Norwegian University of Life Sciences

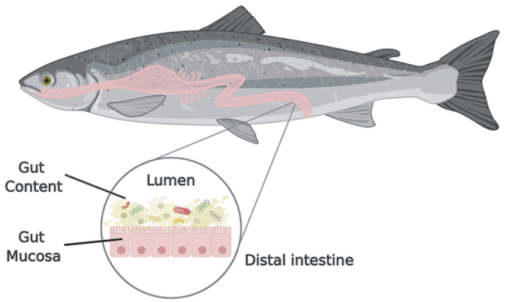
16S data processing and analysis

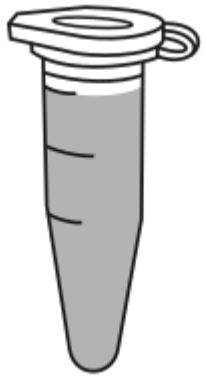
Dr. Sérgio Rocha

24.06.2025

General approach

Norwegian University of Life Sciences



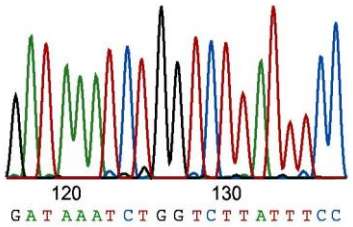


Digesta sample

16S rRNA gene amplification

16S rRNA

16S rRNA gene sequencing



DADA2 Amplicon Sequencing. Exactly. Version 1.14

silva high quality ribosomal RNA databases

Total DNA extraction → 16S amplificantion and library preparation → Sequecing → Data processing

Orion.nmbu.no

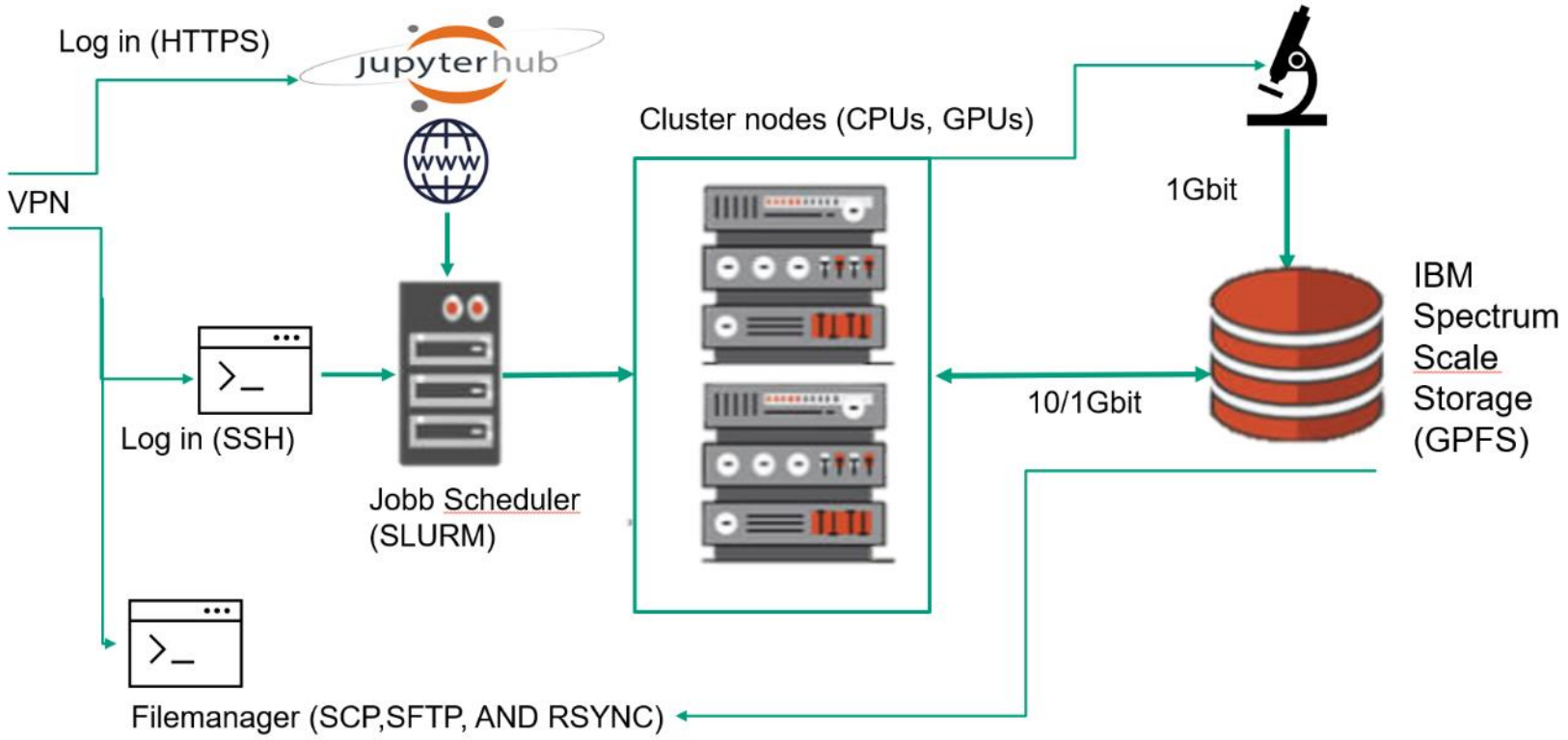
Norwegian University of Life Sciences

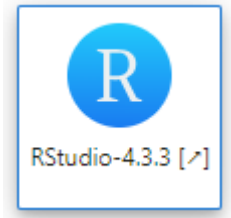
Orion HPC

Computing problems, such as genome sequence analysis, computational chemistry, simulation of universe activity, etc., require a different kind of computing power. These computing problems may need thousands, millions or billions of instructions to be completed.

Main configuration of the Orion HPC

The Orion HPC system currently consists of 1680 processor cores with more than 12 terabytes of RAM and 1 petabyte of storage accessible on a 10/1 Gbit network via NFS. The Orion HPC's computation nodes utilize various processors with Centos Linux 7.9 as an operating environment.



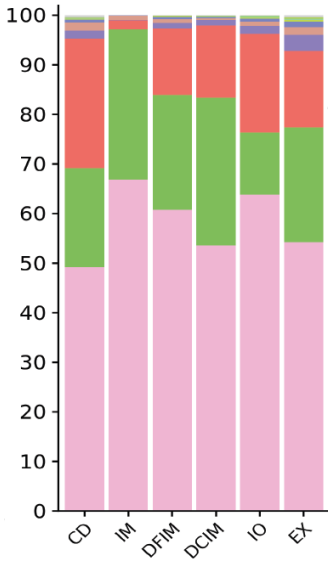


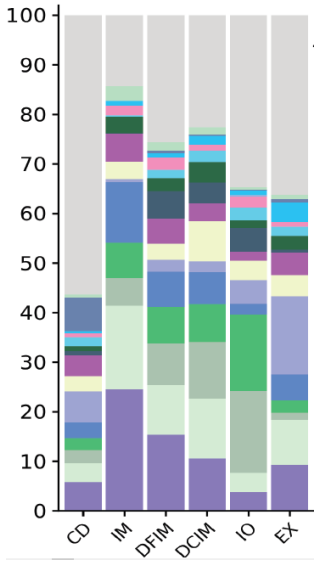
Orion HPC provides a supercomputing environment with thousands of cores to researchers and students to handle their computational problems. The following table summarizes the computing power capacity of Orion HPC nodes.

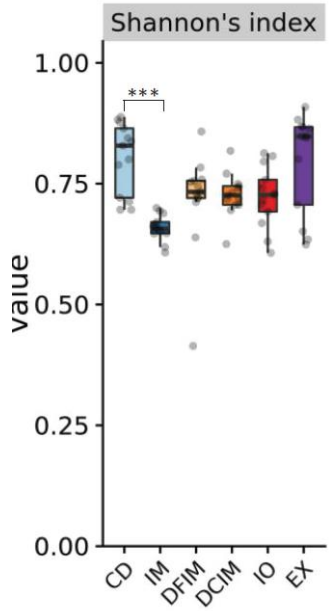
Examples

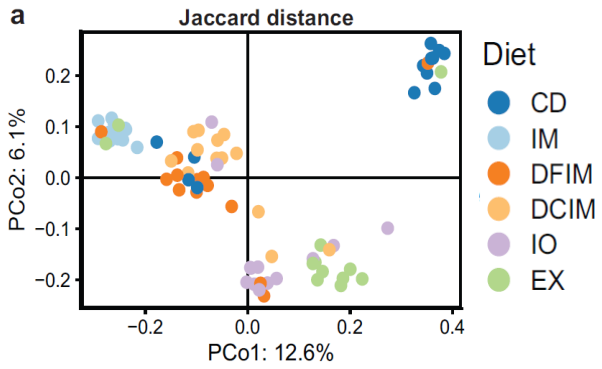
silva
high quality ribosomal RNA databases

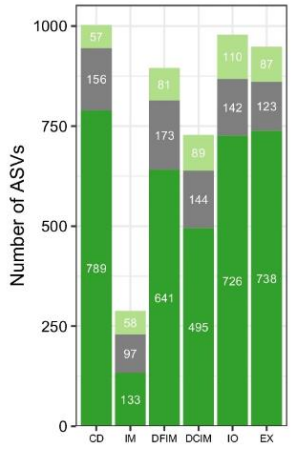
Norwegian University of Life Sciences

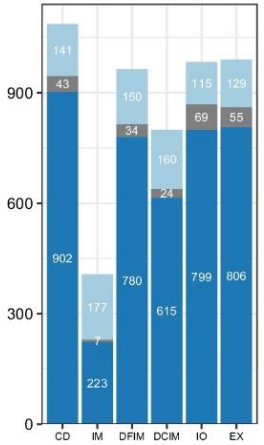


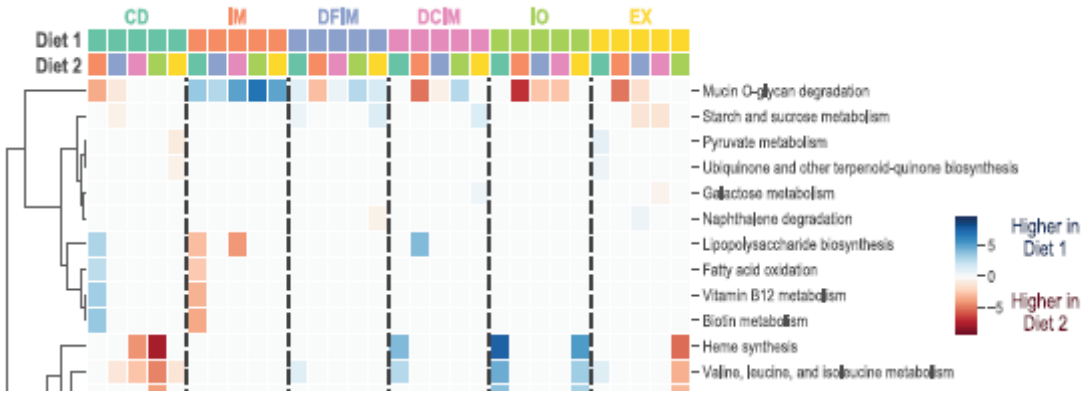




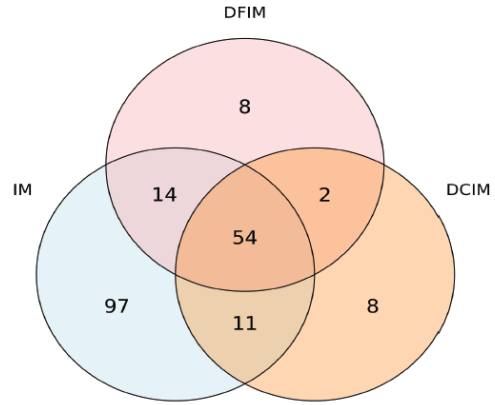








16



Some terminology

Norwegian University of Life Sciences

Amplicon: The resulting sequence of a targeted amplification of genetic material. Targeted meaning primers were used. (different from shotgut)

Marker gene: A gene that can be useful for delineating organisms, like a fingerprint (**16S rRNA**).

OUT: Operational Taxonomic Unit. OTUs are an artificial, arbitrary construct useful for grouping sequences together into units to help us summarize and analyze things, and they also intended to help deal with sequencing error intrinsic to the technology. → 97% or 99% OTU clustering

ASV: Amplicon Sequence Variant. Resulting sequences from newer processing methodologies that attempt to take into account sequencing error rates and are believed to represent true biological sequences.
Moving towards using ASVs over OTUs is supported in recent publications.

Barcodes: sequences ligated to your individual samples' genetic material before they get all mixed together to be sequenced together. These barcodes are then unique to each sample, so you can afterwards identify which sequences came from which samples.

Demultiplex: step in processing amplicon sequence data where you would use the barcode information in order to know which sequences came from which samples after they had all been sequenced together.

OTUs or ASVs?

Norwegian University of Life Sciences

There are several approaches, and you will likely get very similar results regardless of which tool you use (as long as you make similar decisions when processing your sequences like minimum abundance filtering, …).

Don't get too lost on trying to find the "best" tool for processing amplicon data.

There is a tendency on getting away from the traditional OTU approach and use more ASVs.

ASVs approach, is most often **more biologically meaningful** and a more useful unit beyond the current dataset than the traditional OTU clustering methods. *"It allows to find new sequences".*

If you are processing a new amplicon dataset, I would suggest to use DADA2, but if you have good support by using other pipeline, it is still a valid approach.

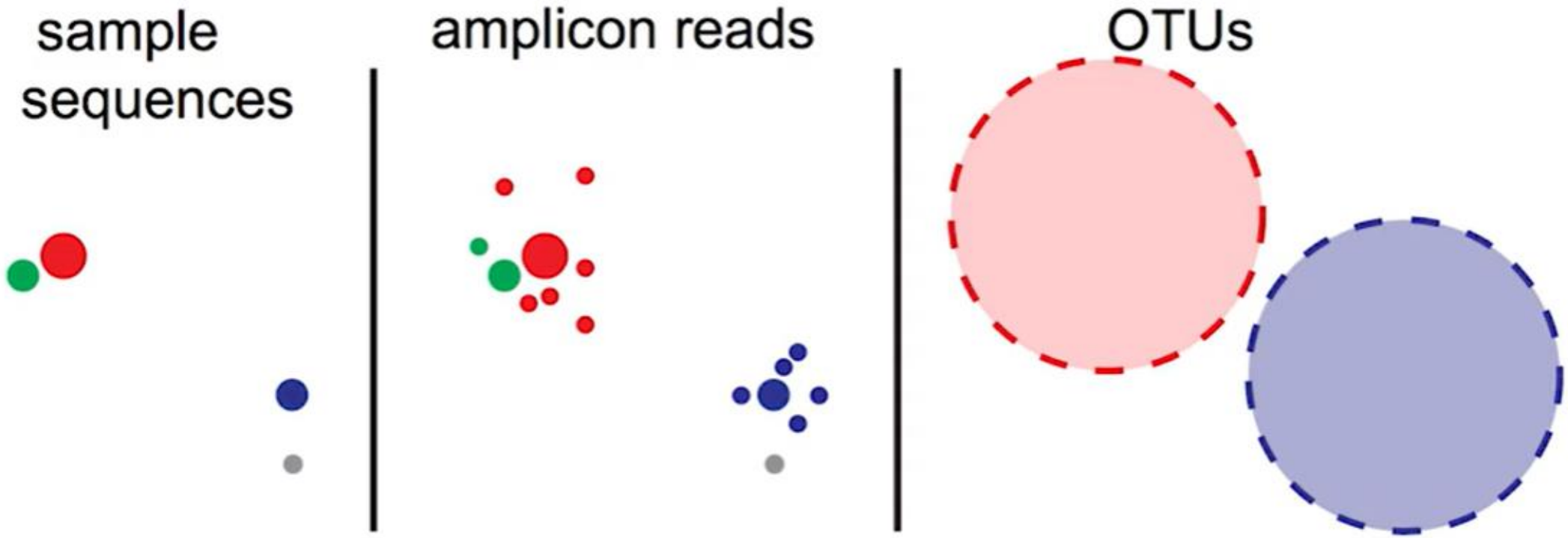
https://astrobiomike.github.io/amplicon/

https://benjjneb.github.io/dada2/tutorial.html

Youtube: Bioinformatics Virtual Coordination Network

OTUs or ASVs?

Norwegian University of Life Sciences



Bioinformatic pipelines

Norwegian University of Life Sciences

Amplicon Sequencing. Exactly. Version 1.14

Mothur

Comparative Study > PLoS One. 2020 Jan 16;15(1):e0227434. doi: 10.1371/journal.pone.0227434. eCollection 2020.

Comparing bioinformatic pipelines for microbial 16S rRNA amplicon sequencing

Andrei Prodan [1], Valentina Tremaroli [2], Harald Brolin [2], Aeilko H Zwinderman [3], Max Nieuwdorp [1], Evgeni Levin [1 4]

Affiliations + expand

PMID: 31945086 PMCID: PMC6964864 DOI: 10.1371/journal.pone.0227434

FULL TEXT LINKS

OPEN ACCESS TO FULL TEXT PLOS ONE

FREE Full text PMC

ACTIONS

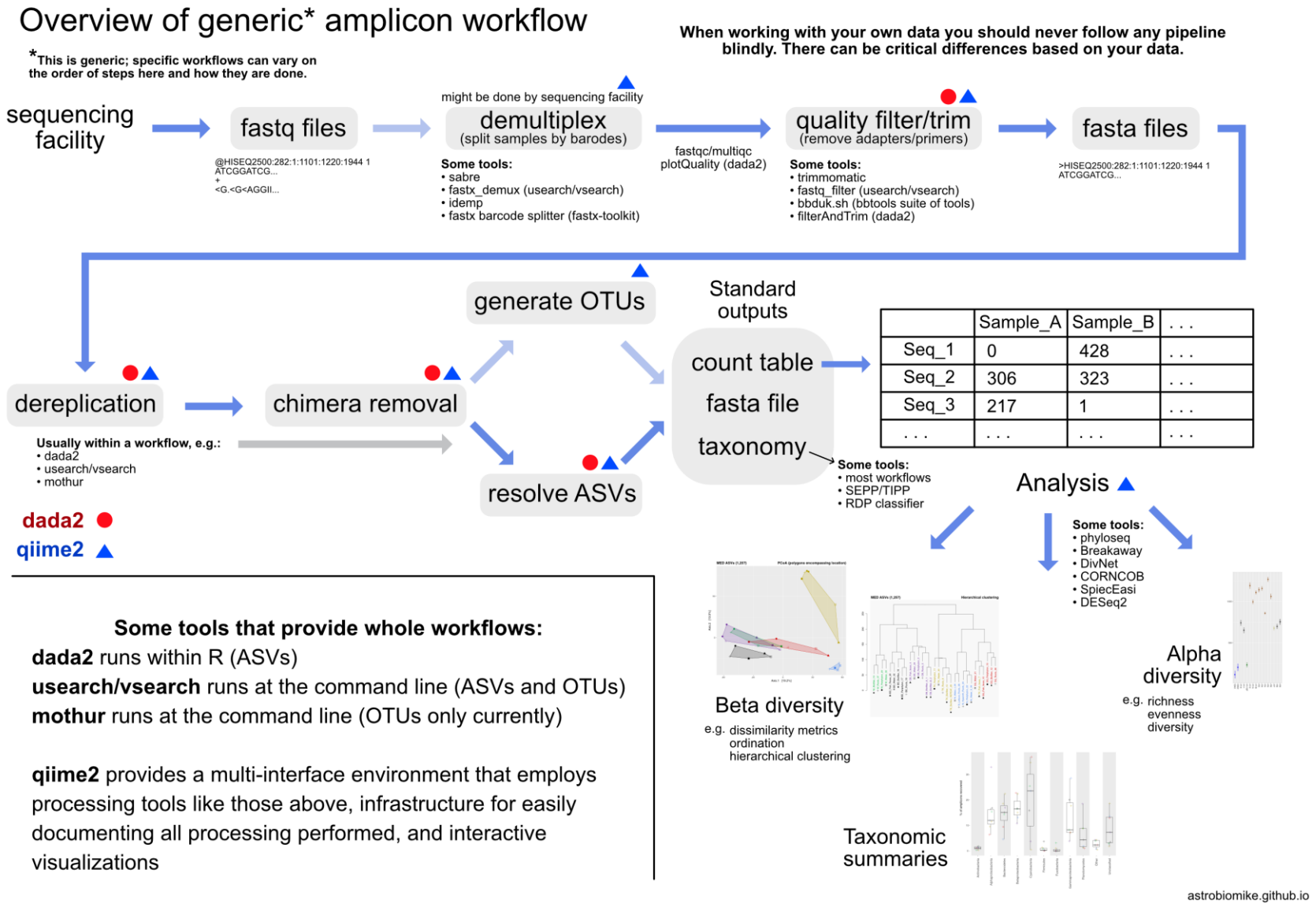
Cite

Collections

but also usearch, vsearch, Minimum Entropy Decomposition, UNOISE …

Amplicon workflow

Norwegian University of Life Sciences



nature methods

Explore content | About the journal | Publish with us | Subscribe

nature > nature methods > brief communications > article

Brief Communication | Published: 23 May 2016

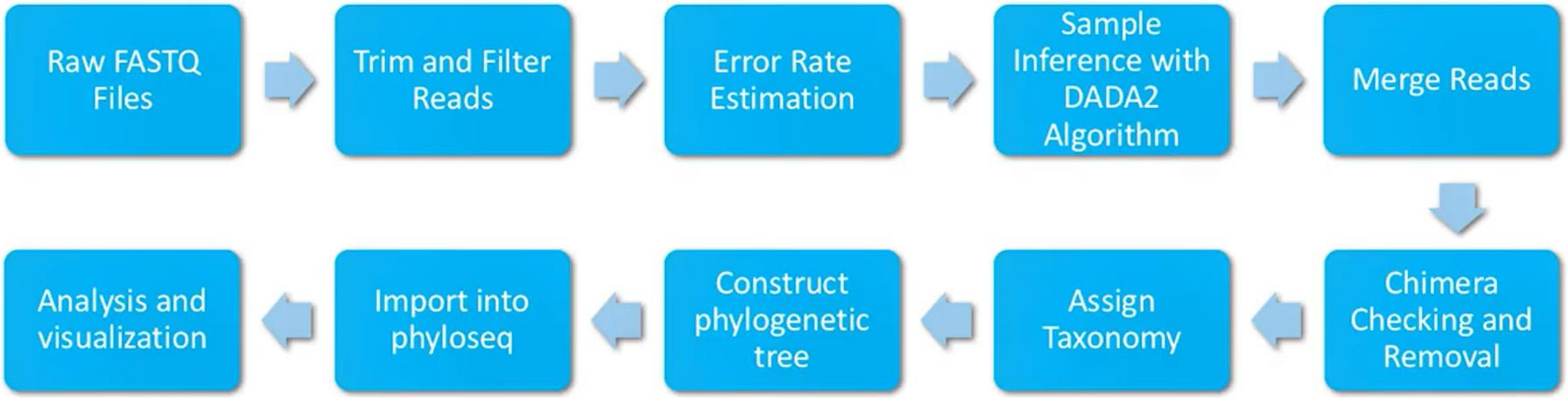
DADA2: High-resolution sample inference from Illumina amplicon data

Benjamin J Callahan, Paul J McMurdie, Michael J Rosen, Andrew W Han, Amy Jo A Johnson & Susan P Holmes

Nature Methods **13**, 581–583 (2016) | Cite this article

102k Accesses | **14k** Citations | **114** Altmetric | Metrics

DADA2 Workflow



https://benjjneb.github.io/dada2/tutorial.html

«FON pipeline»

Norwegian University of Life Sciences

1. Sequencing denoising
2. Pre-processing
3. Taxonomic Analysis
4. alpha-diversity
5. beta-diversity
6. Individual taxa comparisson
7. LEfSe
8. Metabolic pathways

1. Sequencing denoising

Norwegian University of Life Sciences

Instal some basic packages

```
## Installation and loading of dada2

```{r}
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("dada2")

library("Rcpp")

library("dada2")

library("ggplot2")
```
```

1. Sequencing denoising

Norwegian University of Life Sciences

Listing all .fastq files as FWD and REV

```
## Set the path to the raw fastq data

```{r dataset}
path <- "/net/fs-2/scale/OrionStore/Scratch/sergroch/FON48/" # change to the directory containing the fastq files, which may remain gzipped

list.files(path) # To check if the fastq files in the directory

```

Create lists of forward and reverse filenames for the forward and reverse reads and a list of sample names

```{r FRList}
# Forward and reverse fastq filenames have format: SAMPLENAME_R1_001.fastq and SAMPLENAME_R2_001.fastq
LL.fnFs <- sort(list.files(path, pattern="_R1_001.fastq", full.names = TRUE))
LL.fnRs <- sort(list.files(path, pattern="_R2_001.fastq", full.names = TRUE))
# Extract sample names, assuming filenames have format: SAMPLENAME_XXX.fastq
LL.sample.names <- sapply(strsplit(basename(LL.fnFs), "_"), `[`, 1)
```
```

1. Sequencing denoising

Norwegian University of Life Sciences

Make quality scores plots

```
```{r ReadQualityProfilesForward}

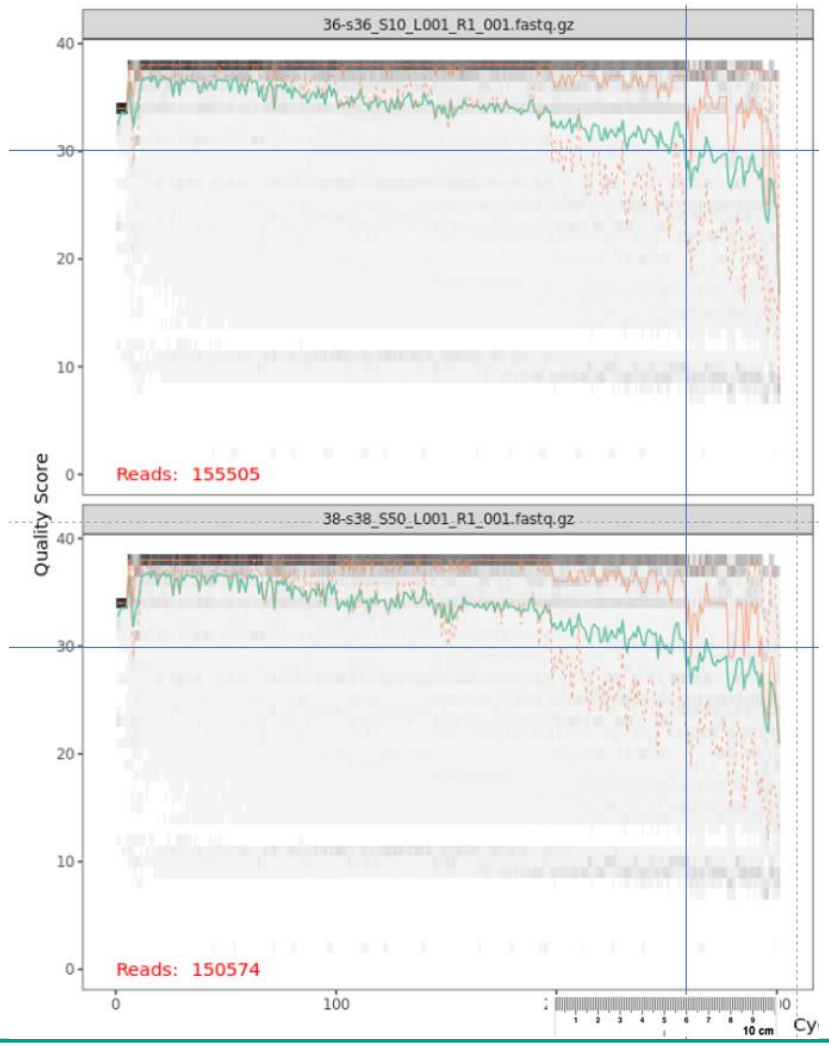
# This displays the forward and reverse quality scores for the samples 93 and 94
plotQualityProfile(c(LL.fnFs[93],LL.fnRs[93],LL.fnFs[94],LL.fnRs[94]))
```
```

To check if the foward/reverse reads from 4 samples show a consistent dropoff in quality toward the end of the reads.

```
```{r ReadQualityProfilesReverse}
plotQualityProfile(LL.fnFs[18:21]) #change to Rev
```
```

1. Sequencing denoising

Make quality scores plots



Green line: mean reads
Solid orange line: median
Dashed orange lines: interquartiles

The grey shading is a heatmap of the frequency of the quality scores at a given position along the read. Darker shading indicates higher frequency.

There are a few low quality scores that bring down the mean (

Oddly, there are some short sequences that are of low quality that bring down the mean at the beginning of the reads.

Note that the quality scores change along the length of the read as well as for the forward vs. the reverse reads.

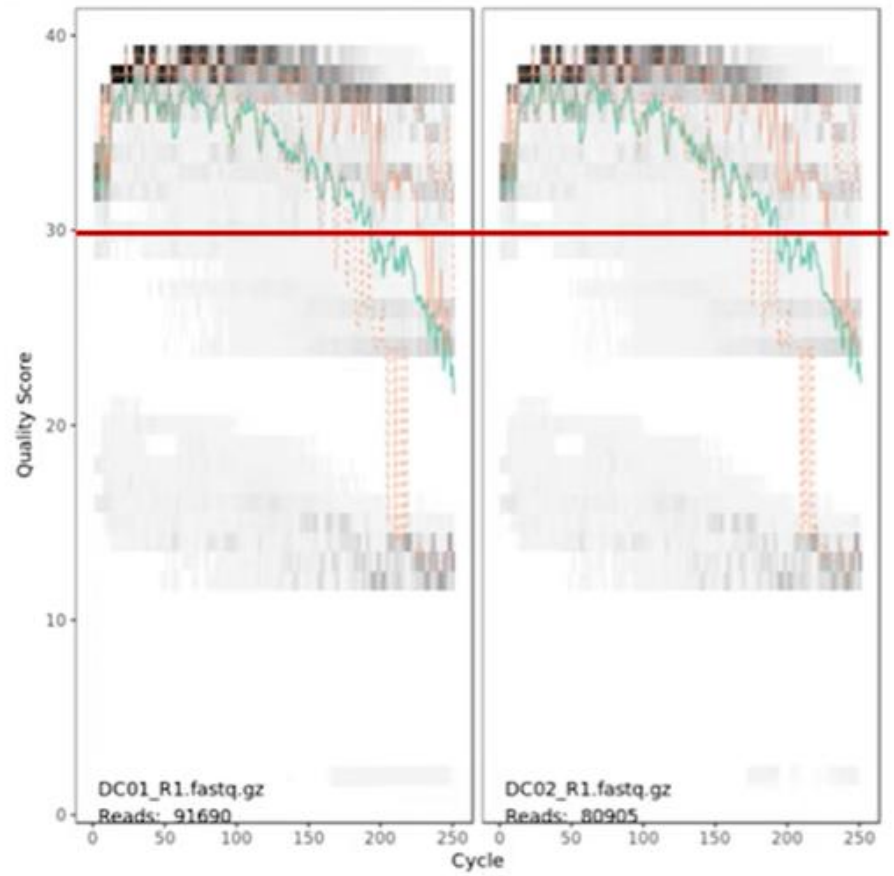
The reverse reads are normally lower quality than the forward reads.

1. Sequencing denoising

Norwegian University of Life Sciences

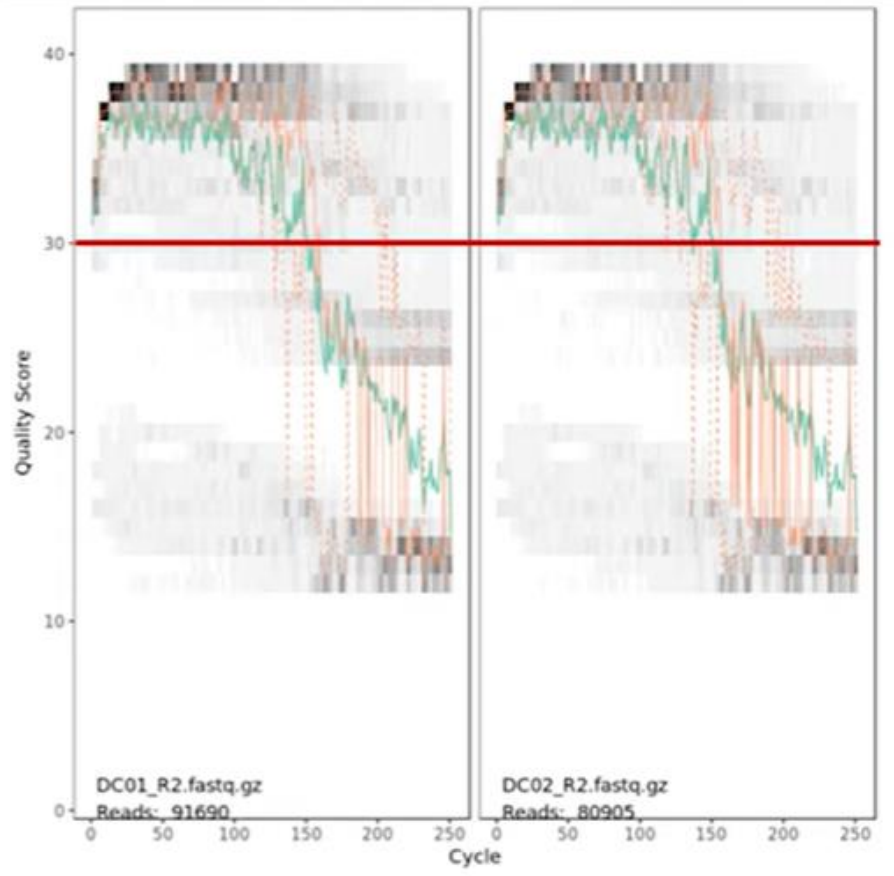
Make quality scores plots

```
#Investigate FWD read quality to determine where to trim
plotQualityProfile(fnFs[1:2])
```



FWD reads drop below Q30 near 200 bp

```
#Investigate Rev read quality to determine where to trim
plotQualityProfile(fnRs[1:2])
```

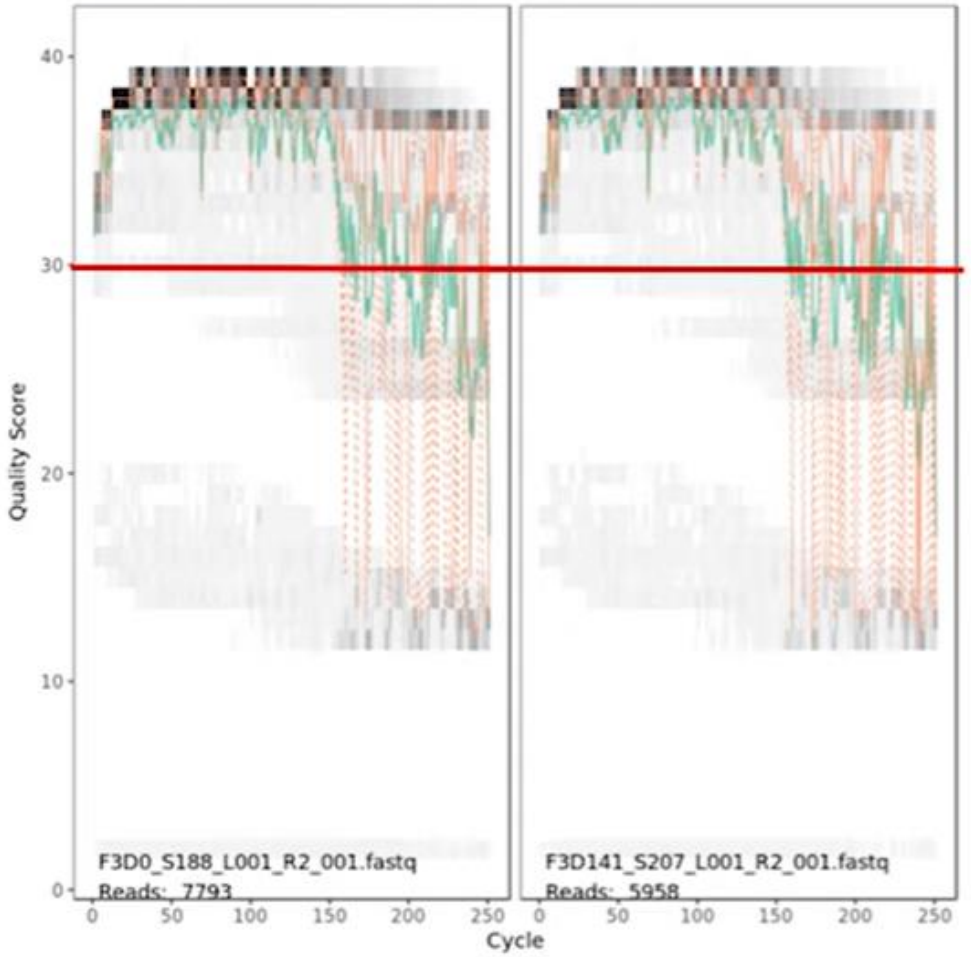


REV reads drop below Q30 near 140 bp

Bioinformatics

Norwegian University of Life Sciences

```
###Plot average quality scores of reverse reads
plotQualityProfile(fnRs[1:2])
```



- The reverse reads are a different story
- This is very common – reverse reads are almost always lower quality
- Overall the scores are pretty decent but they drop off right around 160 bp
- Since we'll have almost complete overlap, we can be aggressive with trimming
  - Cut at 160 bp

1. Sequencing denoising

Norwegian University of Life Sciences

Create filter file path and remane file

```
## Create a list of the filenames (including the path) for the filtered reads

This filename list will be later used by the filterAndTrim function from DADA2 to write the filtered reads into a new
directory called "filtered".

```{r FilteredFilenames}
# Place filtered files in filtered/ subdirectory
LL.filtFs <- file.path(path, "filtered", paste0(LL.sample.names, "_F_filt.fastq.gz"))
LL.filtRs <- file.path(path, "filtered", paste0(LL.sample.names, "_R_filt.fastq.gz"))
```
```

1. Sequencing denoising

Filter and Trim the reads

```
```{r FilterAndTrim}
library(dada2)
LL.FiltOut <- filterAndTrim(LL.fnFs, LL.filtFs, LL.fnRs, LL.filtRs, truncLen=c(275,215),
              maxN=0, maxEE=c(4,4), truncQ=2, rm.phix=TRUE, trimLeft = c(17,21),
              compress=TRUE, multithread=FALSE) # On Windows set multithread=FALSE
head(LL.FiltOut)
saveRDS(LL.FiltOut, "LL.FiltOut.rds")
```
```

The **filterAndTrim** function of DADA2 is the quality control step of the pipeline.

It takes as input the forward and reverse reads, and writes a reduced set of sequences in FASTQ format that excludes low quality sequences which do not meet the criteria specified by the function arguments, **truncLen=c(275,215)**.

The **maxEE** sets the maximum expected errors in a sequence to 4 for both the forward and reverse sequences, and **rm.phix** removes any phiX DNA sequences.

The **trimLeft** argument trims 17 bases corresponding to the length of the Pro341f primer from the beginning of the forward reads, and trims the 21 bases of Pro805r reverse primer from the beginning of the reverse reads.

1. Sequencing denoising

Norwegian University of Life Sciences

Error rates and ploting

```
## Learn the error rates for forward and reverse reads (30min)

This step applies machine learning to develop a model of the error rate for forward and reverse reads.

```{r ErrorTraining}
LL.errF <- learnErrors(LL.filtFs, multithread=FALSE)
LL.errR <- learnErrors(LL.filtRs, multithread=FALSE)

```

Plot the estimated error rates for the transition types

The nucleotide transition error rates will be unique to each run.

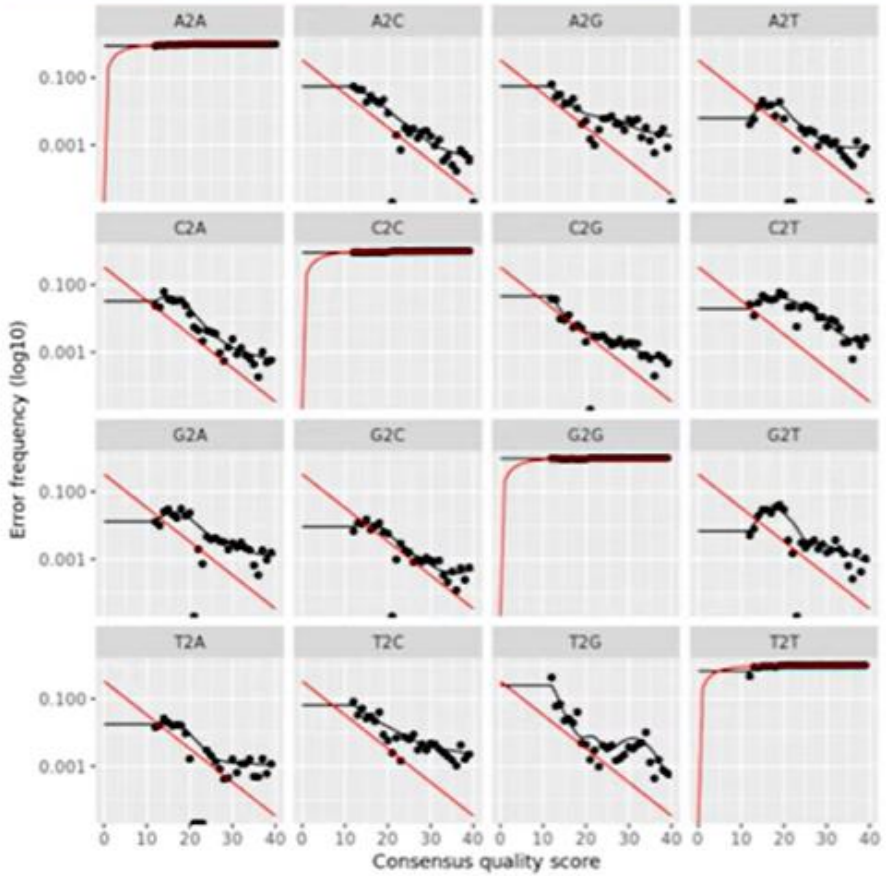
```{r PlotErrorRates}
plotErrors(LL.errF.rds, nominalQ=TRUE)
plotErrors(LL.errR.rds, nominalQ=TRUE)
```
```

1. Sequencing denoising

Error rates and ploting

```
###Plot error rates for all possible base transitions
plotErrors(errF, nominalQ=TRUE)
```

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



- Plots frequency of each possible base transition as a function of quality score
- Black line – observed error rates
- Red line - expected error rate under the nominal definition of the Q-value
- In general, frequency of errors decrease as quality score increases
- These look as expected so we can proceed with data analysis

1. Sequencing denoising

Error rates and ploting

```
###Estimate the error model for DADA2 algorithim using reverse reads
errR <- learnErrors(filtRs, multithread=TRUE)
```

```
Initializing error rates to maximum possible estimate.
Sample 1 - 7113 reads in 1660 unique sequences.
Sample 2 - 5463 reads in 1335 unique sequences.
Sample 3 - 2914 reads in 853 unique sequences.
Sample 4 - 2941 reads in 880 unique sequences.
Sample 5 - 4312 reads in 1286 unique sequences.
Sample 6 - 6741 reads in 1803 unique sequences.
Sample 7 - 4560 reads in 1265 unique sequences.
Sample 8 - 15637 reads in 3414 unique sequences.
Sample 9 - 11413 reads in 2522 unique sequences.
Sample 10 - 12017 reads in 2771 unique sequences.
Sample 11 - 5032 reads in 1415 unique sequences.
Sample 12 - 5299 reads in 1349 unique sequences.
Sample 13 - 18075 reads in 3290 unique sequences.
Sample 14 - 6250 reads in 1390 unique sequences.
Sample 15 - 4052 reads in 1134 unique sequences.
Sample 16 - 7369 reads in 1635 unique sequences.
Sample 17 - 4765 reads in 1084 unique sequences.
Sample 18 - 4871 reads in 1161 unique sequences.
Sample 19 - 6504 reads in 1502 unique sequences.
   selfConsist step 2
   selfConsist step 3
   selfConsist step 4
   selfConsist step 5
   selfConsist step 6

Convergence after  6  rounds.
Total reads used:  135328
```

Notice that error rate estimation takes longer for reverse reads

- This command creates an error model that will be used by the DADA2 algorithm
- Every batch of sequencing will have a different error rate
- Algorithm starts with the assumption that the error rates are the maximum possible
- Alternates error rate estimation and sample composition inference until they converge at a consistent solution

1. Sequencing denoising

Norwegian University of Life Sciences

Dereplication: remove identical sequences

```
## Dereplication

There will be many sequences which are 100% identical. By identifying the sequences which are identical, a process called "dereplication", the size of the dataset may be reduced.
```

```{r Dereplication}

# Dereplicate the forward and then the reverse reads

LL.derepFs <- derepFastq(LL.filtFs, verbose=TRUE)
saveRDS(LL.derepFs, "LL.derepFs.rds")
LL.derepRs <- derepFastq(LL.filtRs, verbose=TRUE)

saveRDS(LL.derepRs, "LL.derepRs.rds")


# Name the derep-class objects by the sample names
names(LL.derepFs) <- LL.sample.names
names(LL.derepRs) <- LL.sample.names
```

1. Sequencing denoising

Norwegian University of Life Sciences

Sample interface *(very long process)* + merging

```
## Sample inference (6h)

At this step we run the principle function of the DADA2 pipeline. The dada function uses the error model along with the set of dereplicated sequences in order to identify which sequences are likely to be real biological sequences and which are likely the result of base-calling errors. Those sequences that are base-calling errors are clustered with the real biological sequence from which they are likely to have derived.

```{r SampleInference}

LL.dadaFs <- dada(LL.derepFs, err=LL.errF.rds, multithread=FALSE)
saveRDS(LL.dadaFs, "LL.dadaFs.rds")

LL.dadaRs <- dada(LL.derepRs, err=LL.errR, multithread=FALSE)
saveRDS(LL.dadaRs, "LL.dadaRs.rds")

LL.dadaFs[[1]]
```

Merge paired reads (10min)

```{r MergeReads}
LL.mergers <- mergePairs(LL.dadaFs, LL.derepFs, LL.dadaRs, LL.derepRs, verbose=TRUE)
# Inspect the merger data.frame from the first sample
head(LL.mergers[[79]])
```
```

1. Sequencing denoising

Norwegian University of Life Sciences

Sample interface *(very long process)* + merging

```
###Merge denoised reads
mergers <- mergePairs(dadaFs, derepFs, dadaRs, derepRs, verbose=TRUE)
# Inspect the merger data.frame from the first sample
head(mergers[[1]])
```

```
6594 paired-reads (in 104 unique pairings) successfully merged out of 7113 (in 254 pairings) input.
5047 paired-reads (in 78 unique pairings) successfully merged out of 5463 (in 199 pairings) input.
2663 paired-reads (in 52 unique pairings) successfully merged out of 2914 (in 142 pairings) input.
2574 paired-reads (in 54 unique pairings) successfully merged out of 2941 (in 163 pairings) input.
3668 paired-reads (in 53 unique pairings) successfully merged out of 4312 (in 203 pairings) input.
```

- Merges paired end reads only if they exactly overlap
  - This is because both forward and reverse reads have been denoised and should be error-free
  - Can be changed by adding maxMismatch option
- By default, the program requires 20 nt of overlap but you can lower it if need be with minOverlap

1. Sequencing denoising

Amplicon Sequence Variant table

Norwegian University of Life Sciences

Make the Amplicon Sequence Variant (ASV) table

DADA2 produces Amplicon Sequence Variants (ASVs). Hence, here we produce an ASV table.

```{r}
LL.seqtab <- makeSequenceTable(LL.mergers)

# This produces the dimensions of the table, with the rows as samples and columns as ASVs.
dim(LL.seqtab)

# Inspect distribution of sequence lengths
table(nchar(getSequences(LL.seqtab)))

# How are the rows named?
rownames(LL.seqtab)[1]

# How are the columns named?
colnames(LL.seqtab)[1]
```

1. Sequencing denoising

Norwegian University of Life Sciences

Remove quimeras

Remove chimeras from the ASV table (1h)

Chimeras occur when primer extentions truncated during a PCR cycle anneal to mismatched 16S template from another taxon and prime the extension of the truncated fragment to create a chimeric sequence consisting of partial 16S sequence from 1 taxon attached to 16S from a different taxon. These chimeric sequences thus do not represent real biological sequences and must be removed.

```{r RemoveChimeras}
LLSeqtab.nochim <- removeBimeraDenovo(LL.seqtab, method="consensus", multithread=FALSE, verbose=TRUE)

# This lists the dimensions of the ASV table following chimera removal. It is the number of samples X number of ASVs.
dim(LLSeqtab.nochim)

# The sum function sums the sequence counts from each combination of ASV and sample. Thus, it gives the number of sequences in each table.
sum(LLSeqtab.nochim)/sum(LL.seqtab)

saveRDS(LLSeqtab.nochim, "LLSeqtab.nochim.rds")
```

1. Sequencing denoising

Norwegian University of Life Sciences

Notes for Chimera Checking

- It is normal for a large majority of unique sequences to be flagged as chimeric
  - They should make up a small proportion of total reads
- If a majority of total reads are chimeric, you have a problem
  - Most likely explanation – primers were not completely removed from reads
    - Using primers with ambiguous bases can cause reads to be flagged as chimeras
    - Re-run filterAndTrim() command using the trimLeft argument
  - If that doesn't fix it, there may be other factors at play
    - Try trimming more low quality bases
    - Think about which hypervariable region you're sequencing (i.e. V4 vs V4V5)
    - How complex is your community?

1. Sequencing denoising

Norwegian University of Life Sciences

Assigning Taxonomy

```
## Assign taxonomy (2.5h)

This step determines the taxonomy of each ASV. Different reference databases may be used, and there are several formatted for use with DADA2 and available at <https://benjjneb.github.io/dada2/training.html>[link](https://benjjneb.github.io/dada2/training.html). Here we use the latest version of the Silva database. Note that there are entries in Silva which do not conform to 7 hierarchy levels. Also note that identifical down to the species level requires a separate reference file and algorithm.

```{r AddTaxonomy}

#silva_nr99_v138.1_train_set.fa.gz
LL.taxa <- assignTaxonomy(LLSeqtab.nochim, paste0("/net/fs-2/scale/OrionStore/Home/sergroch/16S material/silva_nr99_v138.1_wSpecies_train_set.fa.gz"), multithread=FALSE)

# A separate approach is needed for assigning species designations #silva_species_assignment_v138.1.fa.gz
LL.taxa <- addSpecies(LL.taxa, paste0("/net/fs-2/scale/OrionStore/Home/sergroch/16S material/silva_species_assignment_v138.1.fa.gz"))

# Inspect the taxa names
taxa.print <- LL.taxa # Removing sequence rownames for display only
rownames(taxa.print) <- NULL
head(taxa.print)

saveRDS(LLSeqtab.nochim, "Seqtab.nochim.rds")
#Seqtab.nochim <- readRDS("SergioSeqtab.nochim.rds")
saveRDS(LL.taxa, "LL.taxa.rds")
#taxa <- readRDS("LL.taxa.rds")

```

silva
high quality ribosomal RNA databases
```

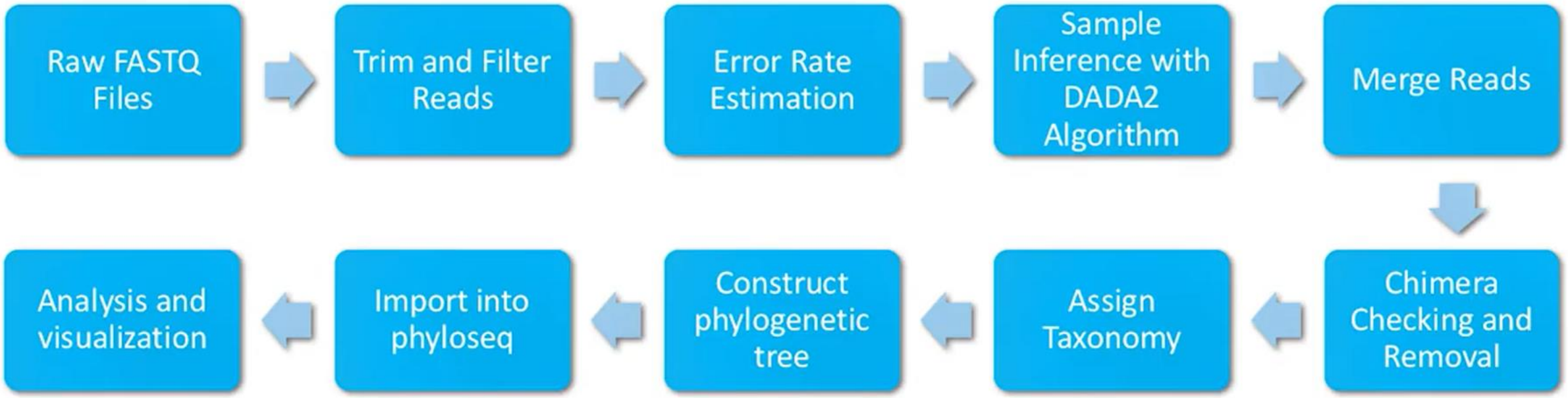
1. Sequencing denoising

Norwegian University of Life Sciences

Taxonomy table

	A	B	C	D	E	F	G	H	J	K	L	M	N	O	P
1	Row.names	Kingdom	Phylum	Class	Order	Family	Genus	Species	S1	S10	S11	S12	S13	S14	S15
2	TAGGGAATC	Bacteria	Firmicutes	Bacilli	Lactobacillales				0	3224	2715	4204	39484	33014	21429
3	TGGGGAATA	Bacteria	Actinobacter	Actinobacter	Micrococcale	Beutenbergi	Salana		0	347	0	400	3900	2976	3608
4	TGGGGAATA	Bacteria	Proteobacte	Gammaprote	Xanthomona	Xanthomona	Silanimonas		0	0	0	0	0	0	0
5	TAGGGAATC	Bacteria	Firmicutes	Bacilli	Lactobacillal	Carnobacteri	Granulicatell	elegans	0	0	0	0	32	0	10
6	TGGGGAATA	Bacteria	Proteobacte	Alphaproteo	Sphingomon	Sphingomon	Sphingomonas		0	0	0	0	0	0	0
7	TGGGGAATT	Bacteria	Proteobacte	Gammaprote	Burkholderia	Comamonad	Comamonas		0	0	0	0	0	0	0
8	TGGGGAATT	Bacteria	Proteobacte	Gammaprote	Burkholderia	Comamonad	Acidovorax	facilis	0	0	0	0	0	0	0
9	TGAGGAATA	Bacteria	Bacteroidota	Bacteroidia	Flavobacteri	Flavobacteri	Myroides	injenensis	0	0	0	0	0	33	0
10	TGGGGAATA	Bacteria	Firmicutes	Clostridia	Clostridiales	Clostridiacea	Hathewaya		0	0	0	0	0	0	0
11	TGAGGAATA	Bacteria	Bacteroidota	Bacteroidia	Bacteroidale	Paludibacteraceae			0	0	0	0	0	0	0
12	TGGGGAATA	Bacteria	Actinobacter	Actinobacter	Micrococcale	Micrococcace	Glutamicibac	creatinolytic	0	0	0	0	0	0	0

DADA2 Workflow



https://benjjneb.github.io/dada2/tutorial.html

2. Pre-processing

Packages needed

```
## Load the required package. Install ggstatsplot before running the script. If you are working with orion, this should be done with R.4.3.1 as this was recently updated for installing the package
```{r loading package, echo=FALSE}
library(Rcpp)
library(dada2)
library(phyloseq)
library(permute)
library(lattice)
library(vegan)
library(ggplot2)
library(tidyverse)
library(ggstatsplot)
library(dplyr)
library(openxlsx)
```
```

2. Pre-processing

Create a phyloseq object

A phyloseq object consists of the 5 main data types needed for complete microbiome analysis.

- An ASV table like the one produced
- The sample metadata table containing, for example, fish number, tank, diet, [DNA],…..
- A reference nucleotide sequence for each ASV
- A phylogenetic tree
- A taxonomy table with the levels of the taxonomic hierarchy for every ASV

2. Pre-processing

Norwegian University of Life Sciences

Create a phyloseq object

````
##Create a phyloseq object
```{r MakePSObject}
# Create the phyloseq object. Note that if there is an "undetermined" file and the sample metadata table doesn't have a sample named "undetermined", then it gets lost from the phyloseq object.

ps <- phyloseq(otu_table(count_tab, taxa_are_rows=FALSE),
               sample_data(LL.samdf),
               tax_table(tax_tab))
```

##Remove the undetermined sample & other if necessary
```{r}
ps <- prune_samples(sample_names(ps) != "Undetermined_F_filt.fastq.gz", ps) # Remove undetermined sample

```


##Transform the phyloseq object into proportion
```{r}
ps_tss <- transform_sample_counts(ps, function(x){x / sum(x)})
```
````

2. Pre-processing

Create a phyloseq object

```
# Taxonomy-based filtering
Remove features without a phylum-level annotation and those assigned as chloroplast or mitochondria. Note that the taxonomic labels are database specific and may change in different versions of the same database. Make sure you're using the correct taxonomic labels to remove chloroplast and mitochondria.
```{r}
ps_tss <- subset_taxa(ps_tss, !is.na(Phylum) & !Phylum %in% c("", "unassigned")) %>%
  subset_taxa(Order != "Chloroplast"|is.na(Order)) %>%
  subset_taxa(Family != "Mitochondria"|is.na(Family))
```

Prevalence-based filtering

Features that show up in only one or a few samples may not represent real biological diversity but rather PCR/sequencing errors (such as PCR chimeras) or reagent contaminants.
```{r}
ps_tss <- subset_samples(ps_tss, !Sample_kind %in% c("negative_control")) %>%
  # remove features present in only one sample
  filter_taxa(., function(x) sum(x > 0) > 1, TRUE) %>%
  taxa_names() %>%
  prune_taxa(ps_tss)
```
```

```
phyloseq-class experiment-level object
otu_table()   OTU Table:         [ 1617 taxa and 96 samples ]
sample_data() Sample Data:       [ 96 samples by 8 sample variables ]
tax_table()   Taxonomy Table:    [ 1617 taxa by 8 taxonomic ranks ]
```

2. Pre-processing

Filter contaminants

```
# Filter contaminants
## Screening of reagent contaminants
The screening of reagent contaminants will be based on two typical characteristics of contaminanting sequences as outlined in the paper [Simple statistical identification and removal of contaminant sequences in marker-gene and metagenomics data](https://microbiomejournal.biomedcentral.com/articles/10.1186/s40168-018-0605-2): they are likely to have frequencies that inversely correlate with sample DNA concentration and are likely to have higher prevalence in control samples than in true samples. The authors developed an R package, [*decontam*](https://github.com/benjjneb/decontam), for removing contaminating sequences in the marker-gene and shotgun metagenomics data. The package, however, does not make use of positive controls for the identification of contaminating sequences. As removing of features may critically affect downstream analyses, we'll do it by manual screening based on the aforementioned principles.

## Identify reagent contaminants
### Data wrangling

Make a dataframe containing features present in the negative controls and mock samples.
```{r}
decontam <- ps_tss %>%
  # the following 4 lines remove features not present in the control samples
  subset_samples(Diet %in% c("FP","Mock", "NC")) %>%
  filter_taxa(., function(x) sum(x > 0) > 0, TRUE) %>%
  taxa_names() %>%
  prune_taxa(ps_tss) %>%
  # convert the phyloseq object into a tidy style dataframe
```
```

2. Pre-processing

Filter contaminants (prevalence)

Norwegian University of Life Sciences

```
### Prevalence-based classification - this generates a graphic of pdf file. Here we use barplots to visualize the abundance and prevalence of the features found in the control samples After the production of the bar plots, manually inspect them to identify the contaminants in the negative control.
```{r, results='hide'}
library(cowplot)
# split the dataframe by feature ID
decontam_spl1 <- group_split(decontam, OTU)
# make barplots

pdf("prevalence_contam.pdf", width = 16, height = 10)

lapply(seq_along(decontam_spl1), function(x){
  # make a bar plot without mock
  p1 <- filter(decontam_spl1[[x]], Sample_type != "pos.control") %>%
    plot_prevalence(x = Diet, y = Abundance, bar_color = Sample_type,
                    xlab = "Sample name", ylab = "Relative abundance (%)",
                    title = unique(decontam_spl1[[x]][, "tax"]))

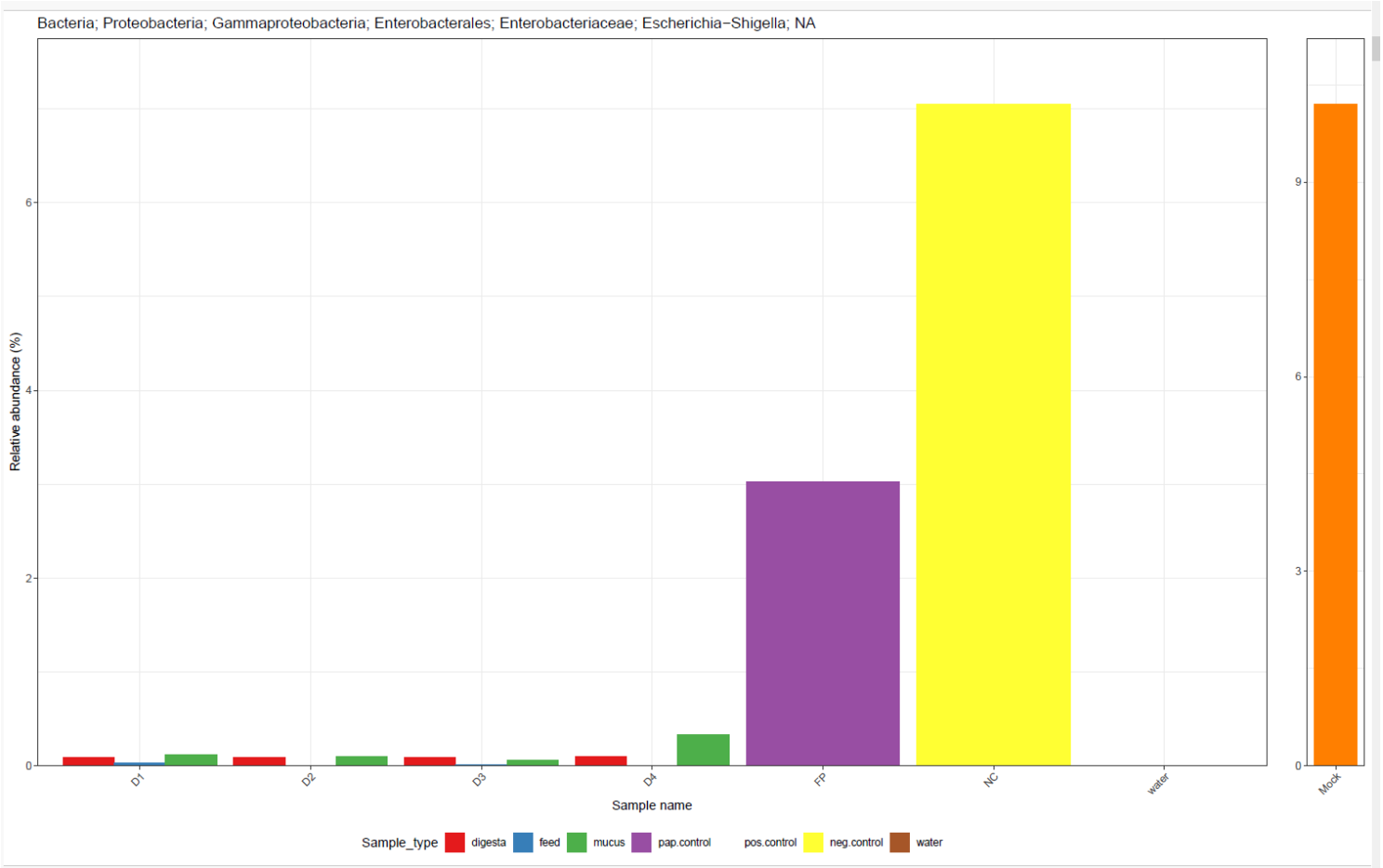
  # make a bar plot with mock only
  p2 <- filter(decontam_spl1[[x]], Sample_type == "pos.control") %>%
    plot_prevalence(x = Diet, y = Abundance, bar_color = Sample_type, xlab = "", ylab = "")

  # assemble plots
  plot_grid(p1, p2 + theme(legend.position = "none"), nrow = 1, align = 'h', axis = "bt", rel_widths = c(13, 1))
  }
)

dev.off()
```
```

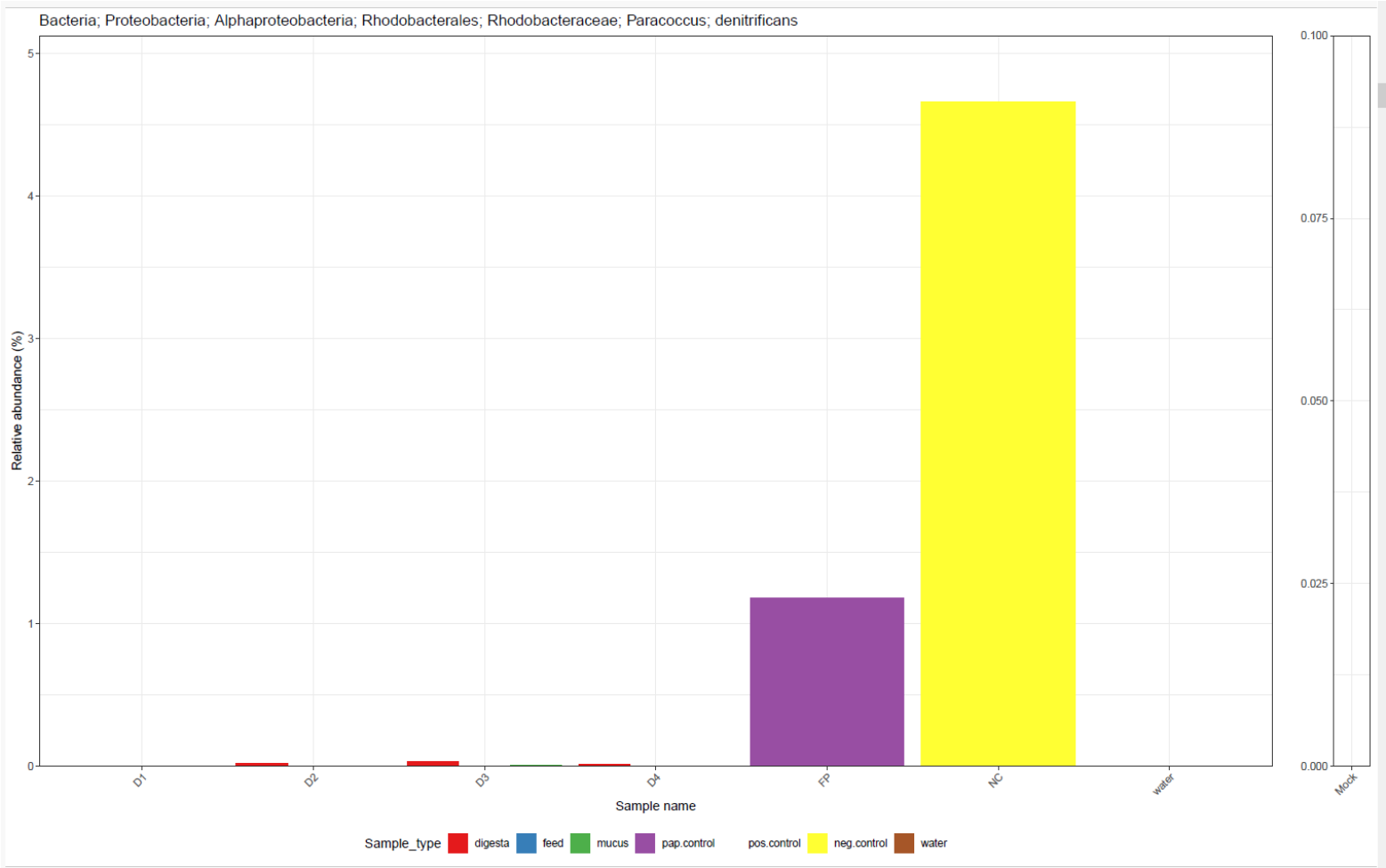
2. Pre-processing

Filtration of contaminants



2. Pre-processing

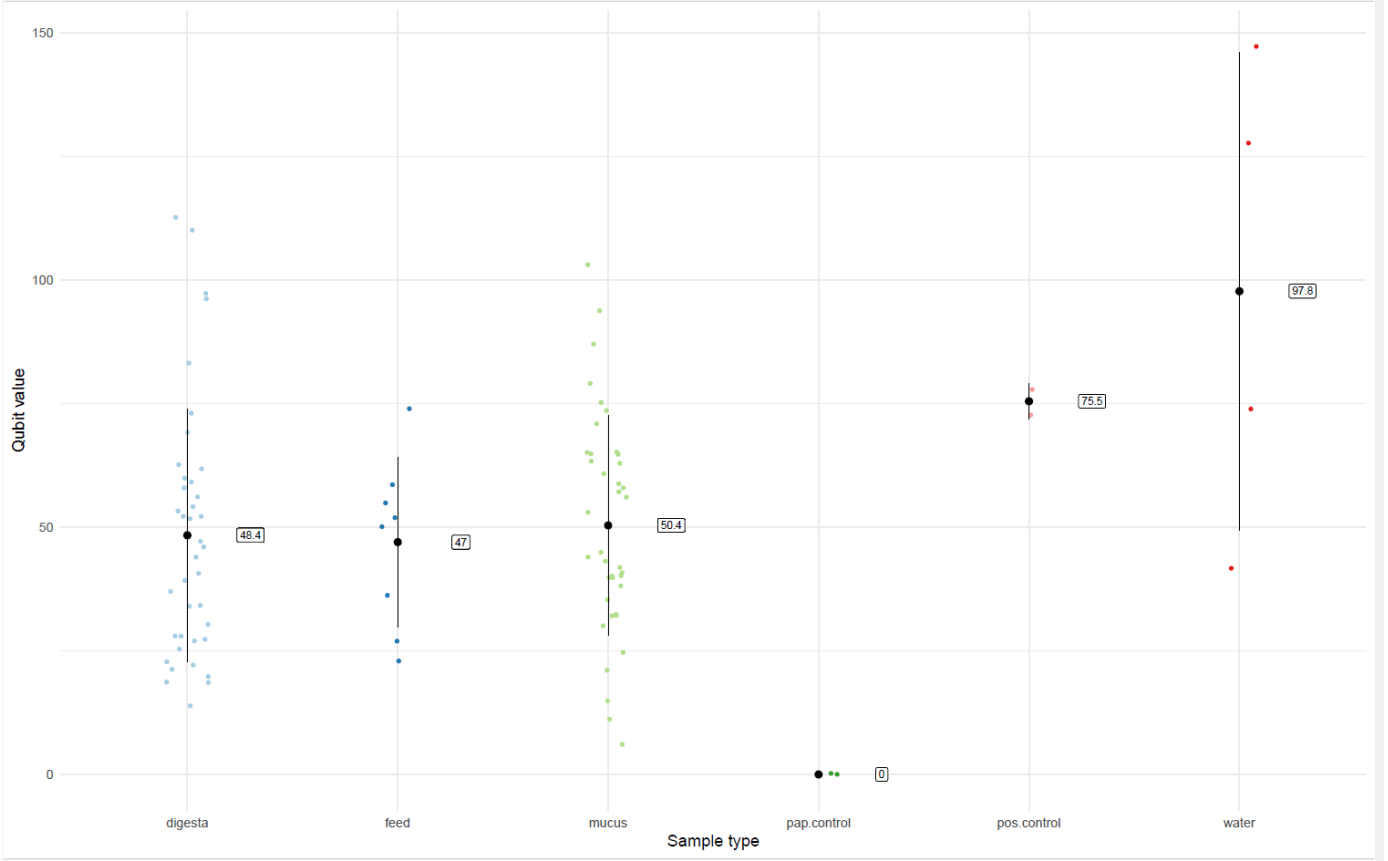
Filtration of contaminants



2. Pre-processing

Filter contaminants (DNA concentration)

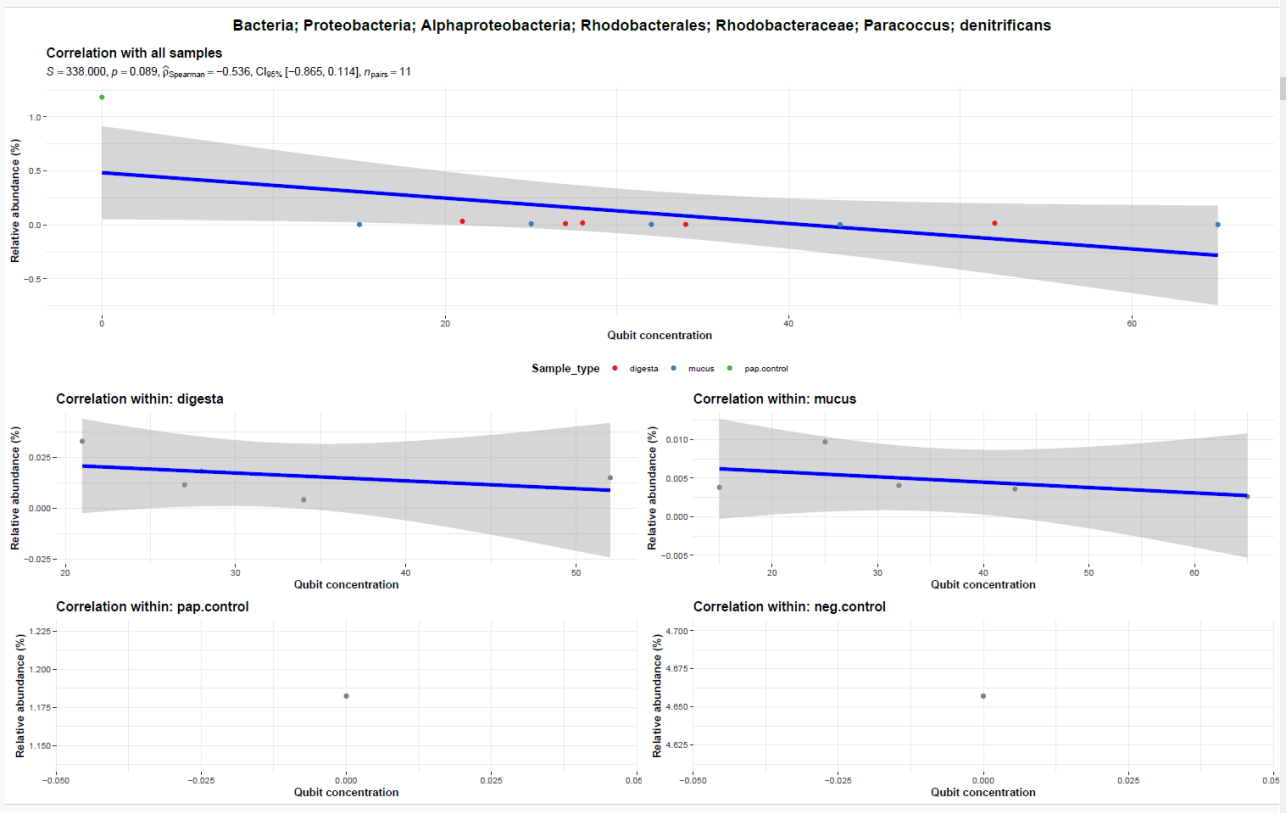
```
## Inspect bacterial DNA concentration
##Before we proceed with the identification of contaminating features, let's look at the DNA concentration as measured by
Qubit values of the DNA templates used for the amplicon PCR.
```



2. Pre-processing

Filter contaminants (frequency)

```
### Frequency-based classification
##Here we visualize correlations between the DNA concentration values and the relative abundance of features found in the
control samples. Features showing inverse correlations with DNA concentration are potential contaminating features
introduced during the amplicon PCR, which is the main source of reagent contamiantion in this study.
```



2. Pre-processing

Determine and remove contaminants

```
### Gather contaminating features
##After inspecting the feature prevalence barplots and the correlation plots, the following features are considered as
reagent contaminants:
```{r}
# gather contaminating features using their taxonomic labels
contam <- select(decontam, OTU, tax) %>%
  distinct() %>%
  filter(grepl("maltaromaticum|rhizophila|lipophiloflavum|Micrococcus|denitrificans|Burkholderia-Caballeronia-Paraburkholder
ia|acnes|cinnamea|Jeotgalicoccus|Parcubacteria|Aestuariimicrobium|sphingomonas|Enhydrobacter", tax))

```
```

2. Pre-processing

Determine and remove contaminants

Norwegian University of Life Sciences

```
##Check the distibution of contaminating features.
```{r, fig.width=16, fig.height=10}
prune_taxa(taxa_names(ps_tss) %in% contam_mock$OTU, ps_tss) %>%
  plot_bar(x = "Sample_type", fill = "Phylum", title = "Cross-contaminating features in the mock samples") +
  scale_y_continuous(expand = expansion(mult = c(0, 0.05))) +
  scale_fill_brewer(palette = "Paired") +
  theme_bw() +
  theme(legend.position = "bottom", axis.text.x = element_text(angle = 90, hjust = 1))
```

Remove reagent and cross-contaminants
```{r}
# remove reagent contaminants from all samples.
ps_nocontam <- prune_taxa(taxa_names(ps_tss_nocontam), ps)
# remove between-sample contaminants from the mock samples
ps_nocontam_mock <- subset_samples(ps_nocontam, Sample_type == "pos.control")
ps_nocontam_mock <- prune_taxa(!taxa_names(ps_nocontam_mock) %in% contam_mock$OTU, ps_nocontam_mock)
# merge phyloseq object
ps_nocontam <- subset_samples(ps_nocontam, Sample_type != "pos.control") %>%
  merge_phyloseq(ps_nocontam_mock)
```
```

2. Pre-processing

Positive control: Mock

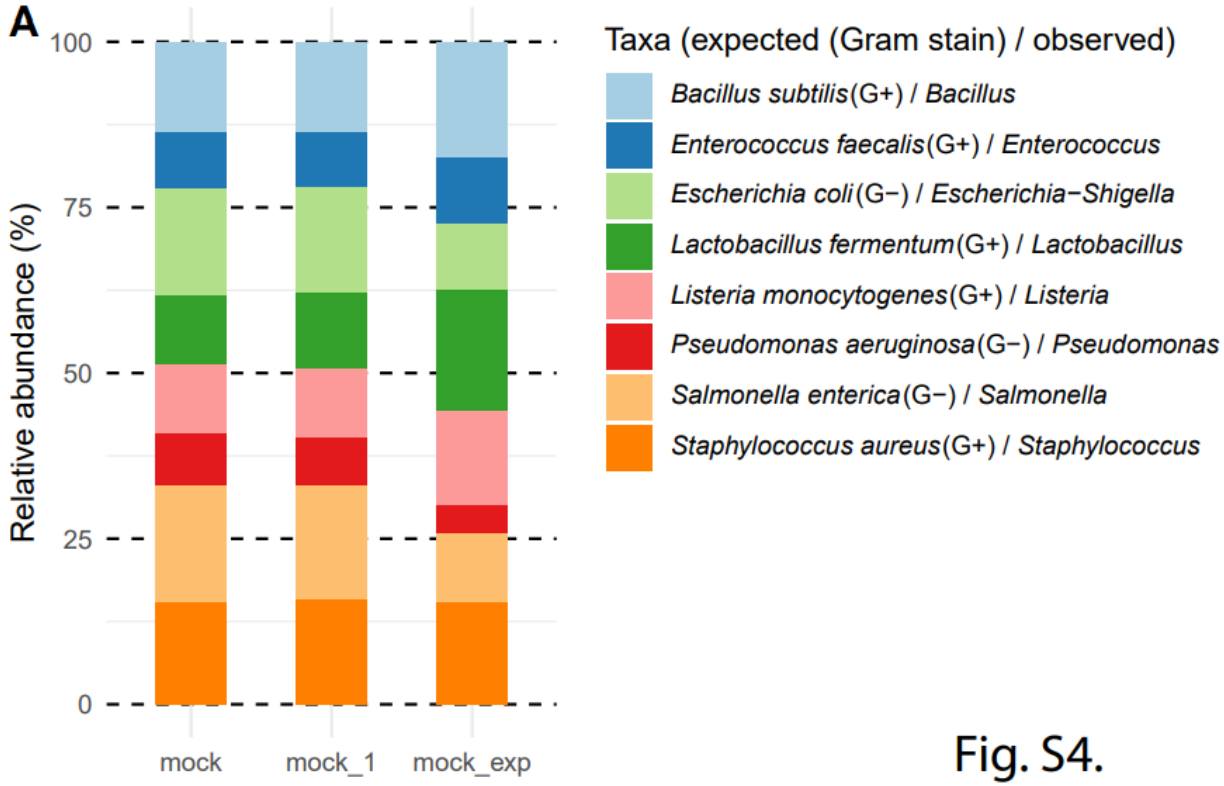


Fig. S4.

Agboola, J.O., Rocha, S.D.C., Mensah, D.D. *et al. et al.* (2023), Animal Microbiome, doi: 10.1186/10.1186/s42523-023-00242-y

3. Taxonomic Analysis

Norwegian University of Life Sciences

More packages to install

```
## Load the required package. Install ggstatsplot before running the script. If you are working with orion, this should be done with R.4.0.4 as this was recently updated for installing the package

```{r loading package, echo=FALSE}
library(Rcpp)
library(dada2)
library(phyloseq)
library(permute)
library(lattice)
library(vegan)
library(ggplot2)
library(tidyverse)
library(ggstatsplot)
library(dplyr)
library(microbiome)
library(microbiomeutilities)
library(knitr)
library(RColorBrewer)
library(DT)
library(gt)
library(cowplot)
library(PerformanceAnalytics)
library(venn)
library(philr)
library(MicrobeR)
library(devtools)
```
```

3. Taxonomic Analysis

Norwegian University of Life Sciences

Extract ps_nocontam and check coverage

````
##extract feature table, taxonomy and metadata from the phyloseq object. Check if your feature table (count_tab) are sample ID as rows. check with taxa_are_rows(phyloseq object). if the answer is FALSE, transpose it as done below. otherwise, don't need to do it.
```{r}
# Extract feature table, taxonomy and metadata from the phyloseq object
count_tab <- as.data.frame(t(otu_table(ps_nocontam))) #to transpose if the observation do not mach to run tab_phy
#count_tab <- as.data.frame((otu_table(ps_nocontam)))
tax_tab <- tax_table(ps_nocontam) %>% as("matrix") %>% as.data.frame()
metadata <- data.frame(sample_data(ps_nocontam), check.names = FALSE)
```

# Overview of taxonomy assignments
First of all, let's look at the coverage of taxonomy assignments at different levels.
```{r}
library(tidyr)
tax_tab %>%
  gather("Rank", "Name", rank_names(ps_nocontam)) %>%
  group_by(Rank) %>%
  # Empty taxonomic ranks may be na or strings containing "uncultured" or "Ambiguous_taxa"
  summarize(ASVs_classified = sum(!is.na(Name) & !grepl("uncultured|Ambiguous|metagenome", Name))) %>%
  mutate(Frac_classified = ASVs_classified / ntaxa(ps_nocontam),
         Frac_classified = ifelse(Frac_classified == 1, "100", round(Frac_classified * 100, 1)),
         Frac_classified = paste(Frac_classified, "%"),
         Rank = factor(Rank, rank_names(ps_nocontam))) %>%
  arrange(Rank) %>%
  datatable(options = list(columnDefs = list(list(className = 'dt-left', targets = c(0:3)))))

##We observed that the 81.1% of ASVs were assigned at the genus level whereas only 12.6% of ASVs got a species-level annotation.
```
````

3. Taxonomic Analysis

Norwegian University of Life Sciences

Extract ps_nocontam and check coverage

Show 10 entries Search:

	Rank	ASVs_classified	Frac_classified
1	Kingdom	1617	100 %
2	Phylum	1617	100 %
3	Class	1584	98 %
4	Order	1558	96.4 %
5	Family	1495	92.5 %
6	Genus	1311	81.1 %
7	Species	203	12.6 %

Showing 1 to 7 of 7 entries Previous 1 Next

3. Taxonomic Analysis

Norwegian University of Life Sciences

Group samples by type (phylum and genus)

```
row.names(count_tab)

openxlsx::write.xlsx(tab_phy, file = "phylum.xlsx", overwrite = TRUE)

##select only digesta sample BY REMOVING the mock, feed , the filter paper, negative control and water samples
##these are positions! by having a wrong sample name. GROUPS WERE CHECKED
#tab_phy1 <- tab_phy %>%
            #select(-c(49,50,51,52,53,54,55,56,57,58,59,60))

##Select ONLY the DIGESTA samples # removed sample 36, fish 10
tab_phy1d <- tab_phy %>%
            select(c(01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,33,34,35,
37,46,47,48)) # removed sample 36, fish 10

##Select ONLY the MUCUS samples
tab_phy1m <- tab_phy %>%
            select(c(38,39,40,41,42,44,45,49,50,51,52,53,54,55,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,
77,78,79,80)) # removed sample 43, fish 10

##Select ONLY the FEED samples
tab_phy2 <- tab_phy %>%
            select(c(81,82,83,84,85,86,87,88))

##Select ONLY WATER samples
tab_phy3 <- tab_phy %>%
            select(c(89,90,91,92))

openxlsx::write.xlsx(tab_phy1d, file = "phylum_digesta.xlsx",overwrite = TRUE)
openxlsx::write.xlsx(tab_phy1m, file = "phylum_mucus.xlsx",overwrite = TRUE)
openxlsx::write.xlsx(tab_phy2, file = "phylum_feed.xlsx",overwrite = TRUE)
openxlsx::write.xlsx(tab_phy3, file = "phylum_water.xlsx",overwrite = TRUE)


tab_phya <- tab_phy %>% rownames_to_column()
openxlsx::write.xlsx(tab_phya, file = "phylum_name.xlsx",overwrite = TRUE)
```

3. Taxonomic Analysis

Norwegian University of Life Sciences

```
##Make taxa barplot at phylum level - DIGESTA
```{r}

# Plot taxa on individual basis for digesta
metadata1d <- subset (metadata, Sample_type == "digesta")
metadata1d$Diet <- factor(metadata1d$Diet, levels =  c("D1", "D2", "D3", "D4"))

p_phy_ind <- make_taxa_barplot(table = tab_phy1d,
                               metadata = metadata1d,
                               group_by = factor(Diet, c("D1", "D2", "D3", "D4")),
                               ntaxa = 10,
                               nrow = 1,
                               plot_mean = FALSE,
                               cluster_sample = FALSE,
                               sample_label = fish,
                               italize_taxa_name = FALSE,
                               colors=colors)

p_phy_ind <- p_phy_ind + labs(x = "fish")

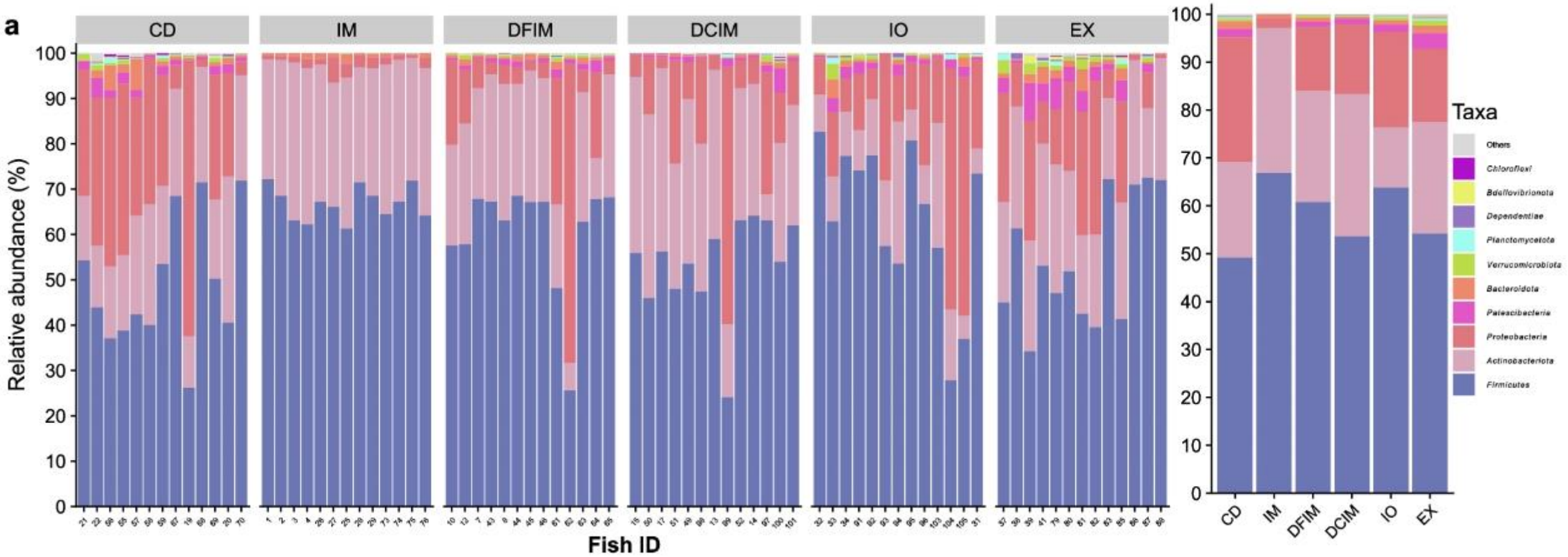
# Plot taxa using group means for digesta
p_phy_meand <- make_taxa_barplot(table = tab_phy1d,
                                 metadata = metadata1d,
                                 group_by = Diet,
                                 ntaxa = 10,
                                 nrow = 1,
                                 plot_mean = TRUE,
                                 cluster_sample = FALSE,
                                 #sample_label = diet,
                                 italize_taxa_name = TRUE,
                                 colors=colors)

p_phy_meand <- p_phy_meand + labs(x = "", y = "")
```
```

3. Taxonomic Analysis

Norwegian University of Life Sciences

Digesta - Phylum

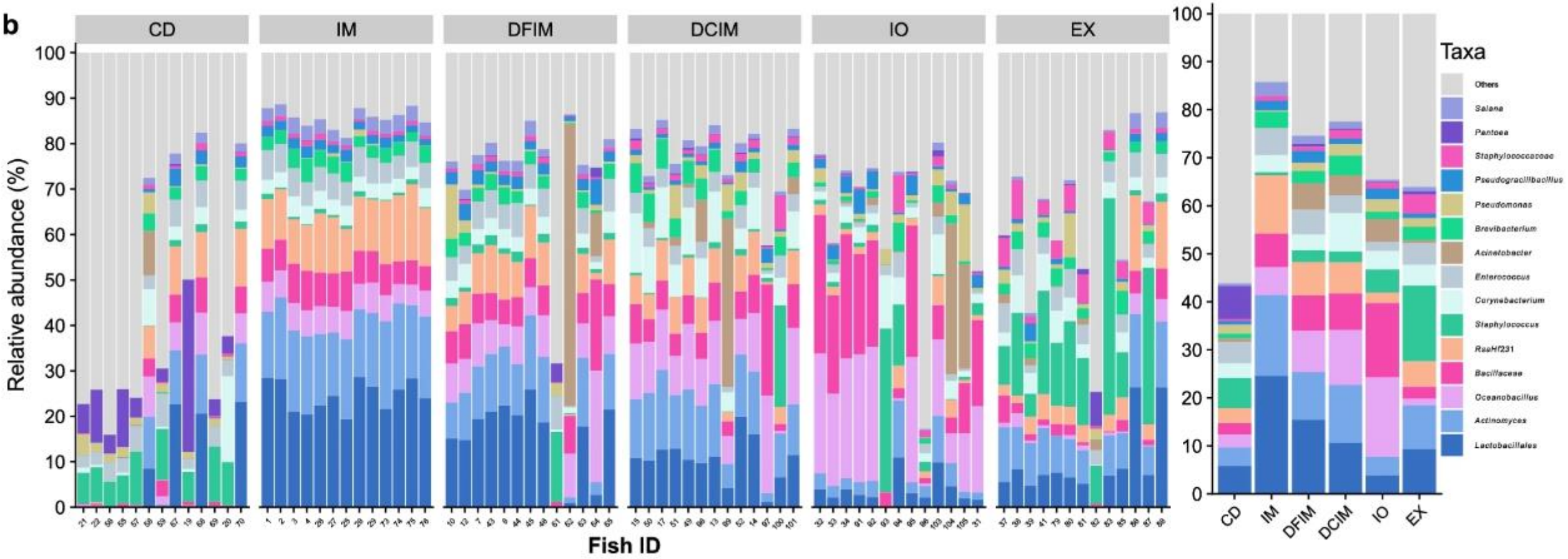


Weththasinghe, P., Rocha, S.D.C., Øyås, O. *et al.* (2022), Animal Microbiome, doi: 10.1186/s42523-021-00161-w

3. Taxonomic Analysis

Norwegian University of Life Sciences

Digesta - Genus

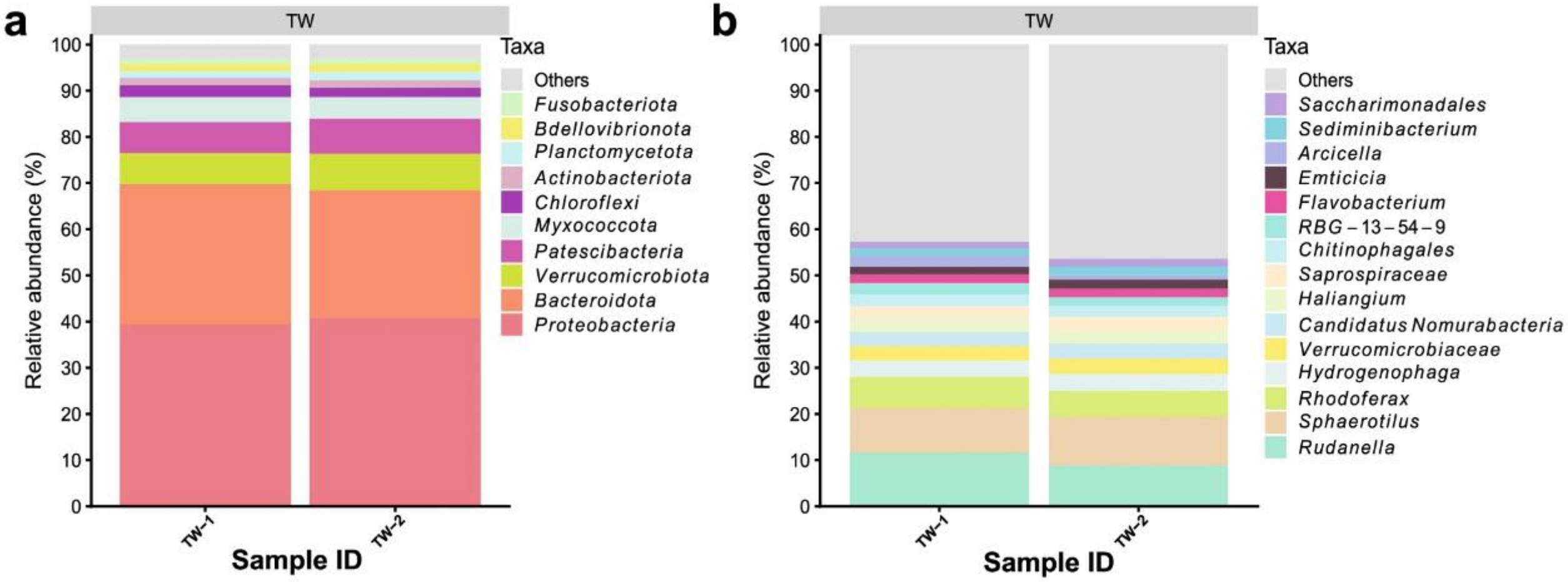


Weththasinghe, P., Rocha, S.D.C., Øyås, O. *et al.* (2022), Animal Microbiome, doi: 10.1186/s42523-021-00161-w

3. Taxonomic Analysis

Norwegian University of Life Sciences

Water samles

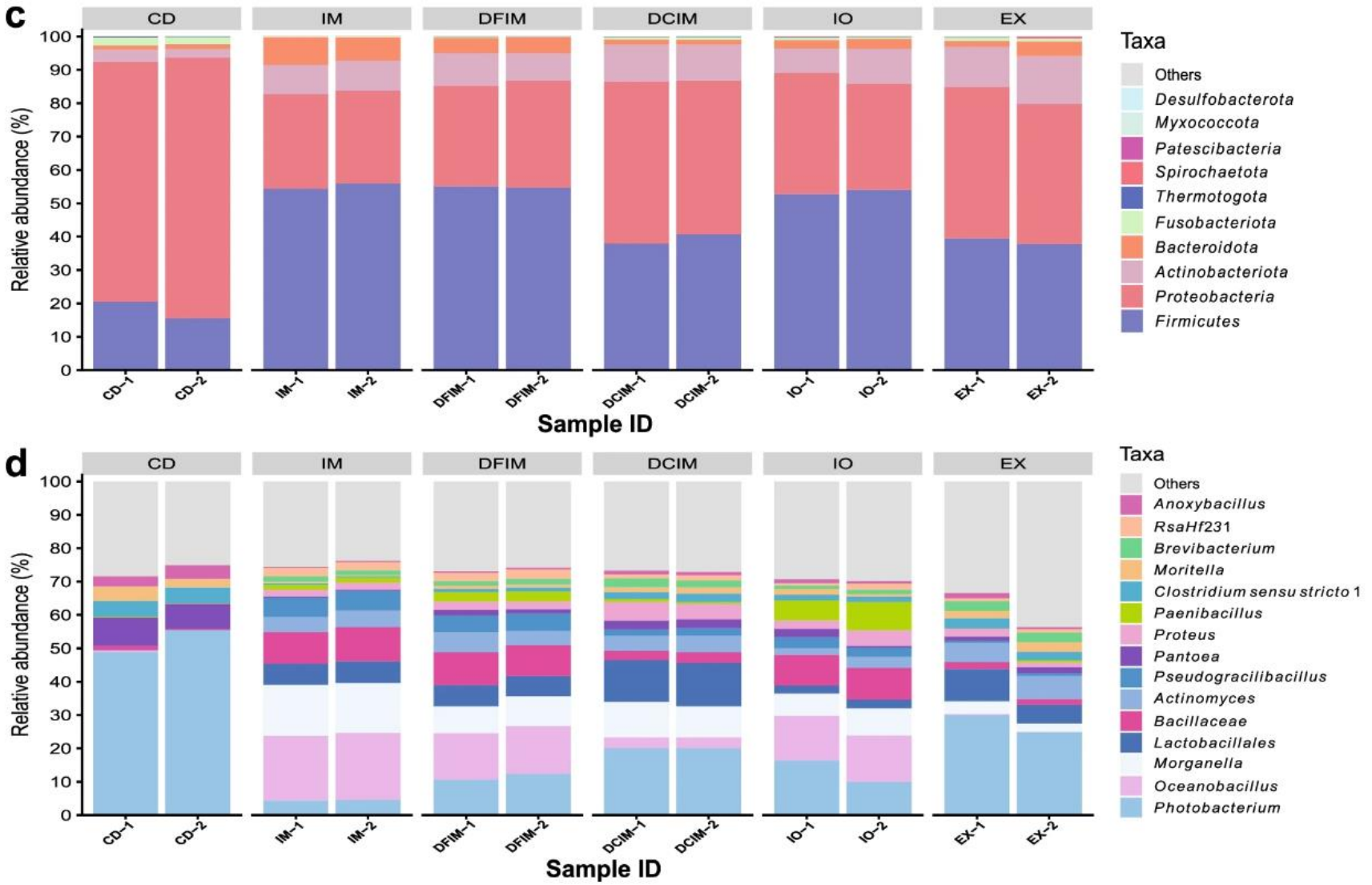


Weththasinghe, P., Rocha, S.D.C., Øyås, O. *et al.* (2022), Animal Microbiome, doi: 10.1186/s42523-021-00161-w

3. Taxonomic Analysis

Norwegian University of Life Sciences

Feed samles



FOODS OF NORWAY

Weththasinghe, P., Rocha, S.D.C., Øyås, O. *et al.* (2022), Animal Microbiome, doi: 10.1186/s42523-021-00161-w

3. Taxonomic Analysis

Core Microbiome

Norwegian University of Life Sciences

Calculate feature prevalence

```
# Compute feature prevalence
CD <- subset_samples(ps_nocontam, Diet == "CD") %>% prevalence()
IM <- subset_samples(ps_nocontam, Diet == "IM") %>% prevalence()
DFIM <- subset_samples(ps_nocontam, Diet == "DFIM") %>% prevalence()
DCIM <- subset_samples(ps_nocontam, Diet == "DCIM") %>% prevalence()
IO <- subset_samples(ps_nocontam, Diet == "IO") %>% prevalence()
EX <- subset_samples(ps_nocontam, Diet == "EX") %>% prevalence()
```

##Get core features that are present in at least 80% samples under the different diets

```
core_taxa <- cbind.data.frame(CD, IM, DFIM, DCIM, IO, EX) %>%
  rownames_to_column("featureID") %>%
  # get core features based on 80% prevalence threshold
  filter(CD >= 0.8|IM >= 0.8|DFIM >= 0.8|DCIM >= 0.8|IO >= 0.8|EX >= 0.8)
```

##Add taxonomy to core features

```
core_taxa_tab <- rownames_to_column(tax_tab, "featureID") %>%
  inner_join(core_taxa, by = "featureID") %>%
  mutate(prev_all = rowSums(.[9:14])) %>%
  arrange(desc(prev_all)) %>%
  mutate_if(is.numeric, ~ifelse(. == 1, 100, round(. * 100, 1))) %>%
```

3. Taxonomic Analysis

Core Microbiome

Norwegian University of Life Sciences

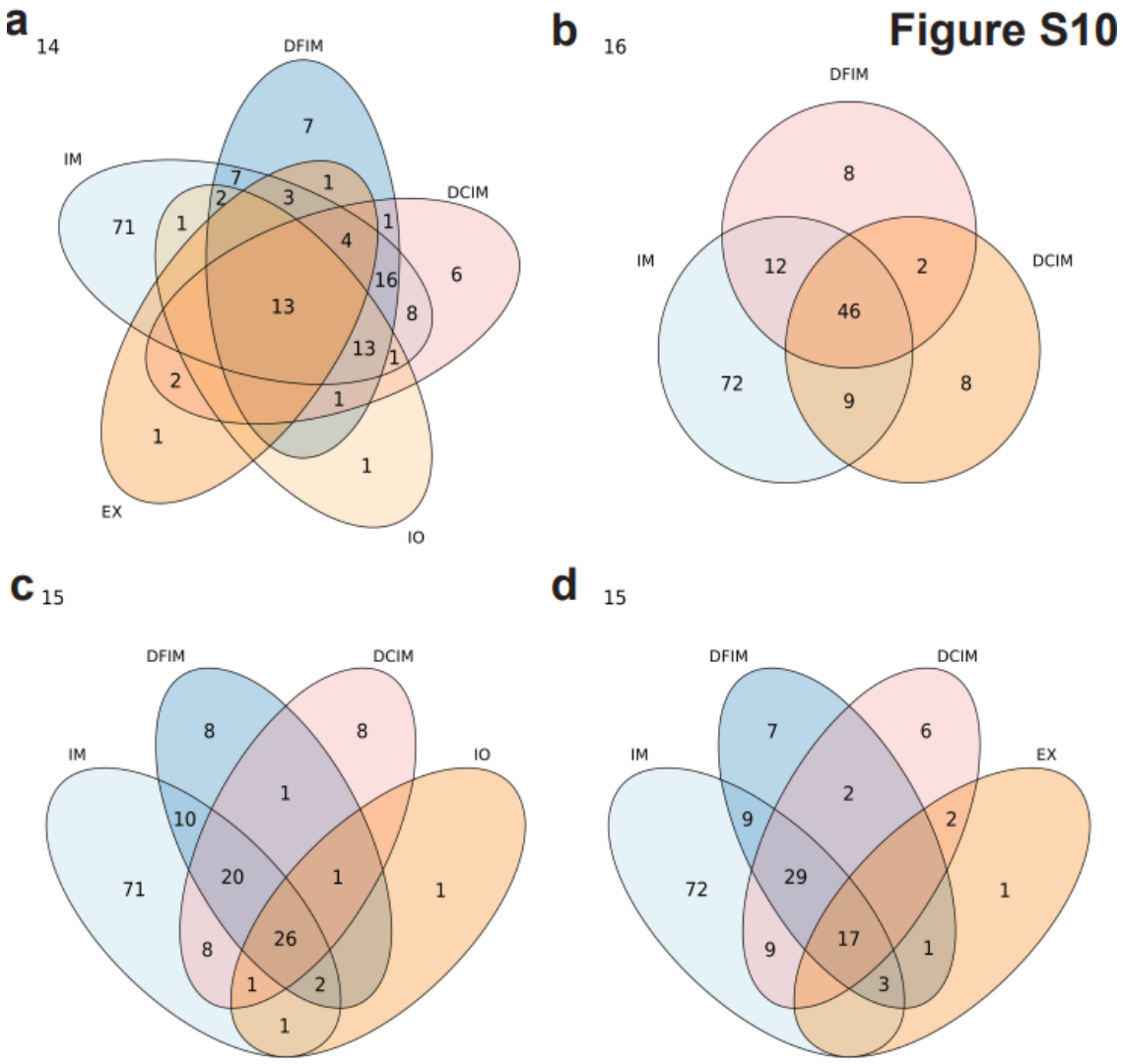


Figure S10

Weththasinghe, P., Rocha, S.D.C., Øyås, O. *et al.* (2022), Animal Microbiome, doi: 10.1186/s42523-021-00161-w

3. Taxonomic Analysis

Digesta, Feed and water overlap

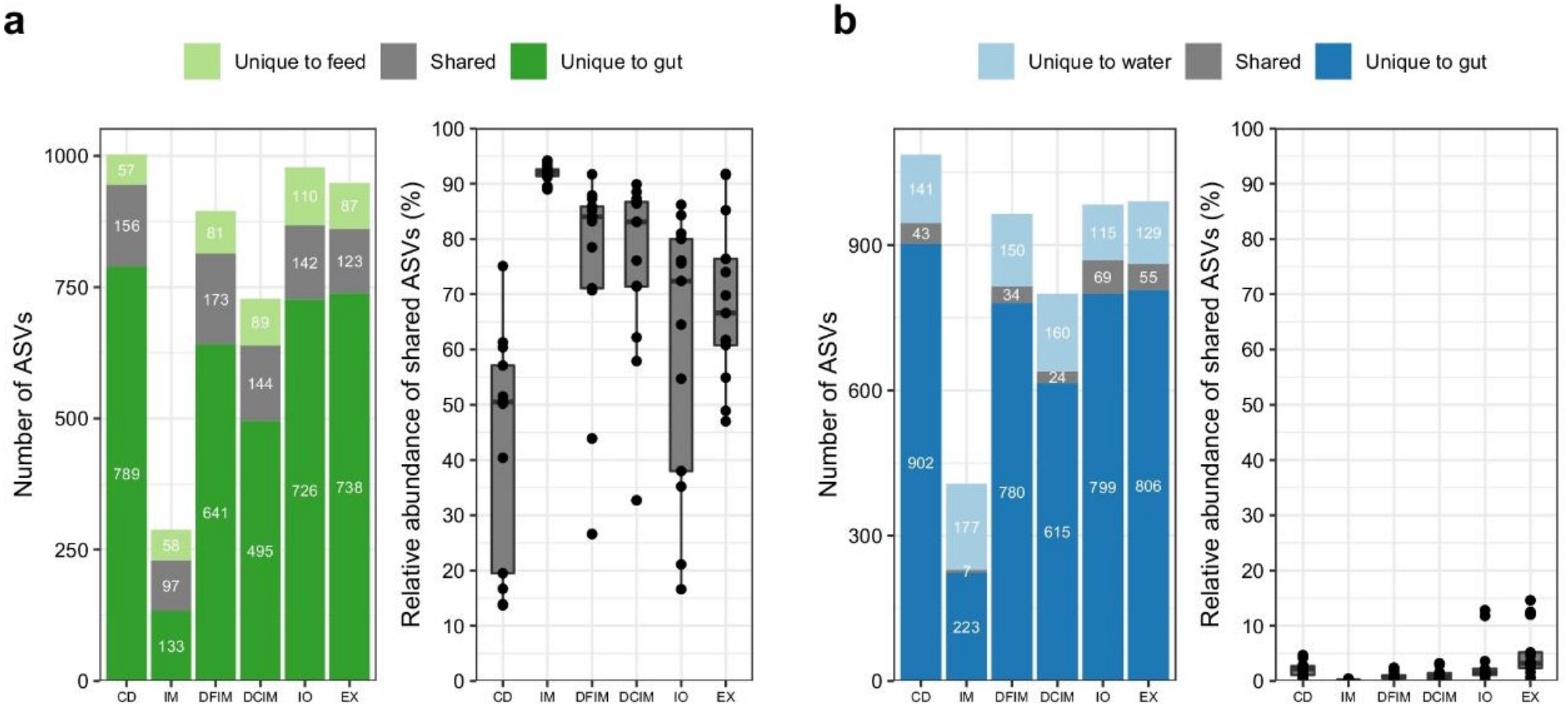
Norwegian University of Life Sciences

Compute ASV overlap

```
ovrl <- bind_cols(prvl) %>%
  pivot_longer("CD":"EX",
               names_to = "Sample",
               values_to = "Gut") %>%
  # compute ASV overlap between water/feed and intestinal samples
  mutate(
    WaterOverlap = case_when(
      Water >  0 & Gut >  0 ~ "Shared",
      Water >  0 & Gut == 0 ~ "Unique to water",
      Water == 0 & Gut >  0 ~ "Unique to gut",
      TRUE ~ "Absent"),
    FeedOverlapCD = case_when(
      FeedCD > 0 & Gut >  0 ~ "Shared",
      FeedCD > 0 & Gut == 0 ~ "Unique to feed",
      FeedCD == 0 & Gut >  0 ~ "Unique to gut",
      TRUE ~ "Absent"),
    FeedOverlapIM = case_when(
      FeedIM > 0 & Gut >  0 ~ "Shared",
      FeedIM > 0 & Gut == 0 ~ "Unique to feed",
      FeedIM == 0 & Gut >  0 ~ "Unique to gut",
      TRUE ~ "Absent"),
    FeedOverlapDFIM = case_when(
      FeedDFIM > 0 & Gut >  0 ~ "Shared",
      FeedDFIM > 0 & Gut == 0 ~ "Unique to feed",
      FeedDFIM == 0 & Gut >  0 ~ "Unique to gut",
      TRUE ~ "Absent"),
    FeedOverlapDCIM = case_when(
      FeedDCIM > 0 & Gut >  0 ~ "Shared",
      FeedDCIM > 0 & Gut == 0 ~ "Unique to feed",
```

3. Taxonomic Analysis

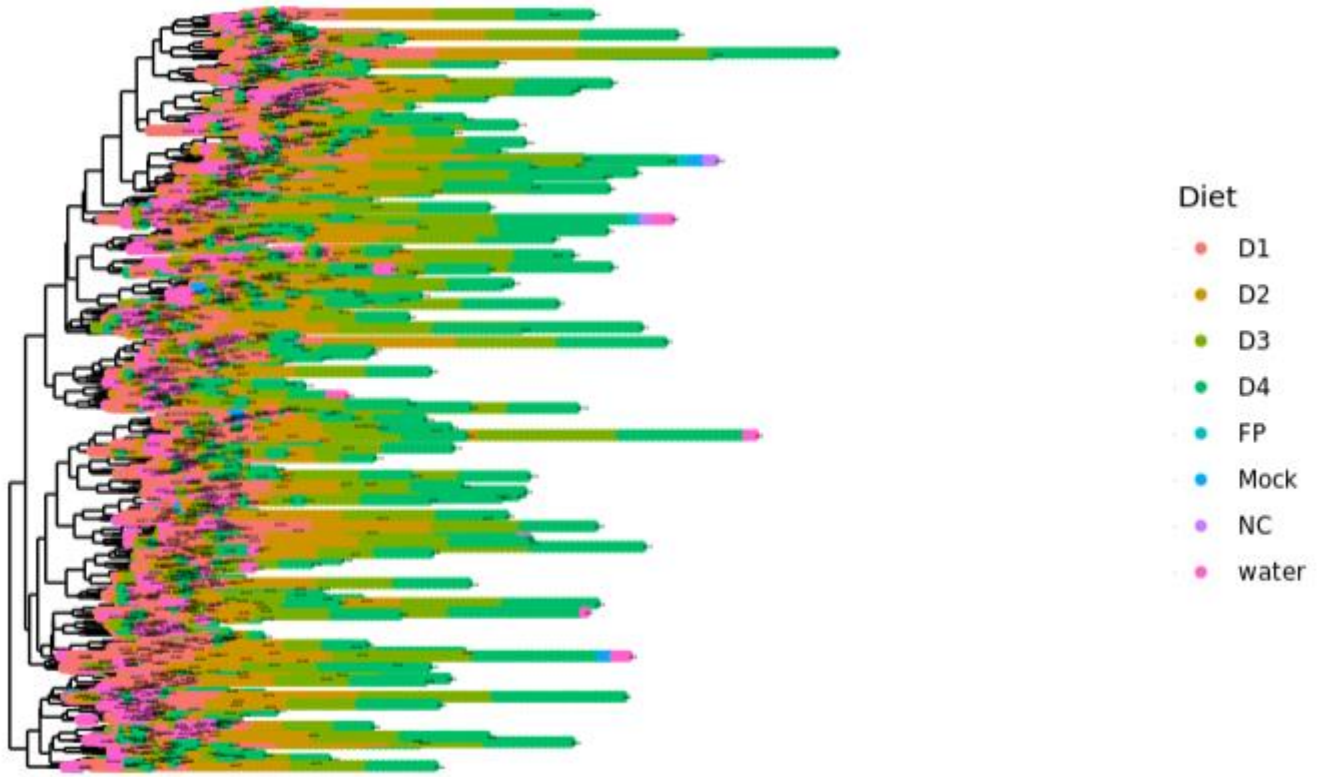
Digesta, Feed and water overlap



Weththasinghe, P., Rocha, S.D.C., Øyås, O. *et al.* (2022), Animal Microbiome, doi: 10.1186/s42523-021-00161-w

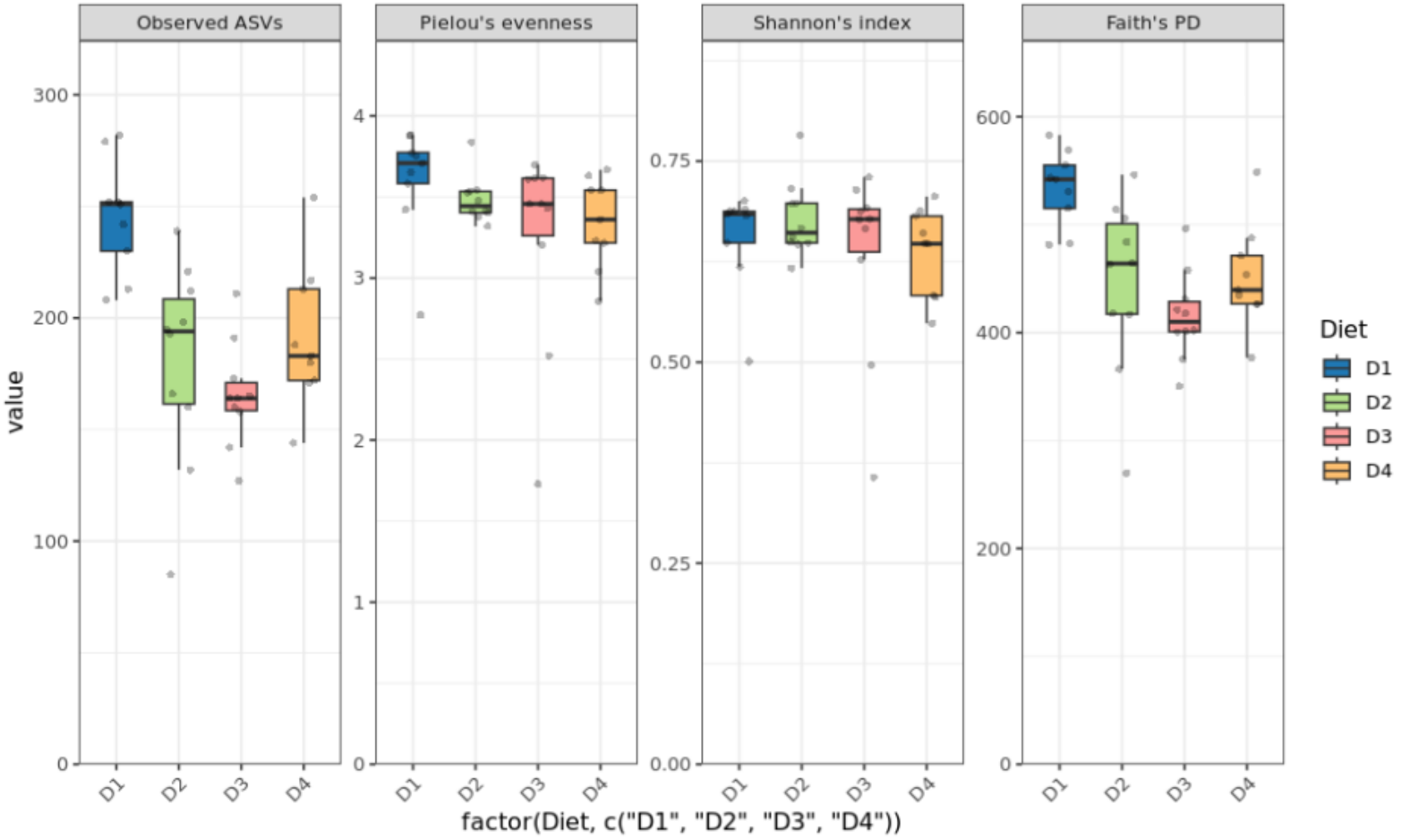
4. alpha-diversity

More and more packages + plot phylogenetic tree



4. alpha-diversity

Plot alpha-diversity indices



4. alpha-diversity

Norwegian University of Life Sciences

statistics

```
# Alpha-diversity group significance

The statisitical difference between the diets for the four alpha diversity measurements were evaluated using kruska-walis test and the significance between diets were identified with pairwise wilcox test.

##Flitering the adiv dataframe into each alpha diversity index

```{r}
##Filter only the dataframe for Observed ASVs
adiv_o <- adiv %>%
            filter(adiv == "Observed ASVs")
##Filter only the dataframe for Pielou's evenness
adiv_p <- adiv %>%
            filter(adiv == "Pielou's evenness")
##Filter only the dataframe for Shannon's index
adiv_s <- adiv %>%
            filter(adiv == "Shannon's index")
##Filter only the dataframe for Faith's PD
adiv_f <- adiv %>%
            filter(adiv == "Faith's PD")
```
```

4. alpha-diversity

Norwegian University of Life Sciences

statistics

```
## Test the diet effect on alpha diversity indices - Kruska-walis test¨

```{r}
##Statistics for observed ASVs
kruskal.test(value ~ Diet, data = adiv_o)

##Statistics for Pielou's evenness
kruskal.test(value ~ Diet, data = adiv_p)

##Statistics for Shannon's index
kruskal.test(value ~ Diet, data = adiv_s)

##Statistics for Faith's PD
kruskal.test(value ~ Diet, data = adiv_f)
```
```

```
        Kruskal-Wallis rank sum test

data:  value by Diet
Kruskal-Wallis chi-squared = 18.146, df = 3, p-value = 0.0004103


        Kruskal-Wallis rank sum test

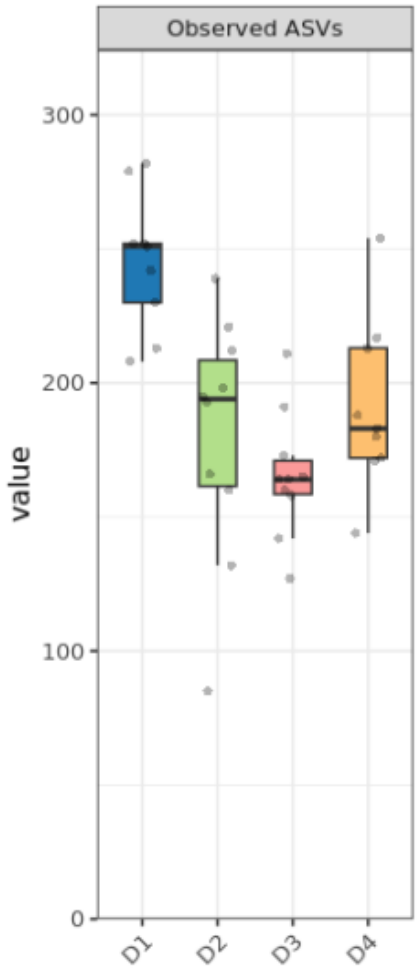
data:  value by Diet
Kruskal-Wallis chi-squared = 6.7913, df = 3, p-value = 0.07886


        Kruskal-Wallis rank sum test

data:  value by Diet
Kruskal-Wallis chi-squared = 1.7168, df = 3, p-value = 0.6332


        Kruskal-Wallis rank sum test

data:  value by Diet
Kruskal-Wallis chi-squared = 16.681, df = 3, p-value = 0.0008218
```



4. alpha-diversity

Norwegian University of Life Sciences

statistics

```
## Pair-wise comparison

from rhe P-values for alpha diversity indices, observed ASVs was the only one below 0.05. Thus, we proceed to use wilcox pairwise comparison to identify differences between diets.

setwd("/net/fs-2/scale/OrionStore/Scratch/sergroch/FON48/")
```

```{r}
##Statistics for observed ASVs
wilcox_o <- compare_means(value ~ Diet, adiv_o, method = "wilcox.test")

##Statistics for Pielou's evenness
wilcox_p <-compare_means(value ~ Diet, adiv_p, method = "wilcox.test")

##Statistics for Shannon's index
wilcox_s <-compare_means(value ~ Diet, adiv_s, method = "wilcox.test")

##Statistics for Faith's PD
wilcox_f <-compare_means(value ~ Diet, adiv_f, method = "wilcox.test")

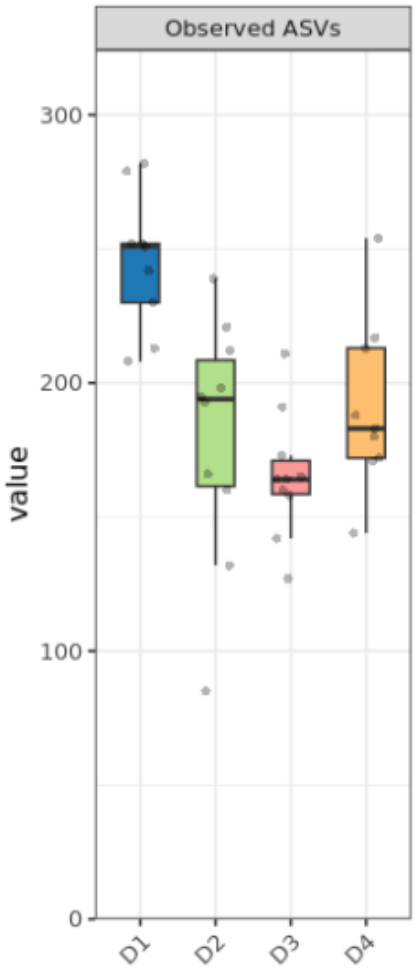
##Write the pairwise wilcox comparison for each alpha diversity indices into a CSV file to annotate Figure 4 using adobe illustrator on any of the editing software later
write.csv(wilcox_o, file = "Wilcox pairwise comparison_Observed ASVs_digesta", row.names = FALSE)
write.csv(wilcox_p, file = "Wilcox pairwise comparison_Pielou's evenness_digesta", row.names = FALSE)
write.csv(wilcox_s, file = "Wilcox pairwise comparison_Shannon's index_digesta", row.names = FALSE)
write.csv(wilcox_f, file = "Wilcox pairwise comparison_Faith's PD_digesta", row.names = FALSE)

openxlsx::write.xlsx(wilcox_o, file = "Wilcox pairwise comparison_Observed ASVs_digesta.xlsx", overwrite = TRUE)
openxlsx::write.xlsx(wilcox_p, file = "Wilcox pairwise comparison_Pielou's evenness_digesta.xlsx", overwrite = TRUE)
openxlsx::write.xlsx(wilcox_s, file = "Wilcox pairwise comparison_Shannon's index_digesta.xlsx", overwrite = TRUE)
openxlsx::write.xlsx(wilcox_f, file = "Wilcox pairwise comparison_Faith's PD_digesta.xlsx", overwrite = TRUE)
```

A tibble: 6 × 8

.y. <chr>	group1 <chr>	group2 <chr>	p <dbl>	p.adj <dbl>	p.format <chr>	p.signif <chr>	method <chr>
value	D4	D1	2.756067e-03	0.01400	0.0028	**	Wilcoxon
value	D4	D3	6.525363e-02	0.20000	0.0653	ns	Wilcoxon
value	D4	D2	1.000000e+00	1.00000	1.0000	ns	Wilcoxon
value	D1	D3	8.660071e-05	0.00052	8.7e-05	****	Wilcoxon
value	D1	D2	5.672346e-03	0.02300	0.0057	**	Wilcoxon
value	D3	D2	1.654939e-01	0.33000	0.1655	ns	Wilcoxon

6 rows



5. beta-diversity

More packages again + rarefraction

```
## Check sample rarefraction curve.

```{r}
p <- ggrare(pseq, step = 1000, color ="Diet", label = "Diet", se = FALSE)
p <- p + facet_wrap(~Diet)
p
pseq1 <- subset_samples(pseq, Diet == "D1")
p1 <- ggrare(pseq1, step = 1000, color ="Diet", label = "Diet", se = FALSE)

pseq2 <- subset_samples(pseq, Diet == "D2")
p2 <- ggrare(pseq2, step = 1000, color ="Diet", label = "Diet", se = FALSE)

pseq3 <- subset_samples(pseq, Diet == "D3")
p3 <- ggrare(pseq3, step = 1000, color ="Diet", label = "Diet", se = FALSE)

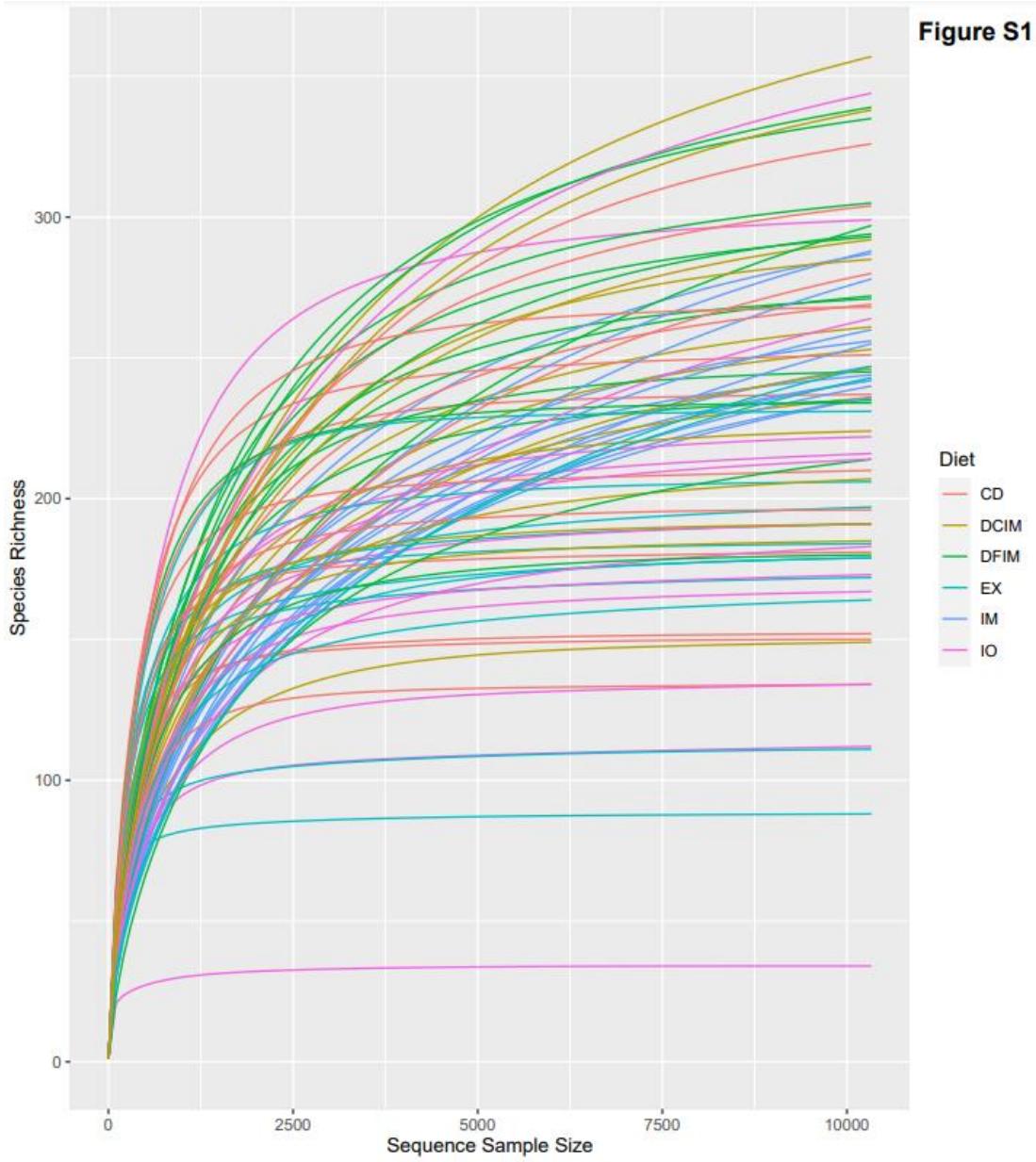
pseq4 <- subset_samples(pseq, Diet == "D4")
p4 <- ggrare(pseq4, step = 1000, color ="Diet", label = "Diet", se = FALSE)

```
```

curves showing the number of unique sequence variants as a function of normalized library size

5. beta-diversity

Rarefraction curve



Norwegian University of Life Sciences

Weththasinghe, P., Rocha, S.D.C., Øyås, O. *et al.* (2022), Animal Microbiome, doi: 10.1186/s42523-021-00161-w

5. beta-diversity

Norwegian University of Life Sciences

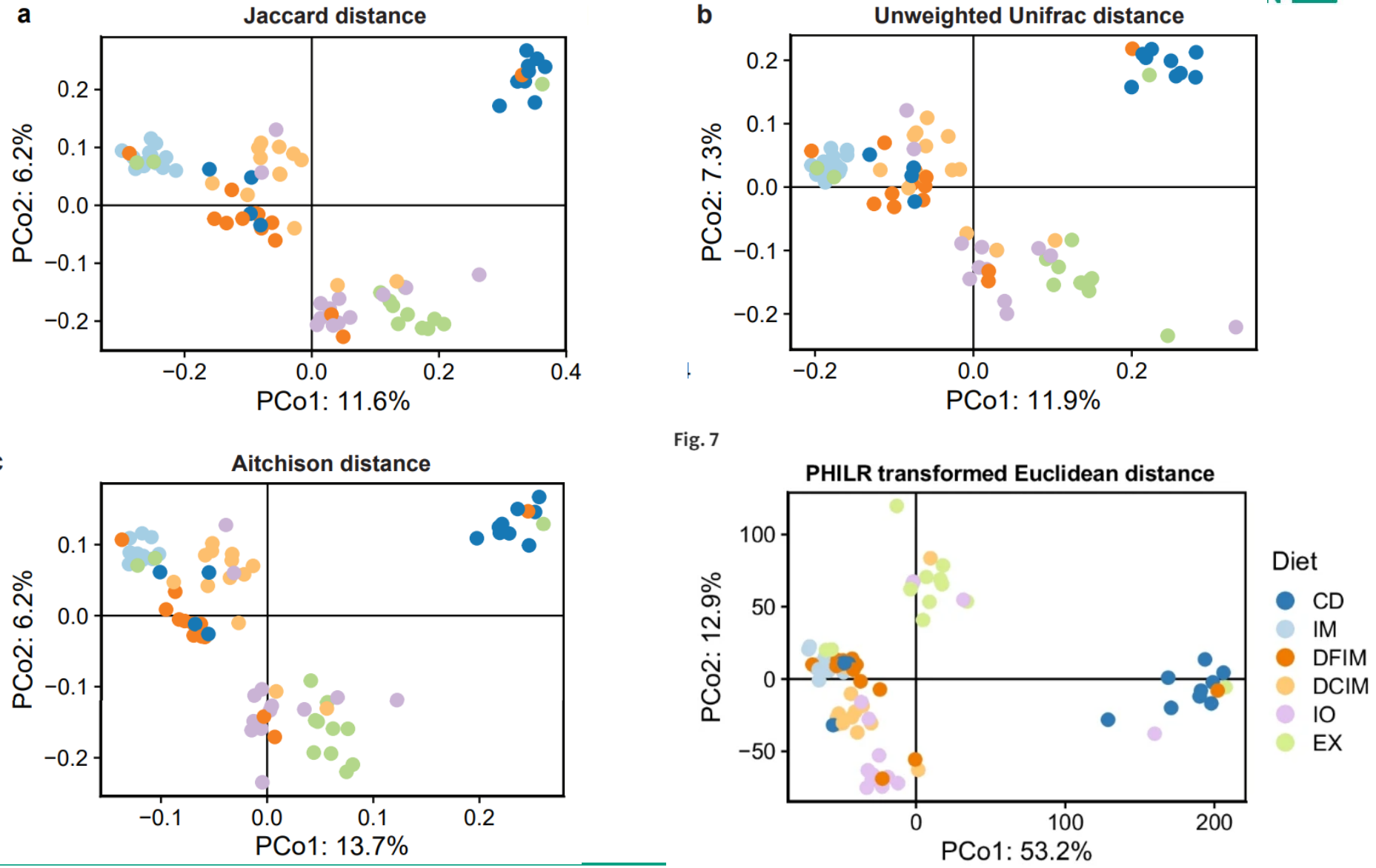
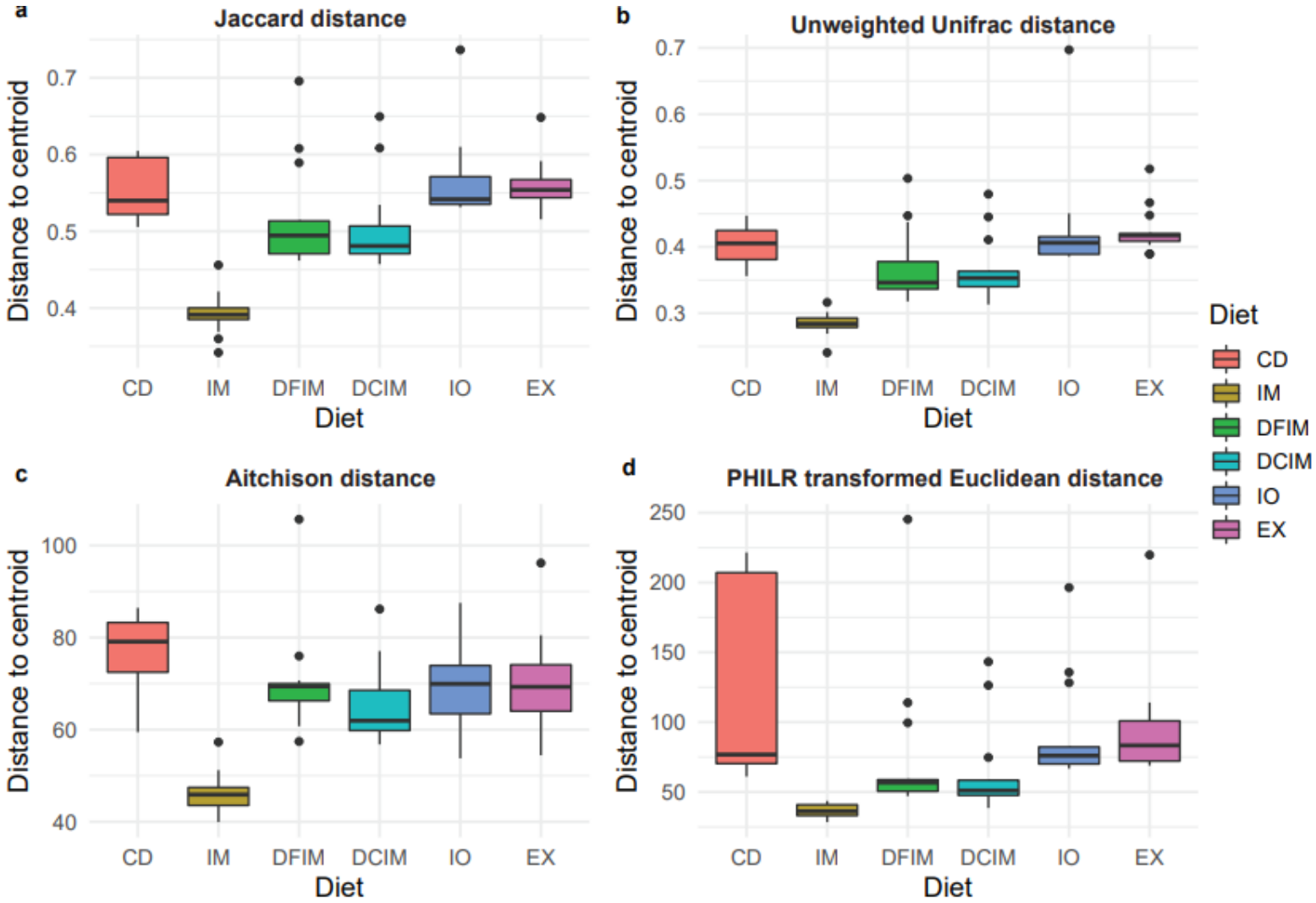


Fig. 7

Weththasinghe, P., Rocha, S.D.C., Øyås, O. *et al.* (2022), Animal Microbiome, doi: 10.1186/s42523-021-00161-w

5. beta-diversity

Norwegian University of Life Sciences



Weththasinghe, P., Rocha, S.D.C., Øyås, O. *et al.* (2022), Animal Microbiome, doi: 10.1186/s42523-021-00161-w

6. Individual taxa comparisson

Tidy taxa table of top 15

```r
taxa_tab <- as.data.frame(taxa_tab) %>%
  rownames_to_column("Taxa") %>%
  separate(
    Taxa,
    sep = ";",
    c("Kingdom", "Phylum", "Class", "Order", "Family", "Genus")) %>%
  mutate(
    Phylum = ifelse(
      is.na(Phylum)|Phylum == "NA"|grepl("uncultured|Ambiguous|metagenome", Phylum),
      Kingdom,
      Phylum),
    Class = ifelse(
      is.na(Class)|Class == "NA"|grepl("uncultured|Ambiguous|metagenome", Class),
      Phylum,
      Class),
    Order = ifelse(
      is.na(Order)|Order == "NA"|grepl("uncultured|Ambiguous|metagenome", Order),
      Class,
      Order),
    Family = ifelse(
      is.na(Family)|Family == "NA"|grepl("uncultured|Ambiguous|metagenome", Family),
      Order,
      Family),
    Genus = ifelse(
      is.na(Genus)|Genus == "NA"|grepl("uncultured|Ambiguous|metagenome", Genus),
      Family,
      Genus)) %>%
  select(-Kingdom, -(Class:Family))


taxa_tab1 <- taxa_tab %>%
  mutate(
    Phylum = gsub("p__", "", Phylum),
    Phylum = factor(Phylum, levels = rev(unique(Phylum))),
    Genus = gsub("g__", "", Genus),
    Genus = factor(Genus, levels = rev(unique(Genus)))) %>%
```

6. Individual taxa comparisson

Norwegian University of Life Sciences

Boxplot of individual groups

```
### Boxplot to abundance of individual dominant taxa in the DIGESTA samples grouped by diets

```{r, fig.width=10}
# define color scheme
col <- c("grey", brewer.pal(n = 10, name = "Paired"))
mtd$Diet <- factor(mtd$Diet, levels =  c("D1", "D2", "D3", "D4"))
taxa_tab1$Genus2 <- reorder(taxa_tab1$Genus, taxa_tab1$Abundance)
# digesta samples
taxa_boxplot_digesta <- filter(taxa_tab1, Sample_type == "digesta") %>%
  ggplot(aes(x = factor(Diet, c("D1", "D2", "D3", "D4")), y = Abundance, fill = Genus)) +
  geom_boxplot(aes(fill = factor(Diet, c("D1", "D2", "D3", "D4"))), alpha = 0.5, width = 0.5) +
  labs(x = "Diets", y = "Relative abundance (%)") +
  ##scale_y_continuous(breaks = 0:10*10, expand = c(0,0)) +
  scale_fill_manual(values = col) +
  facet_wrap(
    ~ Genus2,nrow = 4,
    scale = "free"

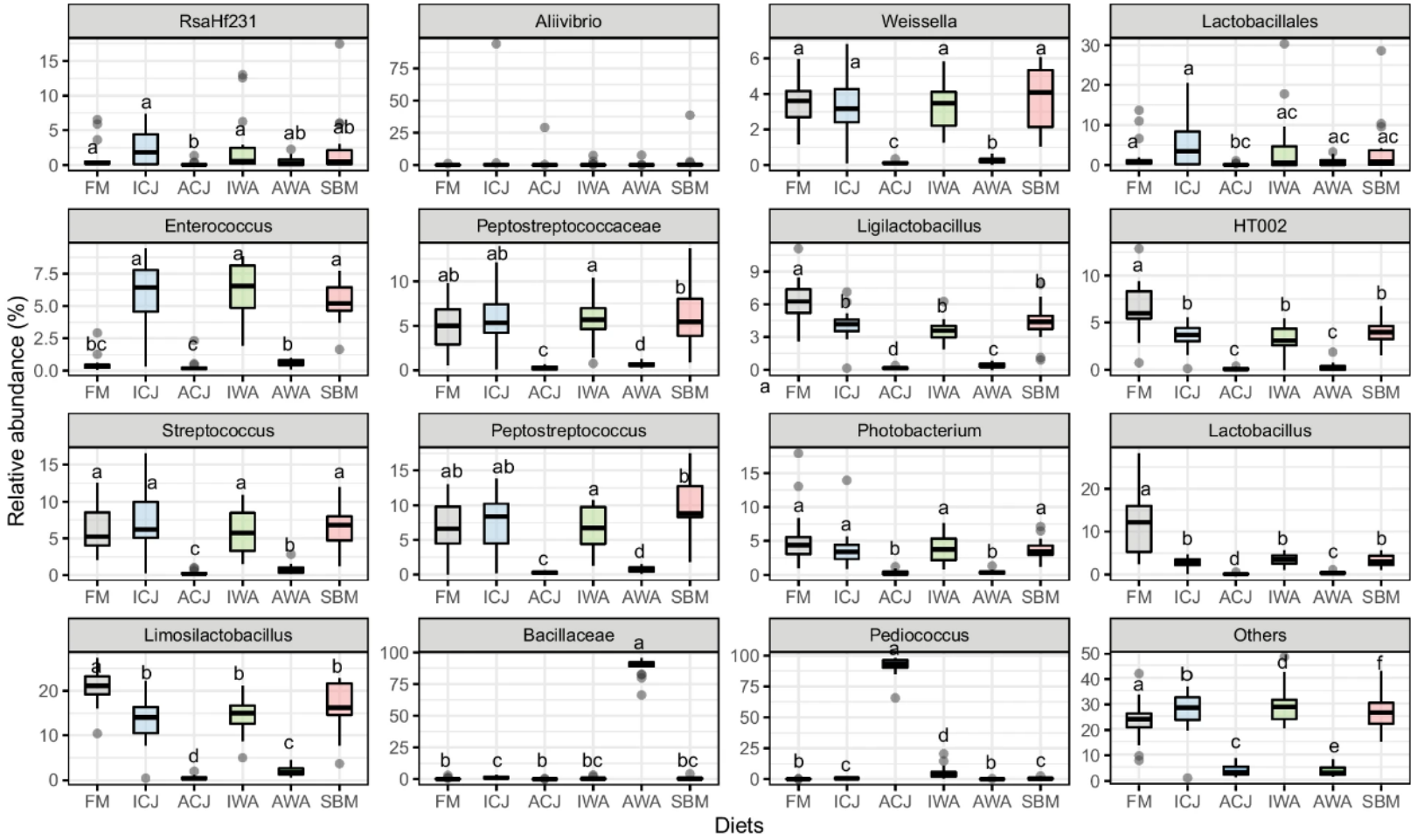
    ) +
  theme_bw() +
  theme(##axis.text.x = element_blank(),
        ##strip.background = element_blank(),
        legend.position = "none")

# Export the plot
ggsave("boxplot_abundance_digesta.tiff", width = 10, height = 6,
       units = "in", dpi = 300, compression = "lzw")
ggsave("boxplot_abundance_digesta.pdf", width = 10, height = 6,
       units = "in", dpi = 300)


```
```

6. Individual taxa comparisson

Boxplot of individual groups



Agboola, J.O., Rocha, S.D.C., Mensah, D.D. *et al. et al.* (2023), Animal Microbiome, doi: 10.1186/10.1186/s42523-023-00242-y

6. Individual taxa comparisson

Filtering groups and perform statistic groups

```
##Flitering the dataframe into each individual taxonomic

```{r}
##Filter only the dataframe for only digesta
taxa_tab2 <- filter(taxa_tab1, Sample_type == "digesta")

##Filter taxa_tab2 dataframe for only the HT002
taxa_h <- taxa_tab2 %>%
            filter(taxa_tab2$Genus == "HT002")
##Filter taxa_tab2 dataframe for only the Corynebacterium
taxa_c <- taxa_tab2 %>%
            filter(taxa_tab2$Genus == "Corynebacterium")

##Filter taxa_tab2 dataframe for only Leuconostoc
taxa_leu <- taxa_tab2 %>%
            filter(taxa_tab2$Genus == "Leuconostoc")

##Filter taxa_tab2 dataframe for only the Furfurilactobacillus
taxa_fur <- taxa_tab2 %>%
            filter(taxa_tab2$Genus == "Furfurilactobacillus")

##Filter taxa_tab2 dataframe for only the Rubritalea
taxa_sec <- taxa_tab2 %>%
            filter(taxa_tab2$Genus == "Secundilactobacillus")

##Filter taxa_tab2 dataframe for only the Lactiplantibacillus
taxa_lac1 <- taxa_tab2 %>%
            filter(taxa_tab2$Genus == "Lactiplantibacillus")

##Filter taxa_tab2 dataframe for only Weissella
taxa_w <- taxa_tab2 %>%
            filter(taxa_tab2$Genus == "Weissella")
```

```
Test the diet effect on individual taxonomy - Kruska-walis tes

```{r}
##Statistics for HT002
kruskal.test(Abundance ~ Diet, data = taxa_h)

##Statistics for Corynebacterium
kruskal.test(Abundance ~ Diet, data = taxa_c)

##Statistics for Leuconostoc
kruskal.test(Abundance ~ Diet, data = taxa_leu)

##Statistics for Furfurilactobacillus
kruskal.test(Abundance ~ Diet, data = taxa_fur)

##Statistics for Secundilactobacillus
kruskal.test(Abundance ~ Diet, data = taxa_sec)

##Statistics for Lactiplantibacillus
kruskal.test(Abundance ~ Diet, data = taxa_lac1)

##Statistics for Weissella
kruskal.test(Abundance ~ Diet, data = taxa_w)
```

7. LEfSe

Norwegian University of Life Sciences

Segata *et al. Genome Biology* 2011, **12**:R60
http://genomebiology.com/2011/11/6/R60

Genome Biology

METHOD Open Access

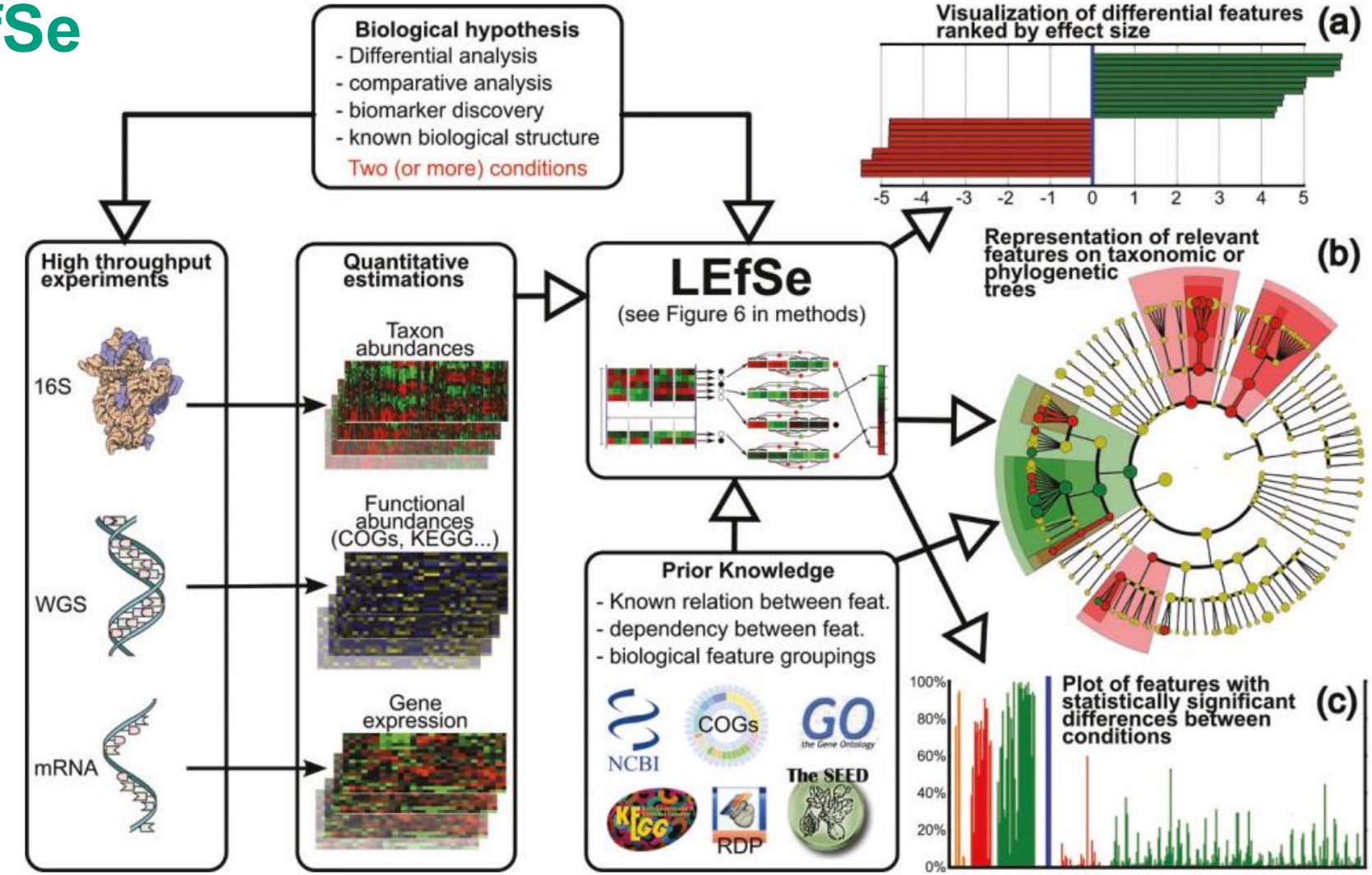
Metagenomic biomarker discovery and explanation

Nicola Segata[1], Jacques Izard[2,3], Levi Waldron[1], Dirk Gevers[4], Larisa Miropolsky[1], Wendy S Garrett[5,6,7] and Curtis Huttenhower[1*]

Abstract

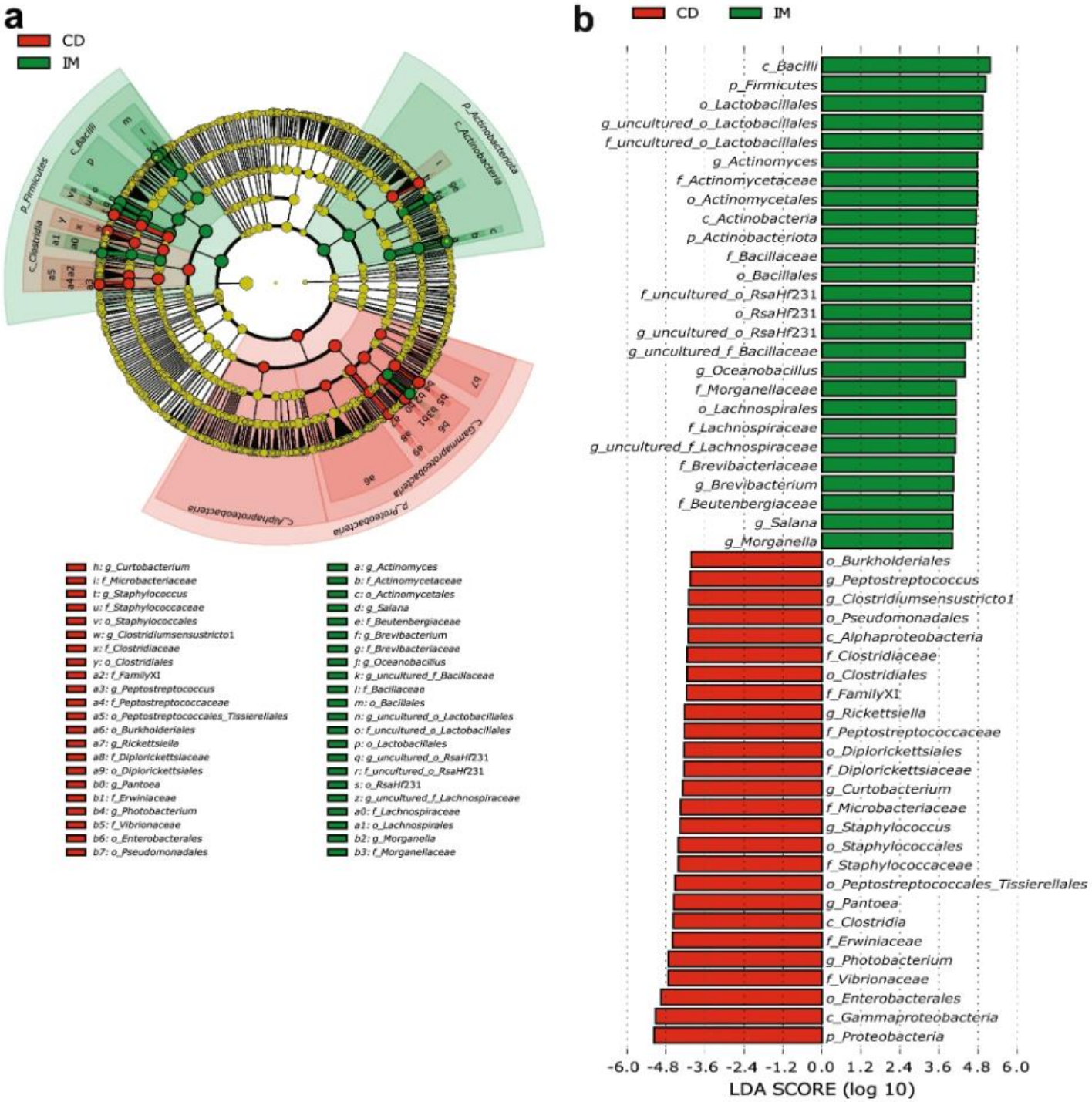
This study describes and validates a new method for metagenomic biomarker discovery by way of class comparison, tests of biological consistency and effect size estimation. This addresses the challenge of finding organisms, genes, or pathways that consistently explain the differences between two or more microbial communities, which is a central problem to the study of metagenomics. We extensively validate our method on several microbiomes and a convenient online interface for the method is provided at http://huttenhower.sph.harvard.edu/lefse/.

7. LEfSe



http://galaxy.biobakery.org/

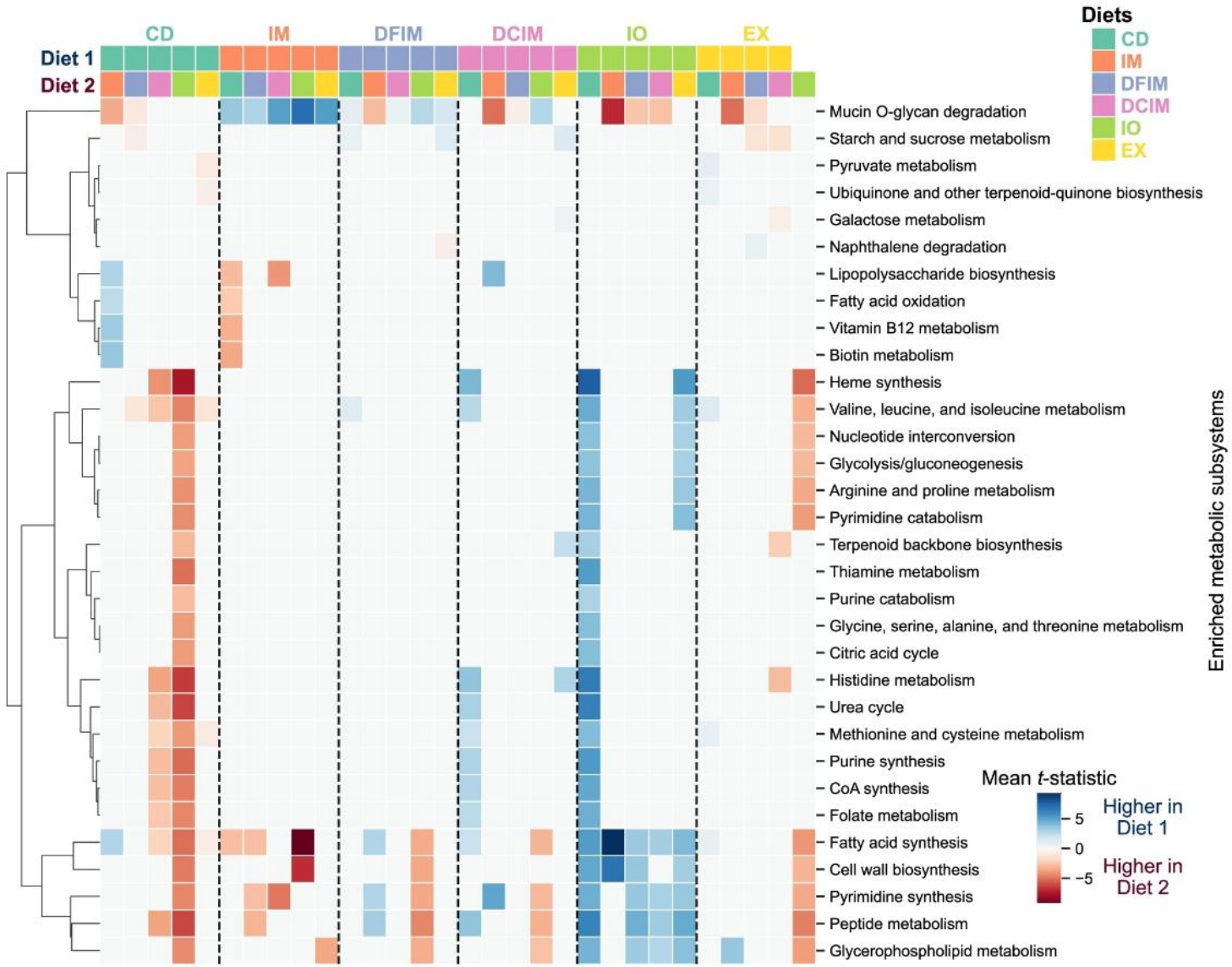
7. LEfSe



Norwegian University of Life Sciences

Weththasinghe, P., *et al.* (2022)

8. Metabolic pathways



Weththasinghe, P., Rocha, S.D.C., Øyås, O. *et al.* (2022), Animal Microbiome, doi: 10.1186/s42523-021-00161-w