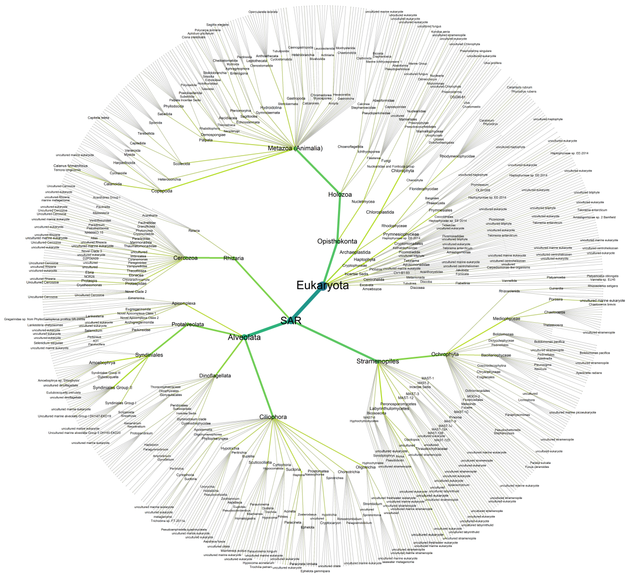
Fiskaaling
Aquaculture Research Station of the Faroes

Methods

Bioinformatic pipeline

A user guide documenting the pipeline used to process amplicon-based next-generation sequencing data at Fiskaaling



Ása Jacobsen
Ása Johannesen

1. February 2023

Fiskaaling rit 2023-01

Fiskaaling P/F
við Áir, FO-430 Hvalvík, Føroyar (Faroe Islands)
Tlf. (+298) 474747, fiskaaling@fiskaaling.fo, www.fiskaaling.fo

Fiskaaling
Aquaculture Research Station of the Faroes
Fiskaaling P/F
við Áir
FO-430 Hvalvík
Føroyar (Faroe Islands)

Tlf. (Phone) (+298) 474747
Fax (+298) 474748
E-mail:fiskaaling@fiskaaling.fo
www.fiskaaling.fo

Methods

Heiti:	Bioinformatic pipeline	
Høvundar:	Ása Jacobsen Ása Johannesen	
Status: Open	Frágr.nr.: 2023-01	Verkætlan: Pipeline
Dato: 1. February 2023	Tal av síðum: 25	Ábyrgdarhavi: Ása Jacobsen
Góðkent: Amanda Vang		Undirritan:

Samandráttur:
This documents the data processing involved in obtaining information from sequencing data. The process starts with reading the raw sequences, identifying them, and adding a taxonomy. It concludes with producing plots (taxonomic trees and diversity plots) and statistics. The document is designed to be used as a modifiable general pipeline guide with all the necessary code to get started on the pipeline for a new project.

Leitiorð:

Bioinformatics, pipeline, Qiime2, eDNA, microbiome

Contents

1 Introduction

In this document, we describe the pipeline developed for processing amplicon-based next-generation sequencing data gathered at Fiskaaling. These data may be from microbiome analysis or eDNA surveys. The pipeline can be used for a variety of primers, reference databases, and applications.
The pipeline documents the process from importing raw paired-end sequencing data with quality scores to producing statistical analysis and plots.
The aim of this document is to provide a dynamic and user-friendly way to gain information from sequencing data, so all code chunks provided are annotated and explanations are also provided. This way, someone can directly copy the code and easily insert their own variable names and directory paths when necessary. The pipeline includes some of the most frequently used code for analysis, but users should expect to build on or adapt according to their specific requirements. The code is available in R and Jupyter notebooks on GitHub (https://github.com/Fiskaaling/NGS_Pipeline/releases/tag/v.1.0).
More information for adapting this code can be found on the Qiime2 website (https://qiime2.org/), which has a friendly and helpful forum and in R documentation, such as the documentation for the MetacodeR package, which is used for much of the work that is carried out in R in this report (https://www.rdocumentation.org/packages/metacoder/versions/0.3.5).

2 Setup

This pipeline was constructed using Qiime2 version 2021.2.0 (Bolyen et al. 2019) running with Python version 3.6.13 (Van Rossum and Drake Jr 1995). Any required packages are noted in the code snippets and while most of them ought to be already bundled with Qiime2, if any are missing, they should be easily installed following documentation for Qiime2 at docs.qiime2.org.
The packages, plugins, formats, and methods used in this pipeline are as follows:

- Qiime2
  - EMPeror (Vázquez-Baeza et al. 2013; Vázquez-Baeza et al. 2017)
  - RESCRIPt (Robeson et al. 2020)
  - FastTree (Price, Dehal, and Arkin 2010)
  - Scikit-learn (Pedregosa et al. 2011)
  - BIOM (McDonald et al. 2012)
  - UniFraq (McDonald et al. 2018; Lozupone and Knight 2005; Lozupone et al. 2011; Hamady, Lozupone, and Knight 2010)
  - MAFFT (Katoh and Standley 2013)
  - DADA2 (Callahan et al. 2016)
  - q2-feature-classifier (Bokulich et al. 2018)
  - Alpha diversity metrics (Faith 1992; Jaccard 1908; Shannon 1948; Pielou 1966; Sørensen 1948)
  - NCBI BLAST (Johnson et al. 2008; NCBI Resource Coordinators 2017)
- R
  - Tidyverse (Wickham et al. 2019)
  - Qiime2R (Bisanz 2018)

– Metacoder (Foster, Sharpton, and Grünwald 2017)

All of the Qiime2 code is written in Jupyter Notebooks for ease of use. This means that the code snippets are preceded with an '!' in order to run in a Jupyter Notebook. These would not be necessary if running from the terminal.

Visualisation and some data manipulation is carried out in R using RStudio and ggplot2. The necessary packages are listed at the top of each notebook, so these need to be installed for that notebook to work. All of the packages are available on CRAN, so these ought to be easy to install using RStudio's package manager.

3 Explanations

Various reference databases are available for taxonomy assignment. The reference database of choice depends on primers used and taxons of interest. In this case we have used a pretrained Silva reference database that we downloaded from the Qiime2 documentation website, but other pretrained naïve bayes classifiers can be used instead. This document does not describe how to train classifiers, but this information can be found at the site of origin or tutorial found on the Qiime2 website. Be ware that sometimes reference databases differ in their taxonomic rank setup, which might require adaptation of some of the described code.

The "4.2 Denoising" section documents how to trim and truncate your sequences based on the length of the primer used for your data as well as the quality plots produced in the "4.1 Data import" section. This means that there are a lot of values that the user needs to set themselves. Also, this is computationally quite a demanding task, so if possible, use several cores (denoted with '–p-n-threads').

The file types that Qiime2 produces are '.qza', which are files that can be used as input files in further analysis and '.qzv' which only can be used for visualization. The '.qza' files can be used both for analysis in Qiime2 and imported into R using a package as noted in the "5.1 Importing to R" section.

In addition to these files, the sample metadata is a text file, usually comma separated or tab separated (.csv or .tsv) and the raw sequence data files.

In the code, the naming of output files (which are also used downstream) is constructed in a neutral consistent way, where user specific details can be added. This makes the process of running the codes easier and minimizes the risk of mixing and confusing files. All instances where users may modify file names or other code have been highlighted using "<>" to designate where the user ought to change a variable or file name.

4 Qiime2 Code

The code written in Qiime2 encompasses importing raw sequence data, estimating quality, performing denoising, assigning taxonomy, filtering data, calculating alpha and beta diversity metrics and performing some beta diversity statistical analysis.

4.1 Data import

First, raw data are imported using the notebook called "1.dataimport.ipynb". Here is the code for importing and validating paired end sequencing data. Here we also summarize and look at quality plots for setting truncation values. The quality plots are then used to estimate the best places to trim sequences in the next notebook.

Choose path or make sure you are in the right folder/directory

```
[ ]: import os
     os.chdir(<your path>)
```

Import data. Here we define:

- the type of raw data “Paired end sequences with quality”
- the path to the relevant directory
- the format of the input data
- the name of the output file and where to put it (currently in the working directory, so no path is defined).

```
[ ]: !qiime tools import \
     --type 'SampleData[PairedEndSequencesWithQuality]' \
     --input-path <your path> \
     --input-format CasavaOneEightSingleLanePerSampleDirFmt \
     --output-path demux-paired-end.qza
```

Take a look at the imported data

```
[ ]: !qiime tools peek demux-paired-end.qza
```

Validate the imported data

```
[ ]: !qiime tools validate demux-paired-end.qza
```

The raw data are summarized and a visualisation file (.qzv) is created.

```
[ ]: !qiime demux summarize \
     --i-data demux-paired-end.qza \
     --o-visualization demux-paired-end.qzv
```

Run the next two cells to looking at the quality plots and setting truncation values. The loaded visualization includes a download link if you want the summarized data, including a table with reads per sample, in a csv format.

```
[ ]: from qiime2 import Visualization
```

```
[ ]: Visualization.load('demux-paired-end.qzv')
```

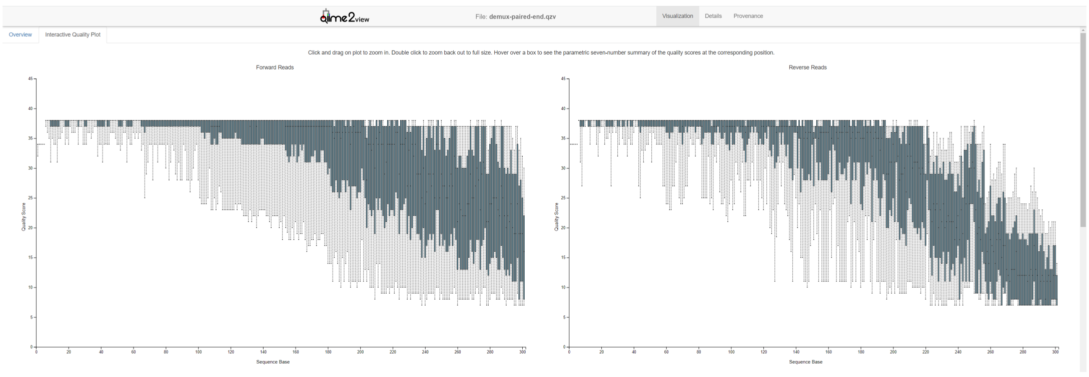

qiime2view
File: demux-paired-end.qzv
Visualization
Details
Provenance
Overview
Interactive Quality Plot
Click and drag on plot to zoom in. Double click to zoom back out to full size. Hover over a box to see the parametric seven-number summary of the quality scores at the corresponding position.
Forward Reads
Reverse Reads
Quality Score
Sequence Base

Figure 1: Example of quality plots of forward and reverse reads visualised when running demux-paired-end.qzv

4.2 Denoising

Here we tidy up the data using dada2 for denoising, and a variety of summarising and filtering.

Choose path or make sure you are in the right folder

```
[ ]: import os
     os.chdir(<your path>)
```

Denoising with dada2.

- input the data created in the data import code
- trim according to the primers that you have used
- truncate according to the quality plots
- output representative sequences
- output a dada2 feature table
- define how many threads you want to use. Useful for big data if you have many cores
- output the denoising stats

```
[ ]: !qiime dada2 denoise-paired \
     --i-demultiplexed-seqs demux-paired-end.qza \
     --p-trim-left-f <value> \
     --p-trim-left-r <value> \
     --p-trunc-len-f <value> \
     --p-trunc-len-r <value> \
     --o-representative-sequences rep-seqs-dada2.qza \
     --o-table dada2-table.qza \
     --p-n-threads <value> \
     --o-denoising-stats dada2-stats.qza
```

Summarize and visualize the feature table. Include the metadata file when you summarize.

```
[ ]: !qiime feature-table summarize \
     --i-table dada2-table.qza \
     --o-visualization dada2-table.qzv \
     --m-sample-metadata-file <your-sample-metadata.tsv>
```

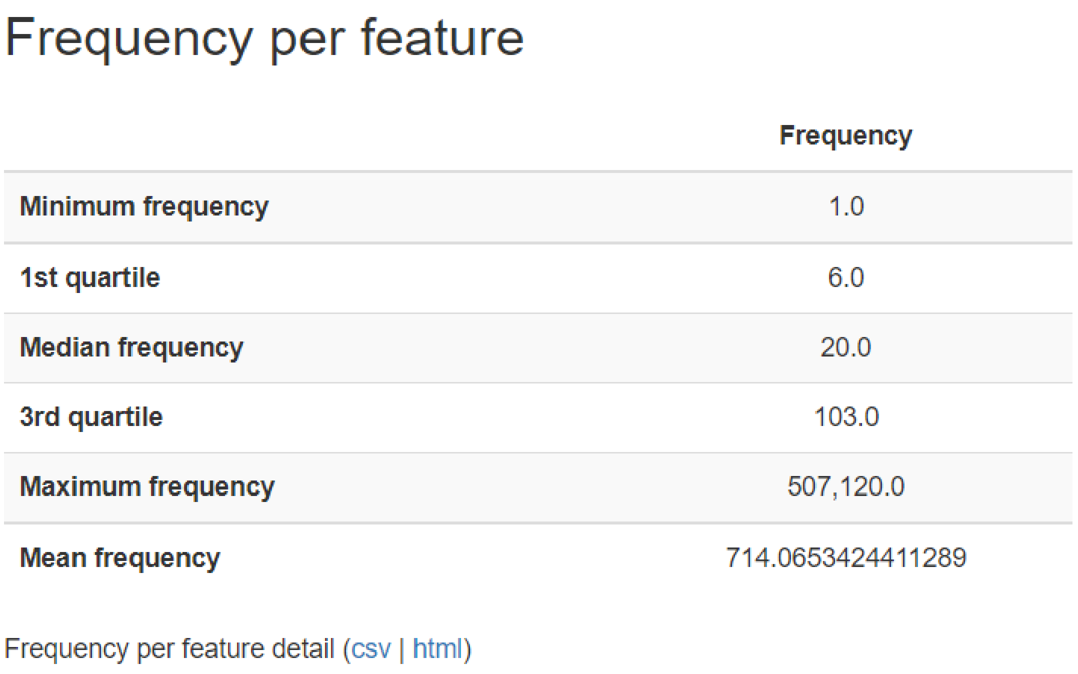
Visualize the representative sequences

```
[ ]: !qiime feature-table tabulate-seqs \
     --i-data rep-seqs-dada2.qza \
     --o-visualization rep-seqs-dada2.qzv
```

Typically, OTUs/ASVs with few reads are removed for quality purposes.
You can use the dada2 feature table produced above and the list of reads per samples, listed in the "demux-paired-end.qzv" from the previous section, as a guide to evaluate what threshold to use according to your data. Again, it is possible to save a csv of the visualization.

```
[ ]: from qiime2 import Visualization
```

```
[ ]: Visualization.load('dada2-table.qzv')
```


Frequency per feature

	Frequency
Minimum frequency	1.0
1st quartile	6.0
Median frequency	20.0
3rd quartile	103.0
Maximum frequency	507,120.0
Mean frequency	714.0653424411289

Frequency per feature detail (csv | html)

Figure 2: Example of a frequency per feature table, which is one of several components displayed in the dada2-table.qzv

Based on the information on your dataset, define a minimum frequency or number of reads per feature ID. This filtrates the dada2 feature table and saves it with the prefix "minreads".

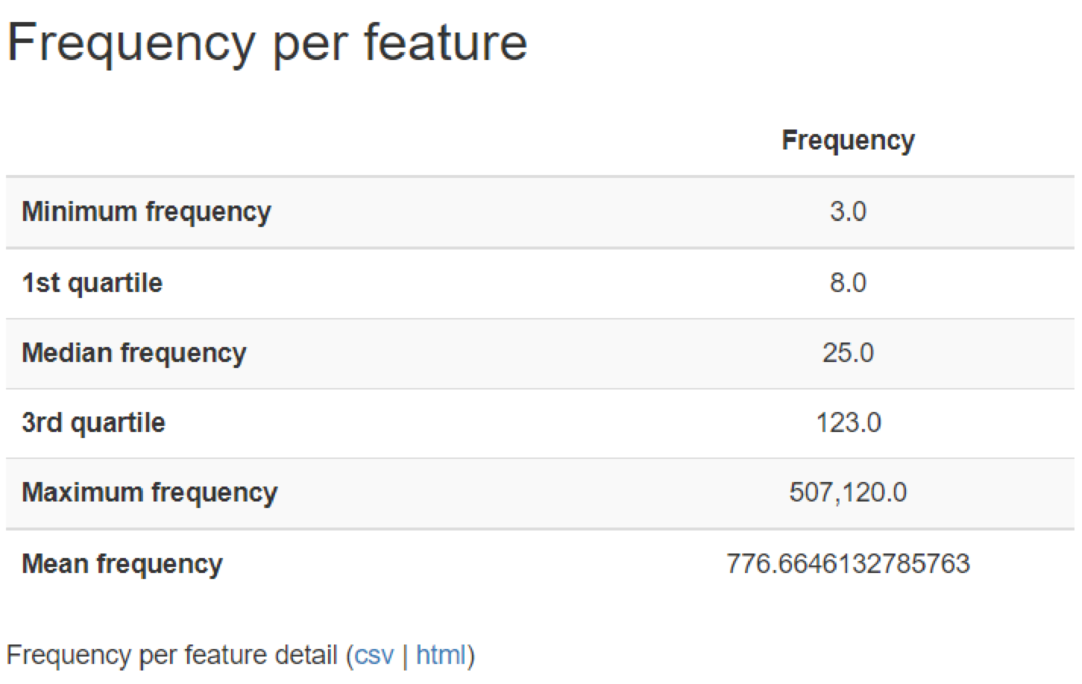
```
[ ]: !qiime feature-table filter-features \
     --i-table dada2-table.qza \
     --p-min-frequency <value> \
     --o-filtered-table minreads-dada2-table.qza
```

Below, data are summarized again and visualized to see how the filtration affected the feature table.

```
[ ]: !qiime feature-table summarize \
     --i-table minreads-dada2-table.qza \
     --o-visualization minreads-dada2-table.qzv \
     --m-sample-metadata-file sample-metadata.tsv
```

```
[ ]: from qiime2 import Visualization
```

```
[ ]: Visualization.load('minreads-dada2-table.qzv')
```



Frequency per feature

	Frequency
Minimum frequency	3.0
1st quartile	8.0
Median frequency	25.0
3rd quartile	123.0
Maximum frequency	507,120.0
Mean frequency	776.6646132785763

Frequency per feature detail (csv | html)

Figure 3: Table of frequences per feature after removing OTUs/ASVs with less than three reads

4.3 Taxonomy

Here we add taxonomic information to our representative sequences. This is based on a pretrained classifier, in this case one created from a Silva reference database. Use a suitable classifier based on the primers that you have used and the taxa of interest.

Define your path

```
[ ]: import os
     os.chdir(<your path>)
```

Start by validating your classifier

```
[ ]: !qiime tools validate silva-138-ssu-nr99-classifier.qza
```

Use your classifier to add taxonomic information to your representative sequences

- input your validated classifier
- input your representative sequences
- name the output file containing the classified sequences

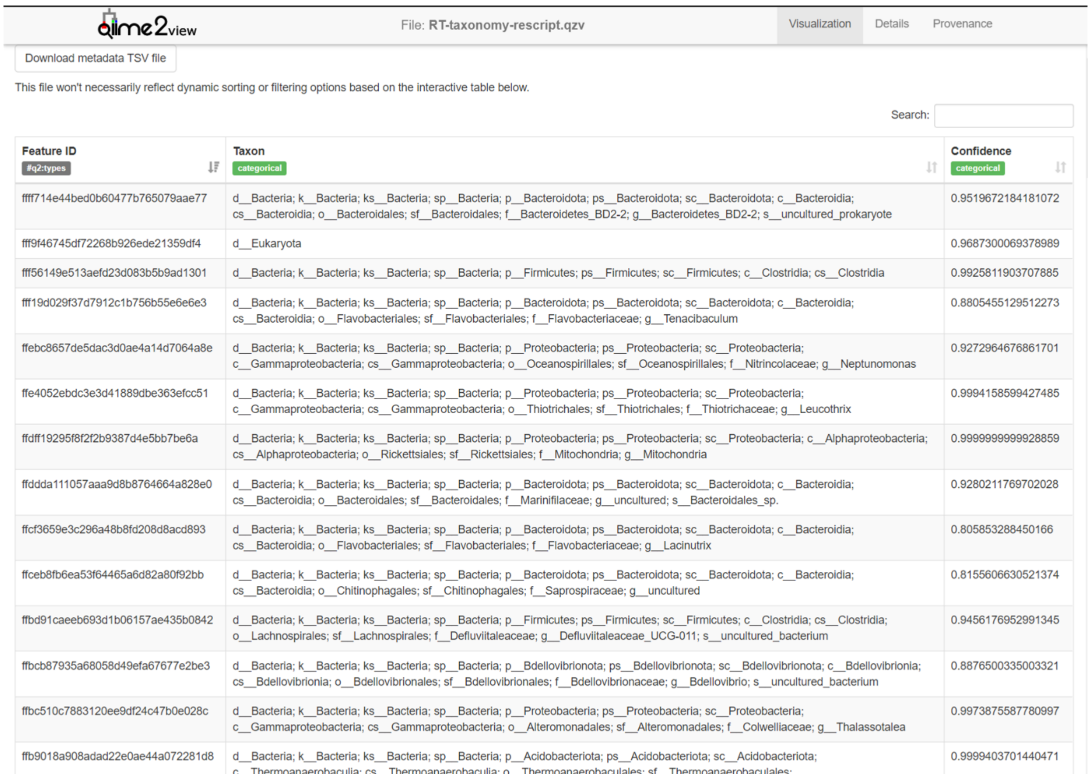
```
[ ]: !qiime feature-classifier classify-sklearn \
     --i-classifier silva-138-ssu-nr99-classifier.qza \
     --i-reads rep-seqs-dada2.qza \
     --o-classification <xx-taxonomy-rescript.qza>
```

Summarize and visualize the output from above

```
[ ]: !qiime metadata tabulate \
     --m-input-file <xx-taxonomy-rescript.qza> \
     --o-visualization <xx-taxonomy-rescript.qzv>
```

```
[ ]: from qiime2 import Visualization
```

```
[ ]: Visualization.load('<xx-taxonomy-rescript.qzv>')
```



File: RT-taxonomy-rescript.qzv

This file won't necessarily reflect dynamic sorting or filtering options based on the interactive table below.

Feature ID #q2:types	Taxon categorical	Confidence categorical
ffff714e44bed0b60477b765079aae77	d__Bacteria; k__Bacteria; ks__Bacteria; sp__Bacteria; p__Bacteroidota; ps__Bacteroidota; sc__Bacteroidota; c__Bacteroidia; cs__Bacteroidia; o__Bacteroidales; sf__Bacteroidales; f__Bacteroidetes_BD2-2; g__Bacteroidetes_BD2-2; s__uncultured_prokaryote	0.9519672184181072
fff9f46745df72268b926ede21359df4	d__Eukaryota	0.9687300069378989
fff56149e513aefd23d083b5b9ad1301	d__Bacteria; k__Bacteria; ks__Bacteria; sp__Bacteria; p__Firmicutes; ps__Firmicutes; sc__Firmicutes; c__Clostridia; cs__Clostridia	0.9925811903707885
fff19d029f37d7912c1b756b55e6e6e3	d__Bacteria; k__Bacteria; ks__Bacteria; sp__Bacteria; p__Bacteroidota; ps__Bacteroidota; sc__Bacteroidota; c__Bacteroidia; cs__Bacteroidia; o__Flavobacteriales; sf__Flavobacteriales; f__Flavobacteriaceae; g__Tenacibaculum	0.8805455129512273
ffebc8657de5dac3d0ae4a14d7064a8e	d__Bacteria; k__Bacteria; ks__Bacteria; sp__Bacteria; p__Proteobacteria; ps__Proteobacteria; sc__Proteobacteria; c__Gammaproteobacteria; cs__Gammaproteobacteria; o__Oceanospirillales; sf__Oceanospirillales; f__Nitrincolaceae; g__Neptunomonas	0.9272964676861701
ffe4052ebdc3e3d41889dbe363efcc51	d__Bacteria; k__Bacteria; ks__Bacteria; sp__Bacteria; p__Proteobacteria; ps__Proteobacteria; sc__Proteobacteria; c__Gammaproteobacteria; cs__Gammaproteobacteria; o__Thiotrichales; sf__Thiotrichales; f__Thiotrichaceae; g__Leucothrix	0.9994158599427485
ffdff19295f8f2f2b9387d4e5bb7be6a	d__Bacteria; k__Bacteria; ks__Bacteria; sp__Bacteria; p__Proteobacteria; ps__Proteobacteria; sc__Proteobacteria; c__Alphaproteobacteria; cs__Alphaproteobacteria; o__Rickettsiales; sf__Rickettsiales; f__Mitochondria; g__Mitochondria	0.9999999999928859
ffddda111057aaa9d8b8764664a828e0	d__Bacteria; k__Bacteria; ks__Bacteria; sp__Bacteria; p__Bacteroidota; ps__Bacteroidota; sc__Bacteroidota; c__Bacteroidia; cs__Bacteroidia; o__Bacteroidales; sf__Bacteroidales; f__Marinifilaceae; g__uncultured; s__Bacteroidales_sp.	0.9280211769702028
ffcf3659e3c296a48b8fd208d8acd893	d__Bacteria; k__Bacteria; ks__Bacteria; sp__Bacteria; p__Bacteroidota; ps__Bacteroidota; sc__Bacteroidota; c__Bacteroidia; cs__Bacteroidia; o__Flavobacteriales; sf__Flavobacteriales; f__Flavobacteriaceae; g__Lacinutrix	0.805853288450166
ffceb8fb6ea53f64465a6d82a80f92bb	d__Bacteria; k__Bacteria; ks__Bacteria; sp__Bacteria; p__Bacteroidota; ps__Bacteroidota; sc__Bacteroidota; c__Bacteroidia; cs__Bacteroidia; o__Chitinophagales; sf__Chitinophagales; f__Saprospiraceae; g__uncultured	0.8155606630521374
ffbd91caeeb693d1b06157ae435b0842	d__Bacteria; k__Bacteria; ks__Bacteria; sp__Bacteria; p__Firmicutes; ps__Firmicutes; sc__Firmicutes; c__Clostridia; cs__Clostridia; o__Lachnospirales; sf__Lachnospirales; f__Defluviitaleaceae; g__Defluviitaleaceae_UCG-011; s__uncultured_bacterium	0.9456176952991345
ffbcb87935a68058d49efa67677e2be3	d__Bacteria; k__Bacteria; ks__Bacteria; sp__Bacteria; p__Bdellovibrionota; ps__Bdellovibrionota; sc__Bdellovibrionota; c__Bdellovibrionia; cs__Bdellovibrionia; o__Bdellovibrionales; sf__Bdellovibrionales; f__Bdellovibrionaceae; g__Bdellovibrio; s__uncultured_bacterium	0.8876500335003321
ffbc510c7883120ee9df24c47b0e028c	d__Bacteria; k__Bacteria; ks__Bacteria; sp__Bacteria; p__Proteobacteria; ps__Proteobacteria; sc__Proteobacteria; c__Gammaproteobacteria; cs__Gammaproteobacteria; o__Alteromonadales; sf__Alteromonadales; f__Colwelliaceae; g__Thalassotalea	0.9973875587780997
ffb9018a908adad22e0ae44a072281d8	d__Bacteria; k__Bacteria; ks__Bacteria; sp__Bacteria; p__Acidobacteriota; ps__Acidobacteriota; sc__Acidobacteriota; c__Thermoanaerobaculia; cs__Thermoanaerobaculia; o__Thermoanaerobaculales; sf__Thermoanaerobaculales;	0.9999403701440471

Figure 4: Example of the visualisation of a table of classified representative sequences. The table is interactive and it is possible to download it as a TAB separated file.

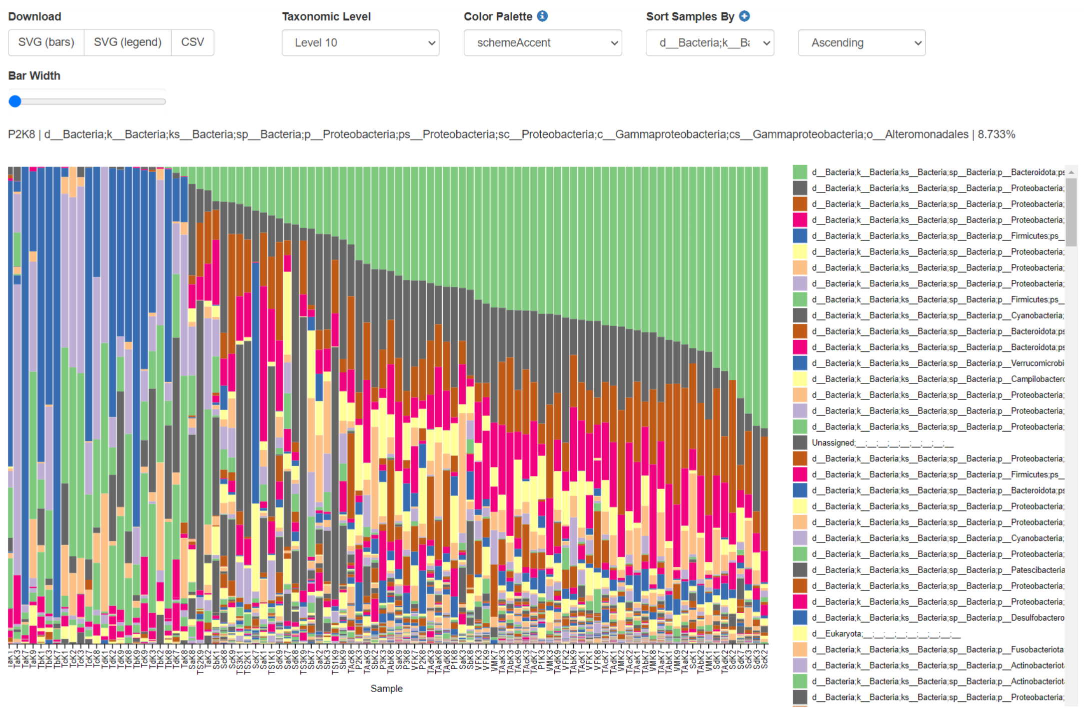
Using the taxonomy gained from the classifier and your dada2 table containing the reads, it is possible to create bar plots. These are interactive so it is possible to filter on metadata as well as looking at specific taxonomic levels. Constructing the barplot requires:

- your filtered dada2 table

- your taxonomy
- your metadata

[ ]:
```
!qiime taxa barplot \
--i-table minreads-dada2-table.qza \
--i-taxonomy <xx-taxonomy-rescript.qza> \
--m-metadata-file <your-sample-metadata.tsv> \
--o-visualization <xx-taxa-barplots.qzv>
```

[ ]:
```
Visualization.load('<xx-taxa-barplots.qzv>')
```



Download: SVG (bars) | SVG (legend) | CSV

Taxonomic Level: Level 10

Color Palette: schemeAccent

Sort Samples By: d__Bacteria;k__B… | Ascending

Bar Width

P2K8 | d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;ps__Proteobacteria;sc__Proteobacteria;c__Gammaproteobacteria;cs__Gammaproteobacteria;o__Alteromonadales | 8.733%

Sample
🟩 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Bacteroidota;ps
⬛ d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;
🟫 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;
🟥 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;
🟦 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Firmicutes;ps__
🟨 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;
🟧 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;
🟪 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;
🟩 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Firmicutes;ps__
⬛ d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Cyanobacteria;
🟫 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Bacteroidota;ps
🟥 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Bacteroidota;ps
🟦 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Verrucomicrobi
🟨 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Campilobacter
🟧 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;
🟪 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;
🟩 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;
⬛ Unassigned;__;__;__;__;__;__;__;__;__
🟫 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;
🟥 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Firmicutes;ps__
🟦 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Bacteroidota;ps
🟨 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;
🟧 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;
🟪 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Cyanobacteria;
🟩 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;
⬛ d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Patescibacteria
🟫 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;
🟥 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;
🟦 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Desulfobactero
🟨 d__Eukaryota;__;__;__;__;__;__;__;__;__
🟧 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Fusobacteriota
🟪 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Actinobacteriot
🟩 d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Actinobacteriot
⬛ d__Bacteria;k__Bacteria;ks__Bacteria;sp__Bacteria;p__Proteobacteria;

Figure 5: Example of a barplot with the taxonomic compositions in the samples

4.4 Filter data

Here are examples for filtering feature tables and subsequently rep-seqs. Tables are filtered using both including and excluding methods based on metadata. Taxonomy based filtering is demonstrated in the final cells.
Define your path

[ ]:
```
import os
os.chdir(<your path>)
```

Filter your reads based on sample metadata (include reads):

- use your filtered dada2 table
- use your metadata file
- filter based on column in your metadata with specific values (these rows are kept)
- name your output file

```
[ ]: !qiime feature-table filter-samples \
     --i-table minreads-dada2-table.qza \
     --m-metadata-file <your-sample-metadata.tsv> \
     --p-where "<[column name]='xx'>" \
     --o-filtered-table <xx-filtered-table.qza>
```

Filter your reads based on sample metadata (exclude reads):

- use your minreads table
- your metadata file
- state the the defined reads should be excluded
- define which reads to filter on (in this case all rows with A-E in "column name" should be excluded)
- name your output

```
[ ]: !qiime feature-table filter-samples \
     --i-table minreads-dada2-table.qza \
     --m-metadata-file <your-sample-metadata.csv> \
     --p-exclude-ids \
     --p-where "<[column name] IN ('A', 'B', 'C', 'D', 'E')>" \
     --o-filtered-table <xx-filtered-table.qza>
```

Summarize your filtered table and create a visualization.

```
[ ]: !qiime feature-table summarize \
     --i-table <xx-filtered-table.qza \
     --o-visualization <xx-filtered-table.qzv>
```

After filtrating your table, you can use this to filter your representative sequences:

- input your representative sequences
- input your filtered table
- name your output

```
[ ]: !qiime feature-table filter-seqs \
     --i-data rep-seqs-dada2.qza \
     --i-table <xx-filtered-table.qza> \
     --o-filtered-data <xx-filtered-rep-seqs.qza>
```

Rather than filtering on sample metadata, you can filter on taxonomy

- input your minreads table
- input your taxonomy file
- define what to exclude
- name your output

```
[ ]: !qiime taxa filter-table \
     --i-table <minreads-dada2-table.qza> \
     --i-taxonomy <xx-taxonomy-rescript.qza> \
     --p-exclude <TaxonA,TaxonB,TaxonC> \
     --o-filtered-table <xx-filtered-tax-table.qza>
```

You can then also filter your sequences based on the taxonomically filtered table:

- input your representative sequences
- input you taxonomy filtered table
- name your output

[ ]:
```
!qiime feature-table filter-seqs \
--i-data <rep-seqs-dada2.qza> \
--i-table <xx-filtered-tax-table.qza> \
--o-filtered-data <xx-filtered-tax-rep-seqs.qza>
```

4.5 Diversity

Here we create phylogenetix trees and calculate diversity core metrics for our data.
Define your path

[ ]:
```
import os
os.chdir(<your path>)
```

Align the representative sequences to build your tree. These may be filtered or not depending on your needs.

[ ]:
```
!qiime alignment mafft \
--i-sequences <xx-rep-seqs.qza> \
--o-alignment <xx-aligned.qza>
```

Mask the aligned sequences. This masks the sequences that are low complexity or otherwise suspected of being low quality.

[ ]:
```
!qiime alignment mask \
--i-alignment <xx-aligned.qza> \
--o-masked-alignment <xx-masked-aligned.qza>
```

Make an unrooted phylogenetic tree based on your aligned and masked data

[ ]:
```
!qiime phylogeny fasttree \
--i-alignment <xx-masked-aligned.qza> \
--o-tree <xx-unrooted-tree.qza>
```

Make a rooted tree from the unrooted tree

[ ]:
```
!qiime phylogeny midpoint-root \
--i-tree <xx-unrooted-tree.qza> \
--o-rooted-tree <xx-rooted-tree.qza>
```

In order to calculate diversity metrics, we will use a dada2 feature table. This one can be filtrated and should probably at least be your minreads table.
The number reads per sample shown in your table can be used to define max sequencing depth in the alpha rarefaction plot below.

[ ]:
```
from qiime2 import Visualization
```

[ ]:
```
Visualization.load('<xx-table.qzv>')
```

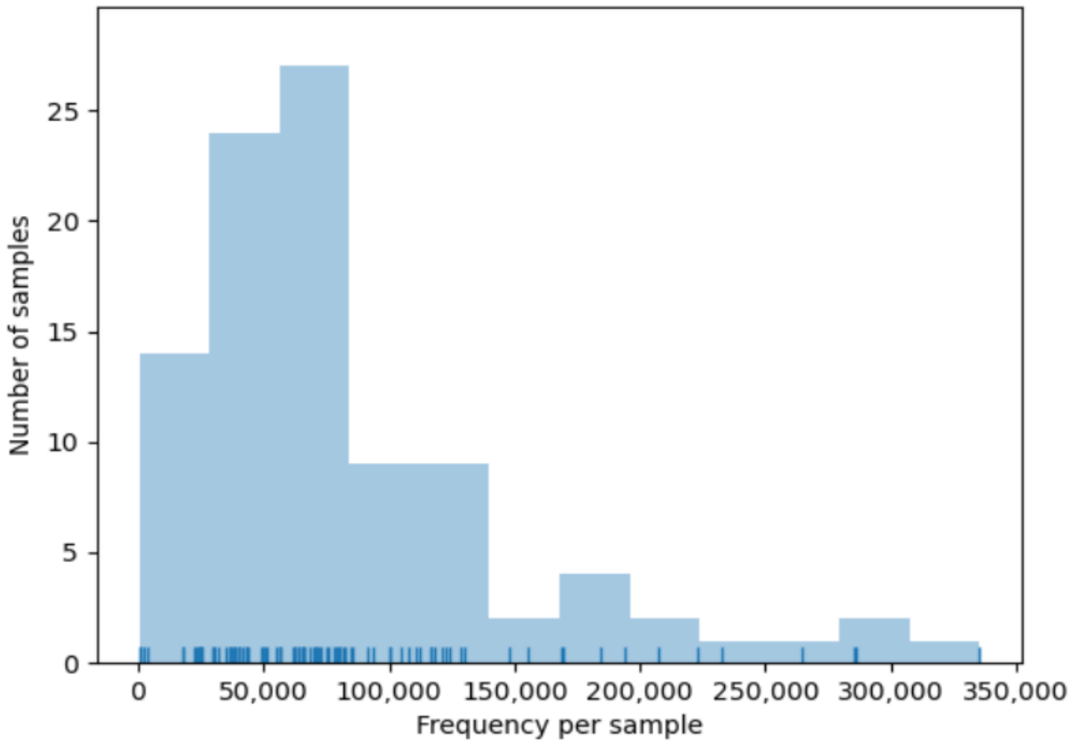


Figure 6: Frequency histogram of reads per sample

To perform alpha rarefaction including phylogenetic data, choose the table of interest, rooted tree, and sample metadata. This can also be done without the phylogenetic data.

- input your table
- input your rooted tree
- the max sequencing depth of your choice
- your sample metadata file
- name the output

[ ]:
```
!qiime diversity alpha-rarefaction \
--i-table <xx-table.qza> \
--i-phylogeny <xx-rooted-tree.qza> \
--p-max-depth <your value> \
--m-metadata-file <your-sample-metadata.tsv> \
--o-visualization <xx-alpha-rarefaction.qzv>
```

[ ]:
```
from qiime2 import Visualization
```

[ ]:
```
Visualization.load('<xx-alpha-rarefaction.qzv>')
```

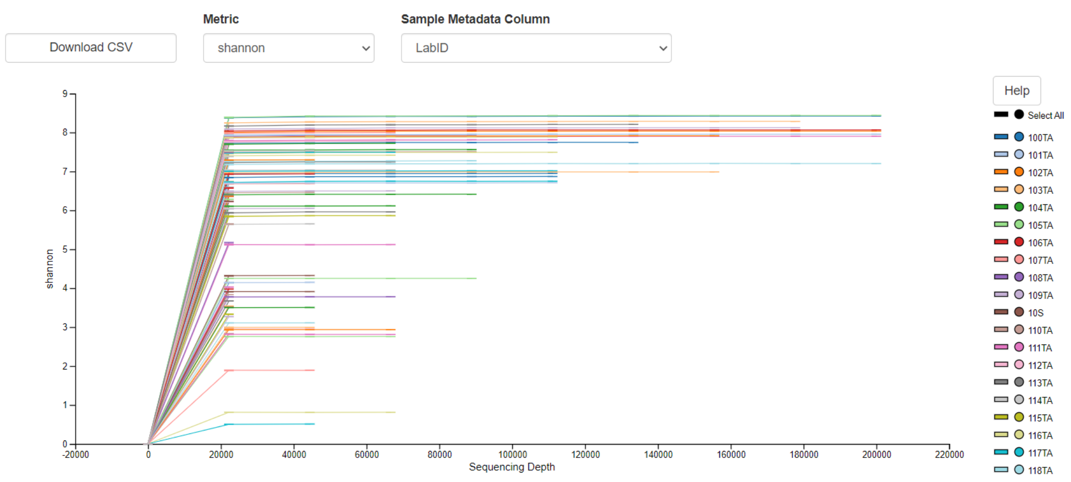


Figure 7: Example of interactive rarefaction curves. These can be used to estimate whether the sequencing depth is sufficient. Also, the sampling depth used for calculating core metrics can be defined based on these plots.

Compute core diversity metrics. The standard output includes:

- Alpha diversity
  - Faith pd
  - Observed features
  - Shannon
  - Evenness
- Distance matrices and PCoA results with emperor plots
  - Unweighted UniFrac
  - Weighted UniFrac
  - Jaccard
  - Bray-Curtis

Use the rarefaction curves to determine a sampling depth.

- input your rooted tree
- input your table
- define sampling depth
- input your sample metadata
- name your output directory

```
[ ]: !qiime diversity core-metrics-phylogenetic \
     --i-phylogeny <xx-rooted-tree.qza> \
     --i-table <xx-table.qza> \
     --p-sampling-depth <your value> \
     --m-metadata-file <your-sample-metadata.tsv> \
     --output-dir <xx-core-metrics-results>
```

```
[ ]: from qiime2 import Visualization
```

The emperor plots are interactive, allowing you to turn the axes and also change shape, size, and colour.

```
[ ]: Visualization.load('xx-core-metrics-results/unweighted_unifrac_emperor.
      ↪qzv')
```

```
[ ]: Visualization.load('xx-core-metrics-results/weighted_unifrac_emperor.
      ↪qzv')
```

```
[ ]: Visualization.load('xx-core-metrics-results/jaccard_emperor.qzv')
```

```
[ ]: Visualization.load('xx-core-metrics-results/bray_curtis_emperor.qzv')
```

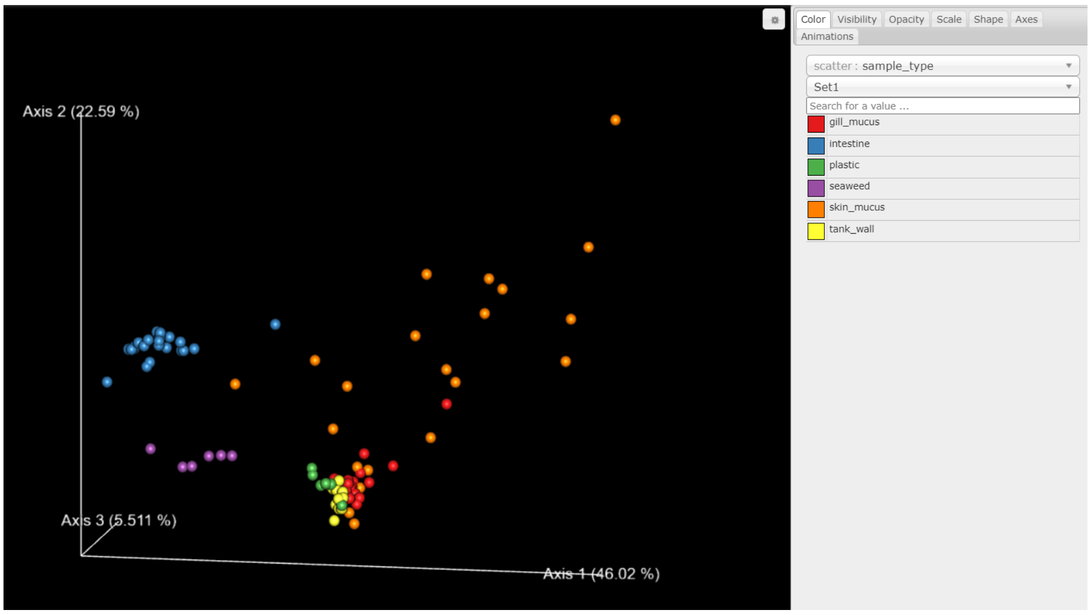


Figure 8: Example of visualisation of an interactive emperor plot

4.6 Statistics

Here we carry out a selection of statistical analysis. The first analysis carried out is Adonis, which is based on distance matrices calculated in the "Diversity" section.

- input your distance matrix of choice
- input your sample metadata file
- the predictor of choice (from your sample metadata file)
- name the output an save it in the directory of your choice

```
[ ]: !qiime diversity adonis \
     --i-distance-matrix <xx-core-metrics-results/xx_distance_matrix.qza> \
     --m-metadata-file <your-sample-metadata.tsv> \
     --p-formula <"Parameter xx"> \
     --o-visualization <xx-core-metrics-results/xx-adonis.qzv>
```

Visualise and download Adonis results. These results are a statistical output table.

```
[ ]: from qiime2 import Visualization
[ ]: Visualization.load('xx-core-metrics-results/xx-adonis.qzv')
```

Here you can run a permanova for Beta group significance.

- input your distance matrix of choice
- input your sample metadata file
- the metadata column (predictor variable)
- the method, in this case permanova
- pairwise, in this case TRUE
- name your output

```
[ ]: !qiime diversity beta-group-significance \
     --i-distance-matrix <xx-core-metrics-results/xx_distance_matrix.qza> \
     --m-metadata-file <your-sample-metadata.tsv> \
     --m-metadata-column <"Parameter xx"> \
     --p-method "permanova" \
     --p-pairwise True \
     --o-visualization <xx-core-metrics-results/xx-betapair.qzv>
```

Visualise and download permanova results table.

```
[ ]: from qiime2 import Visualization
```

```
[ ]: Visualization.load(<'xx-core-metrics-results/xx-betapair.qzv'>)
```

5 R Code

Data is imported into R for easier data manipulation and custom figure creation. The described code used in R demonstrates importing the data and performing alpha diversity statistics and plots. The data are also prepared for filtering and making bar plots, phylogenetic trees, and PCoA plots.

In R, data are stored in objects using "<-" (or alternatively "=", but we use "<-"). Therefore, when a variable that you need to name is placed in front of a "<-", this means that you are naming the output variable of the function that follows.

You will also see the symbol "%>%", which is called a pipe. This "pushes" the data before the pipe to the function after the pipe. Chaining pipes allows you to carry out several functions with your data before storing it in an object.

Some code chunks contain lines of code that are commented out (using #). These lines are optional and often need to be used after the rest of the code has already been run (for example rearranging the levels of a factor for post-hoc analysis).

5.1 Importing to R

This section is for opening packages and importing data.
The packages below are the ones used in the notebooks described in this project.

```
library(tidyverse)
library(qiime2R)
library(metacoder)
```

Import your sample metadata.

```
metadata <- read_tsv("<your directory/sample-metadata.tsv>")
```

Import your taxonomy table, rooted tree, and feature table of choice.

```
<xx_taxonomy> <-read_qza("<your directory/xx-taxonomy-rescript.qza>")$data
<xx_rooted_tree> <-read_qza("<your directory/xx-rooted-tree.qza>")$data
<xx_table> <-read_qza("<your directory/xx-table.qza>")$data
```

Import the relevant Alpha diversity data for your needs. Name variable as needed and path as needed. The code creates the column “SampleID” from the row names in the data.

```
<xx_shannon> <-read_qza(
    "<your directory/xx-core-metrics-results/shannon_vector.qza>")$data %>%
        rownames_to_column("SampleID")
```

Import the relevant Beta diversity data for your needs. Name variable as needed and path as needed.

```
<xx_wunifrac> <-read_qza(
"<your directory/xx-core-metrics-results/weighted_unifrac_pcoa_results.qza>")
```

5.2 Alpha diversity statistics and plots

This section is for preparing plots and calculating statistics for alpha-diversity.

Join alpha diversity with metadata using the sample ID column from both data frames. When joining, first define the metadata column that will define which rows to take from the left joined data.

```
<xx_alpha_data> <-metadata %>%
  left_join(xx_evenness, by=c("<sample ID column"="sample ID column>")) %>%
  left_join(xx_shannon, by=c("<sample ID column"="sample ID column>")) %>%
  left_join(xx_observed_otus, by=c("<sample ID column"="sample ID column>")) %>%
  left_join(xx_faith, by=c("<sample ID column"="sample ID column>"))
```

Use the following chunk as an example for filtering data if necessary. There are two examples of how to filter in dplyr, adjust as needed.

- The rows with "X" and "Y" in "Your parameter" column are kept
- The rows with "your value" in the "Your parameter2" column are kept

```
<xx_alpha_data_f> <-<xx_alpha_data> %>%
  filter(<your parameter> %in% c(<"X", "Y">)) %>%
  filter(<your parameter2> == <"your value">)
```

Below is code used for creating alpha diversity plots. Parameters are defined in the first line. Here Shannon is used as an example. Choose your own predictors.

```
ggplot(<xx_alpha_data>,aes(x=<Parameter1>, y=<xx_shannon>, col=<Parameter2>))+
  stat_summary(geom="errorbar", fun.data=mean_se, width=0) +
  stat_summary(geom="line", fun.data=mean_se) +
  stat_summary(geom="point", fun.data=mean_se) +
  xlab("<Parameter1>") +
  ylab("<Shannon>") +
  theme_q2r() +
  scale_color_viridis_d(name="<Parameter2>")
  ggsave("<name_of_choice>.jpg", height=3, width=4, device="jpg")
```

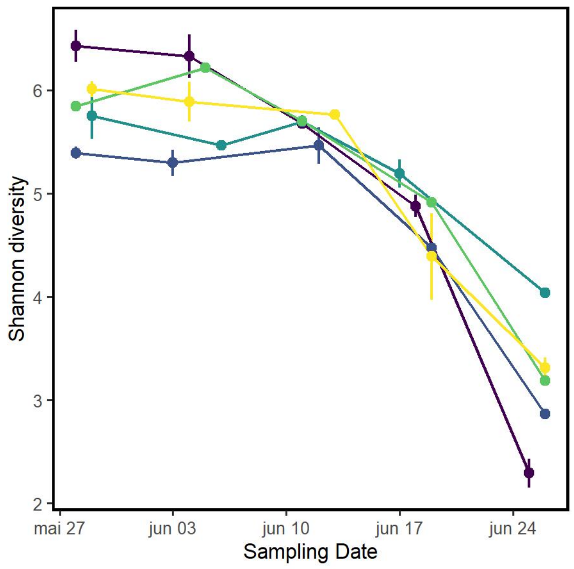


Figure 9: Example of a Shannon diversity plot

You can use an interaction model to check whether there were interacting effects between Parameters. If a significant effect is found, releveling can be used to determine which levels in a Parameter differ.

```
#<xx_alpha_data$Parameter2> <-relevel(
    factor(<xx_alpha_data$Parameter2>),ref="<value>")
<xx_shannonmod_int> <-lm(
    <xx_shannon>~<Parameter2>*<Parameter1>, data=<xx-alpha-data>)
summary(<xx_shannonmod_int>)
anova(<xx_shannonmod_int>)
plot(<xx_shannonmod_int>)
```

You can use an additive model to determine whether any of the parameters affect diversity. Again, releveling can be used for the post-hoc differences.

```
#<xx_alpha_data$Parameter2> <-relevel(
    factor(<xx_alpha_data$Parameter2>),ref="<value>")
<xx_shannonmod_add> <-lm(
    <xx_shannon>~<Parameter2+Parameter1>,data=<xx_alpha_data>)
summary(<xx_shannonmod_add>)
anova(<xx_shannonmod_add>)
plot(<xx_shannonmod_add>)
```

5.3 Taxonomic compositions

In this section we prepare the data for creating taxonomic bar charts and phylogenetic trees.
First we make a taxmap object for easy filtering. The data table readable by "parse_tax_data" is built. Then the taxmap environment is created.

```
<totaxmap_xx_rescript> <- <xx_table> %>%
  as.data.frame() %>%
  rownames_to_column("Feature.ID") %>%
  left_join(<xx_taxonomy>) %>%
  mutate(Feature=Taxon)


<xx_rescript_parsed> <-parse_tax_data(<totaxmap_xx_rescript>,
    class_cols="Taxon",
    class_sep=";",
    class_key=c(tax_rank="taxon_rank",
    tax_name="taxon_name"),
    class_regex = "^(.+)__(.+)")
```

Summarizing data for frequency barplot. The first line calculates total abundance of each taxon. The following lines allow you to sum the counts across your chosen variables.

```
<xx_rescript_parsed>$data$taxon_counts <- calc_taxon_abund(
    <xx_rescript_parsed>,
    data = "tax_data")
<xx_rescript_parsed>$data$taxon_counts$total <- rowSums(
    <xx_rescript_parsed>$data$taxon_counts[<column_no:column_no>])
<xx_rescript_parsed>$data$taxon_counts$<totalA> <- rowSums(
    <xx_rescript_parsed>$data$taxon_counts[grep("<A>",
    names(<xx_rescript_parsed>$data$taxon_counts))])
<xx_rescript_parsed>$data$taxon_counts$<totalB> <- rowSums(
    <xx_rescript_parsed>$data$taxon_counts[grep("<B>",
    names(<xx_rescript_parsed>$data$taxon_counts))])
```

Extract the taxon counts from the taxmap environment

```
taxon_counts<- <xx_rescript_parsed>$data$taxon_counts
```

The taxmap environment is large and complex. Sometimes it can be helpful to print parts of it to see the data.

```
print(<xx_rescript_parsed>$data$tax_data)
```

Your taxmap object can be taxonomically filtered based on taxon name or taxon rank.

- "subtaxa" is used to define whether subtaxa should be included in your filter.
- "invert" defines whether your filter includes or excludes taxa. TRUE means that you exclude what the filter finds.
- "supertaxa" defines whether to include or exclude supertaxa in your filtering.

```
<filtered_xx> <- <xx_rescript_parsed> %>%
  filter_taxa(taxon_names == "<Animalia>", subtaxa = TRUE, invert=TRUE) %>%
  filter_taxa(taxon_ranks == "<p>", supertaxa=FALSE)
```

Look at your filtered data.

```
print(<filtered_xx>$data$class_data)
```

Prepare data for bar chart:

- First join the taxon counts with the class data.
- Filter to your desired taxonomic level. Here, we use phyla.
- Ensure the taxon ids are unique.
- Calculate frequencies based on the summarised data.
- Finally use gather to make your data long with the chosen metadata in a column.
- In the gather function, define the name of your metadata column, the name of your values column, and which columns the data are in.

```
counts<- <filtered_xx>$data$taxon_counts %>%
  left_join(<filtered_xx>$data$class_data, by=c("taxon_id"="taxon_id")) %>%
  filter(tax_rank=="<p>") %>%
  distinct(taxon_id, .keep_all = TRUE) %>%
  mutate(<sumA>=sum(<totalA>)) %>%
  mutate(<sumB>=sum(<totalB>)) %>%
  group_by(tax_name) %>%
  mutate(<freqA>=<totalA/sumA>) %>%
  mutate(<freqB>=<totalB/sumB>) %>%
  gather(<A and B parameter>, <frequency>, <column no:column no>)
```

Pool the smallest taxa so decrease the number of taxa in the plot. Set your own pooling threshold.

```
xxpool<-counts %>%
  group_by(tax_name) %>%
  summarise(pool=max(frequency)< <value>,
            mean=mean(frequency),
            .groups="drop")
```

Make a stacked frequency bar chart.

- First join your counts with your pooled taxa. Then for taxa that have been assigned to "pool", rename them to "Other".
- Summarise your frequencies by tax name and any other relevant variable.
- Make the tax name variable a factor and sort by mean frequency.
- Make your plot.

```
<xx_barchart> <- inner_join(counts,<xxpool>,by="tax_name") %>%
  mutate(tax_name=if_else(pool,"Other",tax_name)) %>%
  group_by(<variable>,tax_name) %>%
  summarise(frequency=sum(frequency),
            mean = min(mean),
            .groups = "drop") %>%
  mutate(tax_name=factor(tax_name),
         tax_name=fct_reorder(tax_name,mean,.desc = FALSE)) %>%
  ggplot(aes(x=<variable>, y=frequency, fill=tax_name))+
  geom_col()+
  scale_fill_brewer(palette = "Paired") +
  theme_classic()+
  labs(x="<Variable>",y="Proportion",fill="<your tax rank>") +
  scale_x_discrete(labels = c('<A>','<B>'))
  #ggsave("<xx_bar>.tiff",dpi=300)
```

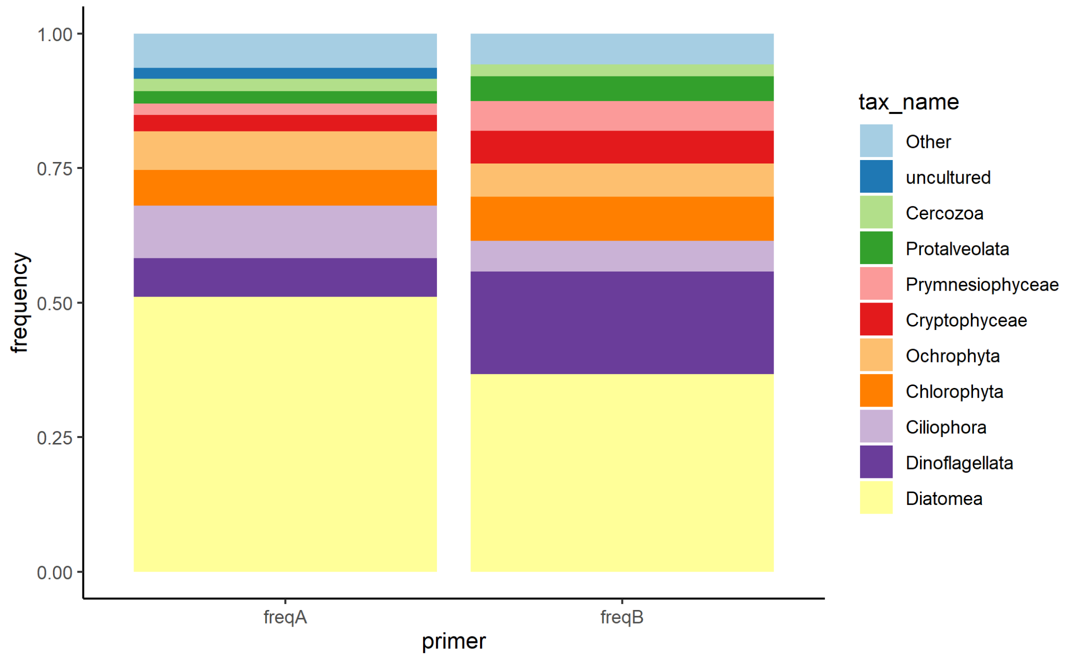


Figure 10: Example of a stacked frequency bar chart showing relative abundance at the taxonomic level chosen

5.4 Phylogenetic trees

The trees are built from taxmap objects. As the previous section ("Taxonomic compositions") describes how to make such an object, this is skipped here. In addition to creating the taxmap object, summarised abundances are used to define the size and colour of tree nodes, and again, this is previously described.

Checking tables

```
print(<xx_rescript_parsed>$data$taxon_counts)
```

Here we build a subtaxa tree. Empty nodes and specific taxa ranks are removed in order to make the tree readable and informative. Insert your summarized variable of choice.

The filter_taxa line that defines the taxon ranks to exclude uses taxon rank data from the reference database. These vary, so the ranks available or how they are denoted may not be the same.

```
subtaxa_tree <- <xx_rescript_parsed> %>%
  filter_taxa(taxon_names %in% c("<subtaxa of choice>"),
              subtaxa=TRUE) %>%
  filter_taxa(<totalx> >0) %>%
  filter_taxa(! taxon_ranks %in% c("d","k","ks","ps","p","c","o","sf","cs")) %>%
  heat_tree(node_label=taxon_names,
            node_size=<totalx>,
            node_color=<totalx>,
            tree_label="<your label>",
            node_label_size_range=c(0.015,0.04), #adjust size range as needed
            node_size_range = c(0.007, 0.02), #adjust size range as needed
            node_color_axis_label="Reads",
            output_file="<subtaxa tree.tiff>")
```

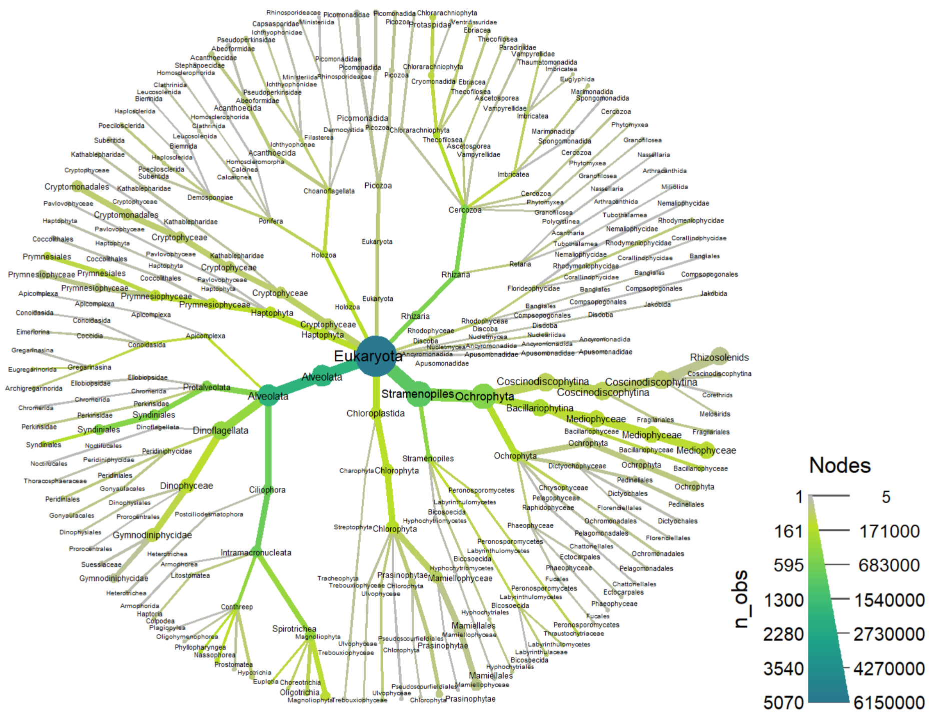


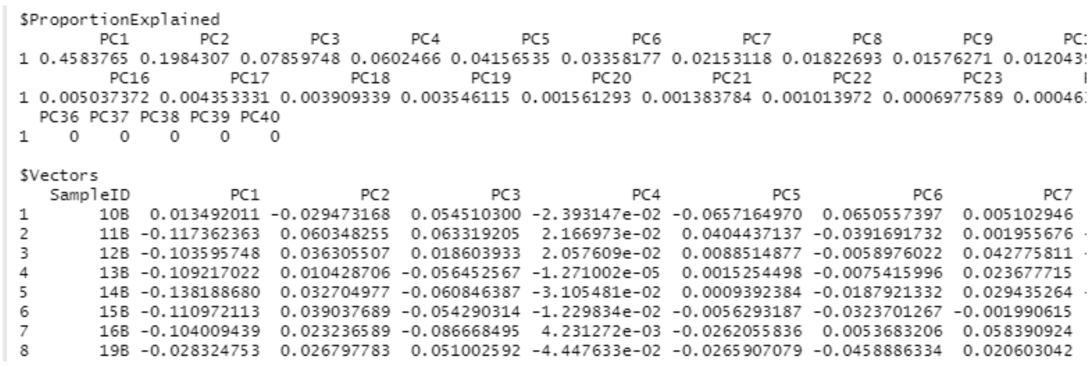
Figure 11: Example of a phylogenetic heat tree

5.5 PCoA plots

In this section, we build beta diversity PCoA plots using the core metrics calculated in Qiime2.

Make any character vectors in your metadata into factors for plotting purposes.

```
metadata$<variable1> <-factor(metadata$<variable1>)
metadata$<variable2> <-factor(metadata$<variable2>)
metadata$<variable3> <-factor(metadata$<variable3>)
```



```
$ProportionExplained
        PC1       PC2        PC3       PC4        PC5        PC6        PC7        PC8        PC9        PC:
1 0.4583765 0.1984307 0.07859748 0.0602466 0.04156535 0.03358177 0.02153118 0.01822693 0.01576271 0.012043
          PC16        PC17        PC18        PC19        PC20        PC21        PC22         PC23         
1 0.005037372 0.004353331 0.003909339 0.003546115 0.001561293 0.001383784 0.001013972 0.0006977589 0.00046
  PC36 PC37 PC38 PC39 PC40
1    0    0    0    0    0

$Vectors
   SampleID          PC1          PC2          PC3           PC4           PC5           PC6          PC7
1       10B  0.013492011 -0.029473168  0.054510300 -2.393147e-02 -0.0657164970  0.0650557397  0.005102946
2       11B -0.117362363  0.060348255  0.063319205  2.166973e-02  0.0404437137 -0.0391691732  0.001955676
3       12B -0.103595748  0.036305507  0.018603933  2.057609e-02  0.0088514877 -0.0058976022  0.042775811
4       13B -0.109217022  0.010428706 -0.056452567 -1.271002e-05  0.0015254498 -0.0075415996  0.023677715
5       14B -0.138188680  0.032704977 -0.060846387 -3.105481e-02  0.0009392384 -0.0187921332  0.029435264
6       15B -0.110972113  0.039037689 -0.054290314 -1.229834e-02 -0.0056293187 -0.0323701267 -0.001990615
7       16B -0.104009439  0.023236589 -0.086668495  4.231272e-03 -0.0262055836  0.0053683206  0.058390924
8       19B -0.028324753  0.026797783  0.051002592 -4.447633e-02 -0.0265907079 -0.0458886334  0.020603042
```

Figure 12: Example of Weighted UniFrac core metric results

Here we plot the Weighted UniFrac metric as an example in the PC-plots.

- Choose metadata variables for size and color differentiation.
- Choose the beta diversity metric and the principal components you want to plot.
- Facet wrap by selected variable to combine plots.

```
<xx_wunifrac>$data$Vectors %>%
  select(SampleID, PC1, PC2, PC3) %>%
  left_join(metadata, by=c("SampleID"="<metadata sample ID>")) %>%
  ggplot(aes(x=PC1, y=PC2, color=<variable1>, size=<variable2>)) +
  geom_point(alpha=0.5) +
  theme_q2r() +
  scale_size_discrete(name="<variable2>") +
  scale_color_viridis_d(name="<variable1>") +
  facet_wrap(.~<variable3>)
#ggsave("PCoA_xx_wu.jpg", height=4.5, width=5.5, device="jpg")
```

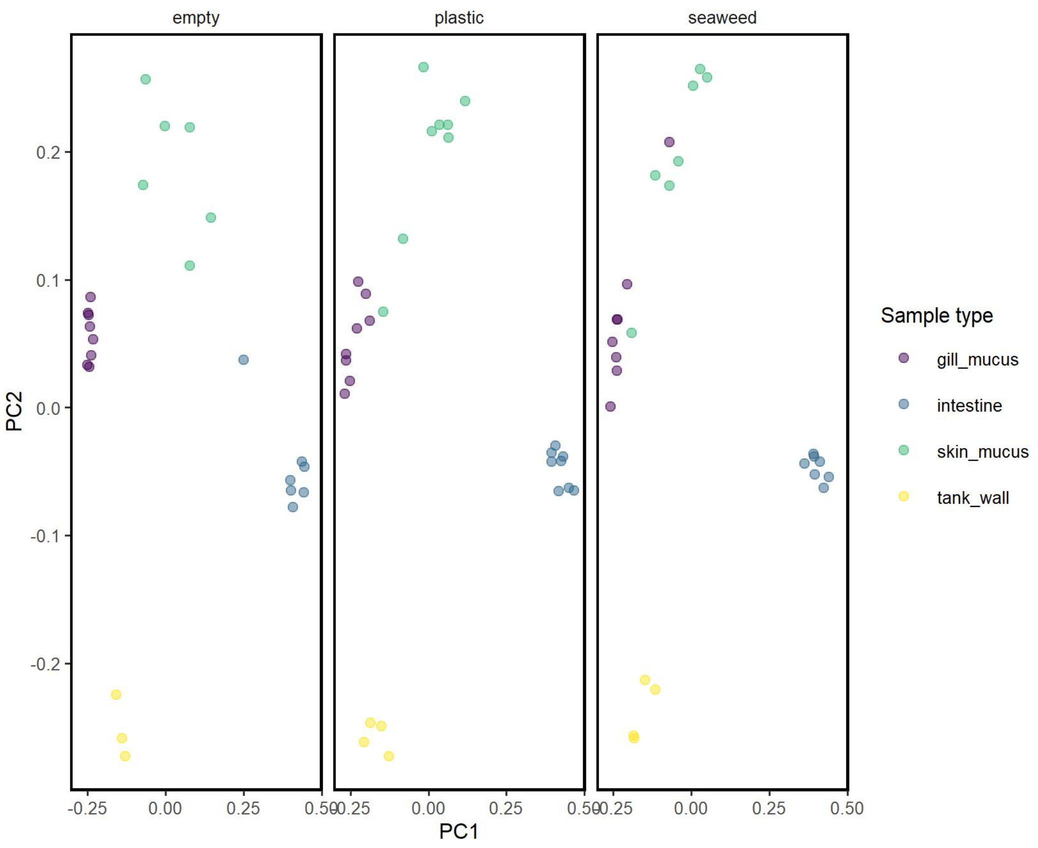


Figure 13: Example of PCoA plot, which has been faceted by a variable in the metadata

References

Bisanz, Jordan E. 2018. “qiime2R: Importing QIIME2 artifacts and associated data into R sessions.”

Bokulich, Nicholas A., Benjamin D. Kaehler, Jai Ram Rideout, Matthew Dillon, Evan Bolyen, Rob Knight, Gavin A. Huttley, and J. Gregory Caporaso. 2018. “Optimizing taxonomic classification of marker-gene amplicon sequences with QIIME 2’s q2-feature-classifier plugin.” *Microbiome* 6 (1): 90. https://doi.org/10.1186/s40168-018-0470-z.

Bolyen, Evan, Jai Ram Rideout, Matthew R. Dillon, Nicholas A. Bokulich, Christian C. Abnet, Gabriel A. Al-Ghalith, Harriet Alexander, et al. 2019. “Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2.” *Nature Biotechnology* 37 (8): 852–857. ISSN: 1546-1696. https://doi.org/10.1038/s41587-019-0209-9.

Callahan, Benjamin J, Paul J McMurdie, Michael J Rosen, Andrew W Han, Amy Jo A Johnson, and Susan P Holmes. 2016. “DADA2: high-resolution sample inference from Illumina amplicon data.” Publisher: Nature Publishing Group, *Nature methods* 13 (7): 581. https://doi.org/10.1038/nmeth.3869.

Faith, Daniel P. 1992. “Conservation evaluation and phylogenetic diversity.” Publisher: Elsevier, *Biological conservation* 61 (1): 1–10. https://doi.org/10.1016/0006-3207(92)91201-3.

Foster, Zachary, Thomas Sharpton, and Niklaus Grünwald. 2017. “Metacoder: An R package for visualization and manipulation of community taxonomic diversity data.” Publisher: Public Library of Science, *PLOS Computational Biology* 13, no. 2 (February): 1–15. https://doi.org/10.1371/journal.pcbi.1005404.

Hamady, Micah, Catherine Lozupone, and Rob Knight. 2010. “Fast UniFrac: facilitating high-throughput phylogenetic analyses of microbial communities including analysis of pyrosequencing and PhyloChip data.” *The ISME Journal* 4 (1): 17–27. ISSN: 1751-7370. https://doi.org/10.1038/ismej.2009.97.

Jaccard, P. 1908. “Nouvelles recherches sur la distribution floral.” *Bull. Soc. Vard. Sci. Nat* 44:223–270.

Johnson, Mark, Irena Zaretskaya, Yan Raytselis, Yuri Merezhuk, Scott McGinnis, and Thomas L. Madden. 2008. “NCBI BLAST: a better web interface.” *Nucleic Acids Research* 36 (suppl_2): W5–W9. https://doi.org/10.1093/nar/gkn201.

Katoh, Kazutaka, and Daron M Standley. 2013. “MAFFT multiple sequence alignment software version 7: improvements in performance and usability.” Publisher: Society for Molecular Biology and Evolution, *Molecular biology and evolution* 30 (4): 772–780. https://doi.org/10.1093/molbev/mst010.

Lozupone, Catherine, and Rob Knight. 2005. “UniFrac: a New Phylogenetic Method for Comparing Microbial Communities.” Publisher: American Society for Microbiology Journals _eprint: https://aem.asm.org/content/71/12/8228.full.pdf, *Applied and Environmental Microbiology* 71 (12): 8228–8235. ISSN: 0099-2240. https://doi.org/10.1128/AEM.71.12.8228-8235.2005.

Lozupone, Catherine, Manuel E Lladser, Dan Knights, Jesse Stombaugh, and Rob Knight. 2011. “UniFrac: an effective distance metric for microbial community comparison.” *The ISME Journal* 5 (2): 169–172. ISSN: 1751-7370. https://doi.org/10.1038/ismej.2010.133.

McDonald, Daniel, Jose C Clemente, Justin Kuczynski, Jai Ram Rideout, Jesse Stombaugh, Doug Wendel, Andreas Wilke, et al. 2012. “The Biological Observation Matrix (BIOM) format or: how I learned to stop worrying and love the ome-ome.” Publisher: BioMed Central, *GigaScience* 1 (1): 7. https://doi.org/10.1186/2047-217X-1-7.

McDonald, Daniel, Yoshiki Vázquez-Baeza, David Koslicki, Jason McClelland, Nicolai Reeve, Zhenjiang Xu, Antonio Gonzalez, and Rob Knight. 2018. “Striped UniFrac: enabling microbiome analysis at unprecedented scale.” *Nature Methods* 15 (11): 847–848. ISSN: 1548-7105. https://doi.org/10.1038/s41592-018-0187-8.

NCBI Resource Coordinators. 2017. “Database Resources of the National Center for Biotechnology Information.” *Nucleic Acids Research* 45 (D1): D12–D17. https://doi.org/10.1093/nar/gkw1071.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. “Scikit-learn: Machine learning in Python.” *Journal of machine learning research* 12 (Oct): 2825–2830.

Pielou, Evelyn C. 1966. “The measurement of diversity in different types of biological collections.” Publisher: Elsevier, *Journal of theoretical biology* 13:131–144. https://doi.org/10.1016/0022-5193(66)90013-0.

Price, Morgan N, Paramvir S Dehal, and Adam P Arkin. 2010. “FastTree 2–approximately maximum-likelihood trees for large alignments.” Publisher: Public Library of Science, *PloS one* 5 (3): e9490. https://doi.org/10.1371/journal.pone.0009490.

Robeson, Michael S, Devon R O’Rourke, Benjamin D Kaehler, Michal Ziemski, Matthew R Dillon, Jeffrey T Foster, and Nicholas A Bokulich. 2020. “RESCRIPt: Reproducible sequence taxonomy reference database management for the masses.” Publisher: Cold Spring Harbor Laboratory _eprint: https://www.biorxiv.org/content/early/2020/10/05/2020.10.05.326504.full.pdf, *bioRxiv,* https://doi.org/10.1101/2020.10.05.326504.

Shannon, Claude E. 1948. “A mathematical theory of communication.” Publisher: Nokia Bell Labs, *The Bell System Technical Journal* 27 (3): 379–423, 623–656. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.

Sørensen, Thorvald. 1948. “A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons.” *Biol. Skr.* 5:1–34.

Van Rossum, Guido, and Fred L Drake Jr. 1995. *Python reference manual.* Centrum voor Wiskunde en Informatica Amsterdam.

Vázquez-Baeza, Yoshiki, Antonio Gonzalez, Larry Smarr, Daniel McDonald, James T Morton, Jose A Navas-Molina, and Rob Knight. 2017. “Bringing the dynamic microbiome to life with animations.” Publisher: Elsevier, *Cell host & microbe* 21 (1): 7–10. https://doi.org/10.1016/j.chom.2016.12.009.

Vázquez-Baeza, Yoshiki, Meg Pirrung, Antonio Gonzalez, and Rob Knight. 2013. “EMPeror: a tool for visualizing high-throughput microbial community data.” Publisher: BioMed Central, *Gigascience* 2 (1): 16. https://doi.org/10.1186/2047-217X-2-16.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy McGowan, Romain François, Garrett Grolemund, et al. 2019. “Welcome to the tidyverse.” Publisher: The Open Journal, *J. Open Source Softw.* 4, no. 43 (November): 1686. ISSN: 2475-9066. https://doi.org/10.21105/joss.01686.