
                   insig = "blank")

#on peut voir que bcp sont correlées
```

```{r}
# on enlève certaines, et no garde S pour PO4, sigma-t, NH4 et NO2. NO3 pour SiOH4. Chla pour PT

permanova_cor_pars <- vegan::adonis2(physeq_clr_dist ~ S + NO3 + NT + Chla + T,
                                     by = "margin",
                                     data = metadata,
                                     perm = 1000)
permanova_cor_pars

#là c'est mieux que le premier test de permanova. On peut voir que la salinité est très significativement related à la variable de réponse, mais on peut voir que le modèle qu'avec la salinité est encore mieux. En gros : les vraies relations entre les variables peuvent etre masked si les variables explicatives sont colinéaires. Cela pose problème dans la création de modèlé qui ammène à des complications dans l'interference de modèle. Prendre le temps d'évaluer la colinéarité c'est le début pour créer des modèles écologiques plus robustes
```

#partie 7.2 : analysis of similarité (ANOSIM)
```{r}
# pour les diff signif entre deux ou plus classes d'objets basé sur n'importe quelle mesure de (dis)similarité. Cela compare les rangs des distances entre les objets des diff classes avec les ranges des distances des objets dans les classes. C'est un peu la meme base que l'ordination NMDS

#ANOSIM
vegan::anosim(physeq_clr_dist, metadata$Geo, permutations = 1000)

#comme permanova le résultat indique un effet signif de la provenance de l'échantillon sur les commu bac
# une approche plus formelle pour tester des hypothèse peut etre faite par une analyse redondante ou une analyse de correspondance canoniques qui utilise directement l'information sur les champs de metadata en même temps que ça genère les ordinations et les test. Ces approches test directement les hypo à propos des variables envtales
```

#partie 8 : Direct gradient analysis
```{r}
#Les ordinations simples analysent qu'une seule data et revèle la strcuture majeur dans un graphique construit depuis un set réduit d'axes orthogonaux. C'est une forme passive d'analyse, et l'utilisateur interprete le résultat de l'ordination à posteriori. Au contraire, les analyses directe de gradient (aussi appelées ordination canonique) associent deux ou plus set de data dans le processus d'ordination lui-même. En conséquence, si l'un veut extraire les structures du set de data et sont related aux structures dans un autre jeu de données, et / ou veulent tester stastistiquement des hypothèses à propos de la signification des relations, l'odination canonique peut etre bien. on va faire un RDA sur note jeu de données, mais pleins d'autres existent.
```

#partie 8.1 : RDA : redundant analysis
```{r}
#c'est une méthode qui combine la regression et la PCA. ça compute les axes qui sont des combinations linéaires des variables explicatives. Dans RDA, un seul peut vraiment dire que les axes expliquent ou modèlisent la variation de matrice de dépendance.

# RDA of the Aitchinson distance
# constrained by all the environmental variables
# contained in metadata
#
# Observe the shortcut formula
spe_rda <- vegan::rda(t(physeq_clr_asv) ~ .,
                      metadata[, 11:21])
head(summary(spe_rda))  # Scaling 2 (default)
```

```{r}
#les variables envtales inclues expliquent 70,44% de la variation sur la compo de la commu bacterienne dans tous les sites. 29,56% de la variation est inexpliquée. CEpendant, on verra que la proportion de variation expliquée est bien plus basse. Le R2 du résumé le mesure la forme des relations canoniques entre les variables de reponses (Y matrice) et les variables explicatives (X matrice) par calculer la proportion de variable de Y expliqué par la variable de X. Cependant cet R2 est biased. On calcule une R2 ajusté, ce qui mesure aussi la force des relations entre X et Y, mais qui appliques une correlation de R2 pour prendre en compte le nombre de variable expliquées. C'est la stat qu'on doit reportée.

# Unadjusted R^2 retrieved from the rda object
R2 <- vegan::RsquareAdj(spe_rda)$r.squared
R2

# Adjusted R^2 retrieved from the rda object
R2adj <- vegan::RsquareAdj(spe_rda)$adj.r.squared
R2adj
```

```{r}
#en réalité, la proportion de variance expliquée descend à 16.25%. La sortie numérique montre que les deux premiers axes canoniques expliquent ensemble 35.1% de la bariance total de la data, le premier axe explique 25.9%. mais ce sont des valeurs non ajustées. R2 ajusté = 16.2%, le % de valeurs propres adj contraintes accumulées montre que le premier axe explique 0.162 * 0.368 = 0.059 soit 5.9% de la variance. Parce que la data écologique sont généralement avec du bruit, on ne devrait jamais avoir une valeur haute de R2. De plus, la premiere valeur propre sans contrainte PC1, le premier axe sans contrainte pour les résidus, est comparativement haut, ce qui veut dire que ça affiche une structure de résidus importante de data de réponse qui n'est pas expliqué par les mesures de paramètres envtaux.
```

```{r}
#l'interpretation de l'ordination avec contraintes doit etre précédé par un test de signification stat. Comme une régression multiple, un résultat non signif ne doit pas etre interprété et doit etre dégagé.

# Global test of the RDA result
anova(spe_rda, step = 1000)
```

```{r}
# Tests of all canonical axes
anova(spe_rda, by = "axis", step = 1000)

#ici on peut voir que tout notre modèle stat n'est pas signif (p=0.08) et que chaque axe canonique préovenant de RDA ne l'est pas non plus (p>0.05). donc on ne peut pas intepréter ce RDA model.
#est-ce qu'on peut dire pourquoi ?
```

```{r}
#sélectionner les variables
# ça arrive parfois qu'on veut réduire le nombre de variables explicatives. les raisons dépendent.
#une approche simple pour identifier une colinéarité dans les variables explicatives c'est d'utiliser le VIF (facteur d'inflation de variance). Les calculs de VIF sont facils à comprendre, plus la valeur est haute plus la colinéarité est haute. VIF mesure la proportion par lesquelles la variance d'un coeff de regression est influencé dans la présence d'autres variables explicatives. VIF plus haut que 20 indique une forte collinéarité. Au dessus de 10 on doit réexaminer, et mis de côté si possible.

# Variance inflation factors (VIF)
vegan::vif.cca(spe_rda)
```

```{r}
#salinité T° et sigma.t ont une très haute VF qui confirme les colinéarités observés avant entre les variables explicatives (permanova). Une réduction du nombre de variables explicatives est justifiée. Pour simplifier ce modèle, on peut faire une sélection "forward", ou "backwards" ou "stepwise". Ces types de sélection nous aide à selectionner des variables qui sont statistiquement importante. Cependant, c'est important de noter que selectionner des variables écologiques est plus important que performer une sélection dans ce sens. Si une variable d'interet écologique n'est pas sélectionnée, ça ne va pas dire qu'on l'enleve de RDA. Ici, on va faire une sélection "forward" sur nos 11 variables envtales. On peut utiliser la fonction ordiR2step()  :

# Forward selection of explanatory variables using vegan's ordiR2step()
step_forward <- vegan::ordiR2step(vegan::rda(t(physeq_clr_asv) ~ 1,
                                  data = metadata[, 11:21]),
                        scope = formula(spe_rda),
                        direction = "forward",
                        pstep = 1000)

#ici on peut ajouter une variable à la fois, et la retenir si ça augmente significativement le modèle ajusté de R2. La sélection qui suit nous montre qu'un modèle qu'avec la salinité a un plus haut R2 ajusté qu'un avec toutes les variables et qui explique 18.4% de la variance.
```

```{r}
#on va calculer la RDA la plus partimonieuse et regardé si c'est signif :

# Parsimonious RDA
spe_rda_pars <- vegan::rda(t(physeq_clr_asv) ~ S, data = metadata[, 11:21])
anova(spe_rda_pars, step = 1000)
```

```{r}
anova(spe_rda_pars, step = 1000, by = "axis")
```

```{r}
R2adj_pars <- vegan::RsquareAdj(spe_rda_pars)$adj.r.squared

# Compare variance inflation factors
vegan::vif.cca(spe_rda)
```

```{r}
vegan::vif.cca(spe_rda_pars)
```

```{r}
#maintenant, chacun des deux, le modèle et le premier axe canonique résulte du RDA sont statistiquement signif (p<0.05). La VIF de la salinité n'est que 1. Ce modèle RDA est interpretable ! on va le plot

# Preparation of the data for the plot
#
# View analysis results
ii <- summary(spe_rda_pars)

# Depending on the drawing result
# the drawing data can be enlarged or
# reduced to a certain extent, as follows
sp <- as.data.frame(ii$species[, 1:2]) * 2
sp_top <- sp[order(abs(sp$RDA1), decreasing = TRUE), ][1:6, ]

st <- as.data.frame(ii$sites[, 1:2])
st <- merge(st,
    metadata["Geo"],
    by = "row.names")

yz <- t(as.data.frame(ii$biplot[, 1:2]))
row.names(yz) <- "Salinity"
yz <- as.data.frame(yz)

eigen_values <- format(100 *ii$cont[[1]][2,], digits=4)

#plot
ggplot() +
 geom_point(data = st, size = 4,
         aes(x = RDA1, y = PC1,
           shape = Geo, fill = Geo)) +
 scale_shape_manual(values = c(21:25)) +
 geom_segment(data = sp_top,
           arrow = arrow(angle = 22.5,
                     length = unit(0.35, "cm"),
                     type = "closed"),
           linetype = 1, size = 0.6, colour = "red",
           aes(x = 0, y = 0, xend = RDA1, yend = PC1)) +
 ggrepel::geom_text_repel(data = sp_top,
                    aes(x = RDA1, y = PC1, label = row.names(sp_top))) +
 geom_segment(data = yz,
           arrow = arrow(angle = 22.5,
                     length = unit(0.35,"cm"),
                     type = "closed"),
           linetype = 1, size = 0.6, colour = "blue",
           aes(x = 0, y = 0, xend = RDA1, yend = PC1)) +
```

```
  ggrepel::geom_text_repel(data = yz, aes(RDA1, PC1, label=row.names(yz)))+
  labs(x = paste("RDA 1 (", eigen_values[1], "%)", sep = ""),
       y = paste("PC 1 (", eigen_values[2], "%)", sep = ""))+
  geom_hline(yintercept = 0,linetype = 3,size = 1) +
  geom_vline(xintercept = 0,linetype = 3,size = 1)+
  guides(shape = guide_legend(title = NULL,
         color = "black"),
         fill = guide_legend(title = NULL))+
  theme_bw() +
  theme(panel.grid = element_blank())
```

```{r}
#l'un des plus fort aspect de RDA est la visualisation simultanée de la réponse et des variables explicatives (sp et variables envtales). Depuis cette ordination, on peut vraiment dire mtn que la salinité est la principale variable envtale mesurée qui faconne les commu bac. Entre toutes les ASV, certaine sont related à ce gradient de salinité. C'est le cas de ASV 11 et 12, dont l'abondance augmente quand la salinité diminue. ASV 7 qui fait l'inverse. On peut analyser de pleins de façon ces diff de pattern, mais ce qui est vraiment puissant avec RDA c'est qu'on fait ressortir les relations de gradients, pas une diff d'abondance entre deux conditions. Cependant, une grande partie de la variance dans la commu bac continue d'être inexpliquée. Les variances dans les commu d'sp peuvent etre expliquées par un processus deterministique comme le "species sorting" (influence de l'envt comme on le voit ici) mais aussi un processus stochastique comme une disperion qui dépend entre autre de la distance entre les commu.
```

```{r}
#Maintenant qu'on a cette info : regardons un pattern commun dans les commu écologique : distance-decay pattern (modèle de décroissance de la distance)
```

#partie 8.2 : MRM : multiple regression on dissimilarity matrices

```{r}
#La décroissance de la similarité des assemblages avec la distance spatiale peut s'expliquer par des mécanismes alternatifs : la limitation de la dispersion et le tri des espèces. Pour comprendre leurs contributions relatives, nous comparons la décroissance de la similarité bac avec la distance spatiale, ainsi qu'indépendamment avec la distance envtale
#Une combinaison de la corrélation de Mantel et de la régression multiple sur les matrices de distance permet une analyse de type régression de deux matrices (dis)similaires ou plus, en utilisant des permutations pour déterminer la signification des coefficients de détermination. Une matrice doit contenir les (dis)similarités calculées à partir des données de réponse, telles que les abondances d'OTU, tandis que les autres matrices doivent contenir les (dis)similarités calculées à partir des données explicatives(par exemple, les paramètres environnementaux ou spatiaux).
```

#Tout d'abord, nous calculons la matrice de distance spatiale. Pour calculer la distance kilométrique entre les points d'échantillonnage à partir des coordonnées géographiques : le package SpatialEpi et la fonction latlong2grid().

```

#soucis de dimension partout ensuite je comprends pas

```{r}
ANF_km <- readRDS(here::here("course-material-
main","data","beta_diversity","spatial_distance.rds"))
ANF_km_dist <- dist(ANF_km)
```

```{r}
#Calculate and add model to the plot

#ANF_decay_exp <- betapart::decay.model(physeq_clr_dist/100,
                            # ANF_km_dist,
                            # y.type="dissim",
                            # model.type="exp",
                            # perm=100)

#Plot Distance decay relationships
#plot(ANF_km_dist, physeq_clr_dist/100,
   # ylim=c(0, max(physeq_clr_dist/100)),
    #xlim=c(0, max(ANF_km_dist)),
    #xlab = "Distance (km)", ylab = "Dissimilarity (CLR)")

#betapart::plot.decay(ANF_decay_exp, col = "blue",
               # remove.dots = TRUE, add = TRUE)

#legend("bottomright",
     # paste("exp: (Beta =", round(ANF_decay_exp$second.parameter, 4),
          #", Rsqr =", round(ANF_decay_exp$pseudo.r.squared, 2),
         # ", p =", round(ANF_decay_exp$p.value, 2)),
      #fill = "blue")
```

```{r}
#Variance partitioning
#Microbiam matrix (response)
physeq_clr_dist_square <- phyloseq::distance(physeq_clr,
                                method = "euclidean",
                                diag = TRUE,
                                upper = TRUE)
```
```

#Spatial matrix (explicative)
ANF_km_dist_square <- dist(ANF_km, diag = TRUE, upper = TRUE)

#environmental matrix (explicative)
envdata <- dist(metadata[,11:21], diag = TRUE, upper = TRUE)
```

```{r}
#Multiple regressions on Matrices (MRM) - attention les colonnes et lignes des matrices doivent correspondrent (pas besoin d'avoir les mêmes noms)

#ecodist::MRM(physeq_clr_dist_square ~ envdata + ANF_km_dist_square, nperm=1000) # 0.366
```

```{r}
#ecodist::MRM(physeq_clr_dist_square ~ envdata, nperm=1000) # 0.212
```

```{r}
#ecodist::MRM(physeq_clr_dist_square ~ ANF_km_dist_square, nperm=1000) # 0.238
```

```{r}
#modEvA::varPart(A = 0.212, B = 0.238, AB = 0.366,
        # A.name = "Environmental",
        # B.name = "Dispersal limitation")
```

```{r}
#En utilisant la régression multiple sur les matrices de distance (MRM), les variables spatiales et environnementales se sont avérées être des prédicteurs significatifs de la diversité bêta et ont ensemble expliqué 36,7 % de la variation dans la dissimilarité des communautés microbiennes. Ensuite, nous avons utilisé la partition de la variance pour répartir la variation en composantes purement spatiales, purement environnementales et environnementales structurées spatialement. Avec 15,4 %, la quantité de variation dans la dissimilarité expliquée par la composante purement spatiale était plus élevée que la variation expliquée par la composante environnementale, ce qui indique que la dispersion est un processus important façonnant nos communautés.

#De manière similaire aux analyses de gradient indirect, de nombreux types d'analyses de gradient direct sont disponibles. Dans le graphique suivant, nous proposons des suggestions de choix appropriés en fonction de la structure des données en entrée et des relations attendues entre les variables.


```
```

#partie 9 : DAA : differential abundance analysis

```{r}
#L'objectif des tests de différence d'abondance (DAA) est d'identifier des taxons spécifiques associés à des variables de métadonnées d'intérêt. Il s'agit d'une tâche difficile et l'une des zones les plus controversées de l'analyse des données microbiomiques, comme illustré dans cette prépublication. Cela est lié à des préoccupations selon lesquelles les approches de normalisation et de test ont généralement échoué à contrôler les taux de découvertes fausses.

#Il existe de nombreux outils pour effectuer des DAA. Les outils les plus populaires, sans entrer dans l'évaluation de leur performance pour cette tâche, sont : (ALDEx2, ANCOM-BC, conrcob, DESeq2, edgeR, LEFse, limma voom, LinDA,MaAsLin2, metagenomeSeq, IndVal,t-test, Test de Wilcoxon)

#des gens ont comparé toutes ces méthodes répertoriées sur 38 ensembles de données différents et ont montré que ALDEx2 et ANCOM-BC produisent les résultats les plus cohérents entre les études. Étant donné que différentes méthodes utilisent différentes approches (paramétriques vs non paramétriques, techniques de normalisation différentes, hypothèses, etc.), les résultats peuvent différer entre les méthodes. Par conséquent, il est fortement recommandé de choisir plusieurs méthodes pour avoir une idée de la robustesse et de la reproductibilité potentielle de vos résultats en fonction de la méthode. trois méthodes utilisées en écologie microbienne (ANCOM-BC, ALDEx2 et LEFse), et nous comparerons les résultats entre elles : package microbiome_marker récent.
```

#partie 9.1 : Linear discriminant analysis Effect Size (LEFse)
```{r}
#LEFSE a été dvp par des gens. LEFSE utilise tout d'abord le test non paramétrique de Kruskal-Wallis (KW) à somme de rangs pour détecter les caractéristiques présentant des abondances différentielles significatives par rapport à la classe d'intérêt ; la cohérence biologique est ensuite examinée en utilisant un ensemble de tests par paires entre les sous-classes à l'aide du test de somme de rangs Wilcoxon (non apparié). En dernière étape, LEfSe utilise l'analyse discriminante linéaire (LDA) pour estimer la taille de l'effet de chaque caractéristique présentant une abondance différentielle.

#LEFSE
mm_lefse <- microbiomeMarker::run_lefse(physeq, norm = "CPM",
                                        wilcoxon_cutoff = 0.01,
                                        group = "Geo",
                                        taxa_rank = "none",
                                        kw_cutoff = 0.01,
                                        multigrp_strat = TRUE,
                                        lda_cutoff = 4)

mm_lefse_table <- data.frame(mm_lefse@marker_table)
```

mm_lefse_table
```

```{r}
p_LDAsc <- microbiomeMarker::plot_ef_bar(mm_lefse)
y_labs <- ggplot_build(p_LDAsc)$layout$panel_params[[1]]$y$get_labels()
p_abd <- microbiomeMarker::plot_abundance(mm_lefse, group = "Geo") +
  scale_y_discrete(limits = y_labs)
gridExtra::grid.arrange(p_LDAsc, p_abd, nrow = 1)

#LEFse identifie 12 biomarqueurs, parmi lesquels ASV 7, 11 et 12 que nous avions déjà identifiés précédemment avec d'autres méthodes.
```

#partie 9.2 : differential analysis of compositions of microbiomes with bias correction (ANCOM-BC)
#fonctionne pas
```{r}
#La méthodologie ANCOM-BC suppose que l'échantillon observé est une fraction inconnue d'un volume unitaire de l'écosystème, et que la fraction d'échantillonnage varie d'un échantillon à l'autre. ANCOM-BC prend en compte la fraction d'échantillonnage en introduisant un terme de correction spécifique à l'échantillon dans un cadre de régression linéaire, qui est estimé à partir des données observées. Le terme de correction sert de correction de biais, et le cadre de régression linéaire en échelle logarithmique est analogue à la transformation en log-ratio pour traiter la composition des données microbiomiques. De plus, cette méthode fournit des valeurs p et des intervalles de confiance pour chaque taxon. Elle contrôle également le taux de découvertes fausses (FDR) et est simple du point de vue de sa mise en œuvre sur le plan computationnel.

#ça marche pas
#ancomBC

install.packages("ANCOMBC")
library(ANCOMBC)
```

```{r}
mm_ancombc <- run_ancombc_patched(
  physeq,
  group = "Geo",
  taxa_rank = "none",
  pvalue_cutoff = 0.001,
  p_adjust = "fdr"
)

mm_ancombc_table <- data.frame(mm_ancombc@marker_table)
mm_ancombc_table
```
```

```

```{r}
an_ef <- microbiomeMarker::plot_ef_bar(mm_ancombc)
y_labs <- ggplot_build(an_ef)$layout$panel_params[[1]]$y$get_labels()
an_abd <- microbiomeMarker::plot_abundance(mm_ancombc, group = "Geo") +
  scale_y_discrete(limits = y_labs)
gridExtra::grid.arrange(an_ef, an_abd, nrow = 1)

#AnCOM-BC identifie 10 biomarqueurs et tout en commun avec les résultats de l'analyse LEFse (si ça fonctionnait)
```

#partie 9.3 : anova-like differential expression (ALDEx2)
```{r}
#ALDEx2 estime la variation technique au sein de chaque échantillon par taxon en utilisant la distribution de Dirichlet. De plus, il applique la transformation du logarithme centralisé sur le rapport (ou des transformations de rapports de logarithmes étroitement liées). Selon la configuration expérimentale, il effectuera un test T de Welch à deux échantillons et un test de Wilcoxon, ou une analyse de variance à un facteur et un test de Kruskal-Wallis. La procédure de Benjamini-Hochberg est appliquée dans tous les cas pour corriger les tests multiples.

mm_aldex <- microbiomeMarker::run_aldex(physeq, group = "Geo",
                                        norm = "CPM",
                                        taxa_rank = "none",
                                        p_adjust = "fdr")

mm_aldex_table <- data.frame(mm_aldex@marker_table)
mm_aldex_table
```

```{r}
#ALDEx2 est beaucoup plus strict et identifie uniquement 1 biomarqueur, l'ASV 27, qui a été identifié par les deux autres méthodes d'analyse de l'abondance différentielle (DAA). Les autres ne parviennent pas à atteindre le seuil de contrôle du taux de découvertes fausses (FDR) utilisé ici, bien qu'ils aient probablement des tailles d'effet assez importantes. Souvent, lorsque je considère la réalisation de tests d'abondance différentielle (DA), j'exécute plusieurs modèles et me concentre sur l'intersection des OTUs (Unités Taxonomiques Opérationnelles) données par au moins deux méthodes. Ici, il s'agirait des 10 ASV identifiés avec ANCOM-BC.
```

#et c'est finiiii, dommage que la fin ne fonctionne pas :(
```