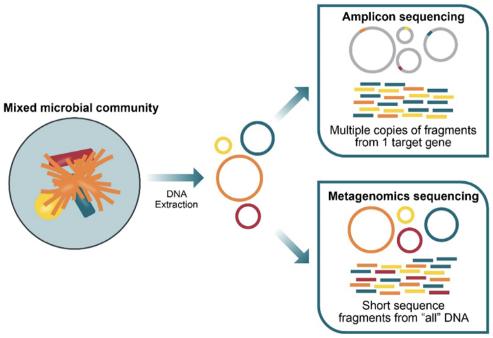
# 2025 NDMC Metagenome workshop

Yincheng Chen

Created on March 28, 2025

## Contents

# Introduction to Microbiota Data


Mixed microbial community
DNA Extraction
Amplicon sequencing
Multiple copies of fragments from 1 target gene
Metagenomics sequencing
Short sequence fragments from "all" DNA

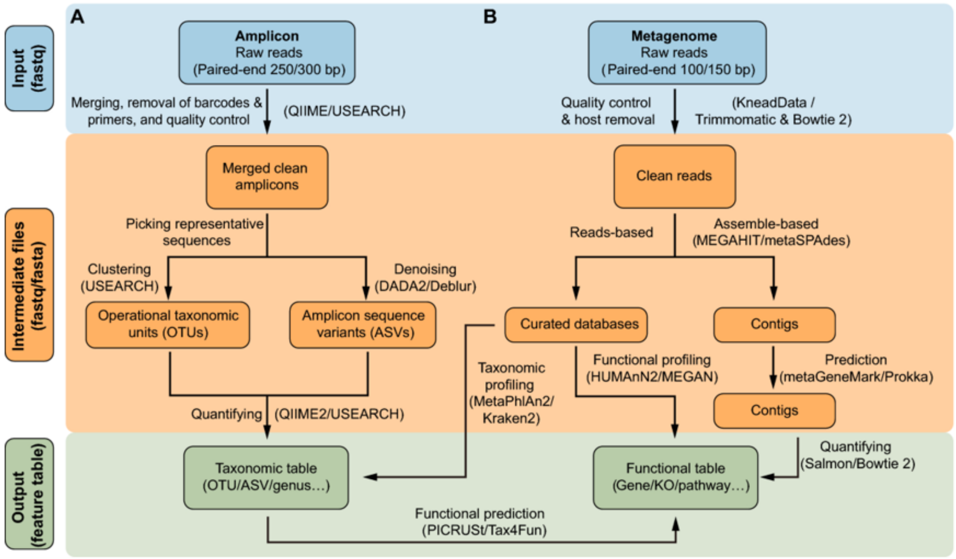
Pros and Cons

- Quantify statistically
- Low resolution (genus level)
- Low computing cost

- Biological functions.
- High resolution (species/strain level)
- High computing cost.

https://blog.crownbio.com/understanding-the-microbiome-using-genomic-sequencing-and-analysis

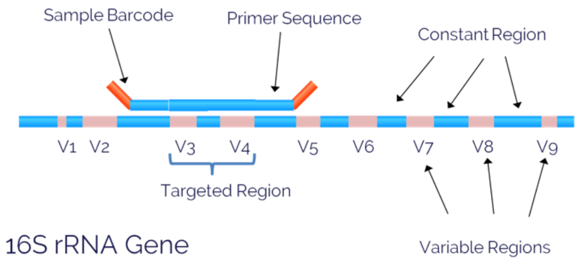
## Practical guide to microbiota data

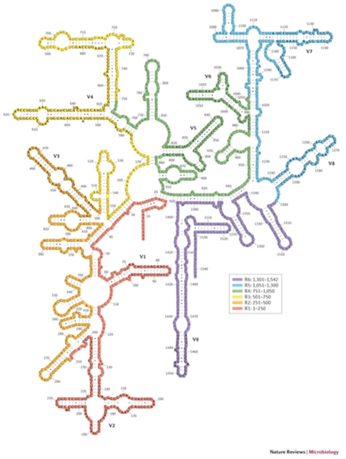

A
B
Input (fastq)
Intermediate files (fastq/fasta)
Output (feature table)
Amplicon Raw reads (Paired-end 250/300 bp)
Merging, removal of barcodes & primers, and quality control
(QIIME/USEARCH)
Merged clean amplicons
Picking representative sequences
Clustering (USEARCH)
Denoising (DADA2/Deblur)
Operational taxonomic units (OTUs)
Amplicon sequence variants (ASVs)
Quantifying
(QIIME2/USEARCH)
Taxonomic table (OTU/ASV/genus…)
Metagenome Raw reads (Paired-end 100/150 bp)
Quality control & host removal
(KneadData / Trimmomatic & Bowtie 2)
Clean reads
Reads-based
Assemble-based (MEGAHIT/metaSPAdes)
Curated databases
Contigs
Taxonomic profiling (MetaPhlAn2/ Kraken2)
Functional profiling (HUMAnN2/MEGAN)
Prediction (metaGeneMark/Prokka)
Contigs
Quantifying (Salmon/Bowtie 2)
Functional table (Gene/KO/pathway…)
Functional prediction (PICRUSt/Tax4Fun)

Liu, Y. X., Qin, Y., Chen, T., Lu, M., Qian, X., Guo, X., & Bai, Y. (2021). A practical guide to amplicon and metagenomic analysis of microbiome data. *Protein & cell*, *12*(5), 315-330.

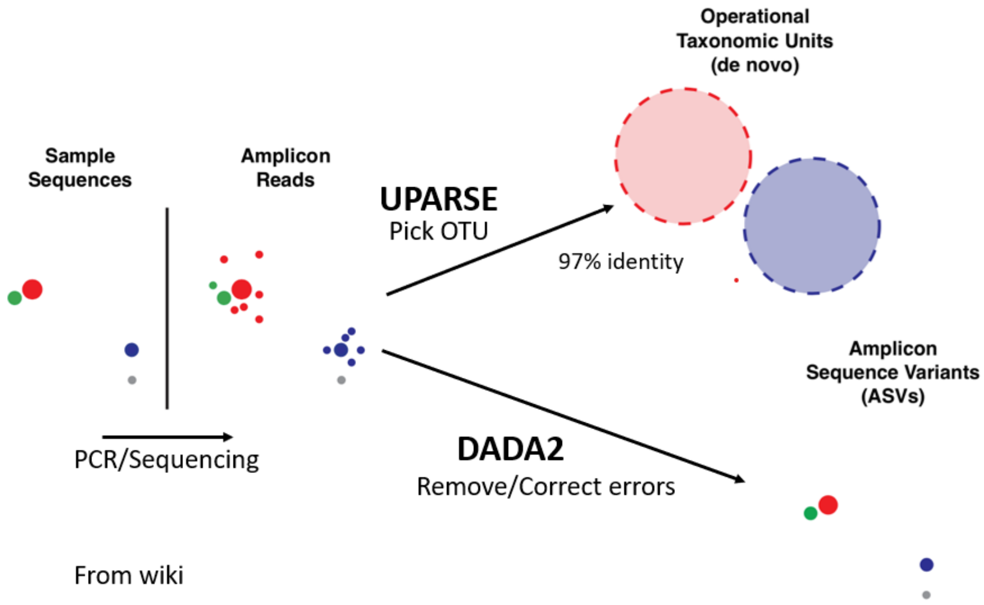
### 16s rRNA amplicon

Highly conserved in bacteria and archaea, but contains nine hypervariable region V1-V9





### Sequence process for amplicon data



Reference database :

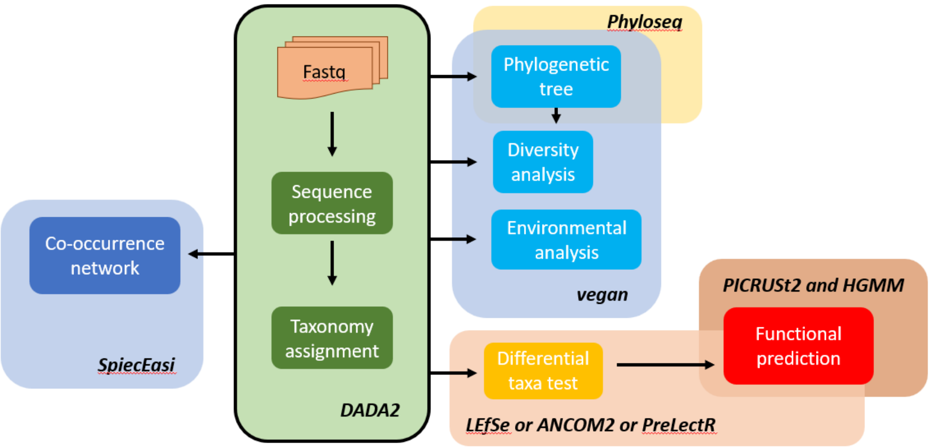
For bacteria

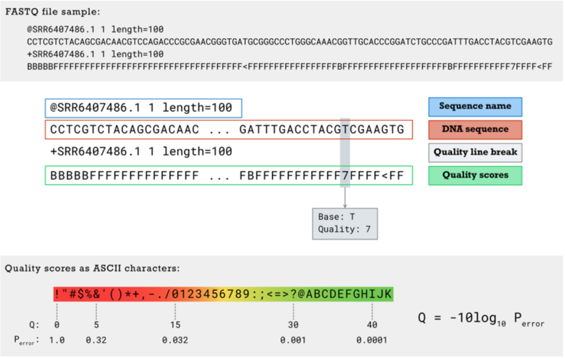
1. **SILVA** v138.2, (2024, Nov.)
2. **RDP** v19, (2023, Aug.)
3. **GreenGenes v2**, (2024, Sep.)

For fungi

- **UNITE v10**, (2025, Feb.)

## Pipeline for 16S rRNA amplicon




FASTQ file sample:

```
@SRR6407486.1 1 length=100
CCTCGTCTACAGCGACAACGTCCAGACCCGCGAACGGGTGATGCGGGCCCTGGGCAAACGGTTGCACCCGGATCTGCCCGATTTGACCTACGTCGAAGTG
+SRR6407486.1 1 length=100
BBBBBFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF<FFFFFFFFFFFFFFFFFBFFFFFFFFFFFFFFFFFFFFBFFFFFFFFFFF7FFFF<FF
```

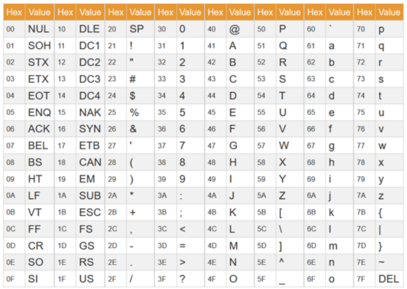
```
@SRR6407486.1 1 length=100
CCTCGTCTACAGCGACAAC ... GATTTGACCTACGTCGAAGTG
+SRR6407486.1 1 length=100
BBBBBFFFFFFFFFFFFFFF ... FBFFFFFFFFFFF7FFFF<FF
```

Quality scores as ASCII characters:

```
     !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJK
Q:       0     5        15            30        40
Perror:  1.0   0.32     0.032         0.001     0.0001
```

$$Q = -10\log_{10} P_{error}$$

ASCII table



| Hex | Value | Hex | Value | Hex | Value | Hex | Value | Hex | Value | Hex | Value | Hex | Value | Hex | Value |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | NUL | 10 | DLE | 20 | SP | 30 | 0 | 40 | @ | 50 | P | 60 | ` | 70 | p |
| 01 | SOH | 11 | DC1 | 21 | ! | 31 | 1 | 41 | A | 51 | Q | 61 | a | 71 | q |
| 02 | STX | 12 | DC2 | 22 | " | 32 | 2 | 42 | B | 52 | R | 62 | b | 72 | r |
| 03 | ETX | 13 | DC3 | 23 | # | 33 | 3 | 43 | C | 53 | S | 63 | c | 73 | s |
| 04 | EOT | 14 | DC4 | 24 | $ | 34 | 4 | 44 | D | 54 | T | 64 | d | 74 | t |
| 05 | ENQ | 15 | NAK | 25 | % | 35 | 5 | 45 | E | 55 | U | 65 | e | 75 | u |
| 06 | ACK | 16 | SYN | 26 | & | 36 | 6 | 46 | F | 56 | V | 66 | f | 76 | v |
| 07 | BEL | 17 | ETB | 27 | ' | 37 | 7 | 47 | G | 57 | W | 67 | g | 77 | w |
| 08 | BS | 18 | CAN | 28 | ( | 38 | 8 | 48 | H | 58 | X | 68 | h | 78 | x |
| 09 | HT | 19 | EM | 29 | ) | 39 | 9 | 49 | I | 59 | Y | 69 | i | 79 | y |
| 0A | LF | 1A | SUB | 2A | * | 3A | : | 4A | J | 5A | Z | 6A | j | 7A | z |
| 0B | VT | 1B | ESC | 2B | + | 3B | ; | 4B | K | 5B | [ | 6B | k | 7B | { |
| 0C | FF | 1C | FS | 2C | , | 3C | < | 4C | L | 5C | \ | 6C | l | 7C | \| |
| 0D | CR | 1D | GS | 2D | - | 3D | = | 4D | M | 5D | ] | 6D | m | 7D | } |
| 0E | SO | 1E | RS | 2E | . | 3E | > | 4E | N | 5E | ^ | 6E | n | 7E | ~ |
| 0F | SI | 1F | US | 2F | / | 3F | ? | 4F | O | 5F | _ | 6F | o | 7F | DEL |

Q=20, error rate = 0.01 (1%)
Q=30, error rate = 0.001 (0.1%)

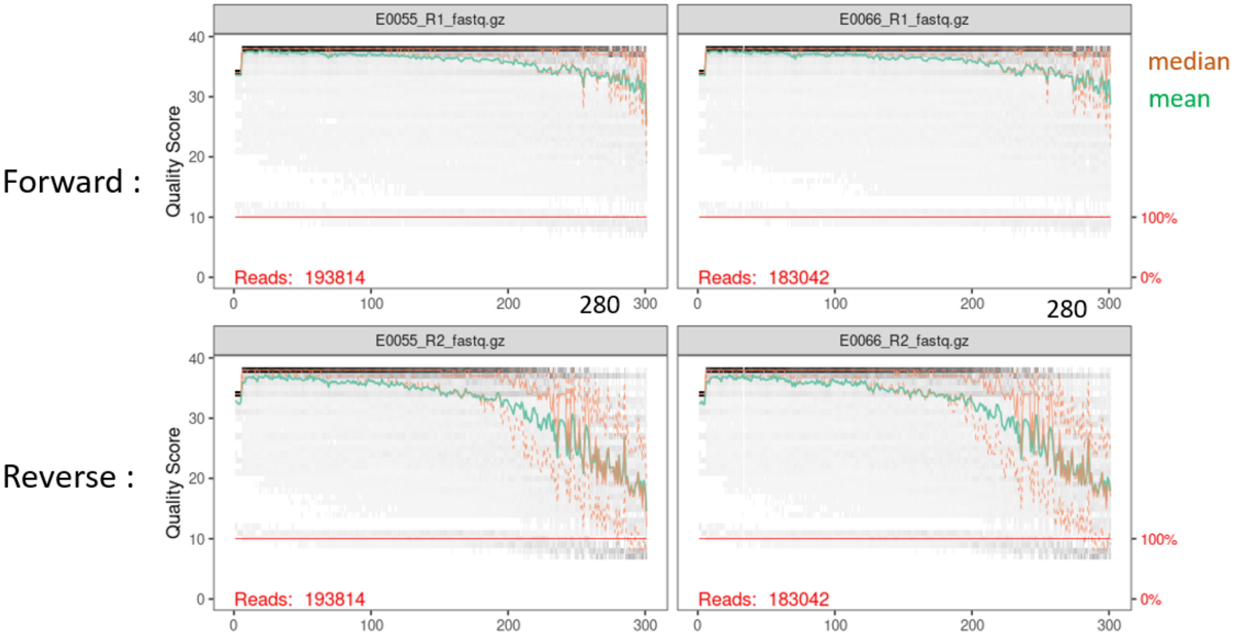
## FASTQ format

---

## DADA2 pipeline tutorial

Please refer directly to the original page

### Step 1. Inspect read quality profiles



### Step 2. Filter and trim

The key parameters for this step are `truncLen` and `trimLeft`.

To prevent adapter sequence contamination, use `trimLeft` to remove bases from the 5' end of the reads.

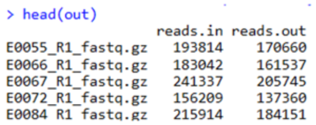
To ensure sequence quality, use `truncLen` to truncate reads after assignment.

There are two important criteria to consider:

- After filtering, at least 80% of the reads should be retained.
- The V3-V4 region typically ranges from 360 to 420 nt in length, so the combined length of paired-end reads after truncation should be at least 400 nt to ensure sufficient overlap.

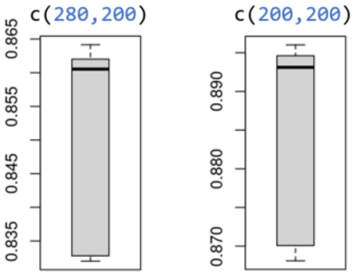
```
out <- filterAndTrim(fnFs, filtFs, fnRs, filtRs,
                     truncLen=c(280,200), trimLeft=10,
                     maxN=0, maxEE=c(2,2), truncQ=2, rm.phix=TRUE,
                     compress=TRUE, multithread=TRUE)
```

```
boxplot(out[,2]/out[,1])
```



```
> head(out)
                  reads.in reads.out
E0055_R1_fastq.gz   193814    170660
E0066_R1_fastq.gz   183042    161537
E0067_R1_fastq.gz   241337    205745
E0072_R1_fastq.gz   156209    137360
E0084 R1 fastq.gz   215914    184151
```

Criterion1 : out[,2]/out[,1] > 0.9   #0.8
Criterion2 : (280-10) + (200-10) > 400   #320



### Step 3. Sequence processing

Follow the standard procedures and verify the length of the merged sequences.

```
# Learn the Error Rates
errF <- learnErrors(filtFs, multithread=TRUE)
...

# Sample Inference
dadaFs <- dada(filtFs, err=errF, multithread=TRUE)
...

# Merge paired reads
mergers <- mergePairs(dadaFs, filtFs, dadaRs, filtRs, verbose=TRUE)
seqtab <- makeSequenceTable(mergers)
dim(seqtab) # [n_smaple, n_ASVs]

# Inspect distribution of sequence lengths
hist(nchar(getSequences(seqtab)))

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

# Assign taxonomy
taxa <- assignTaxonomy(seqtab.nochim, "~/tax/silva_nr_v132_train_set.fa.gz", multithread=TRUE)
taxa <- addSpecies(taxa, "~/tax/silva_species_assignment_v132.fa.gz")
```

### Step 4. Sample QC

```
getN <- function(x) sum(getUniques(x))
track <- cbind(out, sapply(dadaFs, getN), sapply(dadaRs, getN), sapply(mergers, getN), rowSums(seqtab.n
colnames(track) <- c("input", "filtered", "denoisedF", "denoisedR", "merged", "nonchim")
rownames(track) <- sample.names
mitochondria_id <- rownames(taxa)[taxa[,5] %in% "Mitochondria"]
unclassification_id <-rownames(taxa)[taxa[,1] != "Bacteria" | is.na(taxa[,2])]
keep_id <- rownames(taxa)[!rownames(taxa) %in% mitochondria_id)]
get_seqN <- function(table,list){
  if(length(list) == 0) { return(rep(0,nrow(table)))
  } else if(length(list) > 1) { return(rowSums(table[,list]))
  } else { return(sapply(table[,list],sum))}
}
track <- cbind(track, get_seqN(seqtab, mitochondria_id), get_seqN(seqtab,keep_id))
colnames(track) <-  c("input", "filtered", "denoisedF", "denoisedR", "merged", "nonchim","mitochondria"
track <- track[keep_sample,]
keep_sample <- rownames(track)[track[,8]/track[,6] > 0.5]  # remove the sample with host contamination
seqtab <- seqtab[keep_sample,keep_id]
nozeroASV <- colnames(seqtab)[colSums(seqtab) > 0]
seqtab <- seqtab[,nozeroASV]
taxa <- taxa[nozeroASV,]
```

### Step 5. Serial number and output

```
source("source/utils.R")
output_path <- '~/dada2_output'
DADA2Adapter(seqtab.nochim, taxa, output_path)  # DADA2Adapter function is sourced from utils.R
dir(output_path)

data <- read.csv(paste0(output_path,'/ASV_table.txt'), sep = '\t')
taxa <- read.csv(paste0(output_path,'/ASV_taxa_table.txt'), sep = '\t')
```

### Step 6. Phylogenetic tree building (1)

The phylogenetic tree can be directly constructed using `DECIPHER`. However, fitting a GTR model is time-consuming (5hr-10hr), and installing `DECIPHER` can be challenging.

```
library(dada2)
library(DECIPHER)
library(phangorn)

seqs <- getSequences(seqtab.nochim)
magnitude <- ceiling(log10(ncol(seqtab.nochim)))
ASVID <- sprintf(paste0("ASV%0", magnitude, "d"), 1:n_ASVs)


names(seqs) <- ASVID
alignment <- AlignSeqs(DNAStringSet(seqs), anchor=NA,verbose=TRUE)

phangAlign <- phyDat(as(alignment, "matrix"), type="DNA")
dm <- dist.ml(phangAlign)
treeNJ <- NJ(dm)
fit = pml(treeNJ, data=phangAlign)
fitGTR <- update(fit, k=4, inv=0.2)
fitGTR <- optim.pml(fitGTR, model="GTR", optInv=TRUE, optGamma=TRUE,
                    rearrangement = "stochastic", control = pml.control(trace = 0))
fitGTR$tree

write.tree(fitGTR$tree, file = paste0(output_path,'/ASV.tree'), append = FALSE,
           digits = 10, tree.names = FALSE)
```

### Step 6. Phylogenetic tree building (2)

So we provider an alternative approach. Since 16S ribosomal sequences contained the non-coding sequence and secondary structure, the alignment method was considered. We suggest to use ssu-align to do alignment for phylogenetic inference. The detailed process is as follows.

```
# ssu-align installation
$ wget eddylab.org/software/ssu-align/ssu-align-0.1.1.tar.gz;
$ tar xfz ssu-align-0.1.1.tar.gz
$ cd ssu-align-0.1.1
$ ./configure
$ make
```

```
# install the man pages and programs in system-wide directories:
$ make install

$ export PATH="$PATH:/usr/local/bin"
$ export MANPATH="$MANPATH:/usr/local/share/man"
$ export SSUALIGNDIR="/usr/local/share/ssu-align-0.1.1"

# conduct SSU-alignment
$ cd ~/Course/NDMC_2025/tmp
$ ssu-align ASV.fasta ssuout
$ cd ssuout
```

The alignment result is generated in `ssuout/ssuout.bacteria.stk`.

We can convert the `.stk` file to FASTA format using `alignment_transform.R`, located in the source folder.

```
$ Rscript --vanilla ~/Course/NDMC_2025/source/alignment_transform.R ~/Course/NDMC_2025/tmp/ssuout/ssuou
```

In order to shorten the processing time, we use the FastTree for maximum-likelihood phylogenetic tree completion. If time is available, RAxML is suggested to get the better quality tree.

```
$ cd ~/Course/NDMC_2025/tmp
$ ~/bin/FastTree -gtr -nt ./ssuout/ssu_align.fasta > ./ASV.tree
```

## Diversity Analysis

The fecal microbiota data which from PRJEB6070 (Zeller et al., 2016) is used in this demonstration. We subsetted this dataset, including 20 patients each from the normal and cancer groups, and used the DADA2 pipeline to complete the analysis up to the taxonomic assignment step.

Please ensure that the packages listed in `prep_help.R` are installed.

```
library(dada2)
library(dplyr)
library(DECIPHER)
library(phangorn)
library(matrixStats)
library(ggpubr)
library(ggplot2)
library(picante)
library(phyloseq)
library(patchwork)
library(rstatix)
library(tidyverse)
library(vegan)
library(scales)

source("source/utils.R")
```

```
dataset_path <- "~/Course/NDMC_2025/data/Zeller_CRC.RData"
load(dataset_path)
```

```
print(ls())
##  [1] "assignModul"       "DADA2Adapter"      "dataset_path"
##  [4] "get_composition"   "ggrare"            "GMMviz"
##  [7] "LEfSe_preparation" "make_edgetable"    "make_net"
## [10] "make_nodetable"    "meta"              "phyloseq_to_edgeR"
## [13] "seqtab"            "seqtab.nochim"     "taxa"
## [16] "track"             "track_ab"          "track_ta"

# Check patients condition
print(table(meta$Class))
##
## Cancer Normal
##     20     20
print(table(meta$diagnosis))
##
## Adenoma  Cancer  Normal
##       6      20      14

# The clinical factor for time-to-event testing
meta$event <- ifelse(meta$Class == 'Cancer', 1, 0)  # as events
meta$duration <- meta$age                           # as duration
```

We designed the DADA2Adapter function to bridge the data into the PreLect pipeline.

```
# set the working  directory
output_path <- '/home/yincheng23/Course/NDMC_2025/tmp'

# generate the raw count table, taxa table and ASV sequence fasta
DADA2Adapter(seqtab.nochim, taxa, output_path) # DADA2Adapter function is sourced from utils.R
dir(output_path)
## [1] "ASV_table.txt"      "ASV_taxa_table.txt" "ASV.fasta"
## [4] "ASV.tree"           "for_PICRUSt2.tsv"   "PICRUSt2"
## [7] "PreLect_tmp"        "ssuout"

# load the data we need
data <- read.csv(paste0(output_path,'/ASV_table.txt'), sep = '\t')
taxa <- read.csv(paste0(output_path,'/ASV_taxa_table.txt'), sep = '\t')
taxa[is.na(taxa)] <- "unidentified"
```

## Alpha diversity

Alpha diversity describes the species diversity within a community.

### Rarefaction normalization

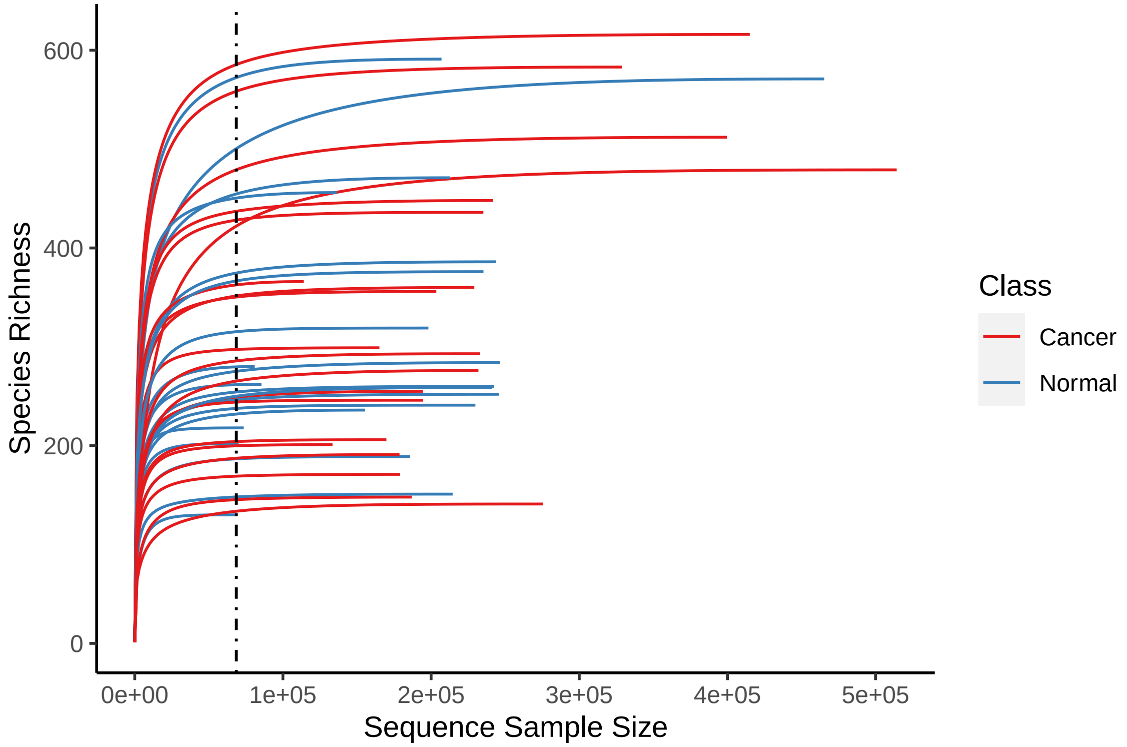
Since various range of sequencing depth in data, we usually conduct the rarefaction normalization to 'fairly' compare the diversity metrics.

```
# Rarefaction curve
ASV = otu_table(data, taxa_are_rows = TRUE)
TAX = tax_table(as.matrix(taxa))
```

```
physeq = phyloseq(ASV, TAX)
sampledata = sample_data(meta)
physeq = merge_phyloseq(physeq, sampledata)

min_depth <- min(colSums(data))
print(min_depth)
```

```
## [1] 68528
```

```
# ggrare function is sourced from utils.R
ggrare(physeq, step = 1500, colour = "Class", se = FALSE) +
  scale_colour_brewer(palette="Set1") +
  geom_vline(xintercept = min_depth, linetype = 'dotdash') +
  theme(axis.line = element_line(linetype = 1,colour = 'black'),
  panel.background = element_rect(I(0)),
  panel.grid.major = element_line(colour = NA),
  panel.grid.minor = element_line(colour = NA))
```



The upper picture show the library size vs. number of observed species, in rarefaction process, we limited the all samples in minimum depth **(black dotdash)**, and then randomly discarding reads from larger samples until the number of remaining samples is equal to this threshold

```
# Rarefaction
print(colSums(data)[1:16])
## ERR475473 ERR475476 ERR475478 ERR475480 ERR475483 ERR475484 ERR475485 ERR475493
##     73475    155494    207049     68528     70074     85513     80933    203534
## ERR475500 ERR475504 ERR475513 ERR475518 ERR475521 ERR475527 ERR475528 ERR475529
##    514255    465384    415085    242616    399663    328836    136463    133480
```

```
min <- min(colSums(data))
data_rarefied <- t(rrarefy(t(data), min))
## Warning in rrarefy(t(data), min): function should be used for observed counts,
## but smallest count is 2
data_rarefied <- data_rarefied[rowSums(data_rarefied) > 0,]
print(colSums(data_rarefied)[1:16])
## ERR475473 ERR475476 ERR475478 ERR475480 ERR475483 ERR475484 ERR475485 ERR475493
##     68528     68528     68528     68528     68528     68528     68528     68528
## ERR475500 ERR475504 ERR475513 ERR475518 ERR475521 ERR475527 ERR475528 ERR475529
##     68528     68528     68528     68528     68528     68528     68528     68528
```

After rarefaction, All the sample size are equality, we can calculate the various alpha index to each sample.

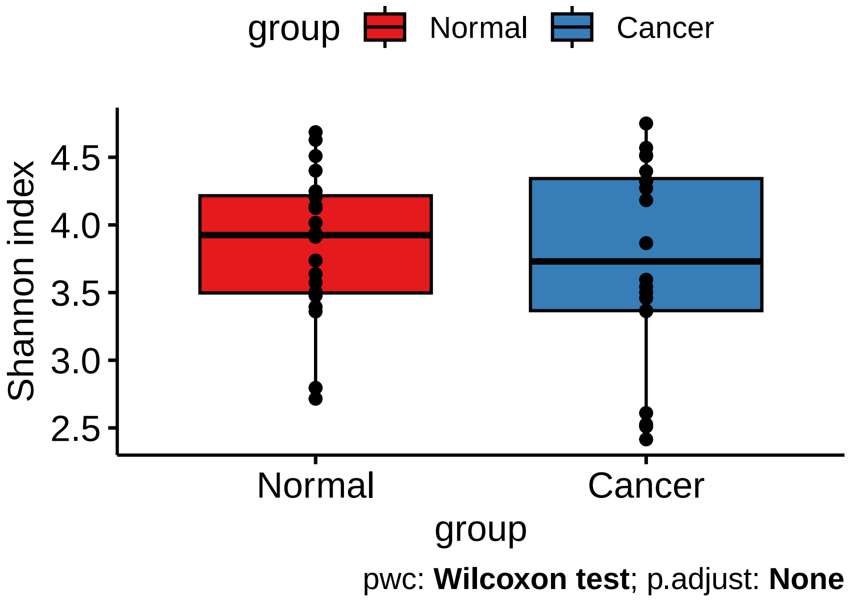
```
dir(output_path)
## [1] "ASV_table.txt"      "ASV_taxa_table.txt" "ASV.fasta"
## [4] "ASV.tree"           "for_PICRUSt2.tsv"   "PICRUSt2"
## [7] "PreLect_tmp"        "ssuout"

# load the data we need
tree <- read.tree(paste0(output_path,'/ASV.tree'))
a <- cophenetic(tree)     #check the root by farthest phylogenetic distance ASV
rowSums(a)[order(rowSums(a), decreasing = T )][1:8]
##  ASV2154  ASV1810  ASV0186  ASV3210  ASV2490  ASV2300  ASV1408  ASV1814
## 3417.648 3349.283 3266.763 3223.351 3205.014 3199.102 3170.137 3145.933
tree <- root(tree, "ASV2154", resolve.root = T)

AlphaIndex <- data.frame(Shannon = diversity(t(data_rarefied) ,index  = "shannon"),
                         Chao1 = estimateR(t(data_rarefied))[2,],
                         Simpson = diversity(t(data_rarefied) ,index  = "simpson"),
                         invSimpson = diversity(t(data_rarefied) ,index  = "invsimpson"),
                         PD = pd(t(data_rarefied), tree)[,1],
                         group = meta$Class)
```

### Shannon diversity

$$H_{sw} = -\sum_{i=1}^{s} (\frac{n_i}{N}) \ln(\frac{n_i}{N})$$

Shannon and Wiener (1963) is base on information theory, $N$ is total number of individual, and $n_i$ is the number of individual belong in species $i$. The more the number of species and the more evenly distributed the individuals are, the higher the index it get. Therefore, $H_{sw}$ can be regarded as `equitability`.

```
pwc <- wilcox_test(Shannon ~ group, paired = F, p.adjust.method = "None", data = AlphaIndex)
pwc <- pwc %>% add_xy_position(x = "group")
ggboxplot(AlphaIndex, x = "group", y = "Shannon", add = "point", fill = "group") +
  scale_fill_brewer(palette = "Set1") + ylab("Shannon index") +
  stat_pvalue_manual(pwc, hide.ns = TRUE) +
  labs(caption = get_pwc_label(pwc))
```

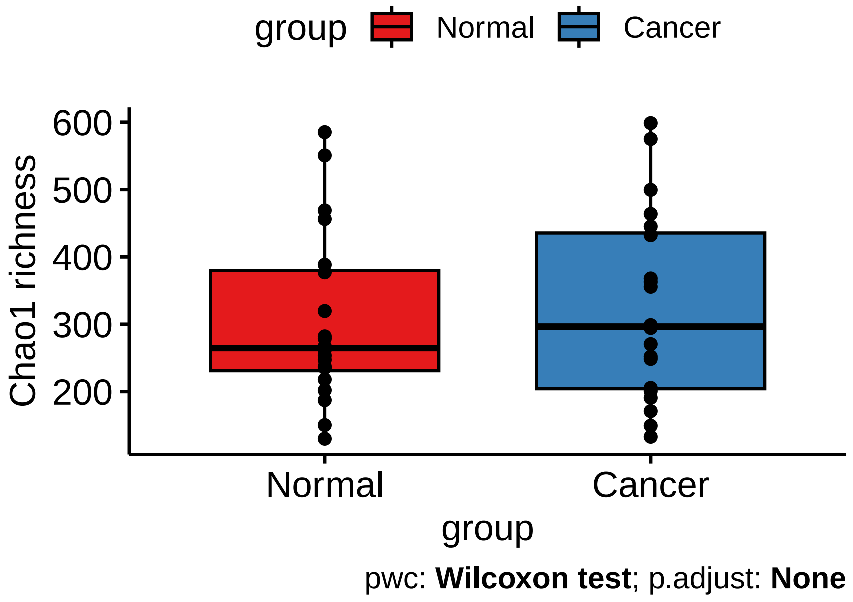


### Chao1 richness

$$S = S_{obs} + \frac{F_1^2}{2F_2}$$

In Chao1 estimator (Chao, A. 1984). $S_{obs}$ is indicated the number of observed species, $F_1$ is the number of species which only once. and $F_2$ is the number of species which twice in community. Higher $F_1$ indicates that the number of non-observed species is likely to be higher. It is expected that the community will have a relatively high abundance of species. but $F_2$ show the species have occurred at least twice, then the chance of new species occurring in the community is low.

```
pwc <- wilcox_test(Chao1 ~ group, paired = F, p.adjust.method = "None", data = AlphaIndex)
pwc <- pwc %>% add_xy_position(x = "group")
ggboxplot(AlphaIndex, x = "group", y = "Chao1", add = "point", fill = "group") +
  scale_fill_brewer(palette = "Set1") + ylab("Chao1 richness") +
  stat_pvalue_manual(pwc, hide.ns = TRUE) +
  labs(caption = get_pwc_label(pwc))
```

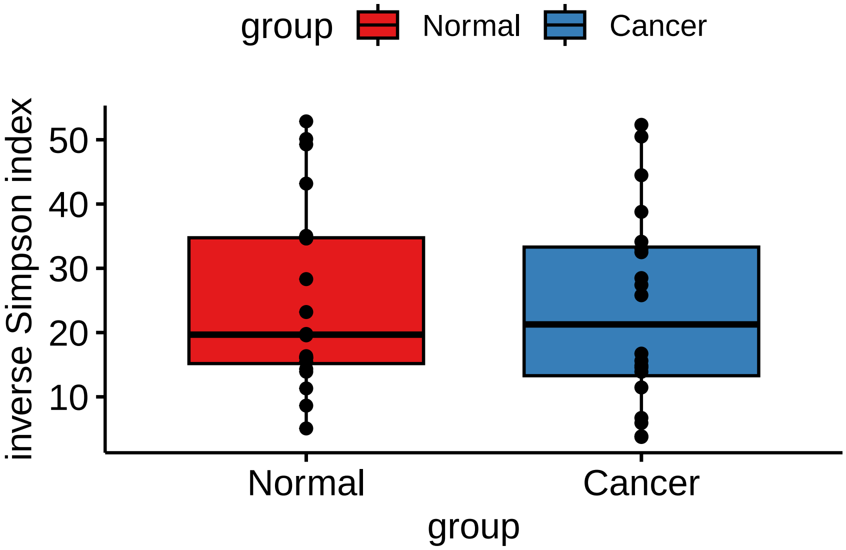


**Simpson index**

$$D_s = \sum_{i=1}^{s} (\frac{n_i}{N})^2 \ ; \quad D_{s'} = \frac{1}{D_s}$$

Simpson (1949) is measure the dominance in single community. if some species is dominant in community, the Simpson will be higher. so it can be regarded as `concentration index`. In other words, the inverse Simpson index is show the `evenness` in community.

```r
pwc <- wilcox_test(Simpson ~ group, paired = F, p.adjust.method = "None", data = AlphaIndex)
pwc <- pwc %>% add_xy_position(x = "group")
ggboxplot(AlphaIndex, x = "group", y = "invSimpson", add = "point", fill = "group") +
  scale_fill_brewer(palette = "Set1") + ylab("inverse Simpson index") +
  stat_pvalue_manual(pwc, hide.ns = TRUE) +
  labs(caption = get_pwc_label(pwc))
```

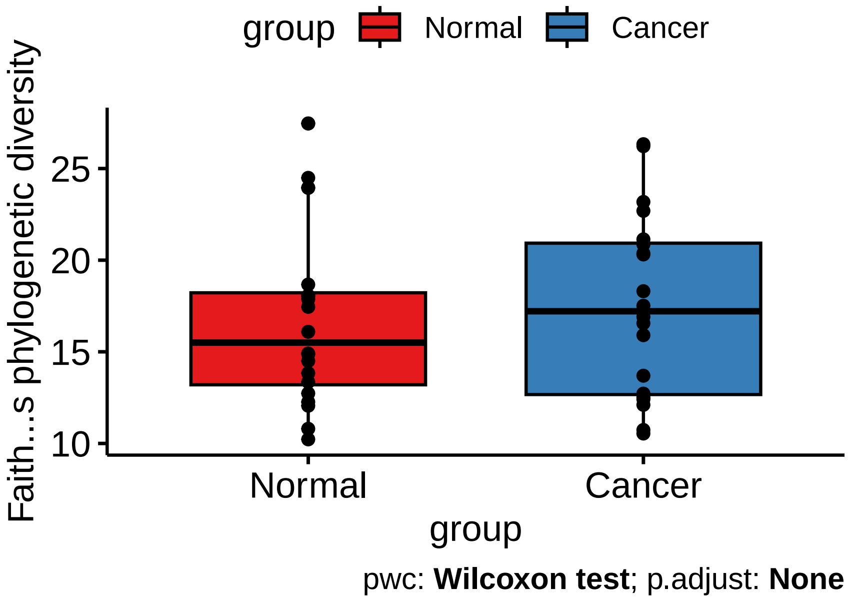


pwc: **Wilcoxon test**; p.adjust: **None**

### Phylogenetic diversity

Faith's Phylogenetic Diversity (Faith D., 1992) which is defined as the sum of the branch lengths of a phylogenetic tree connecting all species, this means that PD indicates `Feature diversity`.

```
pwc <- wilcox_test(PD ~ group, paired = F, p.adjust.method = "None", data = AlphaIndex)
pwc <- pwc %>% add_xy_position(x = "group")
ggboxplot(AlphaIndex, x = "group", y = "PD", add = "point", fill = "group") +
  scale_fill_brewer(palette = "Set1") + ylab("Faith's phylogenetic diversity") +
  stat_pvalue_manual(pwc, hide.ns = TRUE) +
  labs(caption = get_pwc_label(pwc))
```



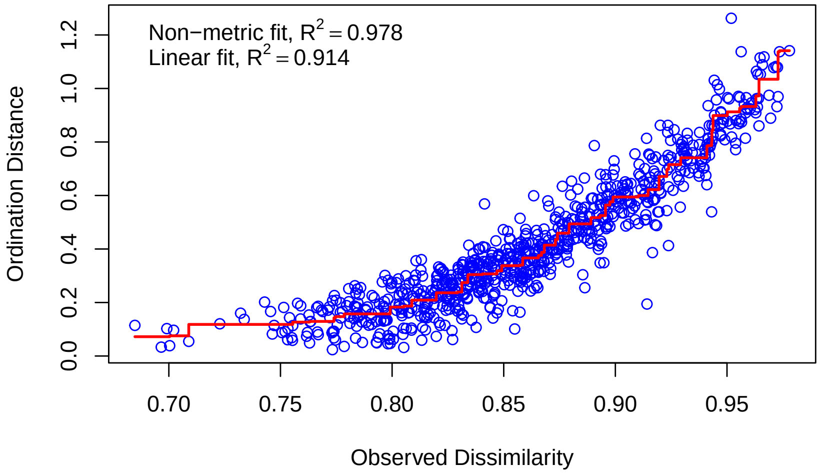
## Beta diversity

### Bray-Curtis distance

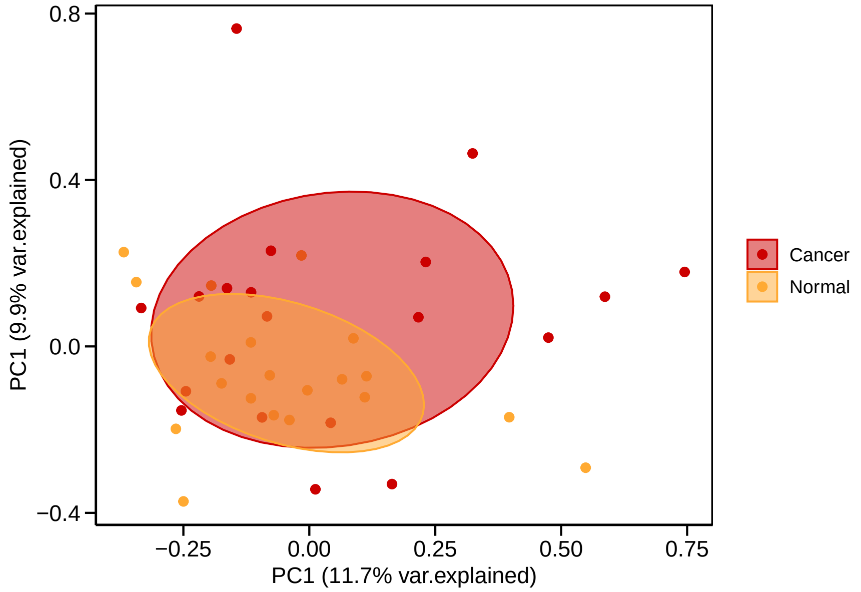
$$D_{BC} = \frac{\sum_{i=1}^{S} |M_{i1} - M_{i2}|}{\sum_{i=1}^{S} M_{i1} + M_{i2}}$$

In Bray-Curtis distance, $S$ is indicates the total number of species in two communities, $M_{i1}$ is means the number of species $i$ in community 1, and so on. This method is similar to Sørensen index. and usually utilizes non-metric multidimensional scaling nMDS for dimension reduction.

```
NMDS=metaMDS(t(data_rarefied), distance = "bray")
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 0.149106
## Run 1 stress 0.1503017
## Run 2 stress 0.1491056
## ... New best solution
## ... Procrustes: rmse 0.0005981917  max resid 0.002768234
## ... Similar to previous best
## Run 3 stress 0.1491057
## ... Procrustes: rmse 1.548925e-05  max resid 7.090235e-05
## ... Similar to previous best
## Run 4 stress 0.1503017
## Run 5 stress 0.1491056
## ... New best solution
## ... Procrustes: rmse 4.216836e-05  max resid 0.0001434376
## ... Similar to previous best
## Run 6 stress 0.1491057
## ... Procrustes: rmse 0.0001441396  max resid 0.0005599521
## ... Similar to previous best
## Run 7 stress 0.1912499
## Run 8 stress 0.1491055
## ... New best solution
## ... Procrustes: rmse 0.0002371133  max resid 0.001049122
## ... Similar to previous best
## Run 9 stress 0.1503017
## Run 10 stress 0.1503017
## Run 11 stress 0.1503084
## Run 12 stress 0.1491058
## ... Procrustes: rmse 0.0001865112  max resid 0.0008890889
## ... Similar to previous best
## Run 13 stress 0.1503084
## Run 14 stress 0.1491059
## ... Procrustes: rmse 0.0002518848  max resid 0.001206103
## ... Similar to previous best
## Run 15 stress 0.1503017
## Run 16 stress 0.1491055
## ... Procrustes: rmse 0.0001621694  max resid 0.0007371812
## ... Similar to previous best
## Run 17 stress 0.2091891
## Run 18 stress 0.1503017
## Run 19 stress 0.1491058
## ... Procrustes: rmse 0.0002246621  max resid 0.00102273
## ... Similar to previous best
## Run 20 stress 0.1503017
## *** Best solution repeated 5 times
NMDSplot <- as.data.frame(NMDS$points)
NMDSplot$group <- meta$Class
prop <- cmdscale(vegdist(t(data_rarefied), method = "bray"), k = 2, eig = T, add = T )
prop <- round(prop$eig*100/sum(prop$eig),1)
print(prop[1:8]) # chick the proportion of variance explained
## [1] 11.7  9.9  7.6  5.7  5.4  4.5  3.8  3.7
stressplot(NMDS) # chick the fitness in nMDS
```



```
ggscatter(NMDSplot, x = "MDS1", y = "MDS2",combine  = T, color = 'group',
          ellipse.type = "norm", ellipse = T,ellipse.level = 0.5, ellipse.alpha = 0.5, repel = TRUE) +
          scale_color_manual(values = c("#CC0000","#FFAA33"))+
          scale_fill_manual(values = c("#CC0000","#FFAA33")) +
          xlab(paste0(c('PC1 (', prop[1],'% var.explained)'), collapse = "")) +
          ylab(paste0(c('PC1 (', prop[2],'% var.explained)'), collapse = "")) +
          theme(panel.background = element_rect(fill = 'transparent'),
                panel.grid = element_blank(),
                axis.ticks.length = unit(0.4,"lines"),
                axis.ticks = element_line(color='black'),
                axis.line = element_line(colour = "black"),
                legend.title=element_blank(),
                legend.position  = 'right')
```



Ellipse type can choose the `convex`, `confidence`, `t`, `euclid`.

### Unifrac distance

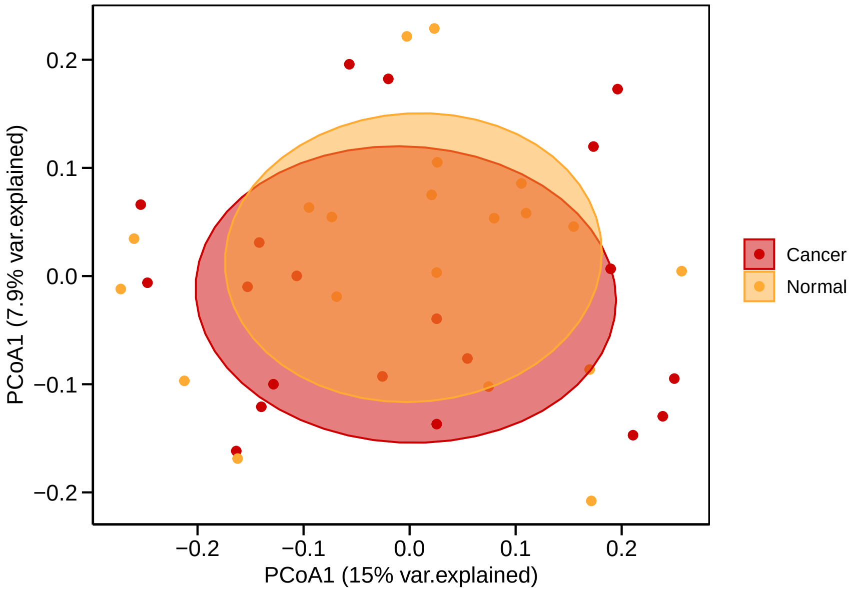
- unWeighted

$$U_{uw} = \frac{\sum_{i=1}^{N} l_i|A_i - B_i|}{\sum_{i=1}^{N} max(A_i + B_i)}$$

- Weighted

$$U_w = \frac{\sum_{i=1}^{n} b_i|\frac{A_i}{A_T} - \frac{B_i}{B_T}|}{\sum_{j=1}^{S} L_j}$$

```
ASV = otu_table(data_rarefied, taxa_are_rows = TRUE)
TAX = tax_table(as.matrix(taxa))
physeq = phyloseq(ASV, TAX, tree)
Unif = UniFrac(physeq, weighted = F, normalized = F, parallel = F)  # if weighted = TRUE; then weighted
Unif_d <- pcoa(Unif)
Unifplot <- data.frame(axis1 = as.numeric(Unif_d$vectors[,1]),
                       axis2 = as.numeric(Unif_d$vectors[,2]))
Unifplot$group <- meta$Class
prop <- cmdscale(Unif, k = 2, eig = T, add = T)
prop <- round(prop$eig*100/sum(prop$eig),1)
print(prop[1:8]) # chick the proportion of variance explained
## [1] 15.0  7.9  6.3  4.0  3.7  3.4  3.3  3.2

ggscatter(Unifplot, x = "axis1", y = "axis2",combine = T, color = 'group',
          ellipse.type = "norm", ellipse = T,ellipse.level = 0.5, ellipse.alpha = 0.5, repel = TRUE) +
          scale_color_manual(values = c("#CC0000","#FFAA33"))+
          scale_fill_manual(values = c("#CC0000","#FFAA33")) +
          xlab(paste0(c('PCoA1 (', prop[1],'% var.explained)'), collapse = "")) +
          ylab(paste0(c('PCoA1 (', prop[2],'% var.explained)'), collapse = "")) +
          theme(panel.background = element_rect(fill = 'transparent'),
                panel.grid = element_blank(),
                axis.ticks.length = unit(0.4,"lines"),
                axis.ticks = element_line(color='black'),
                axis.line = element_line(colour = "black"),
                legend.title=element_blank(),
                legend.position  = 'right')
```



### ANOSIM

Analysis of similarities (ANOSIM) is a non-parametric statistical test widely used in the field of ecology. As an ANOVA-like test, where instead of operating on raw data, operates on a ranked dissimilarity matrix.

Given a matrix of rank dissimilarities between a set of samples, each solely belong to one treatment group, the ANOSIM tests whether we can reject the null hypothesis that the similarity between groups is greater than or equal to the similarity within the groups.

The test statistic R is calculated in the following way:

$$R = \frac{\bar{r_B} - \bar{r_W}}{M/2}$$

where $\bar{r_B}$ is the average of rank similarities of pairs of samples (or replicates) originating from different sites, $\bar{r_W}$ is the average of rank similarity of pairs among replicates within sites, and $M = \mathrm{n}(\mathrm{n}-1)/2$ where n is the number of samples.

```
anosim(vegdist(t(data_rarefied), method = "bray"), meta$Class)
##
## Call:
## anosim(x = vegdist(t(data_rarefied), method = "bray"), grouping = meta$Class)
## Dissimilarity: bray
##
## ANOSIM statistic R: 0.08716
##       Significance: 0.004
##
## Permutation: free
## Number of permutations: 999
anosim(Unif, meta$Class)
##
## Call:
## anosim(x = Unif, grouping = meta$Class)
## Dissimilarity:
##
## ANOSIM statistic R: 0.01125
##       Significance: 0.274
##
## Permutation: free
## Number of permutations: 999
```

#### adonis2

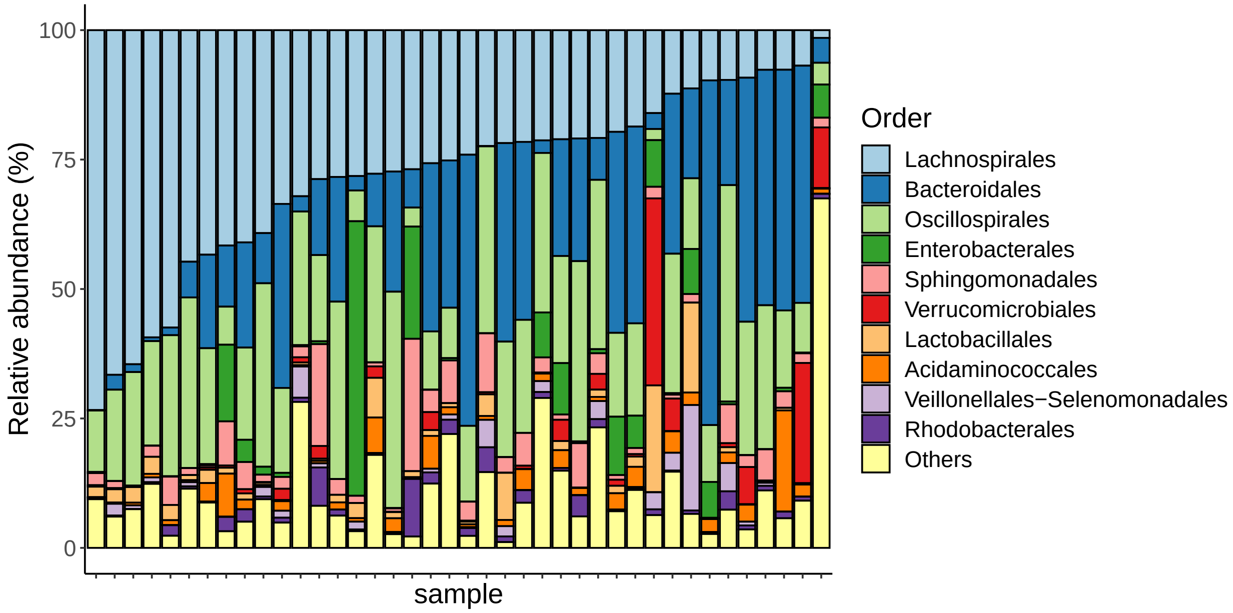
Permutational Multivariate Analysis of Variance (`adonis`),

```
adonis2(t(data_rarefied) ~ Class, data = meta, method= "bray")
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = t(data_rarefied) ~ Class, data = meta, method = "bray")
##          Df SumOfSqs      R2    F Pr(>F)
## Class     1   0.5381 0.04186 1.66  0.015 *
## Residual 38  12.3178 0.95814
## Total    39  12.8559 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
adonis2(Unif ~ Class, data = meta)
## Permutation test for adonis under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = Unif ~ Class, data = meta)
##          Df SumOfSqs      R2      F Pr(>F)
## Class     1   0.1782 0.02849 1.1142  0.244
## Residual 38   6.0787 0.97151
## Total    39   6.2569 1.00000
```

### Community composition

sort by major species decrease

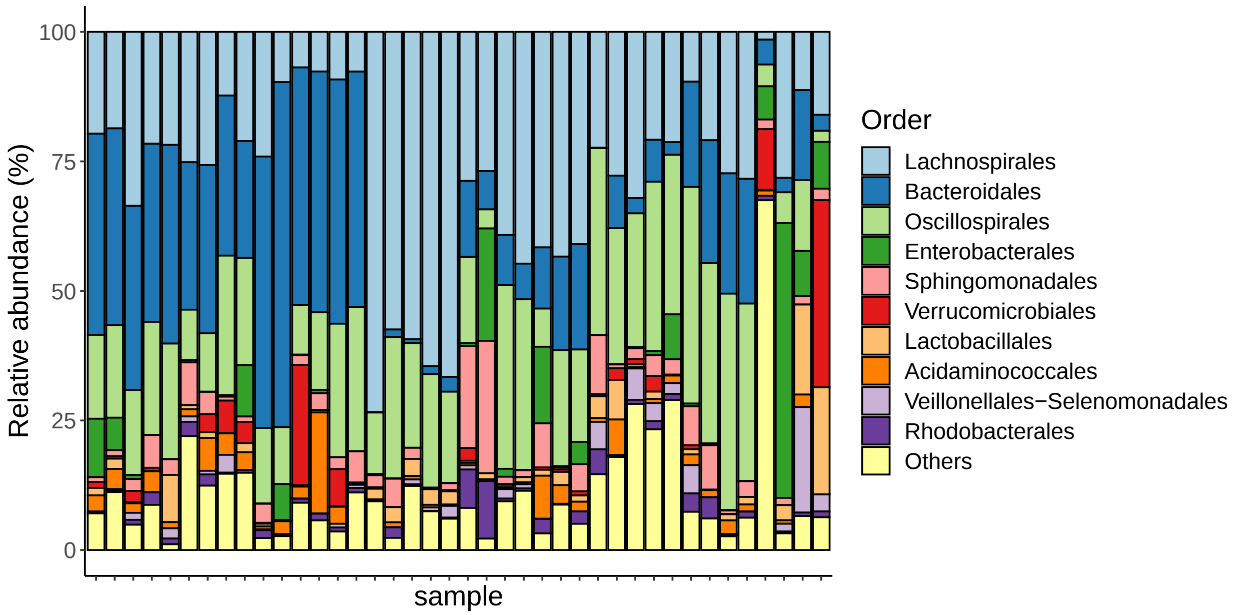
```
# choose the taxonomic rank for visualization
print(colnames(taxa))
## [1] "Kingdom" "Phylum"  "Class"   "Order"   "Family"  "Genus"   "Species"
# get_composition function is sourced from utils.R
compair <- get_composition(data, taxa, 'Order', meta, 'Class')

ggplot(compair, aes(x = sample, y = percentage, fill = taxa)) +
  geom_bar(stat="identity",colour = "black") + scale_fill_brewer(palette = "Paired") +
  labs(fill = "Order") + ylab("Relative abundance (%)") +
  #facet_grid(~group, space="free", scales="free") + # separate the group
  theme(axis.text.x = element_blank(),
        axis.line = element_line(linetype = 1,colour = 'black'),
        panel.background = element_rect(I(0)),
        panel.grid.major = element_line(colour = NA),
        panel.grid.minor = element_line(colour = NA),
        text = element_text(size=16))
```



sort by hierarchical clustering

```
compair_2D <- reshape(compair[,1:3], idvar = "sample", timevar = "taxa", direction = "wide")
rownames(compair_2D) <- compair_2D[,1]
compair_2D <- compair_2D[,2:ncol(compair_2D)]
horder <- hclust(dist(compair_2D), method = 'ward.D')
horder <- horder$labels[horder$order]
compair$sample <- factor(compair$sample, levels = horder)
```

```
ggplot(compair, aes(x = sample, y = percentage, fill = taxa)) +
  geom_bar(stat="identity",colour = "black") + scale_fill_brewer(palette = "Paired") +
  labs(fill = "Order") + ylab("Relative abundance (%)") +
  #facet_grid(~group, space="free", scales="free") + # separate the group
  theme(axis.text.x = element_blank(),
        axis.line = element_line(linetype = 1,colour = 'black'),
        panel.background = element_rect(I(0)),
        panel.grid.major = element_line(colour = NA),
        panel.grid.minor = element_line(colour = NA),
        text = element_text(size=16))
```



# Environmental Analysis

To demonstrate how the environmental factor impacting in microbiota, we use wetland microbiota which characterised by ions concentration. the soil samples were collected from 4 wetland in southern Taiwan and named as AA, BB, CC, DD respectively. and each sample had be measured with LCMS.

```
wetland <- read.table("~/Course/NDMC_2025/data/wetland/ASV_table.txt", sep = "\t", stringsAsFactors = F
ion <- read.table("~/Course/NDMC_2025/data/wetland/ion.txt", sep = "\t", stringsAsFactors = F)
wetland <- wetland[,rownames(ion)]
head(wetland[,1:5])
##                 AA1 AA2 AA3 AA4  AA5
## ASV_V3V400009     0 196   0  13    7
## ASV_V3V400024     0   0   0   0    0
## ASV_V3V400036  2846  26 552   0 2030
## ASV_V3V400040     0   0   0   0    0
## ASV_V3V400042   821 793 947 633  427
## ASV_V3V400043     0   0   0   3    0
head(ion)
##          Na      K      Mg      Ca      Cl    Br    SO4     EC    pH  Salinity
## AA1   538.0  45.76   93.40   78.33   775.3   0.0  321.9   3600 7.143  1.610759
## AA2   691.1  47.13   87.07   49.96   703.3   0.0  164.2   4020 7.675  1.799610
## AA3   716.1  56.79  124.20  151.00   920.1   0.0  418.2   4330 6.862  2.040961
## AA4   651.2  80.48   97.89   89.33   727.9   0.0  244.4   3640 6.330  1.755768
## AA5   576.9   0.00   85.44   83.32   748.3   0.0  326.7   3510 6.670  1.650724
## BB1 23830.0 922.90 2989.00 1494.00 37540.0 266.6 5298.0 137700 6.701 74.292982
```

## Mantel test

The Mantel test computes the Pearson or Spearman correlation between two distance matrices and evaluates its significance using a permutation test.

```
min <- min(colSums(wetland))
wetland_rarefied <- t(rrarefy(t(wetland), min))
wetland_rarefied <- wetland_rarefied[rowSums(wetland_rarefied) > 0,]

manteldf <- data.frame(Factor = 0, rho = 0, p_value = 0)
count <- 1
for(i in 1:ncol(ion)){
  Mantel_res <- mantel(dist(ion[,i]),vegdist(t(wetland_rarefied)), method = "spearman")
  manteldf[count,1] <- colnames(ion)[i]
  manteldf[count,2] <- round(Mantel_res$statistic,4)
  manteldf[count,3] <- round(Mantel_res$signif,4)
  count = count + 1
}
manteldf
##      Factor    rho p_value
## 1        Na 0.7629   0.001
## 2         K 0.1049   0.121
## 3        Mg 0.6716   0.001
## 4        Ca 0.7391   0.001
## 5        Cl 0.7424   0.001
## 6        Br 0.1665   0.033
## 7       SO4 0.6853   0.001
## 8        EC 0.7235   0.001
## 9        pH 0.2262   0.020
## 10 Salinity 0.7553   0.001
```

## BioENV

This finds the best subset of environmental variables, so that the Euclidean distances of scaled environmental variables have the maximum (rank) correlation with community dissimilarities.

```
bio <- bioenv(as.formula(paste("t(wetland_rarefied)~ ", paste(colnames(ion), collapse = " + "))), ion)
## 1023 possible subsets (this may take time...)
bio <- summary(bio)
df <- data.frame(Factor = bio$variables, rank = bio$size, rho = bio$correlation)
df
##                                  Factor rank       rho
## 1                                    Na    1 0.7629216
## 2                           Ca Salinity    2 0.7699462
## 3                       Na Ca Salinity    3 0.7783320
## 4                    Na Ca EC Salinity    4 0.7773259
## 5                 Na Ca Cl EC Salinity    5 0.7741496
## 6              Na Ca Cl EC pH Salinity    6 0.7634334
## 7           Na Mg Ca Cl EC pH Salinity    7 0.7617276
## 8       Na Mg Ca Cl SO4 EC pH Salinity    8 0.7566520
## 9    Na Mg Ca Cl Br SO4 EC pH Salinity    9 0.7420710
## 10 Na K Mg Ca Cl Br SO4 EC pH Salinity   10 0.6766375
```

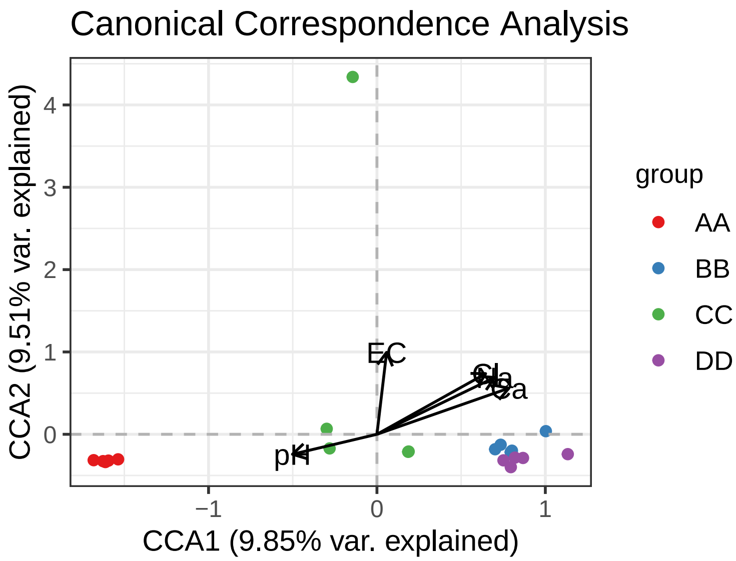
## CCA

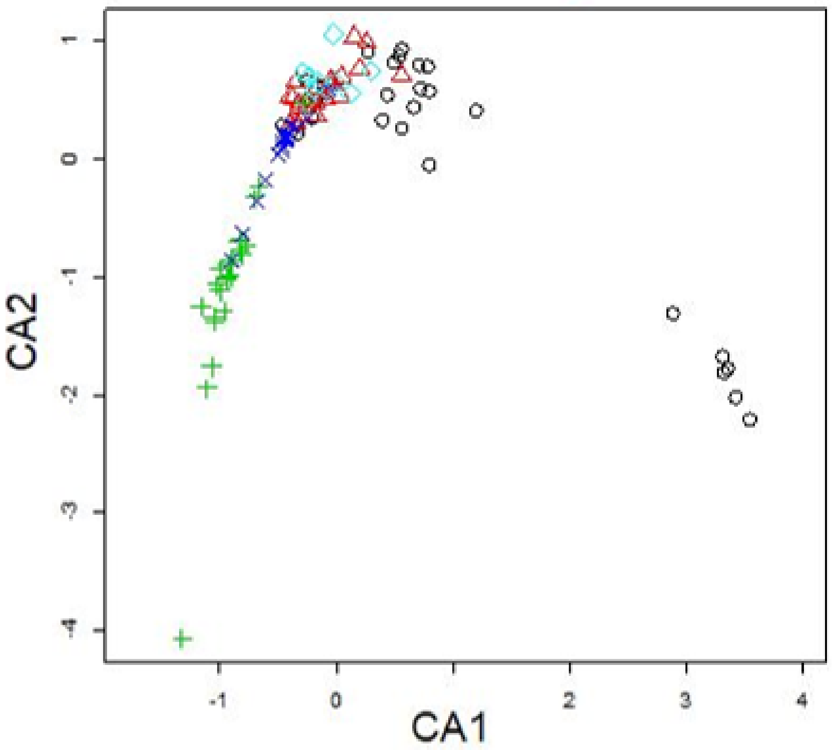
Canonical Correspondence Analysis (CCA) is an ordination technique that directly relates species composition to environmental variables. CCA is based on Chi-square distances, which measure differences in species composition between samples, and uses a constrained ordination approach to maximize the correlation between species distributions and environmental gradients.

Given a species abundance matrix and an environmental variable matrix , CCA seeks to find axes that maximize the weighted correlation between species composition and environmental variables. The species scores are derived based on weighted averaging, ensuring that species are positioned optimally along the environmental gradients. This makes CCA particularly suitable for ecological datasets where species exhibit unimodal responses to environmental gradients.

```
#CCA <- cca(as.formula(paste("t(wetland_rarefied)~", paste(colnames(ion), collapse = " + "))), ion)
CCA <- cca(t(wetland_rarefied)~ Na + Ca + Cl+ EC + Salinity + pH, ion)
#cca_temp <- anova.cca(CCA)
cca_term <- anova.cca(CCA, by="terms")
cca_term
## Permutation test for cca under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## Model: cca(formula = t(wetland_rarefied) ~ Na + Ca + Cl + EC + Salinity + pH, data = ion)
##          Df ChiSquare      F Pr(>F)
## Na        1    0.8836 2.0777  0.001 ***
## Ca        1    0.5973 1.4045  0.029 *
## Cl        1    0.5803 1.3645  0.022 *
## EC        1    0.6755 1.5884  0.002 **
## pH        1    0.5788 1.3611  0.020 *
## Residual 14    5.9536
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

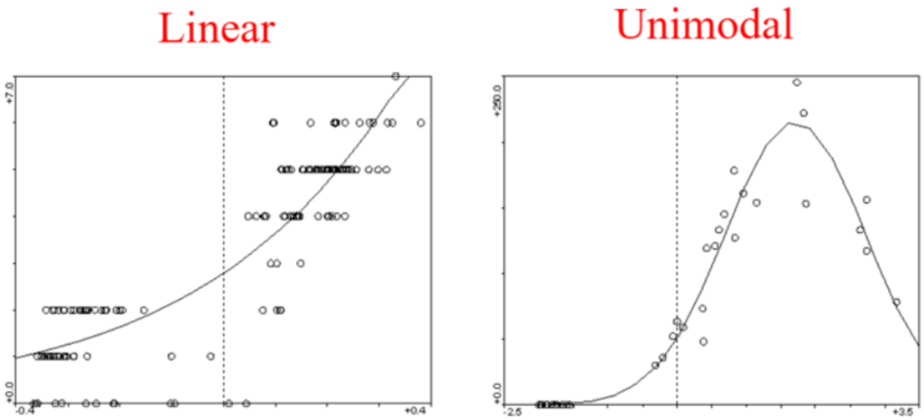
```
CCAscores <- scores(CCA, display = "sites") %>%
               as.data.frame() %>% rownames_to_column("site")
CCAscores$group <- substr(rownames(ion),1,2)
cc <- data.frame(CCA$CCA$biplot)

ggplot() + geom_point(data = CCAscores, aes(x = CCA1, y = CCA2, color=group), alpha=1) +
    geom_vline(xintercept = c(0), color = "grey70", linetype = 2) +
    geom_hline(yintercept = c(0), color = "grey70", linetype = 2) +
    geom_segment(data = cc, aes(x = 0, y = 0, xend = CCA1, yend = CCA2), arrow = arrow(length = unit(0.
    scale_color_brewer(palette = 'Set1') +
    geom_text(data = cc, aes(x = CCA1, y = CCA2, label = rownames(cc))) + theme_bw() +
    labs(x = paste0("CCA1 (",round(CCA$CCA$eig[1] / CCA$tot.chi*100,2), "% var. explained)"),
         y = paste0("CCA2 (",round(CCA$CCA$eig[2] / CCA$tot.chi*100,2), "% var. explained)"),
         title = "Canonical Correspondence Analysis") +
    theme(legend.text.align = 0, legend.title.align = 0,
          legend.title=element_text(size=10),legend.text=element_text(size=10))
```





**The arch effect in CCA**

A common artifact in CCA is the arch effect, a curvature in the ordination space that arises when species distributions follow strong unimodal patterns. This occurs because Chi-square distances emphasize differences in species presence and absence, leading to distortions in the ordination space. As a result, sites along a continuous environmental gradient may appear curved instead of following a linear progression.

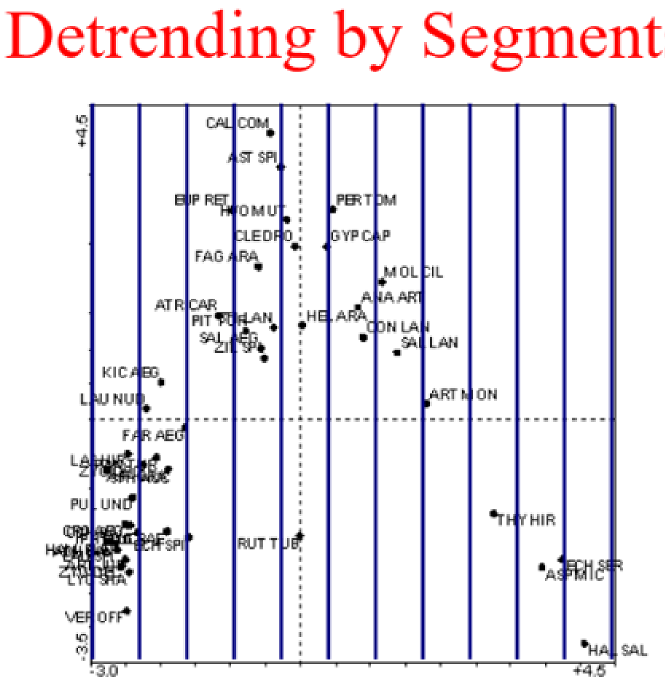


x-axis is environmental gradient and y-axis is abundance of species.

reference :

- David Zelený Lab at NTU
- slideplayer

## DCA



To correct the arch effect, Detrended Correspondence Analysis (DCA) was introduced. DCA is a modification of Correspondence Analysis (CA) that removes the arch effect through segment-wise detrending and local standardization.

**Segment-wise Detrending** DCA first computes an initial ordination using Correspondence Analysis (CA). However, instead of allowing the first axis to curve, DCA splits the axis into equal segments and removes the local mean within each segment. This eliminates the arch effect and ensures that the ordination axis follows a straight gradient.

**Local Standardization** In addition to detrending, DCA adjusts species scores to equalize species variance across the axis. This prevents overemphasis on dominant species and ensures that species with different dispersions are treated equally.

**DCA First Axis and Gradient Length** The `first DCA axis (DCA1)` represents the primary gradient in species composition. The length of the axis, measured in standard deviation (SD) units, indicates the degree of species turnover:

- If first axis length $< 2$ SD:

species distributions follow a linear pattern, making `Redundancy Analysis (RDA)` more appropriate.

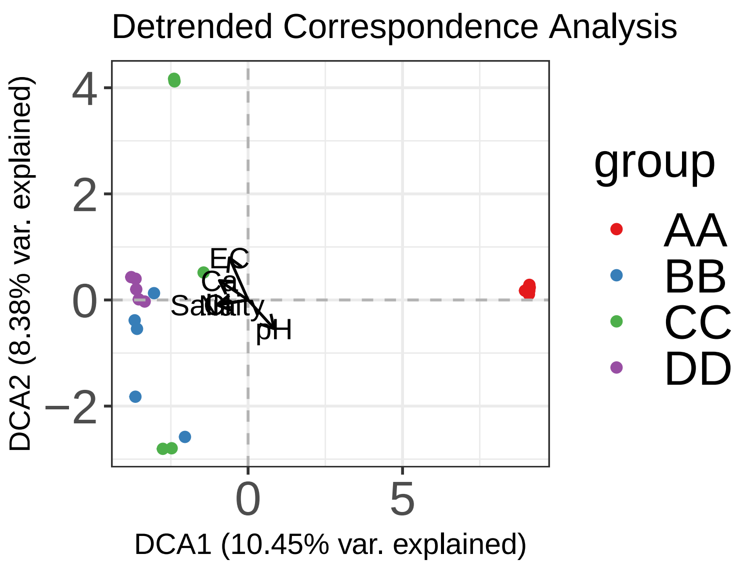
- If first axis length $> 4$ SD:

species distributions follow a unimodal response, meaning `CCA` is the better choice.

```
DCA <- decorana(t(wetland_rarefied))
DCA
##
## Call:
## decorana(veg = t(wetland_rarefied))
##
## Detrended correspondence analysis with 26 segments.
## Rescaling of axes with 4 iterations.
## Total inertia (scaled Chi-square): 9.2689
##
##                        DCA1   DCA2   DCA3   DCA4
## Eigenvalues          0.9682 0.7771 0.6458 0.4624
## Additive Eigenvalues 0.9682 0.7762 0.6419 0.4392
## Decorana values      0.9693 0.8123 0.6090 0.3957
## Axis lengths        12.8999 6.9736 5.7240 3.5045
```

```
#dca_factor <- envfit(as.formula(paste("DCA~ ", paste(colnames(ion), collapse = " + "))), data = ion)
dca_factor <- envfit(DCA~ Na + Ca + Cl+ EC + Salinity + pH, data = ion)
dca_factor
##
## ***VECTORS
##
##              DCA1     DCA2     r2 Pr(>r)
## Na       -0.99542 -0.09556 0.4575  0.008 **
## Ca       -0.93041  0.36653 0.5559  0.002 **
## Cl       -0.99632 -0.08571 0.3943  0.016 *
## EC       -0.60555  0.79581 0.0241  0.773
## Salinity -0.99586 -0.09094 0.4272  0.011 *
## pH        0.84129 -0.54058 0.3720  0.017 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Permutation: free
## Number of permutations: 999
```

```
dcv <- data.frame(dca_factor$vectors$arrows)
DCA_sum <- as.data.frame(summary(DCA)$site.scores)
##
## Call:
## decorana(veg = t(wetland_rarefied))
##
## Detrended correspondence analysis with 26 segments.
## Rescaling of axes with 4 iterations.
## Total inertia (scaled Chi-square): 9.2689
##
##                          DCA1   DCA2   DCA3   DCA4
## Eigenvalues           0.9682 0.7771 0.6458 0.4624
## Additive Eigenvalues  0.9682 0.7762 0.6419 0.4392
## Decorana values       0.9693 0.8123 0.6090 0.3957
## Axis lengths         12.8999 6.9736 5.7240 3.5045
DCAscores <- DCA_sum %>% rownames_to_column("site")
DCAscores$group <- substr(rownames(ion),1,2)

ggplot() + geom_point(data = DCAscores, aes(x = DCA1, y = DCA2, color = group), alpha=1) +
    geom_vline(xintercept = c(0), color = "grey70", linetype = 2) +
    geom_hline(yintercept = c(0), color = "grey70", linetype = 2) +
    geom_segment(data = dcv, aes(x = 0, y = 0, xend = DCA1, yend = DCA2), arrow = arrow(length = unit(0
    geom_text(data = dcv, aes(x = DCA1, y = DCA2, label = rownames(dcv))) + theme_bw() +
    scale_color_brewer(palette = 'Set1') +
    labs(x = paste0("DCA1 (",round(DCA$evals[1] / DCA$totchi*100,2), "% var. explained)"),
         y = paste0("DCA2 (",round(DCA$evals[2] / DCA$totchi*100,2), "% var. explained)"),
         title = "Detrended Correspondence Analysis")+
    theme(legend.text.align = 0, legend.title.align = 0,
      legend.title=element_text(size=18),
      legend.text=element_text(size=18),
      axis.text = element_text(size=18))
```



### RDA

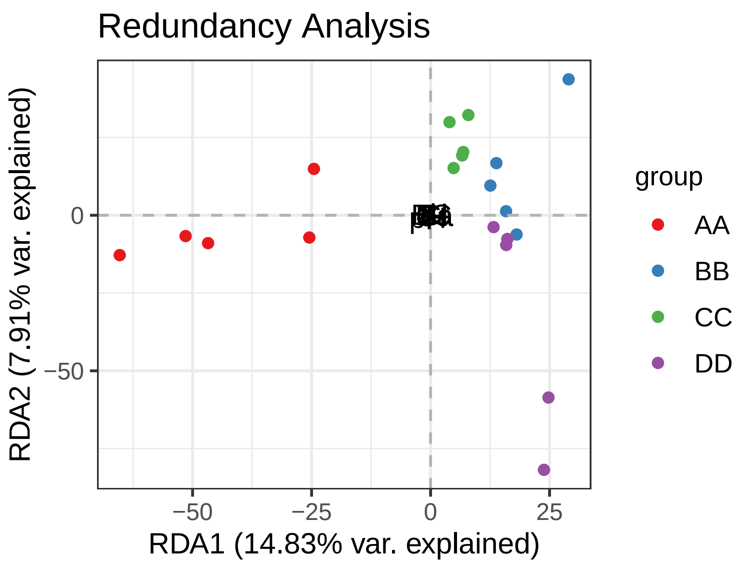
Redundancy Analysis (RDA) is a linear constrained ordination method similar to Principal Component Analysis (PCA), but with environmental constraints. Instead of using Chi-square distances, RDA applies Euclidean distances, making it suitable when species responses are approximately linear with respect to environmental gradients.

Given a species data matrix and an environmental matrix , RDA performs a multivariate regression of on , followed by PCA on the fitted values. This results in ordination axes that explain the maximum variation in species data constrained by the environmental variables.

```
#RDA <- rda(as.formula(paste("t(wetland_rarefied)~ ", paste(colnames(ion), collapse = " + "))), ion)
RDA <- rda(t(wetland_rarefied)~ Na + Ca + Cl+ EC + Salinity + pH, ion)
rda_temp <- anova.cca(RDA)
rda_term <- anova.cca(RDA, by="terms")
rda_term
## Permutation test for rda under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## Model: rda(formula = t(wetland_rarefied) ~ Na + Ca + Cl + EC + Salinity + pH, data = ion)
##          Df Variance      F Pr(>F)
## Na        1   783986 2.3209  0.001 ***
## Ca        1   541133 1.6020  0.051 .
## Cl        1   392550 1.1621  0.269
## EC        1   362663 1.0736  0.371
## pH        1   457295 1.3538  0.134
## Residual 14  4729101
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
RDA_sum <- summary(RDA)
RDAscores <- as.data.frame(RDA_sum$sites[,1:2]) %>% rownames_to_column("site")
```

```
RDAscores$group <- substr(rownames(ion),1,2)
cc <- as.data.frame(RDA_sum$biplot[,1:2])
ggplot() + geom_point(data = RDAscores, aes(x = RDA1, y = RDA2, color=group), alpha=1) +
    geom_vline(xintercept = c(0), color = "grey70", linetype = 2) +
    geom_hline(yintercept = c(0), color = "grey70", linetype = 2) +
    geom_segment(data = cc, aes(x = 0, y = 0, xend = RDA1, yend = RDA2), arrow = arrow(length = unit(0.
    geom_text(data = cc, aes(x = RDA1, y = RDA2, label = rownames(cc))) + theme_bw() +
    scale_color_brewer(palette = 'Set1')  +
    labs(x = paste0("RDA1 (",round(RDA$CCA$eig[1] / RDA$tot.chi*100,2), "% var. explained)"),
         y = paste0("RDA2 (",round(RDA$CCA$eig[2] / RDA$tot.chi*100,2), "% var. explained)"),
         title = "Redundancy Analysis") +
    theme(legend.text.align = 0, legend.title.align = 0,
          legend.title=element_text(size=10),legend.text=element_text(size=10))
```



## Differential Taxa

### ALDEx2

ALDEx2 which generated Monte Carlo samples of Dirichlet distributions for each sample, using a uniform prior, performed CLR transformation of each realization, and then performed Wilcoxon tests on the transformed realizations.

```
library(ALDEx2)
## Loading required package: zCompositions
## Loading required package: MASS
##
## Attaching package: 'MASS'
## The following object is masked from 'package:rstatix':
##
##     select
## The following object is masked from 'package:patchwork':
##
```

```
##     area
## The following object is masked from 'package:dplyr':
##
##     select
## Loading required package: NADA
## Loading required package: survival
##
## Attaching package: 'NADA'
## The following object is masked from 'package:IRanges':
##
##     cor
## The following object is masked from 'package:S4Vectors':
##
##     cor
## The following object is masked from 'package:stats':
##
##     cor
## Loading required package: truncnorm
ALDEx2_result <- aldex(reads=data, conditions = meta$Class,
                       mc.samples = 128, test="t", effect=TRUE,
                       include.sample.summary = FALSE, verbose=T, denom="all")
## aldex.clr: generating Monte-Carlo instances and clr values
## operating in serial mode
## removed rows with sums equal to zero
## computing center with all features
## data format is OK
## dirichlet samples complete
## transformation complete
## aldex.ttest: doing t-test
## running tests for each MC instance:
## |------------(25%)----------(50%)----------(75%)----------|
## aldex.effect: calculating effect sizes
## operating in serial mode
## sanity check complete
## rab.all  complete
## rab.win  complete
## rab of samples complete
## within sample difference calculated
## between group difference calculated
## group summaries calculated
## effect size calculated
## summarizing output

selected_result <- ALDEx2_result[ALDEx2_result$wi.ep < 0.05,]
selected_result <- cbind(rownames(selected_result),selected_result)
head(selected_result)
##         rownames(selected_result)   rab.all rab.win.Cancer rab.win.Normal
## ASV0001                   ASV0001 14.373547      14.630241     13.8742929
## ASV0007                   ASV0007  9.648390      12.282246      7.7414882
## ASV0119                   ASV0119  8.466592       3.945452     10.0351473
## ASV0130                   ASV0130  3.991660       1.643844      9.4433972
## ASV0146                   ASV0146  8.222159       9.565661      7.4461137
## ASV0178                   ASV0178  1.850530       5.716942      0.4256441
##          diff.btw diff.win     effect   overlap       we.ep    we.eBH
## ASV0001 -1.377995 2.364327 -0.5127501 0.3161593 0.009924686 0.9111672
## ASV0007 -4.198266 5.508847 -0.6886866 0.2265625 0.001738385 0.8659766
## ASV0119  4.845204 7.793662  0.6220667 0.2638564 0.008007031 0.8930373
## ASV0130  4.215072 9.212734  0.4781253 0.3099141 0.056589218 0.9238886
## ASV0146 -1.937207 5.932652 -0.3029744 0.3093750 0.149241061 0.9396661
## ASV0178 -5.289013 6.269317 -0.7701952 0.2107729 0.003434822 0.7223806
##               wi.ep    wi.eBH
## ASV0001 0.041240625 0.9270136
## ASV0007 0.003136204 0.8139919
## ASV0119 0.009039480 0.8642710
## ASV0130 0.049115638 0.9198037
## ASV0146 0.046041349 0.9261615
## ASV0178 0.003249730 0.6543478
#write.table(selected_result, "~/Course/NDMC_2025/tmp/ALDEx2.txt", quote=FALSE, sep="\t", col.names = F
```

## ANCOM

ANCOM first examined the abundance table to identify `outlier zeros` and `structural zeros`, Outlier zeros, identified by finding outliers in the distribution of taxon counts within each sample grouping, were ignored during differential abundance analysis, and replaced with NA. Structural zeros, taxa that were absent in one grouping but present in the other, were ignored during data analysis and automatically called as differentially abundant. Using the main function ANCOM, all additive log-ratios for each taxon were then tested for significance using Wilcoxon rank-sum tests, and p-values were FDR-corrected using the BH method. ANCOM-II then applied a detection threshold as described in the original paper, whereby a taxon was called as DA if the number of corrected p-values reaching nominal significance for that taxon was greater than 60% of the maximum possible number of significant comparisons.

```
library(compositions)
## Welcome to compositions, a package for compositional data analysis.
## Find an intro with "? compositions"
##
## Attaching package: 'compositions'
## The following object is masked from 'package:NADA':
##
##     cor
## The following object is masked from 'package:ape':
##
##     balance
## The following objects are masked from 'package:IRanges':
##
##     cor, cov, var
## The following objects are masked from 'package:S4Vectors':
##
##     cor, cov, var
## The following objects are masked from 'package:BiocGenerics':
##
##     normalize, var
## The following objects are masked from 'package:stats':
##
##     anova, cor, cov, dist, var
## The following object is masked from 'package:graphics':
```

```
##
##     segments
## The following objects are masked from 'package:base':
##
##     %*%, norm, scale, scale.default
library(exactRankTests)
##  Package 'exactRankTests' is no longer under development.
##  Please consider using package 'coin' instead.
library(nlme)
source('~/Course/NDMC_2025/source/ancom_v2.1.R')

meta$Sample <- rownames(meta)
prepro <- feature_table_pre_process(feature_table = data, meta_data = meta,
                                    sample_var = 'Sample', group_var = 'Class',
                                    out_cut = 0.05, zero_cut = 0.90,
                                    lib_cut = 1000, neg_lb=FALSE)
feature_table <- prepro$feature_table
metadata <- prepro$meta_data
struc_zero <- prepro$structure_zeros
main_var <- 'Class'
p_adj_method = "BH"
alpha=0.05
adj_formula=NULL
rand_formula=NULL
ANCOM_result <- ANCOM(feature_table = feature_table, meta_data = metadata,
             struc_zero = struc_zero, main_var = main_var, p_adj_method = p_adj_method,
             alpha=alpha, adj_formula = adj_formula, rand_formula = rand_formula)
ANCOM_result <- ANCOM_result$out
ANCOM_result <- ANCOM_result[ANCOM_result$W != 0, ]
head(ANCOM_result)
##     taxa_id   W detected_0.9 detected_0.8 detected_0.7 detected_0.6
## 6   ASV0007 457        FALSE        FALSE         TRUE         TRUE
## 28  ASV0030   1        FALSE        FALSE        FALSE        FALSE
## 40  ASV0042   1        FALSE        FALSE        FALSE        FALSE
## 43  ASV0045   3        FALSE        FALSE        FALSE        FALSE
## 103 ASV0119 467        FALSE        FALSE         TRUE         TRUE
## 113 ASV0130   2        FALSE        FALSE        FALSE        FALSE
#write.table(out, "~/Course/NDMC_2025/tmp/ANCOM.txt", quote=FALSE, sep="\t", col.names = F, row.names =

ANCOM_selected <- ANCOM_result[ANCOM_result$detected_0.6, ]
head(ANCOM_selected)
##     taxa_id   W detected_0.9 detected_0.8 detected_0.7 detected_0.6
## 6   ASV0007 457        FALSE        FALSE         TRUE         TRUE
## 103 ASV0119 467        FALSE        FALSE         TRUE         TRUE
## 155 ASV0178 495        FALSE        FALSE         TRUE         TRUE
## 182 ASV0222 395        FALSE        FALSE        FALSE         TRUE
## 255 ASV0323 Inf         TRUE         TRUE         TRUE         TRUE
## 468 ASV0842 Inf         TRUE         TRUE         TRUE         TRUE
#write.table(out, "~/Course/NDMC_2025/tmp/ANCOM_thr.txt", quote=FALSE, sep="\t", col.names = F, row.nam
```

## edgeR

We added a pseudocount of 1 to the data and used the function `calcNormFactors` from the edgeR to compute relative log expression normalization factors. Negative binomial dispersion parameters were then estimated using the functions `estimateCommonDisp` followed by `estimateTagwiseDisp` to shrink feature-wise dispersion estimates through an empirical Bayes approach. We then used the `exactTest` for negative binomial data to identify features that differ between the specified groups. The resulting p-values were then corrected for multiple testing with the BH method with the function `topTags`.

```
library(edgeR)
## Loading required package: limma
##
## Attaching package: 'limma'
## The following object is masked from 'package:BiocGenerics':
##
##     plotMA

ASV <- phyloseq::otu_table(data, taxa_are_rows = T)
sampledata <- phyloseq::sample_data(meta, errorIfNULL = T)
phylo <- phyloseq::merge_phyloseq(ASV, sampledata)
test <- phyloseq_to_edgeR(physeq = phylo, group = "Class") # phyloseq_to_edgeR function is sourced from
et = exactTest(test)
out = topTags(et, n=nrow(test$table), adjust.method="BH", sort.by="PValue")
edgeR_result <- out@.Data[[1]]
edgeR_selected <- edgeR_result[edgeR_result$FDR < 0.05,]
edgeR_selected <- cbind(rownames(edgeR_selected), edgeR_selected)
head(edgeR_selected)
##         rownames(edgeR_selected)      logFC    logCPM       PValue          FDR
## ASV0075                  ASV0075 -10.005478 11.509109 4.983343e-12 1.089973e-08
## ASV0029                  ASV0029 -11.378337 12.878259 6.000401e-12 1.089973e-08
## ASV0178                  ASV0178  -7.512891  9.946950 1.006243e-11 1.218560e-08
## ASV0073                  ASV0073 -10.009413 11.513042 3.960265e-11 2.895228e-08
## ASV0083                  ASV0083  -9.866453 11.370704 4.697678e-11 2.895228e-08
## ASV0198                  ASV0198  -8.189402  9.708196 5.451478e-11 2.895228e-08

# write.table(subout, "~/Course/NDMC_2025/tmp/edgeR.txt", quote=FALSE, sep="\t", col.names = F, row.nam
```

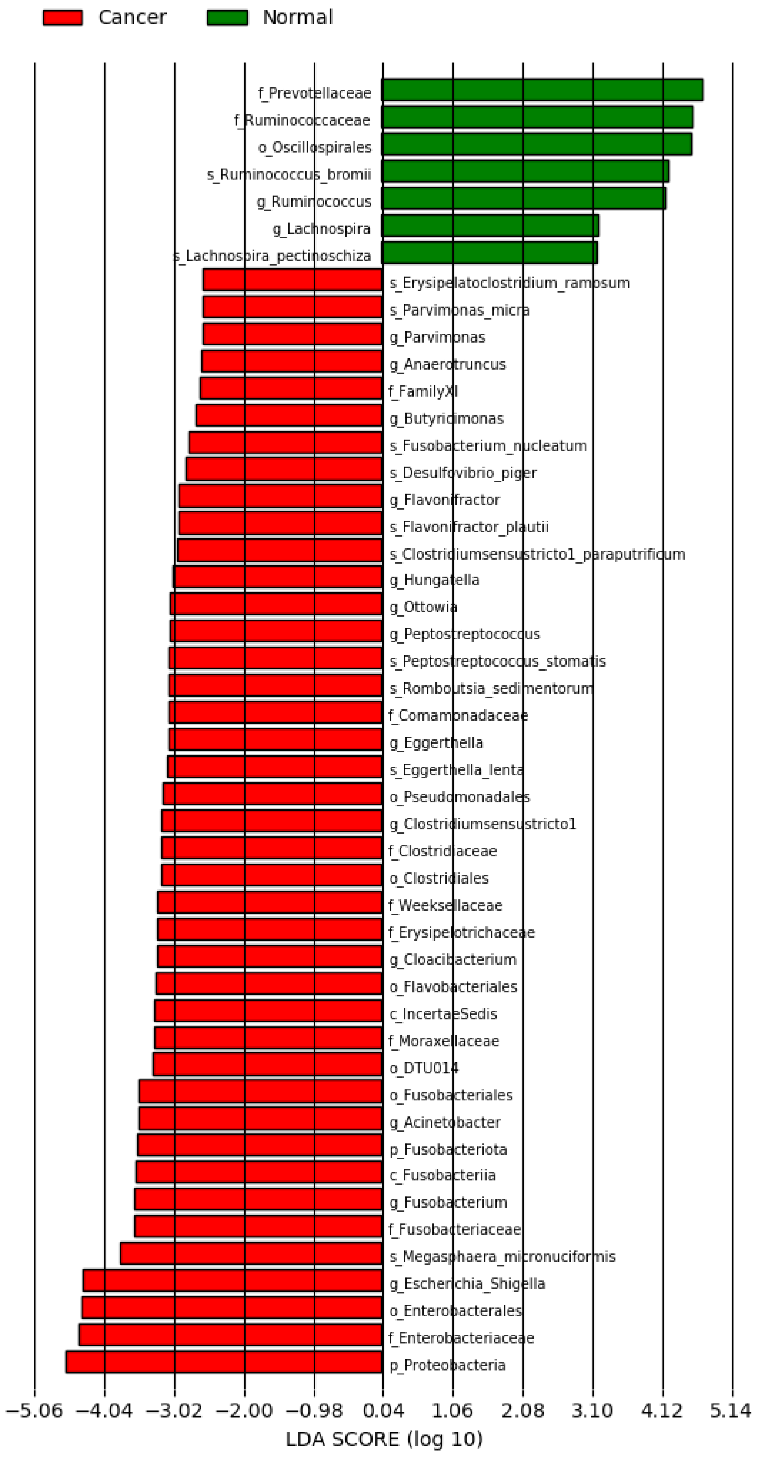
## LEfSe

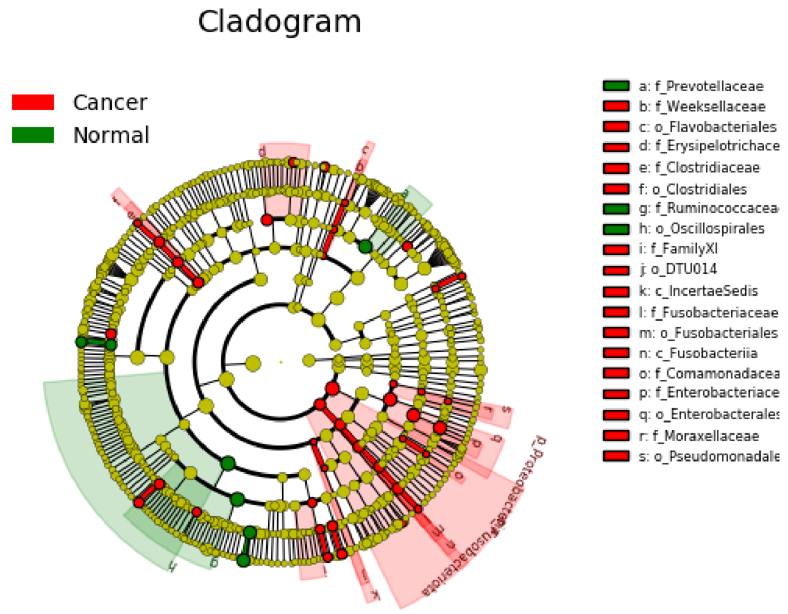
LEfSe performed a Kruskal-Wallis (which in our two-group case reduces to the Wilcoxon rank-sum) hypothesis test to identify potential differentially abundant features, followed by linear discriminant analysis (LDA) of class labels on abundances to estimate the effect sizes for significant features. From these, only those features with scaled LDA analysis scores above the threshold score of 2.0 (default) were called as differentially abundant.

```
# file preparation
ret_tab <- LEfSe_preparation(data, taxa, meta, "Class", pvl_filter = 0.05)
head(ret_tab[,1:5])
##                                 [,1]               [,2]
## Class                           "Normal"           "Normal"
## sampleID                        "ERR475473"        "ERR475476"
## k_Bacteria                      "90.4988091187479" "93.7881847531094"
## k_Archaea                       "0"                "0.0237951303587277"
## k_Bacteria|p_Proteobacteria "13.0343654304185" "6.13142629297594"
## k_Bacteria|p_Firmicutes     "58.7764545763865" "40.1578195943252"
##                              [,3]               [,4]
## Class                        "Normal"           "Normal"
## sampleID                     "ERR475478"        "ERR475480"
## k_Bacteria                   "89.5623741239996" "98.1584169974317"
## k_Archaea                    "3.39050176528261" "0"
## k_Bacteria|p_Proteobacteria "6.6023018705717"  "16.199217837964"
## k_Bacteria|p_Firmicutes     "71.9902052171225" "69.8721690403923"
##                              [,5]
## Class                        "Normal"
## sampleID                     "ERR475483"
## k_Bacteria                   "98.2261609156035"
## k_Archaea                    "0"
## k_Bacteria|p_Proteobacteria "7.60909895253589"
## k_Bacteria|p_Firmicutes     "88.7233496018495"

dir.create("~/Course/NDMC_2025/tmp/LEfSe_tmp")
write.table(ret_tab, "~/Course/NDMC_2025/tmp/LEfSe_tmp/tmp_in.txt", quote=FALSE, sep="\t", col.names =
```

```
# LEfSe execution by docker
$ docker run -u $UID:$(id -g) -it --rm -v /home/yincheng23/Course/NDMC_2025/tmp/LEfSe_tmp:/tmp yincheng
$ format_input.py /tmp/tmp_in.txt /tmp/data_in -s -1 -u 2 -o 1000000
$ run_lefse.py /tmp/data_in /tmp/LEfSe_res.txt
$ plot_res.py /tmp/LEfSe_res.txt /tmp/LEfSe_res.png --dpi 100
$ plot_cladogram.py /tmp/LEfSe_res.txt /tmp/cladogram.png --format png --dpi 100
$ cat /tmp/LEfSe_res.txt | awk '{if($3>2){print $0}}' > /tmp/LEfSe_res_selected.txt  # filtering by |LD
```

```
result <- read.csv("~/Course/NDMC_2025/tmp/LEfSe_tmp/LEfSe_res.txt", sep = '\t', header = F)
colnames(result) <- c('taxa','log10LDA','tendency','U','p')
head(result)
```


Cancer
Normal
f_Prevotellaceae
f_Ruminococcaceae
o_Oscillospirales
s_Ruminococcus_bromii
g_Ruminococcus
g_Lachnospira
s_Lachnospira_pectinoschiza
s_Erysipelatoclostridium_ramosum
s_Parvimonas_micra
g_Parvimonas
g_Anaerotruncus
f_FamilyXI
g_Butyricimonas
s_Fusobacterium_nucleatum
s_Desulfovibrio_piger
g_Flavonifractor
s_Flavonifractor_plautii
s_Clostridiumsensustricto1_paraputrificum
g_Hungatella
g_Ottowia
g_Peptostreptococcus
s_Peptostreptococcus_stomatis
s_Romboutsia_sedimentorum
f_Comamonadaceae
g_Eggerthella
s_Eggerthella_lenta
o_Pseudomonadales
g_Clostridiumsensustricto1
f_Clostridiaceae
o_Clostridiales
f_Weeksellaceae
f_Erysipelotrichaceae
g_Cloacibacterium
o_Flavobacteriales
c_IncertaeSedis
f_Moraxellaceae
o_DTU014
o_Fusobacteriales
g_Acinetobacter
p_Fusobacteriota
c_Fusobacteriia
g_Fusobacterium
f_Fusobacteriaceae
s_Megasphaera_micronuciformis
g_Escherichia_Shigella
o_Enterobacterales
f_Enterobacteriaceae
p_Proteobacteria
−5.06
−4.04
−3.02
−2.00
−0.98
0.04
1.06
2.08
3.10
4.12
5.14
LDA SCORE (log 10)



## PreLect

PreLectR is an R package implementing the PreLect algorithm, which integrates L1 regularization with an inverted prevalence penalty to select universal and informative features in sparse data.

It supports four tasks: binary classification, multi-class classification, regression, and time-to-event analysis.

$$\text{Binary classification} : J(\mathbf{w}) = \text{BCE}(\mathbf{y}, \hat{\mathbf{y}}) + \lambda \sum_j \frac{|\mathbf{w}_j|}{p_j}$$

$$\text{Regression} : J(\mathbf{w}) = \text{MSE}(\mathbf{y}, \hat{\mathbf{y}}) + \lambda \sum_j \frac{|\mathbf{w}_j|}{p_j}$$

$$\text{Multi-class classification} : J(\mathbf{w}) = \frac{1}{c} \sum_{l=1}^{c} \left( \text{BCE}(\mathbf{y}_l, \hat{\mathbf{y}}_l) + \lambda \sum_{j=1}^{d} \frac{|\mathbf{w}_{j,l}|}{p_{j,l}} \right)$$

$$\text{Time-to-event} : J(\mathbf{w}) = h_0(t) \cdot e^{\sum x_i \cdot w} + \lambda \sum_j \frac{|\mathbf{w}_j|}{p_j}$$

Based on previous studies, we recommend using variance-stabilizing transformation (VST) for data normalization.

```r
# Variance Stabilization Transformation
library(DESeq2)
library(PreLectR)
data_pseudo_count <- data + 1
meta$Class <- factor(meta$Class)

dds <- DESeqDataSetFromMatrix(countData = data_pseudo_count,
                              colData = meta,
                              ~ Class)

vst <- varianceStabilizingTransformation(dds, fitType="mean", blind= T)
```

```r
vst_table <- assay(vst)

# feature-wise z-standardization
data_scaled <- t(scale(t(as.matrix(vst_table))))
```

We will only use z-standardized table `data_scaled` and raw count table `data` for the subsequent analyses.

Automatically perform lambda scanning to identify 30 lambda values for examination.

```r
meta$Class <- factor(meta$Class, levels = c("Normal", "Cancer"))  # assign "Normal" as control sample
lrange <- AutoScanning(data_scaled, data, meta$Class, step =30)
##   |                                                                           |
length(lrange)
## [1] 30
exp(lrange)
##  [1] 1.000000e-07 1.487352e-07 2.212216e-07 3.290345e-07 4.893901e-07
##  [6] 7.278954e-07 1.082637e-06 1.610262e-06 2.395027e-06 3.562248e-06
## [11] 5.298317e-06 7.880463e-06 1.172102e-05 1.743329e-05 2.592944e-05
## [16] 3.856620e-05 5.736153e-05 8.531679e-05 1.268961e-04 1.887392e-04
## [21] 2.807216e-04 4.175319e-04 6.210169e-04 9.236709e-04 1.373824e-03
## [26] 2.043360e-03 3.039195e-03 4.520354e-03 6.723358e-03 1.000000e-02

# Examining the testing lambda.
dir.create("~/Course/NDMC_2025/tmp/PreLect_tmp")
output_path <-"~/Course/NDMC_2025/tmp/PreLect_tmp"
tuning_res <- LambdaTuningParallel(data_scaled, data, meta$Class, lrange, n_cores=10, outpath=output_pa
```

```r
head(tuning_res$TuningResult)
```

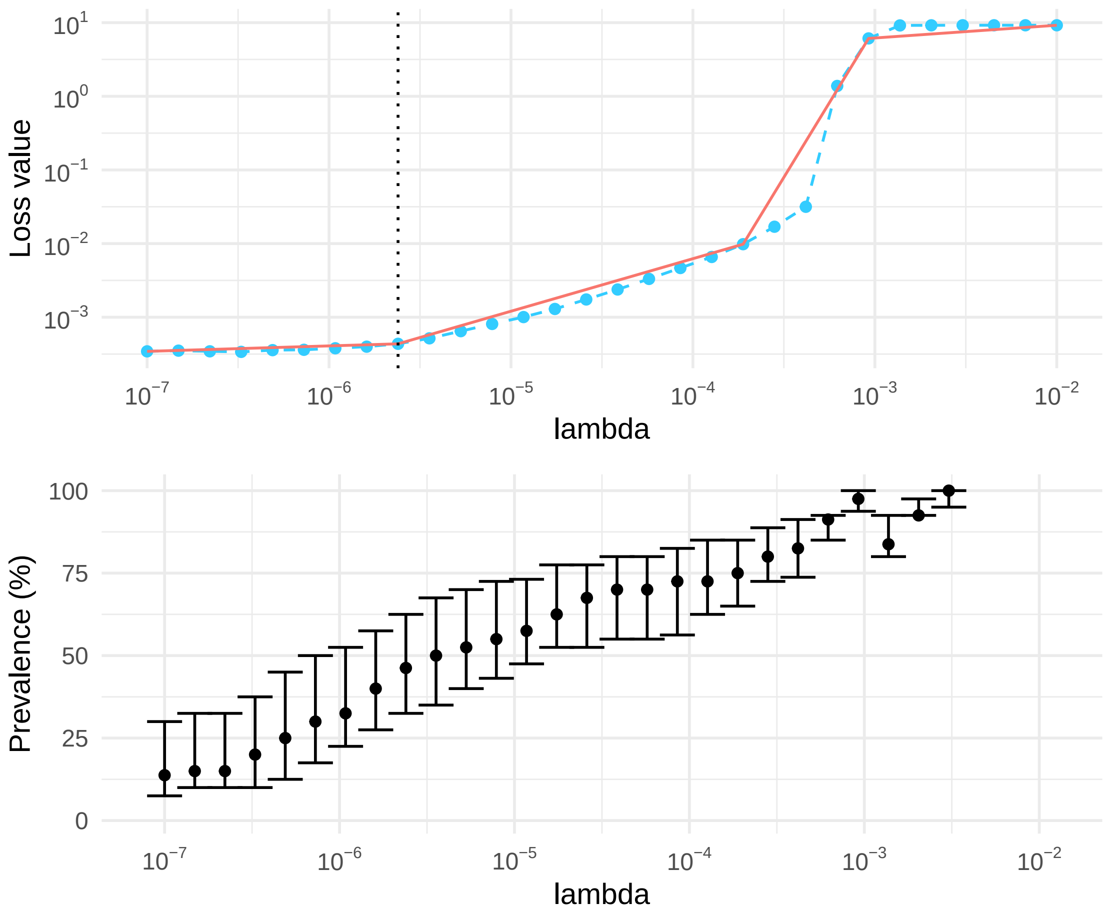
```
##   Feature_number Percentage Prevalence  AUC loss_history error_history
## 1           1051  0.2892926       0.15 0.75 0.0003444818  9.998251e-05
## 2            988  0.2719516       0.15 0.75 0.0003522416  9.997643e-05
## 3            943  0.2595651       0.15 0.75 0.0003449498  9.998842e-05
## 4            805  0.2215800       0.20 0.75 0.0003388232  9.999399e-05
## 5            653  0.1797413       0.25 0.75 0.0003578406  9.999683e-05
## 6            556  0.1530416       0.30 0.75 0.0003616958  9.990256e-05
##     loglmbd
## 1 -16.11810
## 2 -15.72110
## 3 -15.32410
## 4 -14.92710
## 5 -14.53011
## 6 -14.13311
```

```r
head(tuning_res$PvlDistSummary)
```

```
##       llmbd max    q3     q2    q1 min
## 1 -16.11810   1 0.300 0.1375 0.075   0
## 2 -15.72110   1 0.325 0.1500 0.100   0
## 3 -15.32410   1 0.325 0.1500 0.100   0
## 4 -14.92710   1 0.375 0.2000 0.100   0
## 5 -14.53011   1 0.450 0.2500 0.125   0
## 6 -14.13311   1 0.500 0.3000 0.175   0
```

Determine the optimal lambda by partitioning tree.

```
lmbd_picking <- LambdaDecision(tuning_res$TuningResult, tuning_res$PvlDistSummary, maxdepth=5, minbucke

lmbd_picking$selected_lmbd_plot/lmbd_picking$pvl_plot
```



```
print(lmbd_picking$opt_lmbd)
## [1] 2.395027e-06
```

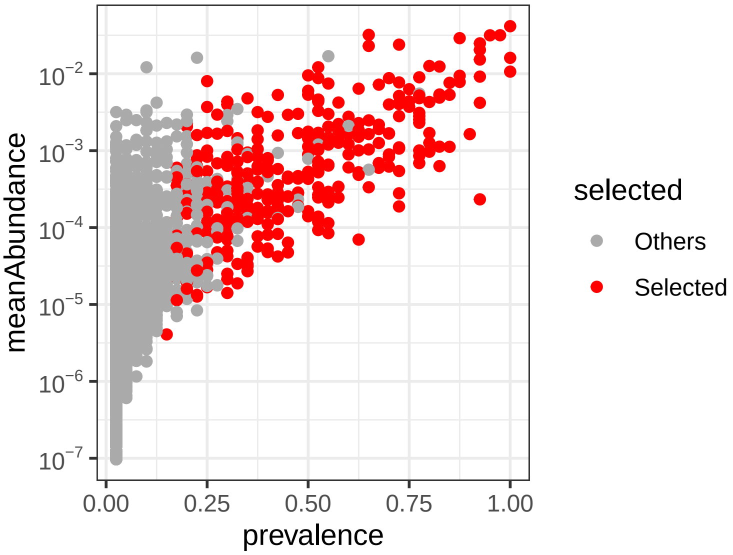
PreLect execution and get the property for each feature

```
# We strongly suggest using 100000 for max_iter to achieve more accurate results.
s=Sys.time()
prevalence <- GetPrevalence(data)
PreLect_out <- PreLect(data_scaled, prevalence, meta$Class, lambda=lmbd_picking$opt_lmbd, max_iter = 10
print(Sys.time()-s)
## Time difference of 2.824755 secs


featpropt <- FeatureProperty(data, meta$Class, PreLect_out, task="classification")

print(paste(nrow(featpropt[featpropt$selected == 'Selected', ]), 'features were selected'))
```

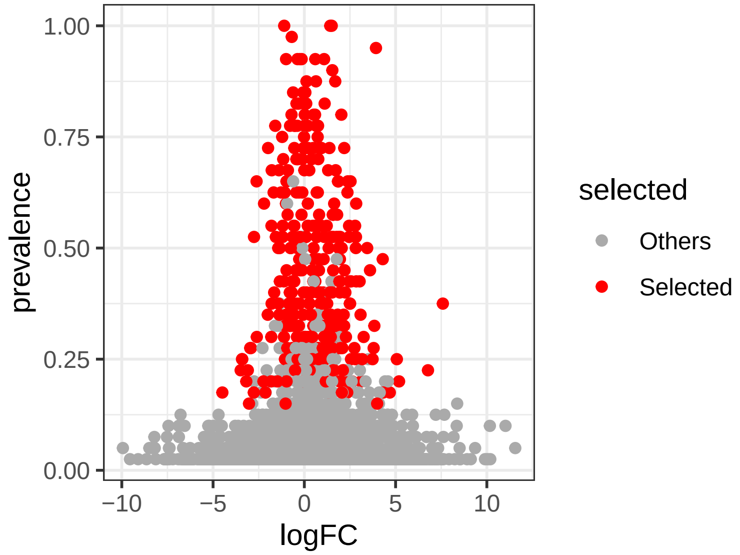
```
## [1] "336 features were selected"

print(paste('median of prevalence :', median(featpropt$prevalence[featpropt$selected == 'Selected'])))
## [1] "median of prevalence : 0.425"

write.table(featpropt, "~/Course/NDMC_2025/tmp/PreLect_tmp/feature_propt.txt", quote=FALSE, sep="\t")
head(featpropt)
##          FeatName        coef tendency selected meanAbundance   variance
## ASV0001  ASV0001  0.54988067   Cancer Selected    0.04150302  117734960
## ASV0002  ASV0002 -0.42573824   Normal Selected    0.03167390   64237504
## ASV0003  ASV0003  0.00000000     <NA>   Others    0.01213962 1555924984
## ASV0004  ASV0004 -0.33145665   Normal Selected    0.03201389  288009324
## ASV0005  ASV0005 -0.32683697   Normal Selected    0.02903994   41768447
## ASV0006  ASV0006 -0.03458693   Normal Selected    0.02485074   72058376
##          prevalence prevalence_case prevalence_control      logFC
## ASV0001       1.000            1.00               1.00  1.4998663
## ASV0002       0.975            0.95               1.00 -0.6884894
## ASV0003       0.100            0.05               0.15 11.0217666
## ASV0004       0.650            0.60               0.70 -2.6154169
## ASV0005       0.875            0.80               0.95  0.6372900
## ASV0006       0.925            0.90               0.95 -0.1468191
```

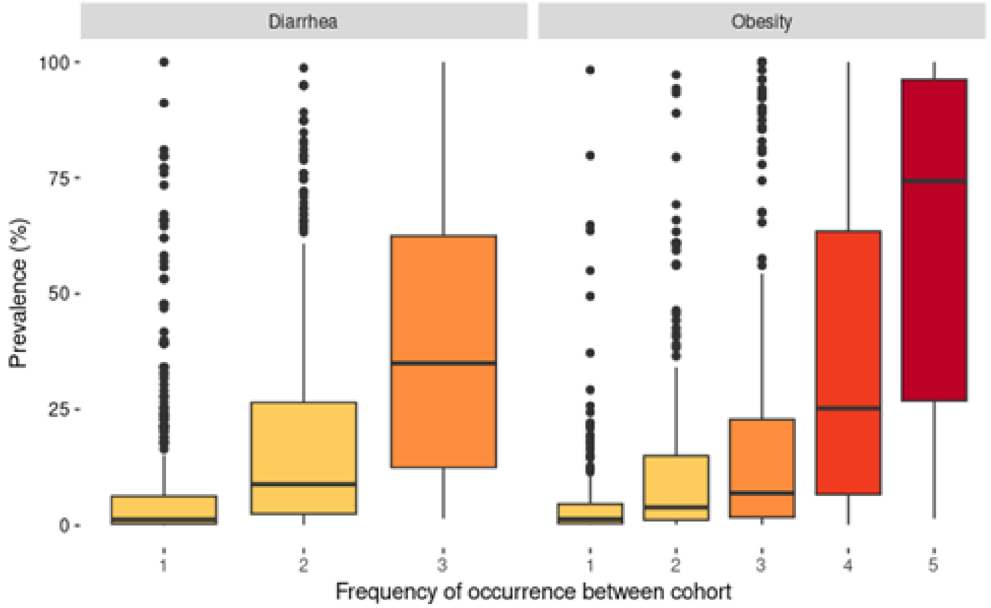
Selection profile visualization and evaluation

```
ggplot(featpropt, aes(x = prevalence, y = meanAbundance, color=selected)) + geom_point() +
  scale_color_manual(values = c('Selected'='red', 'Others'='#AAAAAA')) +
  scale_y_log10(breaks = trans_breaks("log10", function(x) 10^x),
                labels = trans_format("log10", math_format(10^.x))) +
  theme_bw()+ theme(panel.background = element_rect(fill = "white", colour = "white"))
```



```
ggplot(featpropt, aes(x = logFC, y = prevalence, color=selected)) + geom_point() +
  scale_color_manual(values = c('Selected'='red', 'Others'='#AAAAAA')) +
  theme_bw()+ theme(panel.background = element_rect(fill = "white", colour = "white"))
```

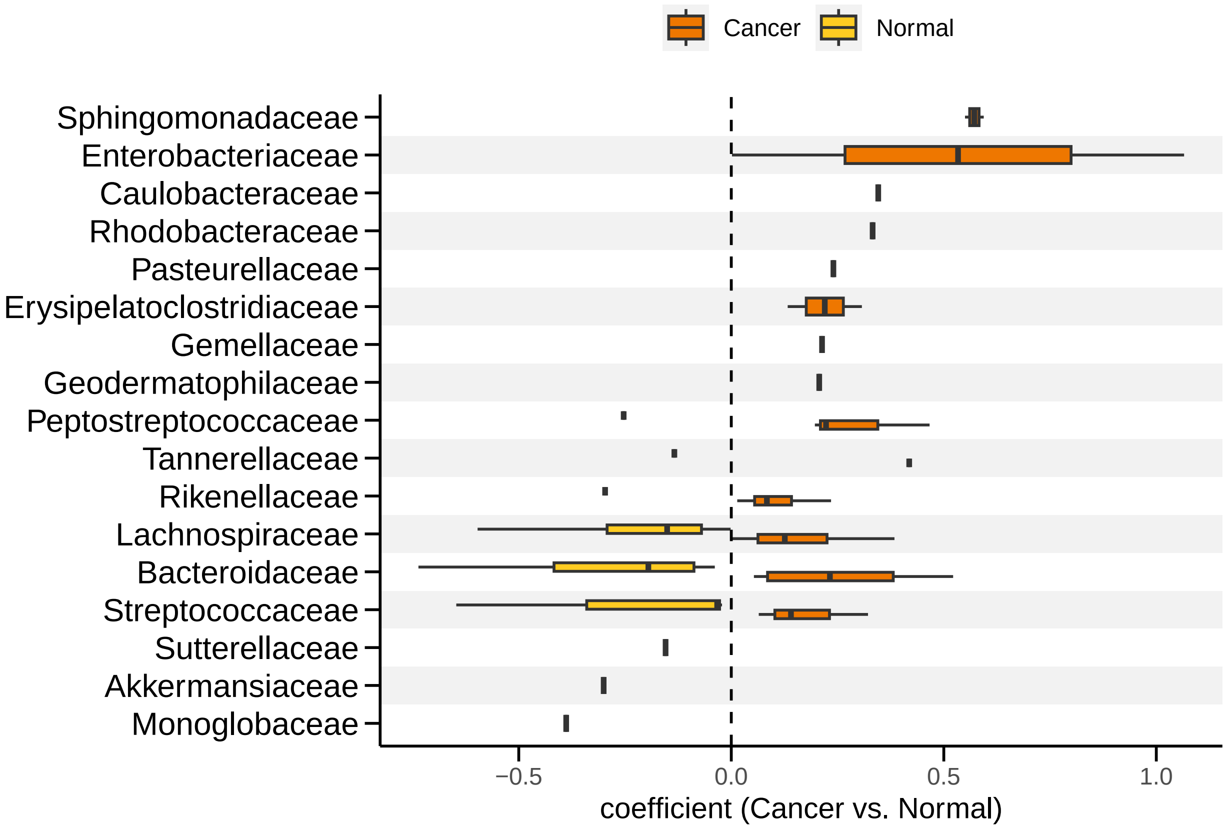




We observed that features present in multiple datasets tend to have higher prevalence within those datasets. This finding suggests that features with higher prevalence in one dataset are more likely to be universal across different datasets.

```
y <- ifelse(meta$Class == "Cancer", 1, 0)

split   <- TrainTextSplit(y, ratio = 0.8)
X_train <- data_scaled[, split$train_idx]
X_test  <- data_scaled[, split$test_idx]
y_train <- y[split$train_idx]
y_test  <- y[split$test_idx]

perf <- evaluation(X_train, y_train, X_test, y_test, featpropt$coef, task='classification')
perf
## $AUC
## [1] 0.875
```

```
result <- TaxaProperty(featpropt, taxa, "Family",  pvl_filter = 0.5)

mycolor <- c("Normal" = "#FFCC22", "Cancer" = "#EE7700")
result$effectSizePlot + scale_fill_manual(values = mycolor) +
   geom_hline(yintercept = 0, color='black', linetype='dashed')
```

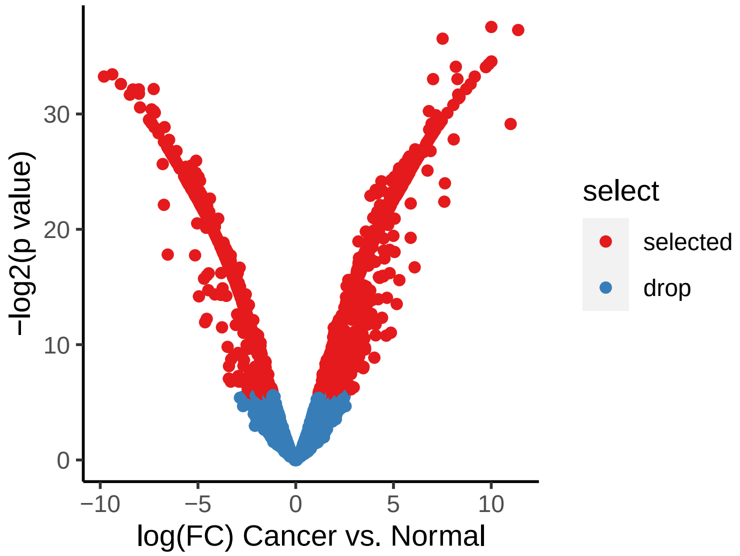


```
head(result$selectedInfo)
##          FeatName       coef tendency selected meanAbundance  variance
## ASV0001  ASV0001  0.5498807   Cancer Selected    0.04150302 117734960
## ASV0002  ASV0002 -0.4257382   Normal Selected    0.03167390  64237504
## ASV0005  ASV0005 -0.3268370   Normal Selected    0.02903994  41768447
## ASV0007  ASV0007  1.0654049   Cancer Selected    0.03158703 178336011
## ASV0008  ASV0008 -0.3002699   Normal Selected    0.02299001 312317781
## ASV0009  ASV0009 -0.4994069   Normal Selected    0.02042962  45214122
##          prevalence prevalence_case prevalence_control      logFC
## ASV0001       1.000            1.00               1.00  1.4998663
## ASV0002       0.975            0.95               1.00 -0.6884894
## ASV0005       0.875            0.80               0.95  0.6372900
## ASV0007       0.950            1.00               0.90  3.9252053
## ASV0008       0.650            0.65               0.65  1.8688587
## ASV0009       0.925            0.85               1.00  0.5991814
##                         taxa
## ASV0001  Sphingomonadaceae
## ASV0002    Lachnospiraceae
## ASV0005     Bacteroidaceae
## ASV0007 Enterobacteriaceae
## ASV0008    Akkermansiaceae
## ASV0009    Lachnospiraceae
```

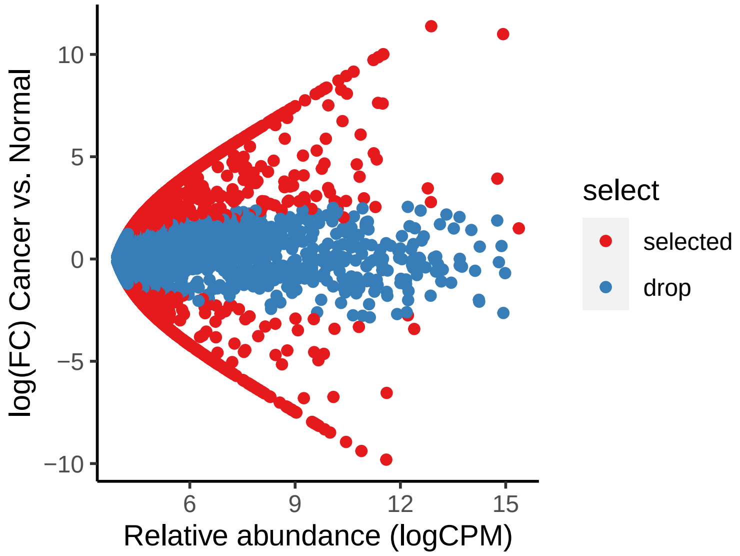
## Basal feature visualization

we use edgeR to demonstrate the several plots.

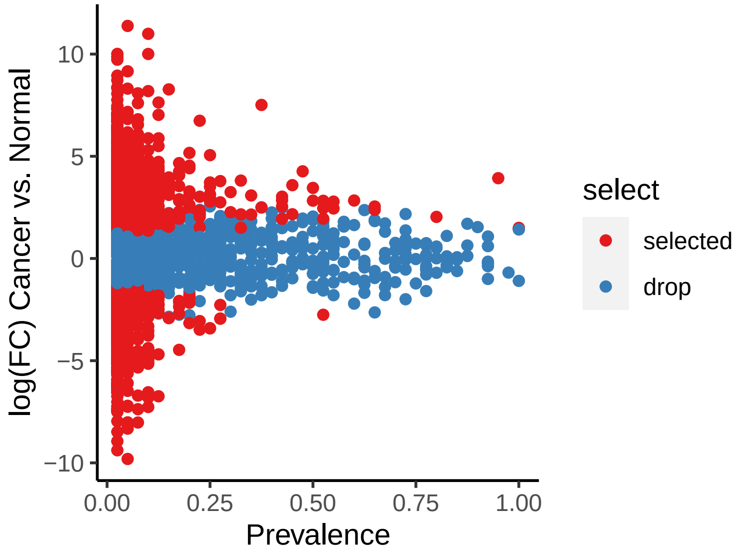
```
library(edgeR)

ASV <- phyloseq::otu_table(data, taxa_are_rows = T)
sampledata <- phyloseq::sample_data(meta, errorIfNULL = T)
phylo <- phyloseq::merge_phyloseq(ASV, sampledata)
test <- phyloseq_to_edgeR(physeq = phylo, group = "Class")
et = exactTest(test)
out = topTags(et, n=nrow(test$table), adjust.method="BH", sort.by="PValue")
edgeR_result <- out@.Data[[1]]
edgeR_selected <- edgeR_result[edgeR_result$FDR < 0.05,]

et = exactTest(test)
et = et$table
et$select <- "drop"
et$select[rownames(et) %in% rownames(edgeR_selected)] = "selected"

data_freq <- data
for(i in 1:ncol(data_freq)){
  data_freq[,i] <- data_freq[,i]/colSums(data_freq)[i]
}


get_pvl <- function(m){
  m_ <- m
  m_[m_ > 0] = 1
  m_ <- as.matrix(m_)
  return(as.numeric(rowSums(m_)/ncol(m_)))
}

variance <- c()
for(i in 1:nrow(data_freq)){
  variance <- c(variance, log(sd(data_freq[i,])**2))
}

et$variation <- variance
et$prevalence <- get_pvl(data)
et$neglog2p <- -log(et$PValue,2)
et$select <- factor(et$select, levels = c('selected','drop'))

ggplot(et, aes(x = logFC, y = neglog2p, color = select)) + geom_point() +
  xlab('log(FC) Cancer vs. Normal') + ylab('-log2(p value)') +
  scale_color_brewer(palette = 'Set1') + theme(axis.line = element_line(linetype = 1,colour = 'black'),
        panel.background = element_rect(I(0)),
        panel.grid.major = element_line(colour = NA),
        panel.grid.minor = element_line(colour = NA))
```



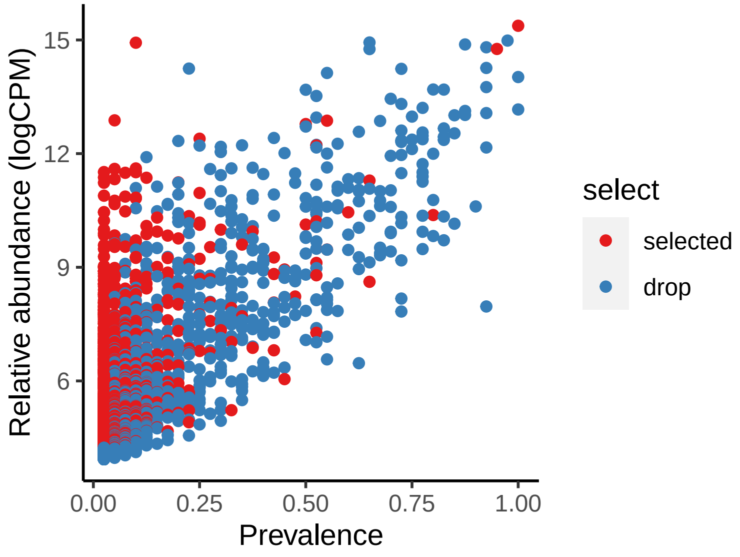
```
ggplot(et, aes(x = logCPM, y = logFC, color = select)) + geom_point() +
  xlab('Relative abundance (logCPM)') + ylab('log(FC) Cancer vs. Normal') +
  scale_color_brewer(palette = 'Set1') +
  theme(axis.line = element_line(linetype = 1,colour = 'black'),
        panel.background = element_rect(I(0)),
        panel.grid.major = element_line(colour = NA),
        panel.grid.minor = element_line(colour = NA))
```



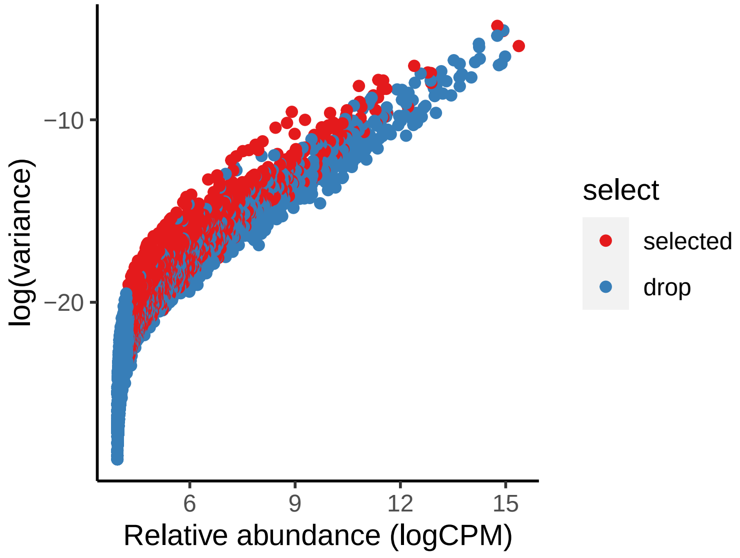
```
ggplot(et, aes(x = prevalence, y = logFC, color = select)) + geom_point() +
  xlab('Prevalence') + ylab('log(FC) Cancer vs. Normal') +
  scale_color_brewer(palette = 'Set1') +
  theme(axis.line = element_line(linetype = 1,colour = 'black'),
        panel.background = element_rect(I(0)),
        panel.grid.major = element_line(colour = NA),
        panel.grid.minor = element_line(colour = NA))
```



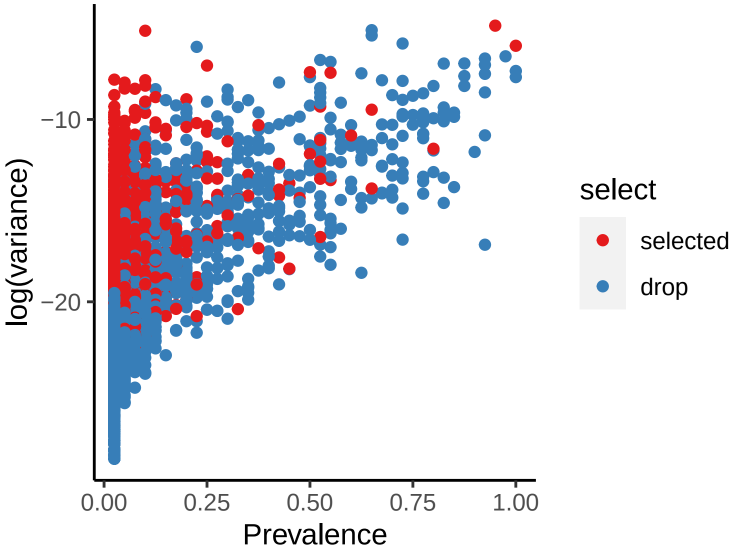
```
ggplot(et, aes(x = prevalence, y = logCPM, color = select)) + geom_point() +
  xlab('Prevalence') + ylab('Relative abundance (logCPM)') +
  scale_color_brewer(palette = 'Set1') +
  theme(axis.line = element_line(linetype = 1,colour = 'black'),
        panel.background = element_rect(I(0)),
        panel.grid.major = element_line(colour = NA),
        panel.grid.minor = element_line(colour = NA))
```



```
ggplot(et, aes(x = logCPM, y = variance, color = select)) + geom_point() +
  xlab('Relative abundance (logCPM)') + ylab('log(variance)') +
  scale_color_brewer(palette = 'Set1') +
  theme(axis.line = element_line(linetype = 1,colour = 'black'),
        panel.background = element_rect(I(0)),
        panel.grid.major = element_line(colour = NA),
        panel.grid.minor = element_line(colour = NA))
```



```
ggplot(et, aes(x = prevalence, y = variance, color = select)) + geom_point() +
  xlab('Prevalence') + ylab('log(variance)') +
  scale_color_brewer(palette = 'Set1') +
  theme(axis.line = element_line(linetype = 1,colour = 'black'),
        panel.background = element_rect(I(0)),
        panel.grid.major = element_line(colour = NA),
        panel.grid.minor = element_line(colour = NA))
```

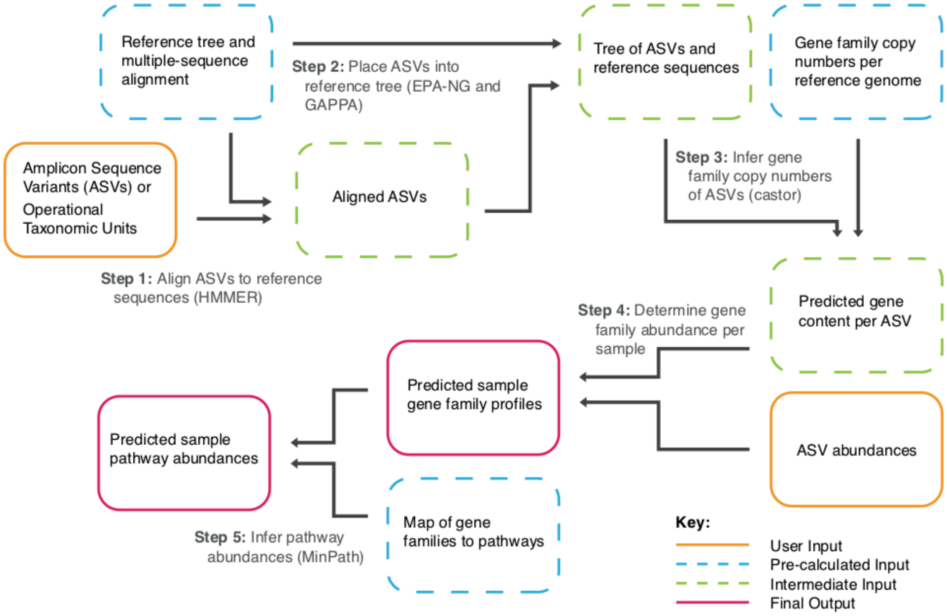


# Functional Prediction

## PICRUSt2

Prepare files for running PICRUSt2 by extracting the selected ASVs.

```
# please ensure the "ASV.fasta" is in "output_path" directory
output_path <- '~/Course/NDMC_2025/tmp'
dir(output_path)
## [1] "ASV_table.txt"      "ASV_taxa_table.txt" "ASV.fasta"
## [4] "ASV.tree"           "for_PICRUSt2.tsv"   "LEfSe_tmp"
## [7] "PICRUSt2"           "PreLect_tmp"        "ssuout"
selected_id <- featpropt$FeatName[featpropt$selected == 'Selected'] # selected by PreLect
data_sub <- data[selected_id, ]  # if you want to use whole ASVs, just data_sub <- data
data_sub <- rbind(colnames(data_sub), data_sub)
colnames(data_sub) <- NULL
rownames(data_sub)[1] <- "#OTU ID"
write.table(data_sub, paste0(output_path,"/for_PICRUSt2.tsv"), sep = "\t", quote=F, col.names   =F)
```


Reference tree and multiple-sequence alignment
Step 2: Place ASVs into reference tree (EPA-NG and GAPPA)
Tree of ASVs and reference sequences
Gene family copy numbers per reference genome
Amplicon Sequence Variants (ASVs) or Operational Taxonomic Units
Aligned ASVs
Step 3: Infer gene family copy numbers of ASVs (castor)
Step 1: Align ASVs to reference sequences (HMMER)
Step 4: Determine gene family abundance per sample
Predicted gene content per ASV
Predicted sample gene family profiles
Predicted sample pathway abundances
ASV abundances
Map of gene families to pathways
Step 5: Infer pathway abundances (MinPath)
Key:
User Input
Pre-calculated Input
Intermediate Input
Final Output

Conduct the PICRUSt2 pipeline via docker

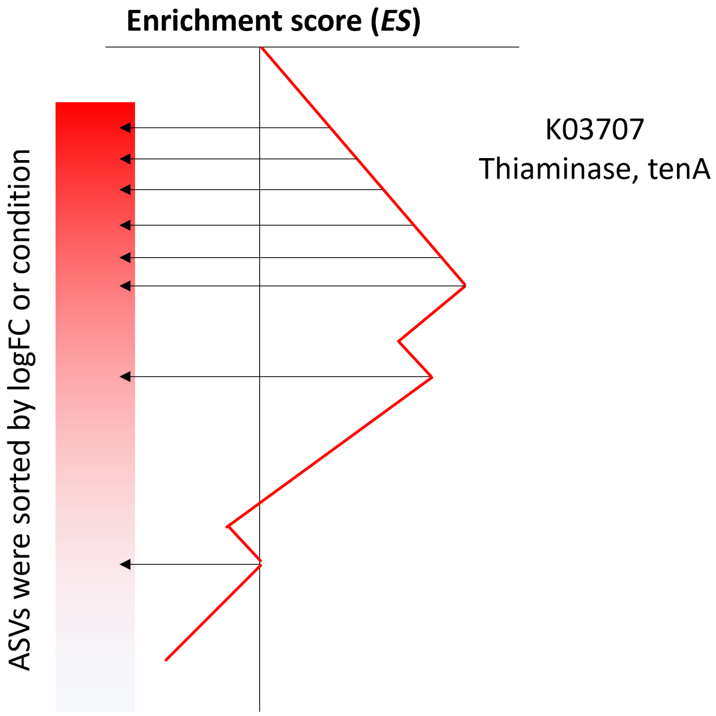
```
# Bash
# run the PICRUSt2 pipeline

# ensure the "ASV.fasta" and "for_PICRUSt2.tsv" files are in the same directory
# and set the working directory to that location before running this script.
$ cd ~/Course/NDMC_2025/tmp
$ docker run \
$    -u $UID:$(id -g) \              # perform this work as the current user.
$    -i -t \                        # interactive terminal mode, allowing you to interact with the cont
$    --rm \                         # automatically remove the container after it exits to save disk sp
$    -v $(pwd):/tmp \               # bind mount the current directory ($(pwd)) to /tmp in the containe
$    yincheng23/picrust2:0.2.0 \    # specify the Docker image to use; it will be pulled from Docker Hu
$    sh /bin/Run_picrust2.sh 10     # use the shell to execute the built-in script (Run_picrust2.sh) wi

# the results will store as ~/Course/NDMC_2025/tmp/PICRUSt2
```

output file : ./PICRUSt2/KO/pred_metagenome_contrib.tsv - Represents the relationship between functions and taxa.

output file : ./PICRUSt2/KO/pred_metagenome_unstrat_descrip.tsv - Contains KO (KEGG Orthology) abundances for each sample along with detailed gene descriptions.

output file : ./PICRUSt2/KO/pred_metagenome_unstrat.tsv.gz - Provides KO abundances for each sample.

Since `PICRUSt2` predicts gene abundance based on taxa abundance, differential expression analysis using this approach may be less convincing. Therefore, we adopted a `GSEA` strategy, ranking taxa by fold change based on their abundance. Using insights provided by `PICRUSt2`, we examined which species carry specific genes to assess the KOs that are actively expressed or suppressed.



Conduct the GSEA with permutation test via `GSEATestwithFC` or `GSEATest`

```
# load the KO-taxa relationship file
# output_path is the directory you save the for_PICRUSt2.tsv file
KOindex <- read.table(paste0(output_path,"/PICRUSt2/KO/pred_metagenome_contrib.tsv"), sep = "\t", heade

# extract the selected ASV identifiers
selected_id <- featpropt$FeatName[featpropt$selected == 'Selected']

# Provide the raw count table for logFC calculation, subsetting with the selected ASVs.
# Additionally, specify the labels for each sample and the name of the case sample group.
GSEAresult <- GSEATestwithFC(KOindex, data[selected_id, ], meta$Class, "Cancer")

## Building the KO-to-taxa mapper...
## Done. In total, 5755 KOs need to be processed.
## Shuffling the labels for GSEA...
## Performing GSEA to identify activated KOs...
##   |==============================================================================================
## Shuffling the labels for GSEA...
## Performing GSEA to identify suppressed KOs...
##   |==============================================================================================
## Done.


# If the grouping variable is continuous (e.g., body mass index), use the following line instead:
# GSEAresult <- GSEATest(KOindex, data[selected_id, ], meta$body.mass.index)
```

Selected the enriched KOs with z-score

```r
Actived_result <- GSEAresult$Actived_KO
Actived_result <- Actived_result[!is.na(Actived_result$z), ]
Actived_result <- Actived_result[Actived_result$z > 2,]
Actived_result <- Actived_result[Actived_result$p < 0.05,]
nrow(Actived_result)
## [1] 630
head(Actived_result)
##        KO        ES        z     p
## 10 K11392 0.5197568 2.016035 0.034
## 29 K07102 0.4521472 2.035035 0.033
## 64 K09160 0.4521472 2.097790 0.032
## 86 K18895 0.9760479 2.131169 0.005
## 93 K03799 0.2475731 2.475009 0.019
## 97 K01879 0.2366376 2.433231 0.019
```

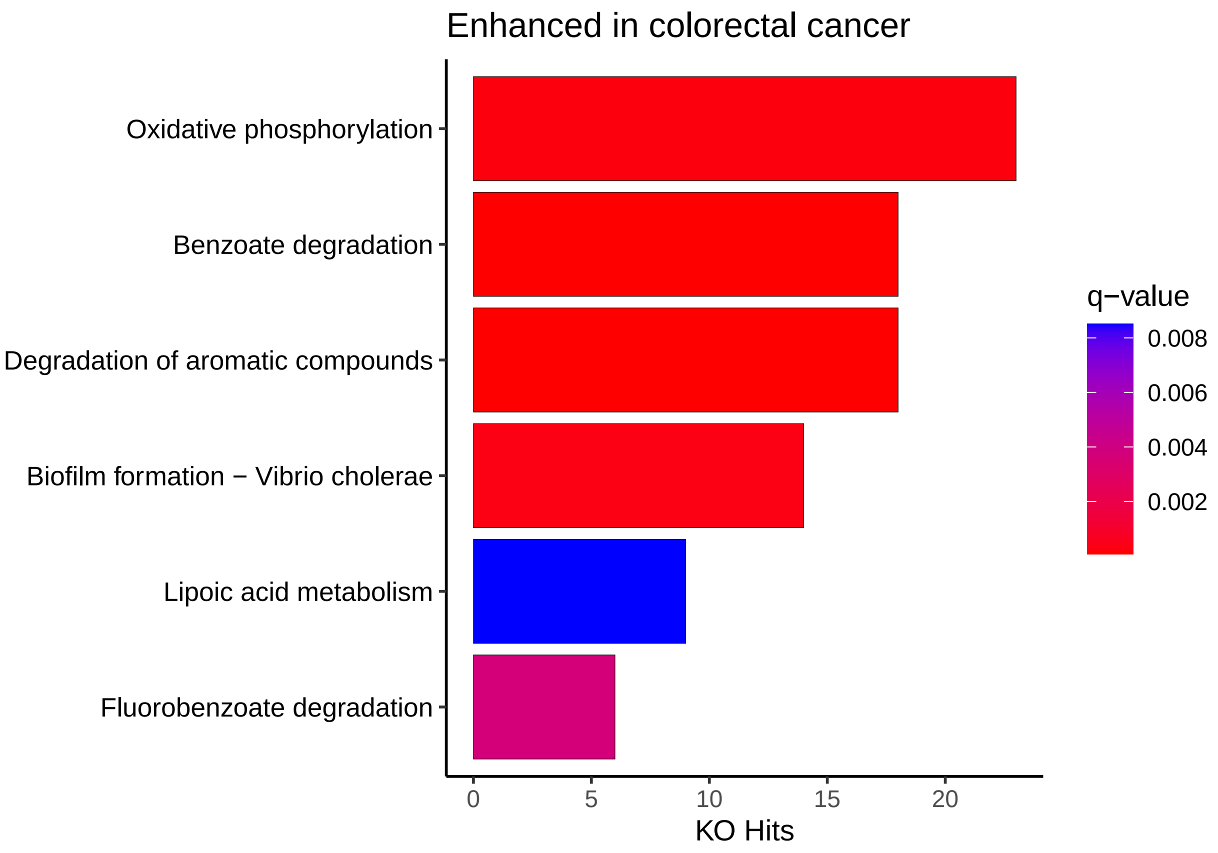
Since PICRUSt2 does not provide detailed information for each KO, we preprocess the KO-pathway table using the KEGG API.

```r
kodb_path <- system.file("exdata", "total_KO_pathway.rds", package = "PreLectR")
KOinfo <- readRDS(kodb_path)
head(KOinfo)
##       KO        symbol                               name    mapid
## 1 K00001 E1.1.1.1, adh alcohol dehydrogenase [EC:1.1.1.1] map00010
## 2 K00001 E1.1.1.1, adh alcohol dehydrogenase [EC:1.1.1.1] map00071
## 3 K00001 E1.1.1.1, adh alcohol dehydrogenase [EC:1.1.1.1] map00350
## 4 K00001 E1.1.1.1, adh alcohol dehydrogenase [EC:1.1.1.1] map00620
## 5 K00001 E1.1.1.1, adh alcohol dehydrogenase [EC:1.1.1.1] map00625
## 6 K00001 E1.1.1.1, adh alcohol dehydrogenase [EC:1.1.1.1] map00626
##                                     pathway
## 1              Glycolysis / Gluconeogenesis
## 2                    Fatty acid degradation
## 3                       Tyrosine metabolism
## 4                       Pyruvate metabolism
## 5 Chloroalkane and chloroalkene degradation
## 6                   Naphthalene degradation
```

Examining the condition enriched pathway with Fisher's exact test via PathwayEnrichment

```r
KOinfo <- KOinfo[KOinfo$KO %in% unique(KOindex$function.), ]
Actived_result <- GSEAresult$Actived_KO
Actived_result <- Actived_result[!is.na(Actived_result$z), ]
Actived_result <- Actived_result[Actived_result$z > 2,]
Actived_result <- Actived_result[Actived_result$p < 0.05,]
selected_KOs <- Actived_result$KO

enrichPW <- PathwayEnrichment(selected_KOs, KOinfo)
enrichPW$q <- p.adjust(enrichPW$p, method = 'fdr')
enrichPW <- enrichPW[enrichPW$q < 0.05, ]
nrow(enrichPW)
## [1] 6

head(enrichPW)
##                                  pathway       id count     ratio            p
## 26    Degradation of aromatic compounds map01220    18 0.3829787 9.848597e-07
## 31               Lipoic acid metabolism map00785     9 0.4090909 2.743792e-04
## 34            Oxidative phosphorylation map00190    23 0.3026316 4.214420e-06
## 64                 Benzoate degradation map00362    18 0.4000000 4.614879e-07
## 112          Fluorobenzoate degradation map00364     6 0.6666667 1.007485e-04
## 133 Biofilm formation - Vibrio cholerae map05111    14 0.4000000 7.841760e-06
##     odds_ratio            q
## 26    5.161160 9.159195e-05
## 31    5.756061 8.505756e-03
## 34    3.605573 2.612941e-04
## 64    5.543134 8.583675e-05
## 112  16.614397 3.747845e-03
## 133   5.572086 3.646418e-04
```

```
enrichPW$pathway <- factor(enrichPW$pathway, levels = enrichPW$pathway[order(enrichPW$count)])
ggplot(enrichPW, aes(x = pathway, y = count, fill = q)) + ggtitle('Enhanced in colorectal cancer') +
  geom_bar(stat="identity",colour = "black",size = 0.1) +  coord_flip() + labs(y = 'KO Hits', fill = "q
  scale_fill_continuous(low = "red", high = "blue", limits = range(enrichPW$q)) +
  theme(axis.text.y = element_text(color='black', size='10'),
        axis.line = element_line(linetype = 1,colour = 'black'),
        axis.title.y = element_blank(),
        panel.background = element_rect(I(0)),
        panel.grid.major = element_line(colour = NA),
        panel.grid.minor = element_line(colour = NA)) + coord_flip()
```



## RPM

Reference Pathways Mapper (RPM) utilizes a reference pathways database, which is a flat file listing pathway/module reactions in sequential order. Tab-separated reactions indicate alternative reactions (OR operation), while line breaks and comma-separated reactions represent reactions that are all required for pathway completion (AND operation).

Below is a snippet from the `human gut metabolic modules (GMMs)` database, as described by Vieira-Silva et al. 2016.

```
MF0001  arabinoxylan degradation
K01209  K15921  K01181  K01198  K15531  K18205
///
MF0003  pectin degradation I
K01051
K01184,K01213   K18650
///
MF0103  mucin degradation
K01186
K05970
K01132  K01135  K01137  K01205
K01207  K12373  K14459
K01205  K01207  K12373  K01227  K13714
K01206
///
```

**Installation** According to original repository

```
$ wget https://github.com/omixer/omixer-rpmR/releases/download/0.3.3/omixerRpm_0.3.3.tar.gz
$ R CMD INSTALL omixerRpm_0.3.3.tar.gz
```

```
library(omixerRpm)
library(dplyr)
library(tidyr)

KOtable <- read.table(paste0(output_path,"/PICRUSt2/KO/pred_metagenome_contrib.tsv"), sep = "\t", heade

colnames(KOtable)[2] <- 'fun'
sumtb <- KOtable %>% group_by(sample, fun) %>% summarise(sum_rel_function_abun=sum(taxon_rel_function_ab
tmp <- spread(sumtb, key = sample, value = sum_rel_function_abun)
colnames(tmp)[1] <- 'entry'
tmp[is.na(tmp)] <- 0

mods <- rpm(tmp, minimum.coverage=0.3, annotation = 1)
## [1] "Loaded GMMs.v1.07"
db <- loadDefaultDB()
## [1] "Loaded GMMs.v1.07"
getNames(db, mods@annotation[1,])
## [1] "ribose degradation"
coverage <- asDataFrame(mods, "coverage")
```

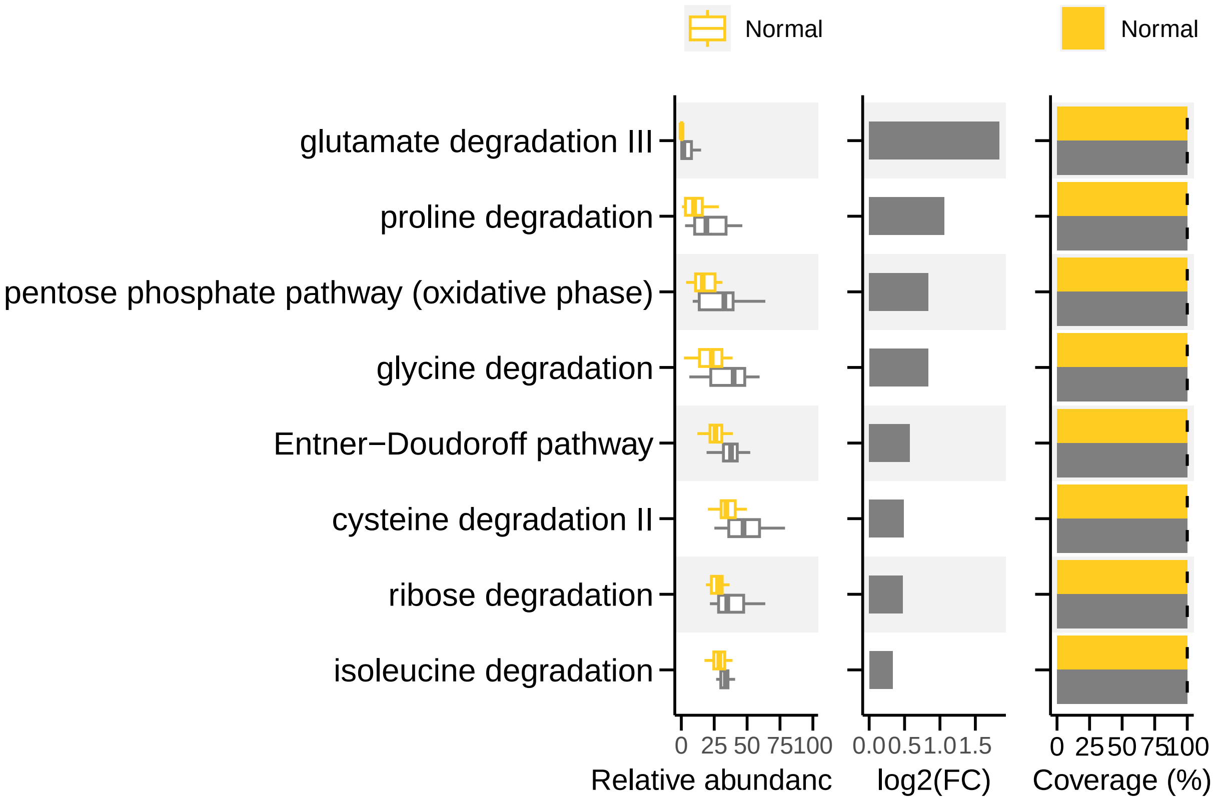
```
module  <- mods@abundance
rownames(module) <- coverage$Description
head(module)
##                                   ERR475527 ERR475549    ERR475504   ERR475529
## ribose degradation              35.16937498  35.17516 3.098085e+01  55.4419850
## tyrosine degradation II          0.00736405   0.00000 0.000000e+00   0.1532172
## aspartate degradation I         56.20638043  68.65780 5.569382e+01  54.3297498
## tryptophan degradation           5.20525631   1.12031 1.160618e+01   3.2142130
## tyrosine degradation I          50.58997819 101.04308 5.424949e+01 104.9491610
## galacturonate degradation II     0.00000000   0.00000 9.507594e-04   0.0000000
##                                  ERR475528   ERR475545   ERR475500   ERR475588
## ribose degradation              27.8606223 27.09122669 63.79307368 50.59532850
## tyrosine degradation II          0.0159962  0.00000000  0.14151356  0.00000000
## aspartate degradation I         79.0703436 87.57625174 87.98962719 63.41611505
## tryptophan degradation          10.6104558 21.77601754  2.87128420 33.18194936
## tyrosine degradation I          51.9888339 60.34117553 53.45470956 41.96002481
## galacturonate degradation II     0.0000000  0.01126817  0.01038632  0.03272184
##                                 ERR475547  ERR475541 ERR475485    ERR475540
## ribose degradation               31.55949 28.4148131 25.326445  23.52926496
## tyrosine degradation II           0.00000  0.1387234  0.000000   0.08120360
## aspartate degradation I          87.24203 59.8965261 49.600167 117.38540049
## tryptophan degradation           15.62138  4.5938590  3.226059  16.50346978
## tyrosine degradation I           52.56882 51.8888387 50.360550  58.77502133
## galacturonate degradation II      0.00000  0.0175545  0.000000   0.02160463
##                                 ERR475562    ERR475584   ERR475521    ERR475565
## ribose degradation               21.27139 22.234725150 21.73869043 18.653580678
## tyrosine degradation II           0.00000  0.019633212  0.08697272  0.095860634
## aspartate degradation I          61.78199 51.502887344 71.82182553 36.918989696
## tryptophan degradation           21.02328  4.870319828 13.28846530 12.576196223
## tyrosine degradation I           56.55310 51.969782663 55.30241396 52.756376669
## galacturonate degradation II      0.00000  0.005960082  0.04027999  0.002396516
##                                   ERR475586  ERR475580    ERR475484 ERR475561
## ribose degradation              46.99615938 34.6893540  27.80085184 36.651911
## tyrosine degradation II          0.43654951  0.0000000   0.03075402  0.000000
## aspartate degradation I         78.71544166 48.5301653  63.21129882 53.275009
## tryptophan degradation          14.27417268  8.0092019   6.06292426  1.126657
## tyrosine degradation I          50.86031496 50.9725785 101.77878685 50.407778
## galacturonate degradation II     0.01942096  0.0221183   0.00000000  0.000000
##                                   ERR475483 ERR475560  ERR475590   ERR475518
## ribose degradation               29.1709203 45.736165 29.9555807 28.04525333
## tyrosine degradation II           0.0000000  0.000000  0.0000000  0.01163850
## aspartate degradation I          58.3060655 41.854488 72.7521866 49.14149961
## tryptophan degradation            0.6992597  2.599723 19.2800891  3.64808639
## tyrosine degradation I          102.9874650 49.899148 54.3941577 53.93726862
## galacturonate degradation II      0.0000000  0.000000  0.1025416  0.02168992
##                                   ERR475534    ERR475478    ERR475555    ERR475577
## ribose degradation              31.02395184 3.450413e+01 3.441916e+01 48.548405549
## tyrosine degradation II          0.05004837 1.083518e-01 0.000000e+00  0.127484993
## aspartate degradation I         55.09630176 5.241802e+01 6.256350e+01 57.795267824
## tryptophan degradation           2.10620205 6.076617e+00 3.097498e-01  1.738492718
## tyrosine degradation I          49.95218205 1.067859e+02 1.007209e+02 51.958954529
## galacturonate degradation II     0.00000000 8.334751e-03 3.997287e-03  0.009393631
##                                   ERR475513   ERR475579   ERR475554    ERR475576
## ribose degradation           31.59691328 22.92837864 18.73800856  37.82823763
## tyrosine degradation II       0.02523416  0.15720108  0.06997963   0.31456905
## aspartate degradation I      90.11875161 91.17260136 76.39798496 148.87959127
## tryptophan degradation       25.45861537 23.86014480  1.94103675  59.91196697
## tyrosine degradation I       57.19484500 45.59180272 50.44075281  74.74717910
## galacturonate degradation II  0.07131394  0.02515217  0.00000000   0.01037918
##                                 ERR475476  ERR475553    ERR475493    ERR475591
## ribose degradation           23.014066166 40.9080176 2.808007e+01 27.068650441
## tyrosine degradation II       0.044911132  0.0000000 1.930299e+00  0.002083388
## aspartate degradation I      67.459882221 85.0328572 4.551332e+01 62.929806811
## tryptophan degradation       12.719718609  0.5422964 5.907959e+00  8.902784854
## tyrosine degradation I       53.791136776 99.3264616 1.021285e+02 56.030165476
## galacturonate degradation II  0.009710515  0.0000000 7.034617e-03  0.048438766
##                               ERR475473   ERR475550  ERR475594    ERR475480
## ribose degradation           19.1812121 32.08871268 62.5142559  31.15863997
## tyrosine degradation II       0.1721522  0.03903622  0.3954753   0.00000000
## aspartate degradation I      58.6562929 78.48988082 94.8305828  35.21222110
## tryptophan degradation        4.5486796  0.62264043  5.3451859   0.16811702
## tyrosine degradation I       58.1781156 49.20786881 61.6465806 106.09704137
## galacturonate degradation II  0.0000000  0.00000000  0.2272061   0.03641655
```

### Pathway mapping

**Module visualization** Please ensure that the dataframe `meta` has a column named `group`, which records the grouping information.

Only binary grouping is supported in the following process.

```
meta$group <- meta$Class # pleas ensure this

# GMMviz function is sourced from utils.R
results <- GMMviz(module, meta, 'Normal') # please assign the control name as the third parameter.
sigdf <- results$significance_df
boxdf <- results$relative_abundance_df
covdf <- results$coverage_df

sigdf <- sigdf[sigdf$p < 0.01, ]
boxdf <- boxdf[boxdf$module %in% sigdf$module,]
covdf <- covdf[covdf$module %in% sigdf$module,]

module_order <- sigdf$module[order(sigdf$log2FC, decreasing = F)]
sigdf$module <- factor(sigdf$module, levels = module_order)
boxdf$module <- factor(boxdf$module, levels = module_order)
covdf$module <- factor(covdf$module, levels = module_order)

mycolor <- c("Normal" = "#FFCC22", "Cancer" = "#EE7700") # color assignment
p1 <- ggplot(boxdf, aes(x = module, y = value, color = group)) +  coord_flip() +
      scale_x_discrete() +
      ylab("Relative abundance (%)") + scale_color_manual(values = mycolor) +
      theme(panel.background = element_rect(fill = 'transparent'),
        panel.grid = element_blank(),
        axis.ticks.length = unit(0.4,"lines"),
        axis.ticks = element_line(color='black'),
        axis.line = element_line(colour = "black"),
        axis.text.y = element_text(colour='black',size=12),
        axis.title.y = element_blank(),
        legend.title=element_blank(),
        legend.position = 'top')
sing = 1
for (i in 1:(nrow(sigdf)-1)){
  sing = sing * -1
  p1 <- p1 + annotate('rect', xmin = i+0.5, xmax = i+1.5, ymin = -Inf, ymax = Inf,
                      fill = ifelse(sing > 0, 'white', 'gray95'))
}
p1 <- p1 + geom_boxplot(outlier.shape = NA, width = 0.5)

p2 <- ggplot(sigdf, aes(x = module, y = log2FC, fill = group)) +  coord_flip() +
      geom_bar(stat = "identity", width=.5, position = "dodge") + scale_x_discrete() +
      scale_fill_manual(values = mycolor) + ylab("log2(FC)") +
      theme(panel.background = element_rect(fill = 'transparent'),
        panel.grid = element_blank(),
        axis.ticks.length = unit(0.4,"lines"),
        axis.ticks = element_line(color='black'),
        axis.line = element_line(colour = "black"),
        #axis.text = element_text(colour='black',size=10),
        axis.text.y = element_blank(),
        axis.title.y = element_blank(),
        legend.title=element_blank(),
        legend.position = 'top')
sing = 1
for (i in 1:(nrow(sigdf)-1)){
  sing = sing * -1
  p2<- p2 + annotate('rect', xmin = i+0.5, xmax = i+1.5, ymin = -Inf, ymax = Inf,
                      fill = ifelse(sing > 0, 'white', 'gray95'))
}
p2 <- p2 + geom_bar(stat = "identity", width=.5, position = "dodge")

p3 <- ggplot(covdf, aes(x = module, y = mean, fill = group)) +
      scale_x_discrete() + coord_flip() + scale_fill_manual(values = mycolor) +
      theme(panel.background = element_rect(fill = 'transparent'),
        panel.grid = element_blank(),
        axis.ticks.length = unit(0.4,"lines"),
        axis.ticks = element_line(color='black'),
        axis.line = element_line(colour = "black"),
        axis.text = element_text(colour='black',size=10),
        axis.text.y = element_blank(),
        axis.title.y = element_blank(),
        legend.title=element_blank(),
        legend.position = 'top')
sing = 1
for (i in 1:(nrow(sigdf)-1)){
  sing = sing * -1
  p3 <- p3 + annotate('rect', xmin = i+0.5, xmax = i+1.5, ymin = -Inf, ymax = Inf,
                     fill = ifelse(sing > 0, 'white', 'gray95'))
}
p3 <- p3 + geom_bar(stat="identity", position=position_dodge()) + ylab("Coverage (%)") +
           geom_errorbar(aes(ymin=mean-std, ymax=mean+std), width=.3, position=position_dodge(.9))

p1 + p2 + p3
```



## Co-Occurrence Network

```
library(Rcpp)
library(igraph)
library(gtools)
library(Matrix)
library(SpiecEasi)
sourceCpp("source/Microbial_network.cpp")
```

### SPIEC-EASI

SPIEC-EASI(**SP**arse **I**nvers**E** **C**ovariance **E**stimation for **E**cological **A**ssociation **I**nference) infers direct microbial interactions from compositional data using a graphical model. It transforms counts into log-ratios, estimates a sparse inverse covariance matrix with regularization (e.g., Graphical Lasso), and builds a network where non-zero elements indicate direct species relationships, overcoming spurious correlations.

```
pvl_filter <- 0.1
SpiecEasi_threshold <- 0.7

# filter low prevalence feature
pdata <- data
pdata[pdata > 0] <- 1
keep_id <- rownames(pdata)[rowSums(pdata) > ncol(pdata)*pvl_filter]
filerted_data <- data[keep_id,]
dim(filerted_data)
## [1] 645  40

# run SpiecEas mb = Neighborhood Selection, glasso = graph LASSO
SPIEC_EASI <- spiec.easi(t(filerted_data), method='mb', lambda.min.ratio=1e-2, nlambda=15)
## Applying data transformations...
## Selecting model with pulsar using stars...
## Fitting final estimate with mb...
## done
adjm <- getOptMerge(SPIEC_EASI)
adjm <- as.matrix(adjm)
adjm[adjm < SpiecEasi_threshold] <- 0

edge_table <- make_edgetable(adjm, filerted_data)
node_table <- make_nodetable(edge_table, data, taxa)
```

## SparCC

SparCC (**S**parse **C**orrelations for **C**ompositional data) infers microbial interactions from compositional data by estimating correlations while accounting for the sum constraint. It assumes most species pairs are uncorrelated, uses log-ratio variance to approximate correlations, and applies a sparse model to focus on strong relationships. Iterative refinement reduces noise, producing a correlation network for ecological insights.

```
pvl_filter <- 0.1
SparCC_threshold <- 0.3
SparCC_conf_threshold <- 0.05

# filter low prevalence feature
pdata <- data
pdata[pdata > 0] <- 1
keep_id <- rownames(pdata)[rowSums(pdata) > ncol(pdata)*pvl_filter]
filerted_data <- data[keep_id,]
dim(filerted_data)

# run SparCC
sparcc_cor <- sparcc(t(filerted_data))
sparcc_boost <- sparccboot(t(filerted_data), R = 100, ncpus = 10)
sparcc_p <- pval.sparccboot(sparcc_boost)
cors <- sparcc_p$cors
pvals <- sparcc_p$pvals

# adjacency matrix constructing
sparCCpcors <- diag(0.5, nrow = dim(sparcc_cor$Cor)[1], ncol = dim(sparcc_cor$Cor)[1])
sparCCpcors[upper.tri(sparCCpcors, diag=FALSE)] <- cors
```

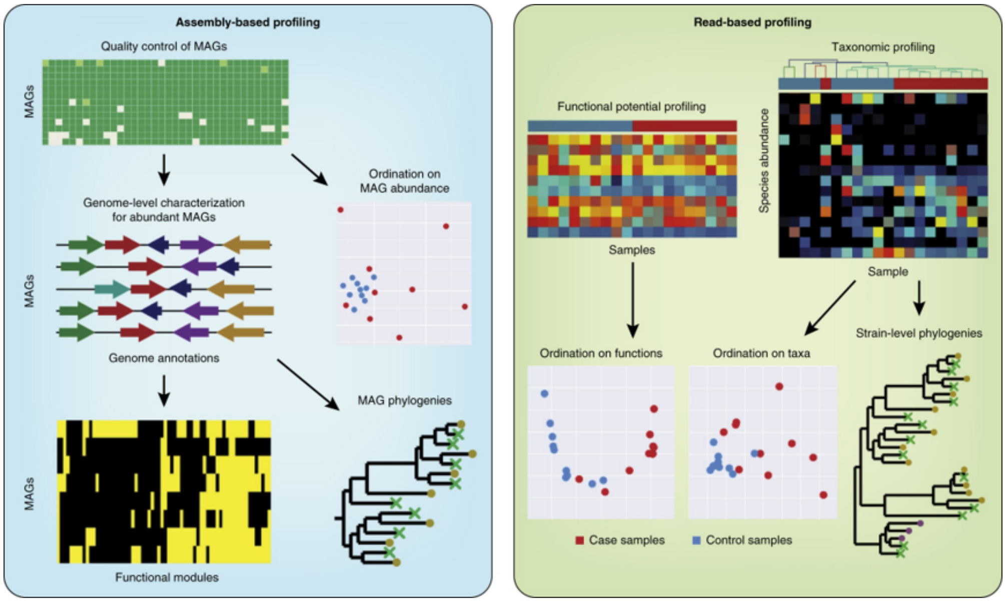
```
sparCCpcors <- sparCCpcors + t(sparCCpcors)
sparCCpval <- diag(0.5, nrow = dim(sparcc_cor$Cor)[1], ncol = dim(sparcc_cor$Cor)[1])
sparCCpval[upper.tri(sparCCpval, diag=FALSE)] <- pvals
sparCCpval <- sparCCpval + t(sparCCpval)
rownames(sparCCpcors) <- rownames(filerted_data)
colnames(sparCCpcors) <- rownames(filerted_data)
rownames(sparCCpval) <- rownames(filerted_data)
colnames(sparCCpval) <- rownames(filerted_data)
sparCCpcors[abs(sparCCpcors) < SparCC_threshold | sparCCpval > SparCC_conf_threshold] = 0

edge_table <- make_edgetable(sparCCpcors, filerted_data)
node_table <- make_nodetable(edge_table, data, taxa)
```

### Cluster identification

```
net <- make_net(edge_table)
net_pack <- graph_from_incidence_matrix(net, mode = "total", weighted = TRUE)
#clustB <- cluster_edge_betweenness(net_pack)
clustW <- cluster_walktrap(net_pack)
clustL <- cluster_louvain(net_pack)
clustG <- cluster_fast_greedy(net_pack)
clustLE <- cluster_leading_eigen(net_pack)
Modularity <- data.frame(Method = c("Random walks", "Louvain", "Greedy", "Non-negative eigenvector"),
                         Modularity = c(modularity(clustW), modularity(clustL),
                                        modularity(clustG),modularity(clustLE)))
Modularity
Modul_number_cutoff <- 3  # if the number of members in a cluster is lower than the cutoff, discard thi
edge_table <- assignModul(edge_table,clustW, "RandomWalks", Modul_number_cutoff)
edge_table <- assignModul(edge_table,clustL, "Louvain", Modul_number_cutoff)
edge_table <- assignModul(edge_table,clustG, "Greedy", Modul_number_cutoff)
edge_table <- assignModul(edge_table,clustLE, "Leading_eigen", Modul_number_cutoff)

write.csv(edge_table, "edge_table.csv", row.names = F,quote = F)
write.csv(node_table, "node_table.csv", row.names = F,quote = F)
```

Next, you can visualize with Gephi.

# Guide to Shotgun Metagenomic Data



## Read bases pipeline

Shotgun metagenomics involves sequencing all genetic material in a sample, generating vast amounts of short DNA fragments known as read bases. These raw reads undergo quality control, filtering, and taxonomic or functional annotation.

The Huttenhower Lab has developed a comprehensive pipeline called `bioBakery Workflows`, which includes:

- **KneadData**: Read quality control and host contamination removal
- **MetaPhlAn**: Taxonomic profiling
- **HUMAnN**: Functional profiling
- **PhyloPhlAn**: Phylogenetic analysis

Our lab also developed a user friendly `bioBakery Workflows` named `BacNex`

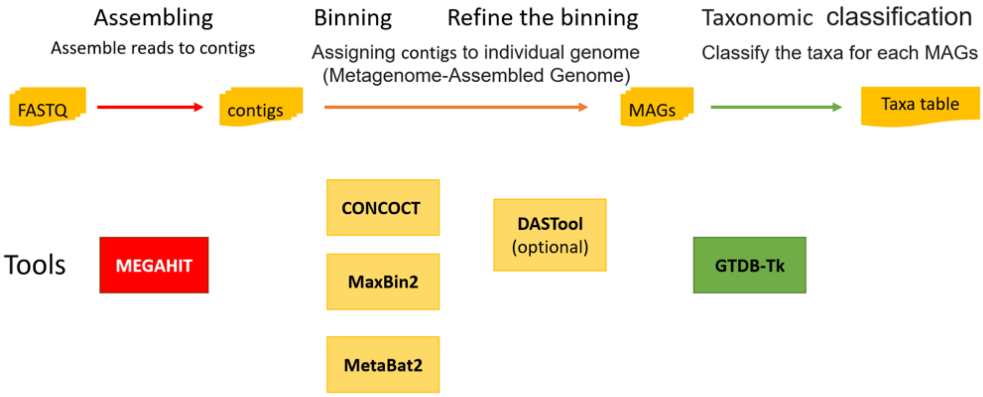
```
# Installation
$ git clone https://github.com/RynoLiu/BacNex.git
$ conda env create -f /env/bacnex_env_v2.yml --name bacnex_env
$ conda activate bacnex_env

# Build database
$ bash BacNex.sh make_database -o $OUTDIR

# Preprocess (KneadData -> MetaPhlAn -> HUMAnN)
$ bash BacNex.sh preprocess -i $seq_dir \
                            -o $outdir \
                            -db $DATABASE \
                            --threads 10
```

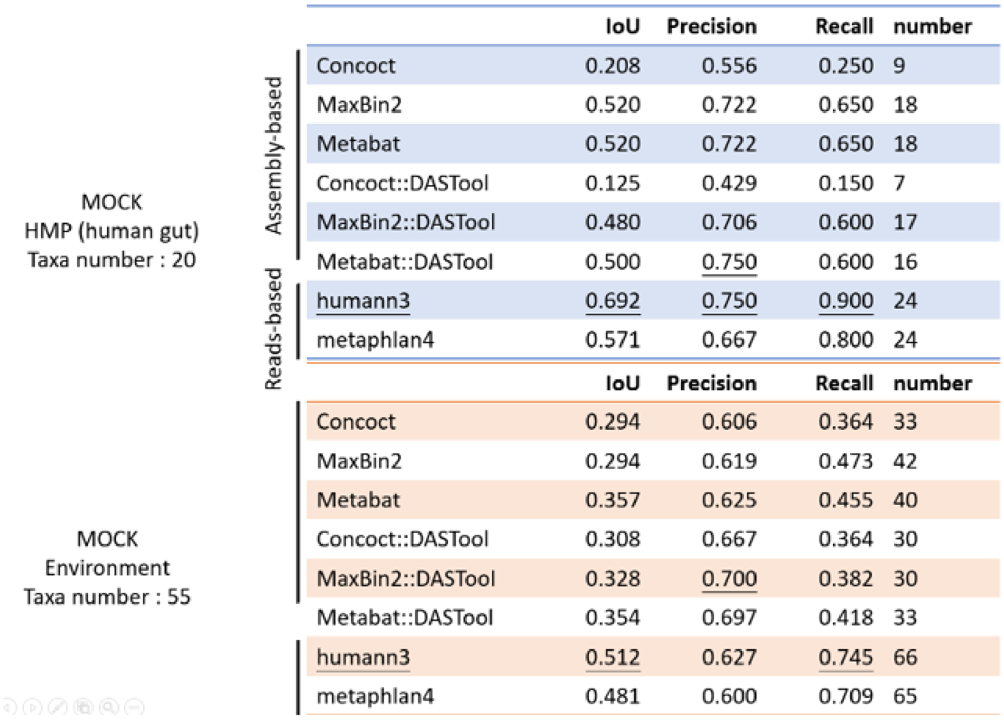
```
# Output HUMAnN table
$ python BacNex.py make_table -i $KO_TSV \
                              -o $OUTDIR

# shiny APP
$ bash BacNex.sh app -w $WORK_DIR \
                     -t 10
```

```
$ git clone https://github.com/RynoLiu/BacNex.git
$ conda env create -f /env/bacnex_env_v2.yml --name bacnex_env
$ conda activate bacnex_env
```

## Assembly bases pipeline

Assembly bases, on the other hand, refer to contigs or scaffolds reconstructed from overlapping reads, providing longer genomic sequences for better functional and phylogenetic analysis.





| MOCK<br>HMP (human gut)<br>Taxa number : 20 | | | IoU | Precision | Recall | number |
|---|---|---|---|---|---|---|
| | Assembly-based | Concoct | 0.208 | 0.556 | 0.250 | 9 |
| | | MaxBin2 | 0.520 | 0.722 | 0.650 | 18 |
| | | Metabat | 0.520 | 0.722 | 0.650 | 18 |
| | | Concoct::DASTool | 0.125 | 0.429 | 0.150 | 7 |
| | | MaxBin2::DASTool | 0.480 | 0.706 | 0.600 | 17 |
| | | Metabat::DASTool | 0.500 | 0.750 | 0.600 | 16 |
| | Reads-based | humann3 | 0.692 | 0.750 | 0.900 | 24 |
| | | metaphlan4 | 0.571 | 0.667 | 0.800 | 24 |

| MOCK<br>Environment<br>Taxa number : 55 | | IoU | Precision | Recall | number |
|---|---|---|---|---|---|
| | Concoct | 0.294 | 0.606 | 0.364 | 33 |
| | MaxBin2 | 0.294 | 0.619 | 0.473 | 42 |
| | Metabat | 0.357 | 0.625 | 0.455 | 40 |
| | Concoct::DASTool | 0.308 | 0.667 | 0.364 | 30 |
| | MaxBin2::DASTool | 0.328 | 0.700 | 0.382 | 30 |
| | Metabat::DASTool | 0.354 | 0.697 | 0.418 | 33 |
| | humann3 | 0.512 | 0.627 | 0.745 | 66 |
| | metaphlan4 | 0.481 | 0.600 | 0.709 | 65 |

While read-based analysis offers high-resolution species profiling, assembly-based approaches improve gene prediction and strain-level characterization. Choosing between them depends on study objectives, sample complexity, and computational resources.