Workflows for processing microbial amplicon sequences

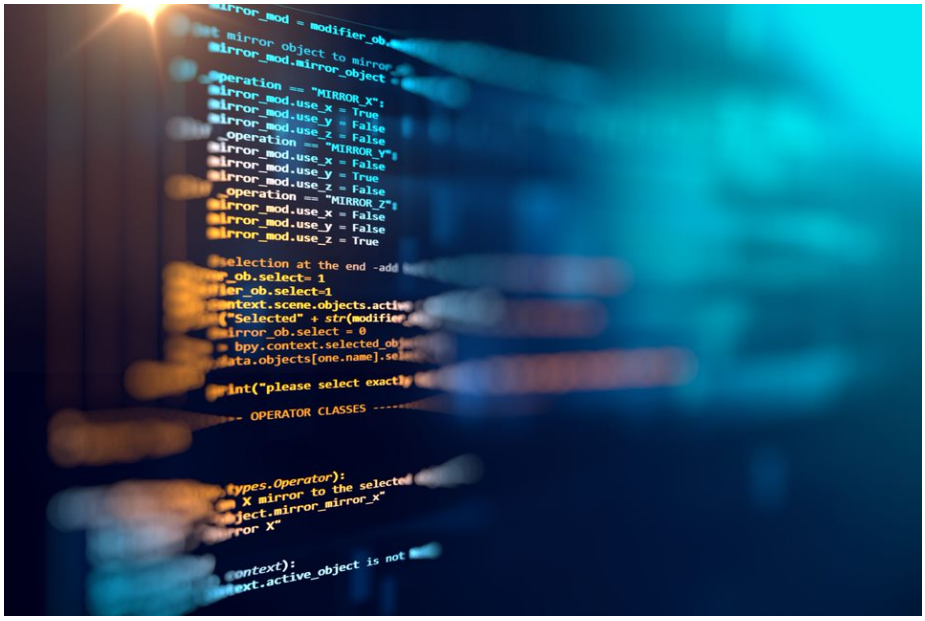
Christian Krohn, PhD, RMIT University

2022-09-27

Contents

Chapter 1

About this GitBook



This GitBook contains workflows for students who want to get started with sequencing microbial amplicons on a Miseq instrument and then process and analyse ASV-based data. It is a compilation of workflows that I have gotten accustomed to during my PhD at La Trobe University with help of many amazing students and colleagues over the years. But it is by no means comprehensive.

One of the biggest hurdles for students that embark on sequencing environmental DNA for the first time, is the effort that is required to understand the various coding languages, file types and formats, packages or platforms that are involved (think Unix, Linux, Slurm, Qiime, R, RMarkdown, python, conda,

ggplot, docker, GitHub, data instances...) before they even can start looking at exploring the data for biological meaning and producing publishable output.

For example, this GitBook is made in Rstudio, using the `rmarkdown` markup language, rendered using the `bookdown` package, and hosted on GitHub. It took me at least five years before I was ready to create and host this content for students. And even then I needed this awesome Bookdown template from Dr. Caspar van Lissa to get started with Bookdown. So it is really helpful to have something to start with.

It was not easy to get over the initial hurdles for me too, so I thought I'd try to compile my workflows. This may help students to get a head start but also helps to scrutinise my approach in the hope that others can point me to better ways of doing things.

Browse to your level

Everyone is at a different level in the journey to exploring microbial diversity. Some have absolutely no idea where to start and others have created their first phylogentic trees with ggtree or similar. The most important prerequisites are patience and persistence... :). Just explore the different chapters and see where it takes you. In fact you will be amazed how much you can achieve with mininal coding knowledge.

Get in touch

We work at the Andy Ball lab, RMIT University, Bundoora, Melbourne. Email me or comment on the discussion section of the GitHub repository for this GitBook. You will need to get a GitHub account to join the discussion. Its free.

This work is licensed under a Creative Commons Attribution 4.0 International License.

Chapter 2

Getting started

Prerequisites will be listed in each sub chapter separately. Following a few general comments:

2.1 General requirements

2.1.1 Access to computational resources

To process data you will need a personal computer or online computational resources with sufficient number of CPUs, working memory and storage. If you want to run most of the processing on a personal computer then I'd recommend a minimum of **4 CPUs, 16 Gb RAM and 500 Gb storage**. However, please note that some alignment tasks require more RAM, hence some of the steps (e.g. taxonomic classifications) may require a more powerful resource. The advantage of using a personal computer is that you can frequently update the latest packages without relying on a university administrator to do that for you. Furthermore, with a personal computer you wont need to share with others, making it easier to run the packages without a workload manager.

Some university may provide their own access to a high performance computer (HPC) for staff and students and likely have certain packages pre-installed. In such cases a batch workload manger such as Slurm may be used, to manage multiple users trying to run computationally heavy jobs. This works fine too but its a bit of a hassle and I would use such an HPC only if really necessary, i.e. smaller jobs on my personal computer and bigger jobs on the HPC.

In Australia the ARDC Nectar Research Cloud provides free computational resources to students and staff at Universities. This is basically like providing access to an online Linux computer with complete freedom to install any packages as if it was your own. If you are a student you can create a trial account with

your student email address (Using your ID only; without the student.edu.au) and check out how it all works, albeit with limited resources. The Nectar service also provides tuturials for starters. However, to get serious with Nectar, you have to request more resources, which is a fairly straight-forward process. Check out the eligibilities and chat to your supervisor if you are interested.

There are also commercial options, which probably offer very similar type of services to Nectar.

In addition, there is the awesome Galaxy Australia platform, which is an open web-based resource that also contains many tutorials and workflows specific to bioinformatics. Definitely worth checking that out too.

This workbook will be based on using the Nectar Research cloud.

2.1.2 File storage

Working with sequencing data may require you do handle large amounts of data. From experience, a run of a 600µl pool (6-10pM) of amplicons in 2x301 Miseq cycles may produce around 10-20 Gb of FASTQ files. Much more can be expected for NextSeq runs or for long-read sequencing.

Once a Miseq run has finished it is typically available to download from the Illumina Basespace account. After a few runs, the data can accumulate quickly. Hence, **persistent storage space** may be required. With an Illumina Basespace account you get 1 TB of storage and you can keep a copy of the FASTQ file on your account. However, Basespace may delete your data if the account has been inactive for more than 6 months. But you will get reminder emails and all you need to do is to log in every now and then so Illumina wont delete your data. Alternatively, data can be upload onto a public repository for persistent storage, such as the Sequence Reads Archives. If you have access to a research cloud or commercial computational resources then you may store data on those. Personally, I prefer a 1TB solid state harddrive.

2.1.3 Command Line

Many of the tools used are managed through command line. Cloud servers are also accessed through command line. That means there is no graphical user interphase and all commands, including installation and running of packages are done through lines of codes on a simple window interphase. This window is called Terminal (Mac, Linux) or Command Prompt application (Windows). There are lots of online sources to help you get started with command line. One example here: https://towardsdatascience.com/a-quick-guide-to-using-command-line-terminal-96815b97b955

2.1.4 Data storage browser

To upload, download, move, rename files on any cloud computer or high performance computer (HPC) at your institution, you will also require a cloud storage browser, such as Cyberduck (Mac) or PuTTY (Windows).

2.1.5 R and R studio

R has become an essential research language. If you want to progress in research it is almost inevitable for you to learn. Just do it. :). Perhaps start here if you dont know where else to start: https://education.rstudio.com/learn/beginner/. It lists some great step-by-step tutorials on how to install R and Rstudio and then explains the basics.

If you are like me, then you simply copy and paste code from other sites and see what happens. This book should provide you with the required information to enable you to follow the workflows. In case you get stuck, you can either chat to me directly or write on the GitHub Discussion Page for this GitBook (chrismitbiz/ABlab-workflows/discussions/).

At the same time you will want to do short courses (eg. from edX Data Science: R Basics) that delve a little deeper into different data structures such as data frames, matrices, lists and the syntax to arrange stuff. Over time you understand what different lines of code mean and can trouble shoot when things don't work. And things often don't work. You will get used to troubleshooting code :).

2.1.6 Environment managers

Package and environment managers are extremely useful for your workflows. They help to install and run software packages such as qiime2, into individual 'environments' independent of your operating system. The environment manager is installed once and from there you use it to install individual packages. The most commonly used environment manager is Conda. Learn more about it here: https://docs.conda.io/projects/conda/en/latest/user-guide/getting-started.html

Conda can be installed by either installing Anaconda or Miniconda. Boths works the same way. Anaconda requires 5 Gb of discspace and installs everything you can possibly need, while Miniconda is just the raw bones and lets you install things one by one.

Most of the commands to manage environments are done in command line using a terminal. There is also a graphical installer that makes handling the environments a little more visual. That would be my go to. Learn more here: www.anaconda.com.

Once conda installed you can always check what environments are installed with the command `conda env list`. In my case the output looks like this:

```
(base) ubuntu@metagenomics:/pvol$ conda env list
# conda environments:
#
base                  *  /pvol/anaconda3
SqueezeMeta              /pvol/anaconda3/envs/SqueezeMeta
anvio-7.1                /pvol/anaconda3/envs/anvio-7.1
cutadaptenv              /pvol/anaconda3/envs/cutadaptenv
qiime2-2021.8            /pvol/anaconda3/envs/qiime2-2021.8
qiime2-2022.2            /pvol/anaconda3/envs/qiime2-2022.2
```

There are several other environment managers. We are also using Docker in our work but the learning curve is a bit steeper so we wont get into it here.

2.1.7 Explore the help functions

Packages such as qiime2 have great resources to help you understand how to run any of the available commands. For example if you dont know what input parameters are available for the qiime command
`qiime feature-classifier classify-sklearn`, you can simply enter this command into your Terminal (with the qiime environment activated, if it is installed with Conda) and add a `--help` at the end: `qiime feature-classifier classify-sklearn --help`.

Or you just want to know what other commands qiime has in its repertoire, you can run `qiime --help`.

For example, this

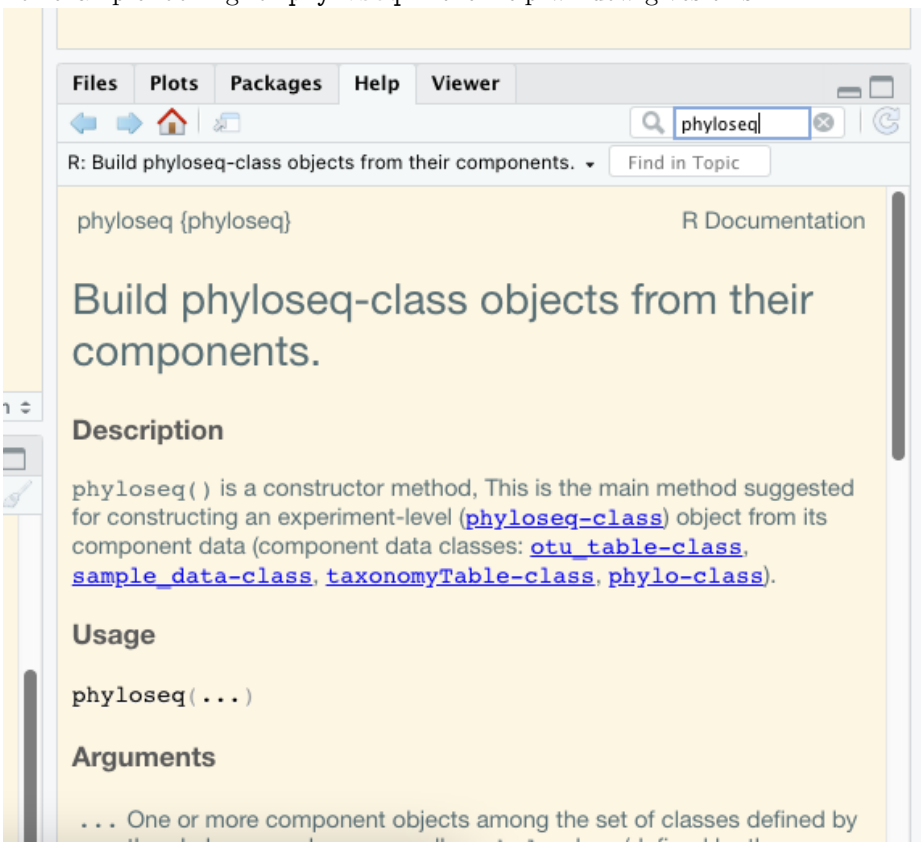
```
conda activate qiime2-2022.2
qiime --help
```

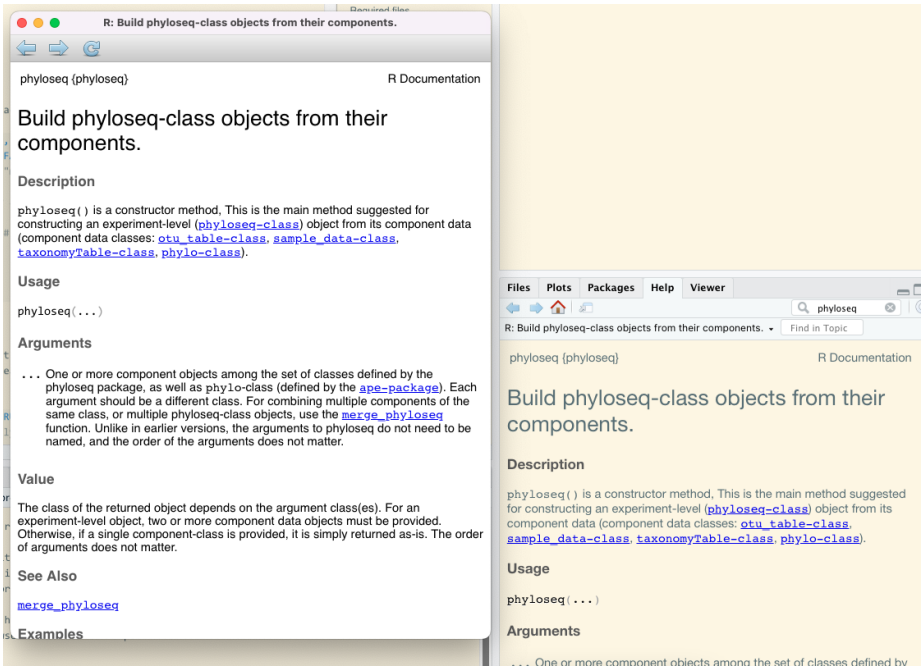
Gives

```
Usage: qiime [OPTIONS] COMMAND [ARGS]...

  QIIME 2 command-line interface (q2cli)
  --------------------------------------

  To get help with QIIME 2, visit https://qiime2.org.

  To enable tab completion in Bash, run the following command or add it to
  your .bashrc/.bash_profile:

      source tab-qiime

  To enable tab completion in ZSH, run the following commands or add them to
  your .zshrc:

      autoload -Uz compinit && compinit
      autoload bashcompinit && bashcompinit
      source tab-qiime

Options:
  --version   Show the version and exit.
  --help      Show this message and exit.

Commands:
  info                Display information about current deployment.
  tools               Tools for working with QIIME 2 files.
  dev                 Utilities for developers and advanced users.
  alignment           Plugin for generating and manipulating alignments.
  composition         Plugin for compositional data analysis.
  cutadapt            Plugin for removing adapter sequences, primers, and
                      other unwanted sequence from sequence data.

  dada2               Plugin for sequence quality control with DADA2.
  deblur              Plugin for sequence quality control with Deblur.
  demux               Plugin for demultiplexing & viewing sequence quality.
  diversity           Plugin for exploring community diversity.
  diversity-lib       Plugin for computing community diversity.
  emperor             Plugin for ordination plotting with Emperor.
  feature-classifier  Plugin for taxonomic classification.
  feature-table       Plugin for working with sample by feature tables.
  fragment-insertion  Plugin for extending phylogenies.
  gneiss              Plugin for building compositional models.
  longitudinal        Plugin for paired sample and time series analyses.
  metadata            Plugin for working with Metadata.
  phylogeny           Plugin for generating and manipulating phylogenies.
  quality-control     Plugin for quality control of feature and sequence data.
  quality-filter      Plugin for PHRED-based filtering and trimming.
  sample-classifier   Plugin for machine learning prediction of sample
                      metadata.

  taxa                Plugin for working with feature taxonomy annotations.
  vsearch             Plugin for clustering and dereplicating with vsearch.
(qiime2-2022.2) ubuntu@metagenomics:/pvol$
```

Within R studio you can access the documentation of individual packages by searching for package or function names in the help windows.

For example looking for `phyloseq` in the Help window gives this:





2.1.8 GitHub account

This GitBook, including all its files, is hosted on one of my GitHub repositories (https://github.com/chrismitbiz/ABlab-workflows). If you have any comments you can ask a question on the Discussion Page of this repository. You require a GitHub account to do that. It is free.

2.2 A note on rarefying (normalising)

Various data normalisation methods exist and opinions on their applicability diverge.

Example of normalisation methods:

- Rarefying data

- Transformation into relative abundances
- Transformation into centered log-ratios
- Cumulative Sum Scaling (CSS)

- Upper Quantile (UQ)

- edgeR-trimmed mean of M values (TMM)
- DESeq-variance stabilizing transformation (VS)

There are others. Check out this link to learn more about the different types of normalisation methods: MicrobiomeAnalyst: FAQs, What are the various normalization methods and which method should I choose?.

Rarefying data is one of the more contested methods. We acknowledge that **rarefying data** (i.e. the random subsampling of abundances to usually the smallest library size) results in loss of data and statistical power and is thus **not recommended for differential abundance analyses**. Check out this well cited paper here to learn why: Waste Not, Want Not: Why Rarefying Microbiome Data Is Inadmissible (McMurdie and Holmes (2014)).

However, rarefying still "*outperformed all the other normalization methods for producing accurate BC (Bray-Curtis) dissimilarities and subsequent PCoAs and PERMANOVAs.*" Paper here: Methods for normalizing microbiome data: An ecological perspective (McKnight et al. (2019)).

For our workflows we may **rarefy data** before creating **alpha diversity** indices (Shannon, Simpsons etc) and **beta diversity** indices (Bray-Curtis dissimilarities).

Other types of normalisation methods should be considered for differential abundance analyses. My preference is to use **centred-log ratios** (Aitchison distances). The **compositional approach** and centred log-ratios are further discussed in the paper Microbiome Datasets Are Compositional: And This Is Not Optional (Gloor et al. (2017)). This type of transformation requires the removal of some of the rare ASVs to remove as many zeros from the abundance table as reasonable.

2.3 Workflows from other lab groups

The following review "*should serve as a starting point for considering what options are available*": Analysing microbial community composition through amplicon sequencing: from sampling to hypothesis testing (Hugerth and Andersson (2017)).

Furthermore, searching for "amplicon-sequencing" under "Topics" on GitHub gave 37 results.

- nf-core/ampliseq
  On the top of that list is the nf-core/ampliseq pipeline developed by the nf-core community. It is based on a software called Nextflow which allows to put different processes into a pipeline. This is great for doing things

a little more reproducible but it requires you to be fairly knowledgeable with Linux, container software, config files etc.. Not great to learn stuff for beginners.

- Tools-Microbiome-Analysis
  Tools-Microbiome-Analysis is a websites containing a comprehensive list of R packages and, more importantly, tutorials related to analysis of microbial amplicons and ecology. Really good reference to go back to every now and then. **Highly recommended**.

- grimmlab/MicrobiomeBestPracticeReview
  Essential paper to read (Current challenges and best-practice protocols for microbiome analysis, 2021 (Bharti and Grimm (2021))) and a great workflow resource on GitHub. **Essential read**. The Grimm lab in Munich published this paper as well as developed a python and R-based workflow that assists with the recommended best practices (amplicon as well as metagenomic workflows).

- KasperSkytte/ampvis2
  Ampviz2 is an R-package to visualise and analyse 16S rRNA amplicon data. It is always more convenient to have packages that have the details and optics worked out for you. Like phyloseq, the ampviz2 package combines different tables from data (E.g. otu table, taxonomic table, phylogenetic tree, sample data etc) and then provides different functions to apply to that combined object to visualise the data.

Chapter 3

Miseq library preps

3.1 Introduction

Some info on getting your run ready

In our lab we usually prepare a 6-10 pM pool of libraries following the 16S Metagenomic Sequencing Library Preparation protocol from Illumina with 10-30% Phix. **Download it, study it and refer to it throughout your lab-work**. This protocol is not limited to 16S amplicons. You can use it for fungal (e.g. ITS) amplicons too. In the last section there are also valuable tips for pre-PCR and post-PCR lab procedures. There is also some additional Illumina guidance here: https://sapac.support.illumina.com/downloads/16s_metagenomic_sequencing_library_preparation.html

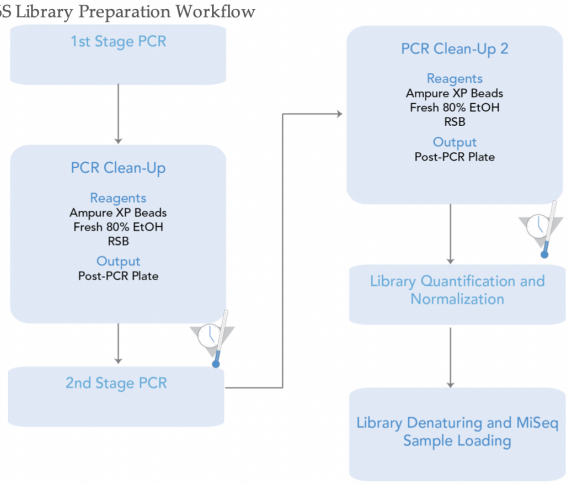
3.1.1 Overview

The protocol encompasses the following steps (copied out of the protocol):

16S Library Preparation Workflow

The following diagram illustrates the workflow using the 16S Library Preparation Protocol. Safe stopping points are marked between steps.

Figure 2 16S Library Preparation Workflow



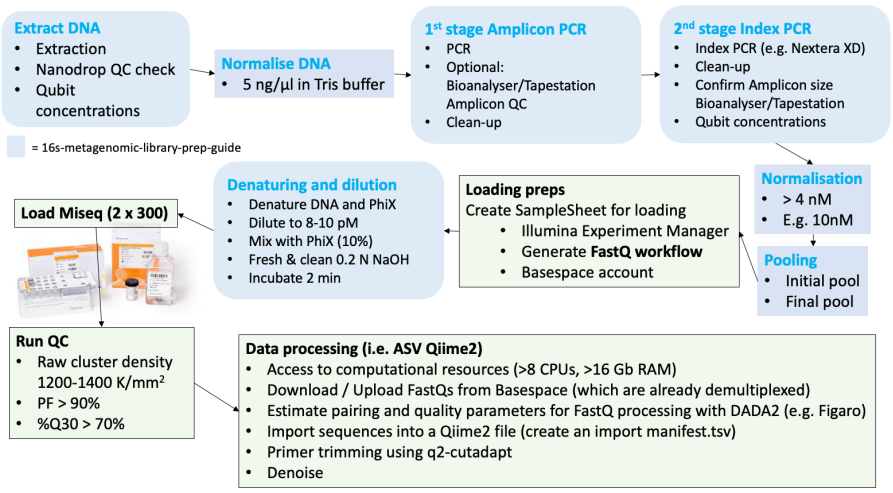
Please note that only **one amplicon target** should be sequenced in each run. That means it is not recommended to sequence for example two different 16S amplicons or one 16S amplicon plus an ITS amplicon. Sequencing different amplicon lengths will result in reduced quality as the the Miseq flowcell will preferentially cluster shorter fragments. However, it is not impossible. We have tried to run four different amplicons on one flowcell before, although with mixed results.

Labwork may take four weeks from DNA extraction to Miseq loading. Double that if you have not done it before. You probably realised that there is quite a bit of labwork required before you finally get FASTQ files and subsequent ASV/taxonomy tables to analyse - the ultimate goal here. The more samples are planned to be included into the pool, the more work and consumables are needed. Illumina recommends no more than 96 samples on one flowcell to get enough depths (i.e. $>$ 100,000 reads per sample). But we have sequenced around 196 samples on one flowcell before, which captured sufficient reads and diversity. This depends on how much detail and rare taxa you want to capture. The less samples you include, the more data and depth for each sample.

Also note that you will require over 20 boxes of tips (mostly 200 µl) for each 96-well plate, so plan your consumables ahead of time. Most of the tips will be used for the PCR clean ups.

After all steps in the above protocol are completed, the Miseq will be loaded

with a 600 µl pool, followed by 56 hours of run-time for 2x301 cycles before FASTQ files are available for download from BaseSpace.



Throughout the 56 hours run you can check how things are tracking either on the BaseSpace website or the BaseSpace phone app. Understanding different QC metrics will help identifying potential problems with the pool or the instrument. It is important to note that this is a bit of a nerve-wrecking moment in the process, as it remains unclear until the metrics come through after around the 21st cycle, whether or not the run will be successful.

3.2 Workflow

3.2.1 DNA extraction

We commonly extract DNA from soils or from wastewater sludges using the *DNeasy Powersoil Pro Kit* (Qiagen) for both, soil and sludge. It results in high quality DNA and includes a bead beating step to make sure the gram positive cells are sufficiently broken up. Although, I don't have evidence for that. In fact, please let us know on the GitHub Discussion Page for this GitBook if you have any comments on that as we are always looking for ways to improve things.

Soil

- For soils, the protocol of the extraction kit is followed, weighing in 0.25 g of soil and making sure to record the exact weights used.

Figure 3.1: "DNA extracts"

- Also make sure to measure the water content of the soil to be able to report DNA concentrations per dry weight of soil.

Sludge

- For DNA extraction of sludge, 0.5 ml is added into an empty 1.7 ml extraction tube (without extraction beads), then centrifuged at 5000 rpm for 5 minutes before removing the supernatent.
- Extraction beads are then added and the Powersoil protocol followed.
- Measure Total Solids (TS) of the sludge to report DNA concentrations per gram of TS.

3.2.2 DNA quality and concentration

Once you have extracted DNA it is necessary to confirm its quality and concentration.

- Quality is assessed using a Nanodrop Spectrophotometer. You are aiming for **260/280 Ratio of 1.8 and a 260/230 Ratio of 2-2.2**. This is explained in the T042-TECHNICAL BULLETIN. The Nanopore also estimates DNA concentrations but it is not recommended to use these concentrations measurements to normalise DNA. It is more accurate and consistent with a Qubit Fluorometer.

- DNA concentrations are best measured using fluorescent dye-based methods such as the **Qubit Fluorometer**. It is easist to use the '*1X*' assays such as the *Qubit 1X dsDNA BR Assay Kit* or the *Qubit 1X dsDNA HS Assay Kit*
  - Broad range (BR) Kit for quantifying extracted environmental DNA and afte the 2nd stage PCR.
  - High Sensitivity (HS) Kit for quantifying the pool at the end.

3.2.3 DNA normalisation

The next step is to normalise all DNA extracts to 5 ng/µl.

- Dilute DNA to equal concentrations, either in a fresh batch of PCR-grade 10mM Tris buffer (pH 8.5) or in PCR-grade water
- Set a pre-define DNA volume, say 2µl for all extracts, and add Tris or water as calculated with

$$C_1V_1 = V_2C_2$$

3.2.4 First and second PCR - Amplicon PCR and Indexing PCR

Follow the 16S Metagenomic Sequencing Library Preparation Guide. Include a negative control and sequence that as well. This allows you to assess background and cross contamination. The whole workflow is best done in 96-well plates, using multi-channel pipettes to avoid indexing mistakes across wells.

Sample list and plate layout

Ensure that all sample IDs, plate-layout and each unique Illumina index carefully planned and printed out before you start your PCRs. Perhaps, stick the layout onto the Bio-safety cabinet where you do most of the pipetting and keep checking you are on the right well and sample. Label all plates even if you discard them afterwards. Labwork can be stressful and you need to trace all your steps across each well of the plate because otherwise you risk that you have indexed the wrong samples and it will become impossible to know which sample you have sequenced.

Primers

Figure 3.2: "The ready-mix working solution for Qubit, which includes dye and buffer. No preparation/premixing required."

Any primers you order for the Miseq will have to include the **Illumina overhang adapter**. The sequence of the overhang adapter is added **in front** of the gene-specific primer sequences. There are two overhang adapters; one for the forward primers and one for the reverse primer. Check out the relevant information on primers from page 3 the 16S Metagenomic Sequencing Library Preparation Guide.

Clean-ups

It is worth doing the clean-ups in a separate deep-well plate ("MIDI" plate, see consumables below) to avoid any blow-out of dried beads at the end of the clean-up. This process is explained in the above mentioned guide too. Tip: Handle the beads extremely carefully after the ethanol has dried. They blow out easily.

Index Kits

The index kits are expensive. Try to handle them with care and replace lids with the provided replacement lids, after every use. Multipipettes are absolutely essential to avoid loosing track on indexes used across the different wells on the plate. The use of the Illumina Plate Fixture comes in handy (see consumables below).

Amplicon QC

After the indexing-PCR products are cleaned it is recommended to run a gel on all samples to confirm that amplification was successful with enough DNA present and that it is the correct amplicon length. It also helps to identify any primer dimers, which may have to be cleaned out.

If you have access to a **Tapestation or a Bioanalyzer** in your lab, perhaps run a random subset of the indexed samples on one of these instruments, just to accurately confirm the average length of the amplicon. This average basepair length of the amplicon is important, as it is the basis for normalising your DNA to nM. In any case, after the index samples are normalised and pooled, it is highly recommended to run the pool on a Bioanalyser too. Otherwise it is too much guesswork in getting the concentrations right. This is further explained below.
Repeat any samples that failed, i.e. that do not show any bands, and ensure you use the same indices.

3.2.5 Normalisation of indexed amplicons

After you finished all PCR work it is time to do another round of Qubitting and normalisation of all samples into a new 96-well plate. The final pool should be 4 nM but we prefer to normalise DNA to a higher concentration first, say 10

nM, then pool and then dilute the pool to 4 nM afterwards.
We are using a very handy excel sheet, which was kindly developed and provided by Sarah Knowler at La Trobe University to prepare and assist in normalisations, converting from ng/µl to nM using the formula:

$$\frac{concentrations\ (ng/ţl)}{660g/mol\ x\ average\ library\ size}$$

Contact us if you would like to get a copy of this very handy library preparation excel sheet.

Some more information here: https://support.illumina.com/content/dam/illumina-support/documents/documentation/system_documentation/miseq/miseq-denature-dilute-libraries-guide-15039740-10.pdf. There's also a pooling calculator here https://support.illumina.com/help/pooling-calculator/pooling-calculator.htm which may be helpful with your calculations.

3.2.6 Pooling

- Pipette equal volumes of the normalised DNA into one 1.5 ml tube.

- Measure its concentration using the High Sensitivity (HS) Qubit 1X dsDNA HS Assay Kit.

- Assuming the pool was created wtih normalised concentration that are higher than 4 nM, then you will need to dilute the pool down to 4nM. In order to do so you will need to accurately measure the average basepair length of your pool. Do not rely on the ladder on a gel. Always get an accurate read. See next point.

- Measure the average base pair lengths of the pool with a **Bioanalyzer or Tapestation**. Use at least three replicates. The average basepair length is then used to calculate the final concentration required in ng/µl to get a 4 nM pool.

- Dilute the pool to the concentration in ng/µl that reflect 4 nM.

3.2.7 Denaturing of pool and loading of the Miseq

Now you are almost ready load the Miseq. The last steps are done right before you plan to load the instrument. First the pool DNA needs to be denatured and diluted further to your preferred loading concentration (6 - 10 pM) before it is mixed with PhiX. The final concentrations impact on cluster densities. That means you get more data with higher concentrations but that is at the expense of potential errors during sequencing at high cluster densities. We recommend to aim for a cluster density of around 900-1000 K/mm^2, which may be achieved with a concentration of around 7-8 pM.

The key for the final steps are that..

- ..the reagent cartridge is properly thawed. Perhaps thaw in fridge 3 hours before starting the denaturing steps. Then put into icy water to ensure final and even thawing across the cartridge. Mix cartrige well at the end and ensure that no bubbles are visible before loading.

- .. the 0.2 N sodium hydroxide (NaOH) is made fresh with biological grade NaOH. For example use the *BioUltra, for molecular biology, 10 M NaOH in H2O* from Sigma Aldrich to make a 0.2 N solution (N = Normality but it is equal to Molarity here because there is only one OH in NaOH). I have used analytical-grade NaOH pellets as well, which worked but the RNAse free solutions are safer in terms of inhibitors etc..

- ..that the pool is cooled down immediately in ice or icy water right after the 2 minutes in the 96 degrees C heatblock and pretty much immediately loaded into the Miseq. The Miseq has to be prepped beforehand (I.e. maintenance wash and sample sheet loaded).

- a sample sheet (or so-called manifest file) is created beforehand using the Illumina Experiment Manager Software. This software assist in producing a `.csv` file that is then loaded into the Miseq instrument via a USB and ensures that the indexes you have used are linked to the relevant samples in your pool. Basically a list of sample IDs and index IDs. An example of a sample sheet from one of our runs is shown below.
  IMPORTANT: Make sure to specify the Workflow as **GenerateFastQ**.

- you have a Illumina BaseSpace account.

[Header]												
IEMFileVersion	5											
Investigator Name	AndyBall Lab											
Experiment Name	PrimerTrial											
Date	6/22/2022											
Workflow	GenerateFASTQ											
Application	Assembly											
Instrument Type	MiSeq											
Assay	Nextera XT											
Index Adapters	Nextera XT v2 Index Kit											
Description	threedigestersfourprimersplusPMA											
Chemistry	Amplicon											
[Reads]												
301												
301												
[Settings]												
ReverseComplement	0											
kmer	31											
Adapter	CTGTCTCTTATACACATCT											
[Data]												
Sample_ID	Sample_Name	Sample_Plat	Sample_Well	I7_Index_ID	index	I5_Index_ID	index2	GenomeFold	Sample_Proj	Description		
PT_01	PT_01	1	A01	N701-C	TAAGGCGA	S513-C	TCGACTAG		PrimerTrial	ARCSR_02_Meso_01V4_UNoPMA		
PT_02	PT_02	1	A02	N702-C	CGTACTAG	S513-C	TCGACTAG		PrimerTrial	ARCSR_02_Meso_02V4_UNoPMA		
PT_03	PT_03	1	A03	N703-C	AGGCAGAA	S513-C	TCGACTAG		PrimerTrial	ARCSR_02_Meso_03V4_UNoPMA		
PT_04	PT_04	1	A04	N704-C	TCCTGAGC	S513-C	TCGACTAG		PrimerTrial	ARCSR_02_Meso_04V4_UNoPMA		
PT_05	PT_05	1	A05	N705-C	GGACTCCT	S513-C	TCGACTAG		PrimerTrial	ARCSR_02_Meso_05V4_UNoPMA		
PT_06	PT_06	1	A06	N706-C	TAGGCATG	S513-C	TCGACTAG		PrimerTrial	ARCSR_02_Meso_01V4_UPMA		
PT_07	PT_07	1	A07	N707-C	CTCTCTAC	S513-C	TCGACTAG		PrimerTrial	ARCSR_02_Meso_02V4_UPMA		
PT_08	PT_08	1	A08	N710-C	CGAGGCTG	S513-C	TCGACTAG		PrimerTrial	ARCSR_02_Meso_03V4_UPMA		
PT_09	PT_09	1	A09	N711-C	AAGAGGCA	S513-C	TCGACTAG		PrimerTrial	ARCSR_02_Meso_04V4_UPMA		
PT_10	PT_10	1	A10	N712-C	GTAGAGGA	S513-C	TCGACTAG		PrimerTrial	ARCSR_02_Meso_05V4_UPMA		
PT_11	PT_11	1	A11	N714-C	GCTCATGA	S513-C	TCGACTAG		PrimerTrial	ARCSR_02_Meso_01V4_UPMAx		
PT_12	PT_12	1	A12	N715-C	ATCTCAGG	S513-C	TCGACTAG		PrimerTrial	ARCSR_02_Meso_02V4_UPMAx		
PT_13	PT_13	1	B01	N701-C	TAAGGCGA	S515-C	TTCTAGCT		PrimerTrial	ARCSR_02_Meso_03V4_UPMAx		
PT_14	PT_14	1	B02	N702-C	CGTACTAG	S515-C	TTCTAGCT		PrimerTrial	ARCSR_02_Meso_04V4_UPMAx		
PT_15	PT_15	1	B03	N703-C	AGGCAGAA	S515-C	TTCTAGCT		PrimerTrial	ARCSR_02_Meso_05V4_UPMAx		
PT_16	PT_16	1	B04	N704-C	TCCTGAGC	S515-C	TTCTAGCT		PrimerTrial	ARCSR_03_Meso_01V4_UNoPMA		
PT_17	PT_17	1	B05	N705-C	GGACTCCT	S515-C	TTCTAGCT		PrimerTrial	ARCSR_03_Meso_02V4_UNoPMA		
PT_18	PT_18	1	B06	N706-C	TAGGCATG	S515-C	TTCTAGCT		PrimerTrial	ARCSR_03_Meso_03V4_UNoPMA		
PT_19	PT_19	1	B07	N707-C	CTCTCTAC	S515-C	TTCTAGCT		PrimerTrial	ARCSR_03_Meso_04V4_UNoPMA		
PT_20	PT_20	1	B08	N710-C	CGAGGCTG	S515-C	TTCTAGCT		PrimerTrial	ARCSR_03_Meso_05V4_UNoPMA		
PT_21	PT_21	1	B09	N711-C	AAGAGGCA	S515-C	TTCTAGCT		PrimerTrial	ARCSR_03_Meso_01V4_UPMA		
PT_22	PT_22	1	B10	N712-C	GTAGAGGA	S515-C	TTCTAGCT		PrimerTrial	ARCSR_03_Meso_02V4_UPMA		
PT_23	PT_23	1	B11	N714-C	GCTCATGA	S515-C	TTCTAGCT		PrimerTrial	ARCSR_03_Meso_03V4_UPMA		
PT_24	PT_24	1	B12	N715-C	ATCTCAGG	S515-C	TTCTAGCT		PrimerTrial	ARCSR_03_Meso_04V4_UPMA		
PT_25	PT_25	1	C01	N701-C	TAAGGCGA	S516-C	CCTAGAGT		PrimerTrial	ARCSR_03_Meso_05V4_UPMA		
PT_26	PT_26	1	C02	N702-C	CGTACTAG	S516-C	CCTAGAGT		PrimerTrial	ARCSR_03_Meso_01V4_UPMAx		
PT_27	PT_27	1	C03	N703-C	AGGCAGAA	S516-C	CCTAGAGT		PrimerTrial	ARCSR_03_Meso_02V4_UPMAx		
PT_28	PT_28	1	C04	N704-C	TCCTGAGC	S516-C	CCTAGAGT		PrimerTrial	ARCSR_03_Meso_03V4_UPMAx		

Figure 3.3: "Example of a manifest file"

Table 3.1: Example of a consumables list we typically include for a Miseq run

Equipment.consumables	Item.number	Supplier	Comment
Qiagen Powersoil Pro test kit, 250	#47016	Qiagen	
Primers	Various	E.g. Sigma	
Magnetic beads, AMPure XP / JetSeq Clean	BIO-68031	Meridian Life Sciences	For clean-up
KAPA HiFi Hotstart ready mix	KK2602, 6.25ml	Millennium Science	For PCR
MiSeq Reagent Kit v3	MS-102-3003	Illumina	Needs to be a
Nextera XT Index Kit v2, Set A, B, C or D	FC-131-2003	Illumina	One set (A, B
PhiX control	FC-110-3001	Illumina	
Qubit 1X dsDNA BR Assay Kit	Q33266	Thermofisher	DNA and PC
Qubit 1X dsDNA HS Assay Kit	Q33230	Thermofisher	Pool quantific
Bioanalyzer Kit (HighSens or 1000 Kit)	5067-4626 or 5067-1504	Integrated Science Tech	
Tween 20	H5151	Promega	For Miseq inst
96-well, 0.2ml plates, non-skirted	AB0600	Thermofisher	For PCR
96 Well 0.8mL Polypropylene Deepwel Plate	AB0859	Thermofisher	For clean-up
Optical Plate Seals	V7840	Promega	For PCR
0.5mL thin-walled PCR tubes	E4942	Promega	For Qubit

3.2.8 Consumables

All consumables and equipment required for the 16S Miseq library prep protocol are available from page 21 of the above mentioned 16S Metagenomic Sequencing Library Preparation Guide.
The **magnetic stand** for 96-well plates and the TruSeq Index **Plate Fixture Kit** are essential.

Here is what we typically include into our purchasing list. The combined total of the below list costs over AUD$ 10,000.

Chapter 4

From BaseSpace to Qiime2 and DADA2

4.1 Introduction

In this chapter you will learn how to process the FASTQ files that are generated from an amplicon sequencing run on a Miseq instrument. We are using Qiime2. Please also refer to the extensive documentation and tutorials available at https://qiime2.org. You can learn almost everything there.

4.1.1 Prerequisites and required files

- FASTQ files available to download on BaseSpace. Two for each sample. One for read 1 (R1) and another for read 2 (R2). For example, `PT-01_S1_L001_R1_001.fastq.gz` and `PT-01_S1_L001_R2_001.fastq.gz` contains all sequences for sample PT-01.
- Personal computer or cloud computational resources with > 4 CPUS, > 16 RAM, > 100 GB storage recommended.
- Cloud storage browser installed (example Cyberduck or PuTTYm see Chapter 2).
- Qiime2 installed. Always use the latest version.
- All required files in correct format and file extension (see below).
- The BaseSpace Downloader software (available on the BaseSpace page. You will be prompted to download once you go to your files).

Table 4.1: Example of a qiime import manifest.tsv file

sample-id	forward-absolute-filepath	rev
PT-01	/pvol/Sequences/20220622_PrimerTrial_RMIT/PT-01_S1_L001_R1_001.fastq.gz	/pv
PT-02	/pvol/Sequences/20220622_PrimerTrial_RMIT/PT-02_S2_L001_R1_001.fastq.gz	/pv
PT-03	/pvol/Sequences/20220622_PrimerTrial_RMIT/PT-03_S3_L001_R1_001.fastq.gz	/pv
PT-04	/pvol/Sequences/20220622_PrimerTrial_RMIT/PT-04_S4_L001_R1_001.fastq.gz	/pv
PT-05	/pvol/Sequences/20220622_PrimerTrial_RMIT/PT-05_S5_L001_R1_001.fastq.gz	/pv
PT-06	/pvol/Sequences/20220622_PrimerTrial_RMIT/PT-06_S6_L001_R1_001.fastq.gz	/pv

Table 4.2: Example of a samplesheet.tsv file

#SampleID	#Sludge	WWTP	SludgeType	PMA_treat	Rep_bio	Sludgecollection
#q2:types	categorical	categorical	categorical	categorical	numeric	categorical
PT-01	1	ETP5	Meso	NoPMA	1	9/5/2022
PT-02	1	ETP5	Meso	NoPMA	2	9/5/2022
PT-03	1	ETP5	Meso	NoPMA	3	9/5/2022
PT-04	1	ETP5	Meso	NoPMA	4	9/5/2022
PT-05	1	ETP5	Meso	NoPMA	5	9/5/2022

`Qiimeimportmanifest.tsv` - A tab separated file with three columns where the first column is the sample ID (The exact same ID that was used in the manifest file for loading the Miseq). The second and third columns are the absolute paths to the forward and reverse FASTQ files, respectively. If you are working on a cloud computer then the FASTQ files should be uploaded to that, and the path needs to contain the full path to the relevant folder, including the full name of the file itself. If you process those files on your local drive then the paths need to change to your local folder containing the FASTQ files. In the below example table, the files are located on a Linux cloud computer (Nectar Research Cloud).

It might take some time to compile the import manifest. Each file path has to exact. However, in case you made a mistake in the file names etc, Qiime is generally pretty good at reporting where the error occured.

`samplesheet.tsv` - file containing sample metadata. This file should contain all other environmental measurements you may have done on each sample (pH, EC...), which you want to relate to microbial taxonomy and composition. The same file and format will be used to import metadata into the R package Phyloseq later too.

IMPORTANT: Create the `.tsv` file as a tab separated `.csv` file in Excel first and then manually change the file name from `.csv` to `.tsv`. Not sure how else to easily build the `.tsv` file.

`scriptfile.txt` - A text file that contains your qiime2 scripts. Example below. Code is compiled here and then only the file itself is executed to run any commands in that text. The hash (#) in front of any line of code stops it from being executed. That means all code is normally 'hashed'. If one wants to run the code the lines are 'un-hashed' and the text file is executed in the command line to run the 'un-hashed' code. On the Linux cloud computer the command would be: `sh scriptfile.txt`.

Note that a script file is meant to make life easy and the process replicable. But you could run each of the processes directly by adding the commands into the terminal without a script file.

Briefly on script files for cloud-based high performance computers (HPC) that are using Slurm to manage multiple user runs. If the HPC at your institution is using Slurm, then the script file has to be called with `sbatch scriptfile.txt`. The script files for HPCs using slurm contains a header that determins the allocations (number of CPUs, RAM etc..) for your task. The first lines of the .txt file typically look similar to this:

```
#!/bin/bash
#SBATCH --cpus-per-task=6
#SBATCH --mem-per-cpu=6000
#SBATCH --time=24:00:00
#SBATCH --partition=compute
#SBATCH --job-name=Q_fun
```

Commands, such as this:

```
echo "Starting at: $(date)"
echo "Finished at: $(date)"
```

.. are less important but help to report the start and end times of the executed commands from beginning (where the `echo` command is placed at the beginning of the script) to end (`echo...` placed at the end of the script) into the Terminal Windows. They dont affect the qiime commands in any way.

Slurm and `sbatch` are not needed if you manage your own computer or your 'own' cloud computer. In the example below we are using a Linux cloud computer, provided by the Nectar Research Cloud so we dont need Slurm.

```
sce#!/bin/bash
#SBATCH --cpus-per-task=8
#SBATCH --mem-per-cpu=6000
#SBATCH --time=96:00:00
#SBATCH --partition=compute
#SBATCH --job-name=Q_bac

start=$(date +'%Y-%m-%d %H:%M:%S')
echo "start time $start"


### 24/06/2022 Experiment 1

### USE THIS FILE TO IMPORT FASTQ FILES INTO A qiime2 file called 'demux-paired-end.qza' or demux-sgle.qza" (or
similar)
### Place this script into the same folder that contains the fastq files
### Output files go into another folder for further processing
### Sequences compiled in /pvol/Sequences/20220622_PrimerTrial_RMIT/


# Load qiime
eval "$(conda shell.bash hook)"
conda activate qiime2-2022.2


##### ----- V4_U

###---------------------------- PAIRED END MANIFEST IMPORT (Step 1)

#qiime tools import --type 'SampleData[PairedEndSequencesWithQuality]' \
#       --input-path /pvol/workdirs/Project1C/202206_PrimerTrial/PreProcessingFastQ/V4_U/Importmanifest.tsv \
#       --output-path /pvol/workdirs/Project1C/202206_PrimerTrial/PreProcessingFastQ/V4_U/demux-paired-end.qza \
#       --input-format PairedEndFastqManifestPhred33V2

#qiime demux summarize \
#       --i-data /pvol/workdirs/Project1C/202206_PrimerTrial/PreProcessingFastQ/V4_U/demux-paired-end.qza \
#       --o-visualization /pvol/workdirs/Project1C/202206_PrimerTrial/PreProcessingFastQ/V4_U/demux-paired-
end.qzv



###----------------------------- SINGLE ENDED MANIFEST IMPORT (Step 1)

##### ----- V4_U
# qiime tools import --type 'SampleData[SequencesWithQuality]' \
#       --input-path /pvol/workdirs/Project1C/202206_PrimerTrial/PreProcessingFastQ/V4_U/
Importmanifest_single.tsv \
#       --output-path /pvol/workdirs/Project1C/202206_PrimerTrial/PreProcessingFastQ/V4_U/demux_single.qza \
#       --input-format SingleEndFastqManifestPhred33V2

# qiime demux summarize \
#       --i-data /pvol/workdirs/Project1C/202206_PrimerTrial/PreProcessingFastQ/V4_U/demux_single.qza \


conda deactivate
stop=$(date +'%Y-%m-%d %H:%M:%S')
```

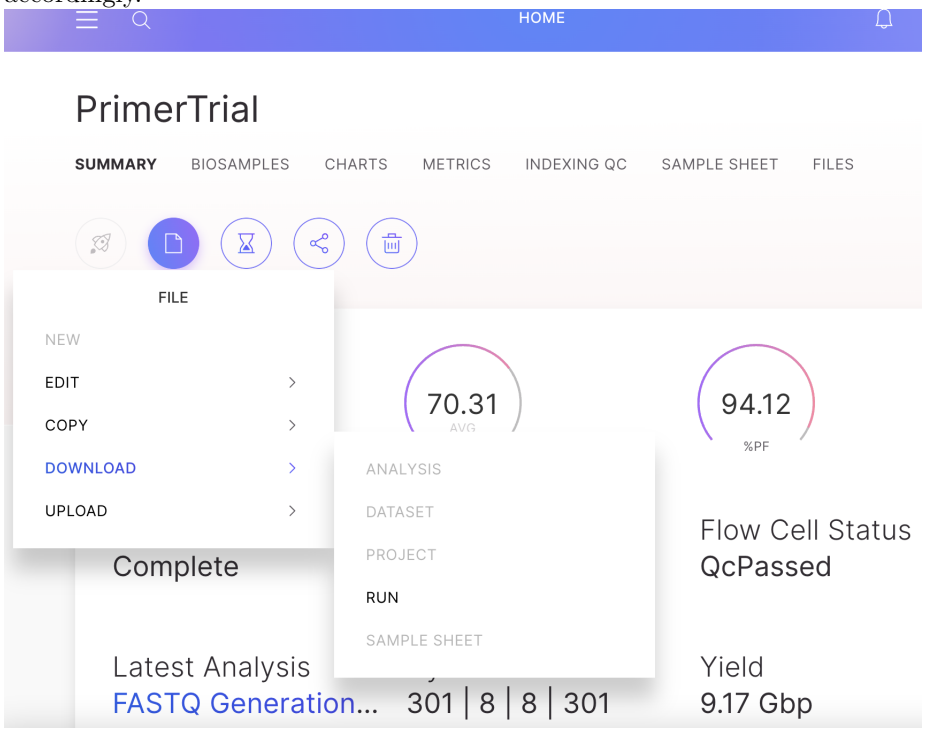
Figure 4.1: “Example of a scriptfile.txt”

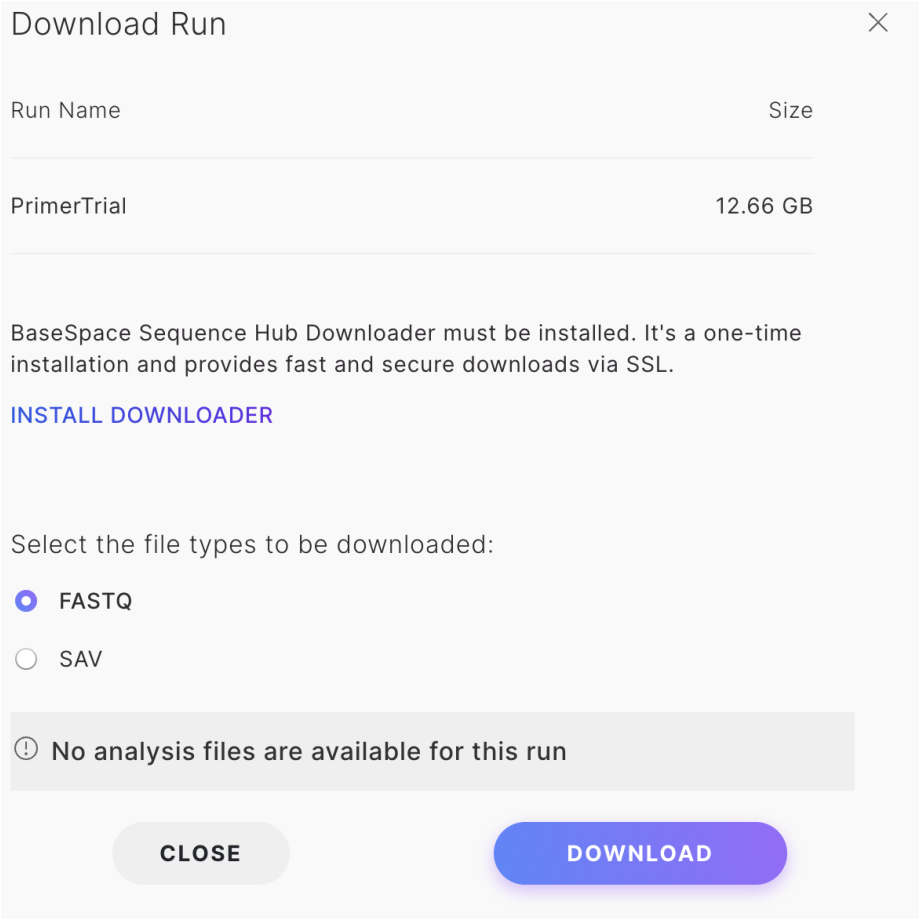
4.2 Workflow

4.2.1 Download FASTQ files

In this example, Fastq files were produced from sequencing 16SrRNA marker genes on an Illumina Miseq instrument. If you ran your own library on your own BaseSpace account, the files should be available to download from that BaseSpace account. If the run was created by another user, that person can invite you to the run, which gives you access to download the files too.

Log in to BaseSpace and into your project. Go to files, run and download FastQ. You will be asked to install a downloader software. Follow instructions accordingly.





The FASTQ files will come in separate folders for each sample. It is necessary to move all files into just ONE folder. That folder will then be used to import the sequences into a qiime object (described below).

4.2.2 Paired end manifest import (Step 1)

Go to the folder that contains all the FASTQ files (all files in one folder). I.e. in Unix/Linux use `cd path/to/folder/` to navigate to it. The scriptfile.txt should also be located here. The following qiime command that will look for the files and then create a qiime object using the sequences found in each FASTQ file. The `--output-path` instructs qiime where to place and how to name the output file, here demux-paired-end.qza is placed into the same directory.
Remove any # in front of your code (already removed in the below example) and

execute the scriptfile.txt with `sh scriptfile.txt` (or `sbatch scriptfile.txt` in case you are using Slurm).

```
qiime tools import --type 'SampleData[PairedEndSequencesWithQuality]' \
  --input-path Qiimeimportmanifest.tsv \
  --output-path demux-paired-end.qza \
  --input-format PairedEndFastqManifestPhred33V2
```

4.2.3 Cutadapt (Step 2)

If all goes well you should now have a demux-paired-end.qza file in the same directory as your FASTQ files , which contains all your sequences. You could move demux-paired-end.qza to another folder and continue all other steps there if you wish. The FASTQ files are not needed anymore. Dont forget to move your script file to where it is most convenient for executing the commands. If you moved the demux-paired-end.qza to another folder it is probably easiest to also move the script file that the same directory (or create a second script file containing the remaining commands).

Go to the folder containing demux-paired-end.qza and use cutadapt to trim out the primer sequences. Unhash relevant lines in your script file, which should be located in the same folder. Run `sh scriptfile.txt`.

In this example we used the V4 primers. Change primer sequence to the exact primers that you used in the amplicon PCR.

```
qiime cutadapt trim-paired \
  --i-demultiplexed-sequences demux-paired-end.qza \
  --p-front-f GTGYCAGCMGCCGCGGTAA \
  --p-front-r GGACTACNVGGGTWTCTAAT \
  --o-trimmed-sequences trimmed_demux-paired-end.qza
```

4.2.4 Read quality assessment

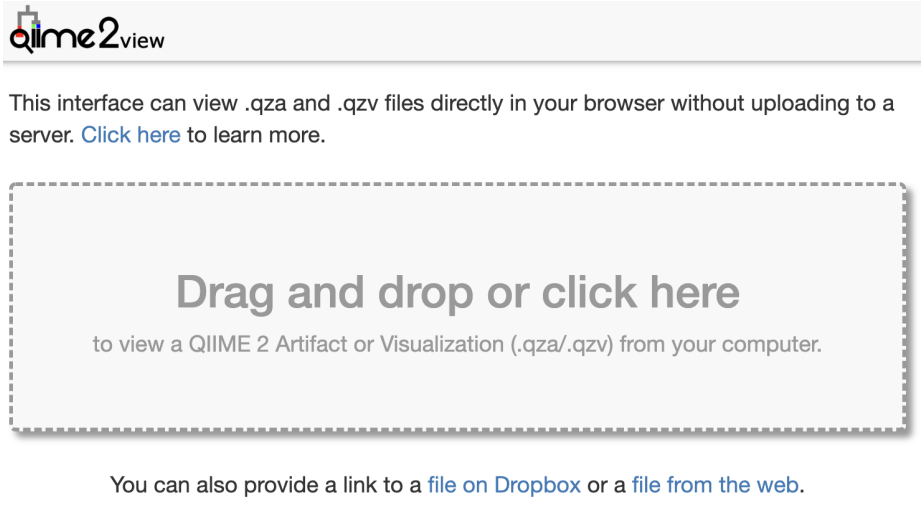
Visualise the output with a .qzv file

The .qza files can be visualised by 'converting' them into .qzv files.
Here we take the trimmed_demux-paired-end.qza and create a trimmed_demux-paired-end.qzv.

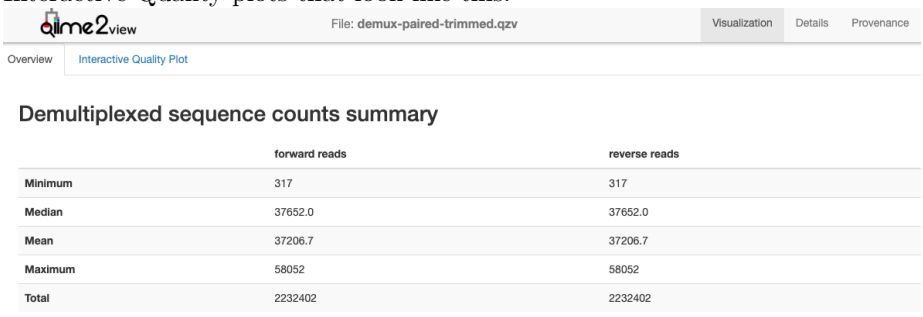
NOTE: View any .qzv file on https://view.qiime2.org. Drag and drop the qzv file into the browser window and inspect the results.

```
qiime demux summarize \
  --i-data trimmed_demux-paired-end.qza \
  --o-visualization trimmed_demux-paired-end.qzv
```

Go to https://view.qiime2.org and drag and drop to visualise your .qzv file in the browser.



Now click on the 'Interactive Quality Plot' tab. You will see exactly that: Interactive Quality plots that look like this:



	forward reads	reverse reads
Minimum	317	317
Median	37652.0	37652.0
Mean	37206.7	37206.7
Maximum	58052	58052
Total	2232402	2232402

Note the total number of sequences. You want to retain a high percentage of these after quality filtering with DADA2.

STOP HERE. Inspect the visualisation and decide on location and maximum expected error.
From the output decide where to truncate the forward and reverse reads with

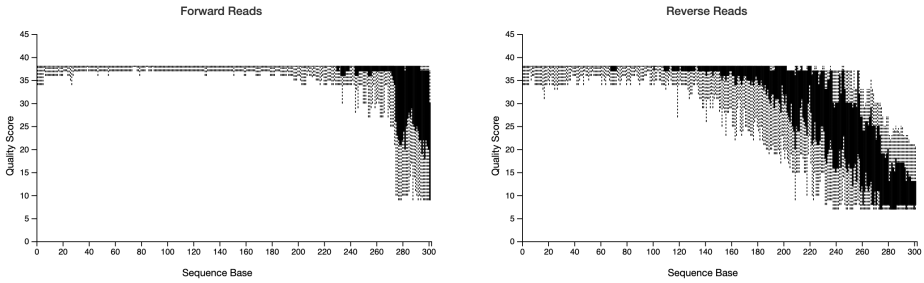


Figure 4.2: "Example of a demux.qzv visualisation"

`p-trunc-len-f`, `p-trunc-len-r`,`--p-max-ee-f` and `--p-max-ee-r` in DADA2 below.

It can take some trial and error to get these setting right. We are using a docker-based package called FIGARO to help us estimate those parameters (Not shown here). But essentially you want to capture high quality reads and be confident about the Amplicon Sequence Variants (ASV), while also capture sufficent depth of ASVs and reads without unnecessarily filtering out too much. In the below example, we have trimmed the forward reads at 272 base pairs with a maximum expected error (max-ee) of 2 (which is the default) and the reverse reads at 151 with a max-ee of 3. I think it is o.k. to relax the max-ee for the reverse reads (which are ALWAYS lower in quality) as I feel more confident about the fact that the reverse reads are paired with the forward reads. Pairing in itself provides increased confidence that the reads do in fact represent a biological relevant sequence. As always please comment on our GitHub discussion page if you have any suggestions here. Thanks!

A minimum overlap between the forward and reverse primer of 20 base pairs is recommended. To overlap can be calculated as following:

length forward read + length reverse read - length amplicon - truncated base-pairs forward read - truncated basepairs reverse reads = overlap

So, for example, if we picked `--p-trunc-len-f 272` and `--p-trunc-len-r 151`, we get

$$301 + 301 - 292 - 29 - 150 = 131\ bp\ overlap$$

In cases where the quality of the reverse reads is very poor, or the amplicon is too long for pairing to work, it is also acceptable to import, trim and denoise only the forward reads. The V4 primer of this example, is nice and short at 292 basepairs, so is great for pairing even at lower reverse-read qualities.

4.2.5 Denoise paired end sequences with dada2 (Step 3)

Once the trimming and max-ee parameters are decided, it is time to run the DADA2 function. This may take a while, depending on total number of samples.

The output will be a feature_table.qza and sample_rep_seqs.qza, containing the ASV abundances and their sequences respectively.

```
qiime dada2 denoise-paired \
  --i-demultiplexed-seqs trimmed_demux-paired-end.qza \
  --o-table feature_table.qza \
  --o-representative-sequences sample_rep_seqs.qza \
  --p-trim-left-f 0 --p-trim-left-r 0 \
  --p-trunc-len-f 271 \
  --p-trunc-len-r 151 \
    --p-max-ee-f 2 \
    --p-max-ee-r 3 \
  --output-dir dada2 \
  --verbose
```

Summarise and visualise the ASV abundances (feature_table.qza) in a .qzv file.

```
qiime feature-table summarize \
  --i-table feature_table.qza \
  --o-visualization feature_table.qzv \
  --m-sample-metadata-file samplesheet.tsv
```

Note: Look at the feature_table.qzv and record median reads per sample. Compare the total frequency after denoising with the total sequence count from the trimmed_demux-paired-end.qza. You hope to retain a high percentage of total sequences after denoising with DADA2. It is prudent to report that percentage in your methods.

4.2.6 Taxonomic classifier and assignment (Step 4)

The next step is to assign taxonomies to the sequences in the denoised sample sequences. Here, we use a pre-trained classifier that is based on the Silva database. This pre-trained classifier is available on the data resource page of Qiime (Most current link at the time of writing: https://docs.qiime2.org/2022.2/data-resources).

However, in case you used a different primer you would have to create the classifier yourself. Again there is great resource avaiable on https://docs.qiime2.org.

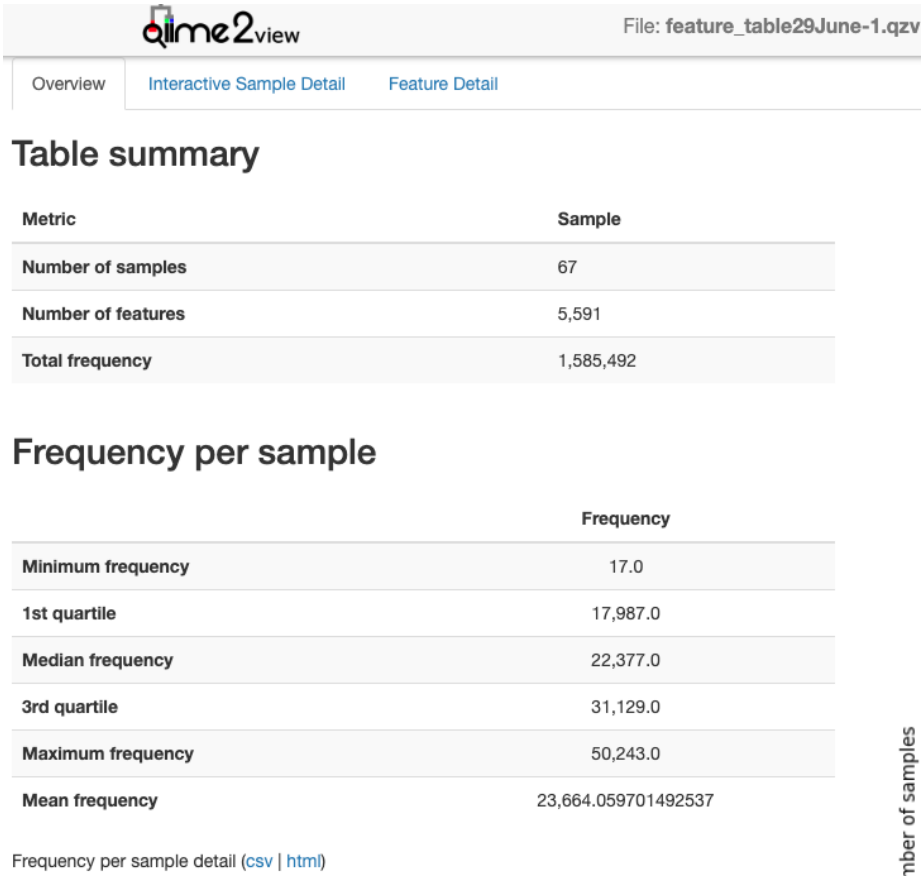


Figure 4.3: "feature_table.qzv output"

The process is fairly straight forward but takes computational time. Briefly, extract reference reads from a database (i.e. Silva here) based on the primers used. Then use those extracted sequences and fit or train them onto a taxonomy. Basically, predict which amplicon sequence should be what phylym/class/order/family/genus etc.... This trained file becomes a 'classifier' that is used to assign taxonmies on your sequences.

Here we have a pre-trained classifier, silva-132-99-515-806-nb-classifier.qza: The output is a file called taxonomy_silva.qza and taxonomy_silva.qzv in this case.

```
qiime feature-classifier classify-sklearn \
  --i-classifier silva-132-99-515-806-nb-classifier.qza \
  --p-reads-per-batch 10000 \
  --i-reads sample_rep_seqs.qza \
  --o-classification taxonomy_silva.qza \
  --quiet

# Then summarise and visualise the output into a .qza file

qiime metadata tabulate \
--m-input-file taxonomy_silva.qza \
--o-visualization taxonomy_silva.qzv
```

4.2.7 Build phylogenetic tree (Step 5)

The next step is not essential but really good to have. Creating a phylogenetic tree from the amplicon sequences.

In this case we are using the insertion tree method. See https://library.qiime2.org/plugins/q2-fragment-insertion/16/ for more information in this method.

As not all ASVs will be inserted we will filter the feature_table.qza again to keep only those ASVs that are in the tree. You will need the reference file from silva or greengenes. In this case we are using `sepp-refs-silva-128.qza`.

```
qiime fragment-insertion sepp \
  --i-representative-sequences sample_rep_seqs.qza \
  --i-reference-database sepp-refs-silva-128.qza \
  --o-tree insertion-tree.qza \
  --o-placements insertion-placements.qza

qiime fragment-insertion filter-features \
  --i-table feature_table.qza \
```

```
  --i-tree insertion-tree.qza \
  --o-filtered-table feature_table_insertiontreefiltered.qza \
  --o-removed-table removed_features.qza
```

Done!

Everything else including further quality filtering happens with `phyloseq` in R where we will import the following files: `feature_table_insertiontreefiltered.qza`, `taxonomy_silva.qza` and `insertion-tree.qza`.

This will be covered in the next chapter.

4.2.8 All steps combined

Copy and paste this into your script file if needed.

```
# Manifest Import
qiime tools import --type 'SampleData[PairedEndSequencesWithQuality]' \
  --input-path Qiimeimportmanifest.tsv \
  --output-path demux-paired-end.qza \
  --input-format PairedEndFastqManifestPhred33V2

# Cutadapt
qiime cutadapt trim-paired \
  --i-demultiplexed-sequences demux-paired-end.qza \
  --p-front-f GTGYCAGCMGCCGCGGTAA \
  --p-front-r GGACTACNVGGGTWTCTAAT \
  --o-trimmed-sequences trimmed_demux-paired-end.qza

qiime demux summarize \
  --i-data trimmed_demux-paired-end.qza \
  --o-visualization trimmed_demux-paired-end.qzv

# Denoise
qiime dada2 denoise-paired \
  --i-demultiplexed-seqs trimmed_demux-paired-end.qza \
  --o-table feature_table.qza \
  --o-representative-sequences sample_rep_seqs.qza \
  --p-trim-left-f 0 --p-trim-left-r 0 \
  --p-trunc-len-f 271 \
  --p-trunc-len-r 151 \
    --p-max-ee-f 2 \
    --p-max-ee-r 3 \
  --output-dir dada2 \
  --verbose
```

```
qiime feature-table summarize \
  --i-table feature_table.qza \
  --o-visualization feature_table.qzv \
  --m-sample-metadata-file samplesheet.tsv

# Taxonomic assignment
qiime feature-classifier classify-sklearn \
  --i-classifier silva-132-99-515-806-nb-classifier.qza \
  --p-reads-per-batch 10000 \
  --i-reads sample_rep_seqs.qza \
  --o-classification taxonomy_silva.qza \
  --quiet

# Phylogenetic tree
qiime fragment-insertion sepp \
  --i-representative-sequences sample_rep_seqs.qza \
  --i-reference-database sepp-refs-silva-128.qza \
  --o-tree insertion-tree.qza \
  --o-placements insertion-placements.qza

# Final filtering
qiime fragment-insertion filter-features \
  --i-table feature_table.qza \
  --i-tree insertion-tree.qza \
  --o-filtered-table feature_table_insertiontreefiltered.qza \
  --o-removed-table removed_features.qza
```

Qiime2 reference:
Bolyen E, Rideout JR, Dillon MR, Bokulich NA, Abnet CC, Al-Ghalith GA, Alexander H, Alm EJ, Arumugam M, Asnicar F, Bai Y, Bisanz JE, Bittinger K, Brejnrod A, Brislawn CJ, Brown CT, Callahan BJ, Caraballo-Rodríguez AM, Chase J, Cope EK, Da Silva R, Diener C, Dorrestein PC, Douglas GM, Durall DM, Duvallet C, Edwardson CF, Ernst M, Estaki M, Fouquier J, Gauglitz JM, Gibbons SM, Gibson DL, Gonzalez A, Gorlick K, Guo J, Hillmann B, Holmes S, Holste H, Huttenhower C, Huttley GA, Janssen S, Jarmusch AK, Jiang L, Kaehler BD, Kang KB, Keefe CR, Keim P, Kelley ST, Knights D, Koester I, Kosciolek T, Kreps J, Langille MGI, Lee J, Ley R, Liu YX, Loftfield E, Lozupone C, Maher M, Marotz C, Martin BD, McDonald D, McIver LJ, Melnik AV, Metcalf JL, Morgan SC, Morton JT, Naimey AT, Navas-Molina JA, Nothias LF, Orchanian SB, Pearson T, Peoples SL, Petras D, Preuss ML, Pruesse E, Rasmussen LB, Rivers A, Robeson MS, Rosenthal P, Segata N, Shaffer M, Shiffer A, Sinha R, Song SJ, Spear JR, Swafford AD, Thompson LR, Torres PJ, Trinh P, Tripathi A, Turnbaugh PJ, Ul-Hasan S, van der Hooft JJJ, Vargas F, Vázquez-Baeza Y, Vogtmann E, von Hippel M, Walters W, Wan Y, Wang M, Warren J, Weber KC, Williamson CHD, Willis AD, Xu ZZ, Zaneveld JR, Zhang Y, Zhu

Q, Knight R, and Caporaso JG. 2019. Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2. Nature Biotechnology 37: 852–857. https://doi.org/10.1038/s41587-019-0209-9

Chapter 5

From Qiime2 into R - Initial diagnostics

5.1 Introduction

In this chapter you will learn how to import Qiime2-produced ASV tables, taxonomy tables and tree files into R. For this exercise we will us publicly available FastQ files that were generated by a research group in Denmark. The files are available in the Sequence Read Archives (SRA) under accession number PRJNA645373.
Reference: [1] C. Jiang, S.J. McIlroy, R. Qi, F. Petriglieri, E. Yashiro, Z. Kondrotaite, P.H. Nielsen, Identification of microorganisms responsible for foam formation in mesophilic anaerobic digesters treating surplus activated sludge, Water Res. 191 (2021) 116779. https://doi.org/10.1016/j.watres.2020.116779.

Sample IDs
You decide on the samplesIDs before you start your library prep and sequencing. For this chapter we use a different dataset to the previous chatper and its sampleIDs are longer, e.g. "SRR12204258" and "SRR12204269", compared to "PT-01", "PT-02" in the previous chapter. It does not matter how you label each of your indexed samples as long as the IDs are unique. And each unique sample ID in the `samplesheet.csv` needs to match the sample IDs of the `feature_table.qza`. You may recall those IDs originate from your Miseq run and are the same sample IDs that are used on the loading manifest file to prepare the run. They subsequently are part of the FastQ filenames and flow through into any relevant Qiime2 data. If a sample ID does not match or if there is a different number of samples between these two input files, then phyloseq will complain.

Here the first three Sample IDs ("SRR12204258" and "SRR12204269" etc..) in the ASV table from this data set.:

	SRR12204258	SRR12204269	SRR12204280
2a4a6962ccd9e5d76122cf3b3335671a	1354	2778	26
04135cecc84e61a9192cd07cf8201b37	106	512	26
7851df9af5c9f989afcc8c4b71da3c58	114	0	467
72f4c77a923b4a2bc3d0a0eaed7b659c	561	192	203

ASV IDs
The same principle applies to each ID of the ASVs. The ASVs IDs have to match between the `feature_table.qza` and the `taxonomy_silva.qza`.

Following the first three ASV IDs (Feature ID) and related taxon assignments in taxonomy_silva.qza of this dataset:

	Feature.ID	Taxon	Confidence
1	2a4a6962ccd9e5d76122cf3b3335671a	d__Bacteria; p__Chloroflexi; c__Anaerolineae; o__Anaeroline...	1.0000000
2	04135cecc84e61a919	2a4a6962ccd9e5d76122cf3b3335671a __Bacteroidota; c__Bacteroidia; o__Bacteroidal...	0.9927922
3	7851df9af5c9f989afcc8c4b71da3c58	d__Bacteria; p__Chloroflexi; c__Anaerolineae; o__Anaeroline...	0.9215569
4	72f4c77a923b4a2bc3d0a0eaed7b659c	d__Bacteria; p__Bacteroidota; c__Bacteroidia; o__Bacteroidal...	0.9994765

NOTE: After the denoising, taxonomic classification and tree alignments in Qiime2, I prefer to do all subsequent analysis in R. However, it is also possible to do much of the diversity analysis in Qiime2. In fact, some interesting plug-ins and functions in Qiime2 are not available as packages in R, hence for some specific tasks you may need to keep using Qiime2. It is up to the goals and preferences of the investigator.

Now let's work in R!

Required files

- `samplesheet.tsv` - same file that was used in the visualisation steps in Qiime2

- `feature_table.qza` - if you created a phylogenetic tree from the sequences as described in the previous chapter then the ASVs in the feature table were filtered to match the ASVs in the tree. Hence the files was named `featuretable-insertiontree-filtered.qza` to differentiate it.

- `taxonomy_silva.qza`

- `insertion-tree.qza`

5.2 Workflow

5.2.1 Packages

First install the required packages. Some packages are stored on 'remote' repositories such as GitHub, hence before you can install them you need helper packages such as `remotes`. Other packages, such as `phyloseq` are managed by BioConductor project. They provide their own package manager package for R called `BiocManager`, which helps to install from the same release.

Learn more about phyloseq here: https://joey711.github.io/phyloseq/

Both, remotes and BiocManager are already installed in the below example. Once the packages are installed, load them into your working space.

```
# install.packages("remotes")
# if (!requireNamespace("BiocManager", quietly = TRUE))
#  install.packages("BiocManager")
# remotes::install_github("jbisanz/qiime2R")
# BiocManager::install("phyloseq")

library(qiime2R)  # to import qiime.qza into an R object
library(phyloseq) # To combine all relevant data objects into one object for easy data management
library(tidyverse) # Compilation of packages for data management
library(stringr)  # to change some of the strings in taxonomic names
library(vegan)   # A commonly used package for numerical ecology
```

5.2.2 Import qiime-files and create a phyloseq object

Now that packages are loaded we can import the Qiime2 `.qza` and the `.tsv` files. The aim is to import everything as R objects, which are then combined into one phyloseq object. Once we have a phyloseq object, any filtering and visualisations can be run from the one object.

In the previous chapter, we used a `samplesheet.tsv` to visualise and summarise the `feature_table.qza`. That exact same samplesheet.tsv is imported here and slightly changed to make it phyloseq friendly. In this example, all relevant Qiime2 outputs were copied into a folder named 'qiimefiles'.

```
# Sample sheet
metadata <- read_tsv("qiimefiles/samplesheet.tsv")  # lets call the R object 'metadata
# Inspect the metadata object. The second row is qiime-specific information and has to

metadata2 <- metadata[c(2:nrow(metadata)),] %>%     # remove the top row and convert c
  rownames_to_column("spl")%>% # this just adds the rownames to a column and calles in
  mutate_all(type.convert) %>%
  mutate_if(is.factor, as.character) %>%  # reformatting columns to avoid any problems
  column_to_rownames("spl") %>%  # this puts the column "spl" back into rownames
  as_tibble() %>%
  column_to_rownames("#SampleID")

# str(metadata2)  # use the `str` command to inspect the data

# Then decide which of the columns you want to be factors and in which order these fac
metadata2  <- metadata2 %>%
  mutate(Reactor = factor(Reactor)) %>%  # replace column Reactor with the same values
  mutate(Location = factor(loc))     # create a new column 'Location', which is a facto

# Qiime import
SVs <-  qiime2R::read_qza("qiimefiles/featuretable-insertiontree-filtered.qza")
ASVtable <- as.data.frame(SVs$data)   # extract the data table, this has to be a dataf

# Taxonomy
taxonomy <- qiime2R::read_qza("qiimefiles/taxonomy_silva.qza") # import the qiime obje
taxonomy <- taxonomy$data  # extract the taxonomy data
## re-format the taxonomy file to split the taxonomy into columns
## remove the confidence column
taxtable <- taxonomy %>% as_tibble() %>%
  separate(Taxon, sep=";", c("Kingdom","Phylum","Class","Order","Family","Genus","Speci
  dplyr::select(-Confidence) %>%
  column_to_rownames("Feature.ID") %>%
  as.matrix()

# Tree
tree <- read_qza("qiimefiles/insertion-tree.qza")
tree <- tree$data  #extract data

# Create the phyloseq object
ps <-phyloseq(
  otu_table(ASVtable, taxa_are_rows = T),
  sample_data(metadata2),
  phyloseq::tax_table(taxtable),
  phyloseq::phy_tree(tree)
)
```

```
# ps  (not run)
# phyloseq-class experiment-level object
# otu_table()   OTU Table:         [ 4218 taxa and 51 samples ]
# sample_data() Sample Data:       [ 51 samples by 14 sample variables ]
# tax_table()   Taxonomy Table:    [ 4218 taxa by 7 taxonomic ranks ]
# phy_tree()    Phylogenetic Tree: [ 4218 tips and 4217 internal nodes ]

# remove things out of the R environment you dont need.
rm(metadata, metadata2, SVs, taxonomy, taxtable, tree, ASVtable)

## Remove those annoying short codes in front of taxa names (i.e. p__ etc) as they
## dont look good in visualisation
tax_table(ps)[, "Kingdom"] <- str_replace_all(tax_table(ps)[, "Kingdom"], "d__", "")
tax_table(ps)[, "Phylum"] <- str_replace_all(tax_table(ps)[, "Phylum"], " p__", "")
tax_table(ps)[, "Class"] <- str_replace_all(tax_table(ps)[, "Class"], " c__", "")
tax_table(ps)[, "Order"] <- str_replace_all(tax_table(ps)[, "Order"], " o__", "")
tax_table(ps)[, "Family"] <- str_replace_all(tax_table(ps)[, "Family"], " f__", "")
tax_table(ps)[, "Genus"] <- str_replace_all(tax_table(ps)[, "Genus"], " g__", "")
tax_table(ps)[, "Species"] <- str_replace_all(tax_table(ps)[, "Species"], " s__", "")
```

From the output of the `ps` object we can see that there are 4281 ASVs in 51 samples and associated with 14 variables in the metadata.

If you wish, you can access each individual data set from the ps object with functions `otu_table(ps)@.Data`, `sample_data(ps)` `tax_table(ps)@.Data` or `phy_tree(ps)`. This can become handy if you want to change something, such as adding columns to the sample_data or changing them to a factor etc..

```
ASVs <- otu_table(ps)@.Data
metadata <- data.frame(sample_data(ps))
taxtable <- data.frame(tax_table(ps)@.Data)
tree <- phy_tree(ps)

# Inspect individual objects in your own time.
```

5.2.3 Save the `ps` object as an `.rds` file into your working directory

Optional
This step allows you to save and store the R object as a RDS file and load it back

into any future R environment, should you require it. This would be helpful if your workflow is separated into different Rscripts and you want to just load in the RDS file instead of having to import qiime files and create the phyloseq object every time.

```
saveRDS(ps, file = "ps_ProjectX_2022July")

# to load the file back into your R environment:
# ps <- readRDS(file = "ps_ProjectX_2022July")
```

5.2.4 Initial filtering

Keep the orginal `ps` object unchanged. Create new objects for subsequent filtering steps.

Often we simply filter out any ASVs that have less than x number of reads and are present in less than x number of samples. This helps to reduce noise and the overload of zeros in the dataset, which some differential abundance tests can't deal with. Perhaps, it also helps with removing ASVs that were incorrectly denoised with DADA2 in the first place. However, it also means that rare ASVs with a low prevalence are not considered. So filtering has to be done with consideration to the intended analysis.

Then filter out any phyla that came up as uncharacterized and taxa that came up as mitochondria and chloroplast.

```
# filter as needed. This will be your final otu-table.
# Although you can still filter for specific analysis if needed.
# minimum of reads per feature
ps.flt = prune_taxa(taxa_sums(ps) >= 5, ps) #minimum reads per feature

# ps.flt (not run)
# phyloseq-class experiment-level object
# otu_table()   OTU Table:         [ 3911 taxa and 51 samples ]
# sample_data() Sample Data:       [ 51 samples by 14 sample variables ]
# tax_table()   Taxonomy Table:    [ 3911 taxa by 7 taxonomic ranks ]
# phy_tree()    Phylogenetic Tree: [ 3911 tips and 1694 internal nodes ]

#filter any "NA"-phyla that have not been classified i.e.
# contain nothing in the phylum column of the taxtable (just a <NA>)

ps.flt  = subset_taxa(ps.flt , !is.na(Phylum) & !Phylum %in% c(""))

# ps.flt (not run)
# phyloseq-class experiment-level object
```

```
# otu_table()   OTU Table:         [ 3892 taxa and 51 samples ]
# sample_data() Sample Data:       [ 51 samples by 14 sample variables ]
# tax_table()   Taxonomy Table:    [ 3892 taxa by 7 taxonomic ranks ]
# phy_tree()    Phylogenetic Tree: [ 3892 tips and 1687 internal nodes ]

# Filter out non-bacteria, mitochondia and chloroplast taxa
ps.flt <- ps.flt %>%
  subset_taxa(Kingdom == "Bacteria" & Family  != "Mitochondria" & Class   != "Chloroplast")

# ps.flt (not run)
# phyloseq-class experiment-level object
# otu_table()   OTU Table:         [ 3853 taxa and 51 samples ]
# sample_data() Sample Data:       [ 51 samples by 14 sample variables ]
# tax_table()   Taxonomy Table:    [ 3853 taxa by 7 taxonomic ranks ]
# phy_tree()    Phylogenetic Tree: [ 3853 tips and 1673 internal nodes ]

# We can filter more for running beta diversity or differential abundance analysis later. This wi
# For example:
# minimum presence in x% of samples (51 * 0.04 = 2 samples)
# ps.flt = filter_taxa(ps.flt, function(x) sum(x > 0) > (0.04*length(x)), TRUE)
# Would result in
# phyloseq-class experiment-level object
# otu_table()   OTU Table:         [ 1674 taxa and 51 samples ]
# sample_data() Sample Data:       [ 51 samples by 14 sample variables ]
# tax_table()   Taxonomy Table:    [ 1674 taxa by 7 taxonomic ranks ]
# phy_tree()    Phylogenetic Tree: [ 1674 tips and 1673 internal nodes ]
```

5.2.5 Proportion of ASVs identified

Check what proportion of ASVs could be identified on phylum and genus-level. This is best done with the handy phyloseq function `ntaxa()`, which gives the number of taxa present in the phyloseq object.

This will indicate how many reads you would be discarding, in case you remove ASVs, which were not assigned a taxon.

It is also an indicator of how well the database you have been using (e.g. Silva here) has performed.You could compare that with other databases if you wish.

```
#  percent of phyla identified
ps.flt  = subset_taxa(ps , !is.na(Phylum) & !Phylum %in% c(""))
```

```
# percent of phyla identified
ntaxa(ps.flt) / ntaxa(ps)
```

```
## [1] 0.9947843
```

```
ps.flt  = subset_taxa(ps , !is.na(Genus) & !Genus %in% c(""))
# percent of phyla identified
ntaxa(ps.flt) / ntaxa(ps)
```

```
## [1] 0.940256
```

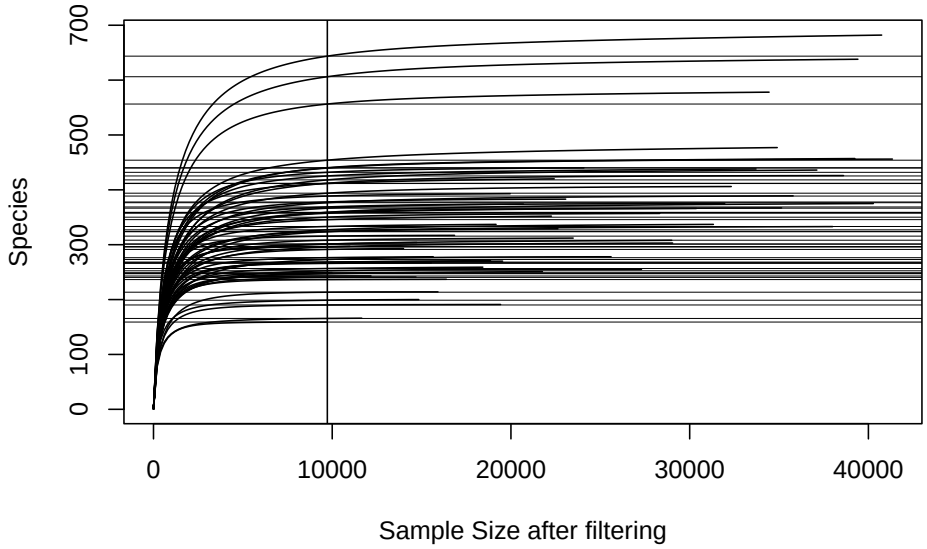
5.2.6 Rarefaction curve

Check if you cover enough depth for diversity analyses. Some samples may have low species richness, including a negative or samples that failed during PCR for example. The rarefaction curve will help assess which samples you may have to exclude from subsequent diversity calculations. It also provides a general view if samples cover enough of the potential richness (Number of ASVs).

The curve shows you the number of 'Species' (i.e. ASVs) on the y axis and the total reads on the x axis for each sample. So basically, it shows how rich each sample is and indicates if the number of reads that are captured in a samples covers enough of the ASVs. If the curve flattens, it indicates that the sample would not get any richer even with more reads...

I typically use the rarefaction curve to check if there are samples that need removing before establishing alpha diversity metrics. In this example, the vertical line helps identify the sample with lowest number of reads. We can check that by running `min(colSums(otu_table(ps.flt)))`, which tells us that they are 9690 reads in the sample with lowest reads. If this sample is kept in and we measure subsequent alpha diversity metrics, where each of the sample-reads are randomly resampled to the lowest read-number (here 9690), then we may not capture the diversity of samples with higher richness. The grey horizontal lines indicate how much richness we may lose if we include the sample with lowest number of reads (the difference between grey horizontal line and the curve of each sample at the highest number of reads on x-axis).

```
vegan::rarecurve(t(otu_table(ps.flt)),
                 step=200, sample = min(colSums(otu_table(ps.flt))),
                 label = FALSE, xlab = "Sample Size after filtering")
```




```
# show the lowest and highest number of sample reads.
max(colSums(otu_table(ps.flt)))
```

```
## [1] 41342
```

```
min(colSums(otu_table(ps.flt)))
```

```
## [1] 9726
```

```
# Filtering options
# filter options (not run), you can filter out specific sample or treatments from the phyloseq ob
# `%notin%` = Negate(`%in%`)  # create a %notin% filter operator for filtering

# Filtering out treatments, e.g. any sample that was labeled "Negative" in the metadata
# ps.flt.flt <-  prune_samples(sample_data(ps.flt)$Treatment %notin% c("Negative"), ps.flt)

# Filtering out any taxa that have zero reads because they were only present in the removed sampl
# ps.flt.flt <- prune_taxa(taxa_sums(ps.flt.flt) != 0, ps.flt.flt)
```

Once you identified the samples you would like to remove (e.g. samples with too few reads), you can filter out these specific samples or treatments from the phyloseq objects. For example, with the `prune_samples(sample_data(ps.flt)$Treatment %notin% c("Negative"), ps.flt)` command here we would sample out any samples labeled as "Negative" in the metadata. Afterwards make sure to sample out any taxa that now contain zero reads because they were only present in the removed samples. Perhaps, create new phyloseq objects to differentiate from

your different filtered object `ps.flt.flt <- prune_taxa(taxa_sums(ps.flt) != 0, ps.flt)`. See also Chapter 7 where the phyloseq object is pre-filtered before ggplotting.

In this example, we are keeping the sample with lowest reads. No further filtering required at this stage.

5.2.7 Prevalence table

Create a prevalence table to get an overview of phyla with low to high prevalence.

The output is a dataframe showing all phyla in order of total abundance. The mean prevalence defines how often the phylum appears across all samples on average. E.g. a mean prevalence of 10.22 means that ASVs in this phylum were present on average in 10.22 samples. Higher values indicates a 'core' relevance to the sum of samples.

This table is handy to decide if you want to remove certain phyla from some visualisations as they may not contribute to the overall analysis. It also allows for a comparison with filtered and unfiltered data for due diligence. Here it is apparent that phyla that were unclassified (i.e.) contributed 3,522 reads to the total of 1,299,791 reads. That is `3522 / sum(phyloseq::otu_table(ps)) * 100` = 0.27%. Chances are that these NA sequences may just be noise and may not represent biological relevant amplicons. Hence, I would keep them out of any further analysis.
However, in environments where you get higher contributions of unknown amplicons, it may indicate that the taxonomic database does not capture your samples well. In that case it would be wise to keep the NAs in for measuring alpha and beta diversity.

```
prevalencedf = apply(X = phyloseq::otu_table(ps.flt),
                     MARGIN = 1,
                     FUN = function(x){sum(x > 0)})

prevalencedf  = data.frame(Prevalence = prevalencedf ,
                           TotalAbundance = taxa_sums(ps.flt),
                           phyloseq::tax_table(ps.flt))

prevalencedf <- prevalencedf %>%
    dplyr::group_by( Phylum ) %>%   # choose phylym level
    dplyr::summarise(Mean_prevalence = mean(Prevalence),
                     Total_abundance = sum(TotalAbundance)) %>%
    dplyr::mutate(Rel_abundance = (Total_abundance / sum(Total_abundance) *100)) %>%
    arrange(desc(Total_abundance)) %>%
```

```
    dplyr::mutate(Cumulated = cumsum(Rel_abundance))

# print table from highest to lowest abundance
knitr::kable(prevalencedf, digits = 1) %>%
  kableExtra::kable_styling(bootstrap_options = c("striped", "condensed", "responsive"), full_wid
```

```
## Compare with the original, unfiltered ps object
prevalencedf = apply(X = phyloseq::otu_table(ps),
                     MARGIN = 1,
                     FUN = function(x){sum(x > 0)})

prevalencedf = data.frame(Prevalence = prevalencedf,
                          TotalAbundance = taxa_sums(ps),
                          phyloseq::tax_table(ps))

prevalencedf <- prevalencedf %>%
    dplyr::group_by( Phylum ) %>%   # choose phylym level
    dplyr::summarise(Mean_prevalence = mean(Prevalence),
                     Total_abundance = sum(TotalAbundance)) %>%
    dplyr::mutate(Rel_abundance = (Total_abundance / sum(Total_abundance) *100)) %>%
    arrange(desc(Total_abundance)) %>%
    dplyr::mutate(Cumulated = cumsum(Rel_abundance))

knitr::kable(prevalencedf, digits = 1) %>%
  kableExtra::kable_styling(bootstrap_options = c("striped", "condensed", "responsive"),full_widt
```

5.2.8 Export ASV and taxa tables to Excel

This might be useful if you would like to share your abundance and taxa tables with someone that does not use R.

```
# extract ASV table from ps object
ASVs <- data.frame(otu_table(ps.flt)) %>%
  rownames_to_column("FeatureID")

# extract taxonomy table from ps object
taxtable <- data.frame(tax_table(ps.flt))
# combine the taxonomy levels into one colum, separated by a ;
taxtable <- taxtable %>%
  rownames_to_column("FeatureID") %>% as_tibble() %>%
  unite(Taxon, sep=";", c("Kingdom","Phylum","Class","Order","Family","Genus","Species"))
# combine ASVs and taxatable into one object
```

Phylum	Mean_prevalence	Total_abundance	Rel_abundance	Cumu
Chloroflexi	4.9	228035	18.1	
Bacteroidota	4.5	199720	15.9	
Firmicutes	4.3	179454	14.3	
Actinobacteriota	5.1	161701	12.9	
Proteobacteria	4.1	151650	12.1	
Desulfobacterota	4.9	57139	4.5	
Synergistota	6.7	51049	4.1	
Cloacimonadota	6.7	43842	3.5	
Acidobacteriota	3.8	35158	2.8	
Thermotogota	10.2	33489	2.7	
Spirochaetota	4.5	24593	2.0	
Verrucomicrobiota	3.0	16246	1.3	
Fermentibacterota	35.0	14812	1.2	
Patescibacteria	3.2	14454	1.1	
Caldatribacteriota	5.3	8934	0.7	
Planctomycetota	2.4	5158	0.4	
Armatimonadota	2.6	4462	0.4	
Marinimicrobia_(SAR406_clade)	20.0	3818	0.3	
Hydrogenedentes	4.0	3411	0.3	
Nitrospirota	8.8	2853	0.2	
SAR324_clade(Marine_group_B)	3.3	2622	0.2	
WS1	3.6	2559	0.2	
Campilobacterota	3.9	2465	0.2	
Fibrobacterota	2.7	1874	0.1	
Myxococcota	2.8	1366	0.1	
Bdellovibrionota	2.4	1133	0.1	
Caldisericota	3.6	1071	0.1	
WPS-2	2.7	897	0.1	
Sumerlaeota	3.6	686	0.1	
Cyanobacteria	9.0	647	0.1	
Fusobacteriota	2.9	565	0.0	
Coprothermobacterota	2.8	511	0.0	
Gemmatimonadota	2.7	304	0.0	
Elusimicrobiota	2.8	225	0.0	
CK-2C2-2	4.0	162	0.0	
Methylomirabilota	4.0	159	0.0	
Dependentiae	1.9	90	0.0	
Latescibacterota	2.0	89	0.0	
Zixibacteria	5.0	74	0.0	
Sva0485	2.0	69	0.0	
Acetothermia	2.0	59	0.0	
WS4	1.2	46	0.0	
Hydrothermae	1.0	26	0.0	
Deinococcota	1.0	11	0.0	
FCPU426	1.0	3	0.0	
Margulisbacteria	1.0	3	0.0	
MBNT15	1.0	2	0.0	

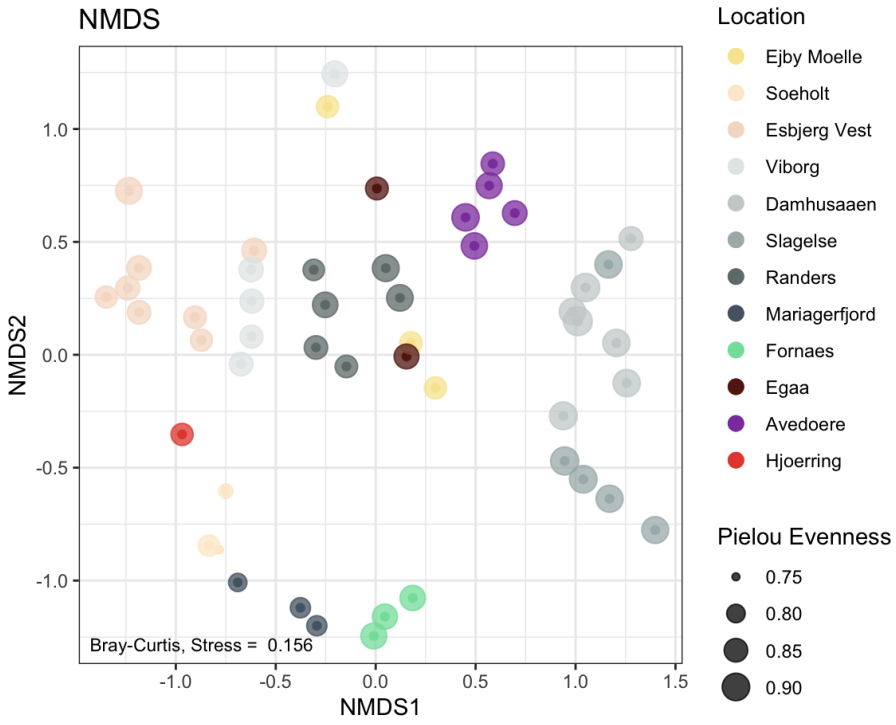
Phylum	Mean_prevalence	Total_abundance	Rel_abundance	Cumulated
Chloroflexi	4.9	228276	17.6	17.6
Bacteroidota	4.5	202022	15.5	33.1
Firmicutes	4.3	197782	15.2	48.3
Actinobacteriota	5.0	165791	12.8	61.1
Proteobacteria	4.0	161897	12.5	73.5
Desulfobacterota	4.8	57187	4.4	77.9
Synergistota	6.5	51227	3.9	81.9
Cloacimonadota	6.4	45296	3.5	85.4
Acidobacteriota	3.8	35318	2.7	88.1
Thermotogota	10.2	33489	2.6	90.7
Spirochaetota	4.4	25028	1.9	92.6
Verrucomicrobiota	3.0	16649	1.3	93.9
Fermentibacterota	35.0	14812	1.1	95.0
Patescibacteria	3.1	14501	1.1	96.1
Caldatribacteriota	5.3	8934	0.7	96.8
Planctomycetota	2.4	5478	0.4	97.2
Armatimonadota	2.6	4462	0.3	97.6
Marinimicrobia_(SAR406_clade)	20.0	3818	0.3	97.9
NA	3.6	3522	0.3	98.1
Hydrogenedentes	4.0	3411	0.3	98.4
Nitrospirota	8.8	2853	0.2	98.6
Campilobacterota	4.2	2710	0.2	98.8
SAR324_clade(Marine_group_B)	3.3	2622	0.2	99.0
WS1	3.6	2559	0.2	99.2
Fibrobacterota	2.8	1942	0.1	99.4
Myxococcota	2.7	1369	0.1	99.5
Bdellovibrionota	2.4	1133	0.1	99.6
Caldisericota	3.6	1071	0.1	99.6
WPS-2	2.7	897	0.1	99.7
Sumerlaeota	3.6	686	0.1	99.8
Cyanobacteria	9.0	647	0.0	99.8
Fusobacteriota	2.9	565	0.0	99.9
Coprothermobacterota	2.8	511	0.0	99.9
Gemmatimonadota	2.7	304	0.0	99.9
Elusimicrobiota	2.8	225	0.0	99.9
CK-2C2-2	4.0	162	0.0	100.0
Methylomirabilota	4.0	159	0.0	100.0
Dependentiae	1.8	94	0.0	100.0
Latescibacterota	2.0	89	0.0	100.0
Zixibacteria	5.0	74	0.0	100.0
Sva0485	2.0	69	0.0	100.0
Acetothermia	2.0	59	0.0	100.0
WS4	1.2	46	0.0	100.0
Hydrothermae	1.0	26	0.0	100.0
Deinococcota	1.0	11	0.0	100.0
FCPU426	1.0	3	0.0	100.0
Margulisbacteria	1.0	3	0.0	100.0
MBNT15	1.0	2	0.0	100.0

```
ASVTable <- ASVs %>% left_join(taxtable, by = "FeatureID")

# write .csv file
# write.csv(ASVTable, "ASVTable.csv") not run
```

Note:
This code is an amalgamation from various sources. Apart from putting it together into this workflow I do not take credit for it.

Chapter 6

Plotting diversity



Some visualisations of alpha and beta diversity.

6.1 Introduction

This chapter introduces you to a basic workflow for plotting alpha and beta diversity using phyloseq and ggplot. It provides examples for loading the phyloseq object from a previously saved .rds file, then how to calculate and include alpha diversity metrics into the sample_data of the phyloseq object, and finally how to plot some of these metrics, individually or combined.

Furthermore, beta diversity, i.e. four different type of ordinations are plotted. The examples, include non-metric dimensional scaling on Bray-Curtis dissimilaries, Principal Component Analysis (Aitchison distance), Redundancy Analysis (Aitchison) and a Principal Coordinate Analysis on unweighted uniFrac distances.

The stats behind the individual metrics are not discussed. If you want to learn more about the various options to analyse your amplicon-derived diversity, perhaps start here: https://sites.google.com/site/mb3gustame/home, the GUide to STatistical Analysis in Microbial Ecology (GUSTA ME). This is one of many other useful links and resources. Some others are listed in chapter 2 under "Workflows from other lab groups".

As mentioned in Chapter 2, prior to establishing alpha diversity indices, as well as Bray-Curtis dissimilarities, we will stick to rarefying sequences to a specific library size (library size = sum of sequence counts of a sample) to normalise the data and prevent bias due to library size. Usually, this is done using the smallest library size of all samples. If you want to learn more about rarefying - this paper explains the process: Enhancing diversity analysis by repeatedly rarefying next generation sequencing data describing microbial communities (Cameron et al. (2021)).
We will follow the recommendations from the authors of this paper to resample without replacement.

Note that rarefying is generally not recommended for statistical inferences in differential abundance analyses due to the loss of statistical power. Learn why here: Waste Not, Want Not: Why Rarefying Microbiome Data Is Inadmissible (McMurdie and Holmes (2014)).

Other normalisation methods (see Chapter 2) are also possible and it is up to you, the analyst, to understand and decide on the appropriate methods.

My preference is to use centred-log ratios (Aitchison distances). Aitchison distances are used in the workflow below for the PCA and RDA. The compositional approach and centred log-ratios are further discussed in the paper Microbiome Datasets Are Compositional: And This Is Not Optional (Gloor et al. (2017)).
In fact, comparing the results after applying different types of normalisation methods, dissimilarities or distances may complement your analysis and help in interpretation of data.

6.1.1 Required files

- A `phyloseq` object; either read in from a pre-saved `.rds` file or created as described in chapter 5 under "Import qiime-files and create a phyloseq object".
- If you dont have your own data you can download the pre-saved `.rds` object from this Chapter and follow the below steps: ps_ProjectX_2022July.

Time to get started...

6.2 Workflow

6.2.1 Packages

Install the following packages if not already.

```
# BiocManager::install("microbiome") If needed
library(phyloseq)
library(ggpubr)       # a handy helper package for ggplots
library(tidyverse)
theme_set(theme_bw())  # setting the theme for ggplots
library(vegan)
library(microbiome)
```

6.2.2 Load phyloseq object

For detail on how to create a phyloseq object please see chapter 5.

```
# reading in a previously saved phyloseq object
ps <- readRDS('ps_ProjectX_2022July')
ps  # get an overview of number of taxa and samples contained in the phyloseq object
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 4218 taxa and 51 samples ]
## sample_data() Sample Data:       [ 51 samples by 15 sample variables ]
## tax_table()   Taxonomy Table:    [ 4218 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 4218 tips and 4217 internal nodes ]
```

Check the metadata contained in the phyloseq object. This is the original metadata from the public repository where the FastQ files were downloaded from (Sequence Read Archives).

There are 15 columns containing some character (chr), factor or integer (int) variables.

```
metadf <- data.frame(sample_data(ps))
str(metadf)
```

```
## 'data.frame':    51 obs. of  15 variables:
##  $ Assay.Type     : chr  "AMPLICON" "AMPLICON" "AMPLICON" "AMPLICON" ...
##  $ Bases          : int  13254234 24196788 15758554 20048406 37634632 43555904 3510
##  $ BioProject     : chr  "PRJNA645373" "PRJNA645373" "PRJNA645373" "PRJNA645373" ..
##  $ Bytes          : int  9235913 16404070 10758456 13009806 24207581 27963491 22585
##  $ Organism       : chr  "anaerobic digester metagenome" "anaerobic digester metage
##  $ collection_date: chr  "30/11/16" "9/3/16" "17/3/16" "5/12/16" ...
##  $ loc            : chr  "Ejby Moelle" "Soeholt" "Esbjerg Vest" "Viborg" ...
##  $ Reactor        : Factor w/ 10 levels "1A","1B","2B",..: 4 4 6 7 7 5 5 5 4 4 ...
##  $ Type           : chr  NA NA NA "DS" ...
##  $ Purpose        : chr  "reactorfoampotentials" "reactorfoampotentials" "reactorfo
##  $ Experiment     : chr  "SRX8715180" "SRX8715169" "SRX8715158" "SRX8715151" ...
##  $ lat_lon        : chr  "55.398077 N 10.415041 E" "56.175302 N 9.582588 E" "55.488
##  $ ReleaseDate    : chr  "2021-01-01T00:00:00Z" "2021-01-01T00:00:00Z" "2021-01-01T
##  $ Sample.Name    : chr  "16SAMP-17236" "16SAMP-17231" "16SAMP-17233" "MQ170915-14"
##  $ Location       : Factor w/ 12 levels "Avedoere","Damhusaaen",..: 4 11 5 12 12 12
```

6.2.3 Pre-filter, rarefy and calculate diversity

Now the diversity indices can be calculated and added to the metadata of the phyloseq object for plotting. First, the abundances are pre-filtered (as shown in the previous chapter), then the ASV abundances are randomly resampled with `phyloseq::rarefy_even_depth`, such that all samples have the same number of ASVs.
The diversity indices are then calculated with `phyloseq::estimate_richness` based on these evenly resampled abundances. The dataframe that is created containing the range of indices is then added to the existing metadata of the phyloseq for plotting.

```
# some additional quality filtering - repeating steps from previous chapter.

ps.flt = prune_taxa(taxa_sums(ps) >= 5, ps) #minimum reads per feature
ps.flt  = subset_taxa(ps.flt , !is.na(Phylum) & !Phylum %in% c(""))
ps.flt <- ps.flt %>%
  subset_taxa(Kingdom == "Bacteria" & Family  != "Mitochondria" & Class   != "Chloroplast")
ps.flt
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 3853 taxa and 51 samples ]
## sample_data() Sample Data:       [ 51 samples by 15 sample variables ]
## tax_table()   Taxonomy Table:    [ 3853 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 3853 tips and 3852 internal nodes ]
```

```
# confirming minimum sample size
min(colSums(otu_table(ps.flt)))
```

```
## [1] 9770
```

```
# quantile summary of sample sizes
quantile(colSums(otu_table(ps.flt)))
```

```
##      0%     25%     50%     75%    100%
##  9770.0 16278.5 23035.0 34173.0 42345.0
```

```
# rarefy - required before measuring alpha diversity.
# This step is to create an abundance table, where the all samples are randomly
# resamples such that all samples have the same number of ASVs.
# It removes any bias but number of re-samples is based on the sample with
# smallest number of ASVs so one has to check and filter beforehand in case
# there are samples with very different number of ASVs. I am generally going
# by a 10x rule. If the lowest number of ASVs is 10x lower than the sample with
# highest number of ASVs, then I keep that sample in for alpha diversity assessments.

ps_rare <- phyloseq::rarefy_even_depth(ps.flt,
    sample.size = min(colSums(otu_table(ps.flt))),
    rngseed = TRUE,
    replace = FALSE, #without replacement
    trimOTUs = TRUE)

# check rarefied phyloseq object
# ps_rare (not run)
```

```
# phyloseq-class experiment-level object
# otu_table()   OTU Table:         [ 3801 taxa and 51 samples ]
# sample_data() Sample Data:       [ 51 samples by 15 sample variables ]
# tax_table()   Taxonomy Table:    [ 3801 taxa by 7 taxonomic ranks ]
 #phy_tree()    Phylogenetic Tree: [ 3801 tips and 3800 internal nodes ]

# Create diversity indices
div.df <- phyloseq::estimate_richness(ps_rare)
# add sample names
div.df$`#SampleID` <- phyloseq::sample_names(ps.flt)
# add diversity indices to phyloseq object
metadf <- data.frame(sample_data(ps.flt))  # export metadata first
metadf <- metadf %>%
  rownames_to_column("#SampleID") %>%
  left_join(div.df, by = "#SampleID") %>%
  column_to_rownames("#SampleID") # add the diversity indices (columns) to the metadat
# add pilou evenness (see calculation in "Numerical Ecology with R - Bocard et al")
metadf$pielou_ev <- div.df$Shannon/log(div.df$Observed)
# replace metadata with temp meta.df which includes the diversity indices
ps.flt@sam_data <- sample_data(metadf)
ps_rare@sam_data <- sample_data(metadf)
# this replaced the old metadata with new metadata that includes the diversity
# indices for this phyloseq object
str(data.frame(sample_data(ps.flt)))
```

```
## 'data.frame':    51 obs. of  25 variables:
##  $ Assay.Type     : chr  "AMPLICON" "AMPLICON" "AMPLICON" "AMPLICON" ...
##  $ Bases          : int  13254234 24196788 15758554 20048406 37634632 43555904 3510
##  $ BioProject     : chr  "PRJNA645373" "PRJNA645373" "PRJNA645373" "PRJNA645373" ..
##  $ Bytes          : int  9235913 16404070 10758456 13009806 24207581 27963491 22585
##  $ Organism       : chr  "anaerobic digester metagenome" "anaerobic digester metage
##  $ collection_date: chr  "30/11/16" "9/3/16" "17/3/16" "5/12/16" ...
##  $ loc            : chr  "Ejby Moelle" "Soeholt" "Esbjerg Vest" "Viborg" ...
##  $ Reactor        : Factor w/ 10 levels "1A","1B","2B",..: 4 4 6 7 7 5 5 5 4 4 ...
##  $ Type           : chr  NA NA NA "DS" ...
##  $ Purpose        : chr  "reactorfoampotentials" "reactorfoampotentials" "reactorfoa
##  $ Experiment     : chr  "SRX8715180" "SRX8715169" "SRX8715158" "SRX8715151" ...
##  $ lat_lon        : chr  "55.398077 N 10.415041 E" "56.175302 N 9.582588 E" "55.488
##  $ ReleaseDate    : chr  "2021-01-01T00:00:00Z" "2021-01-01T00:00:00Z" "2021-01-01T
##  $ Sample.Name    : chr  "16SAMP-17236" "16SAMP-17231" "16SAMP-17233" "MQ170915-14"
##  $ Location       : Factor w/ 12 levels "Avedoere","Damhusaaen",..: 4 11 5 12 12 12
##  $ Observed       : num  161 354 247 265 389 454 406 304 278 371 ...
##  $ Chao1          : num  161 356 247 266 395 ...
##  $ se.chao1       : num  0 2.2689 0.0832 1.0272 3.914 ...
##  $ ACE            : num  161 357 247 266 395 ...
```

```
##  $ se.ACE         : num  4.6 9.28 7.43 7.72 9.83 ...
##  $ Shannon        : num  4.24 4.84 4.74 4.78 5.03 ...
##  $ Simpson        : num  0.966 0.973 0.982 0.983 0.986 ...
##  $ InvSimpson     : num  29.6 37.7 56.8 57.8 70.6 ...
##  $ Fisher         : num  27.4 72 46.1 50.2 81 ...
##  $ pielou_ev      : num  0.834 0.825 0.861 0.857 0.843 ...
```

6.2.4 Colors

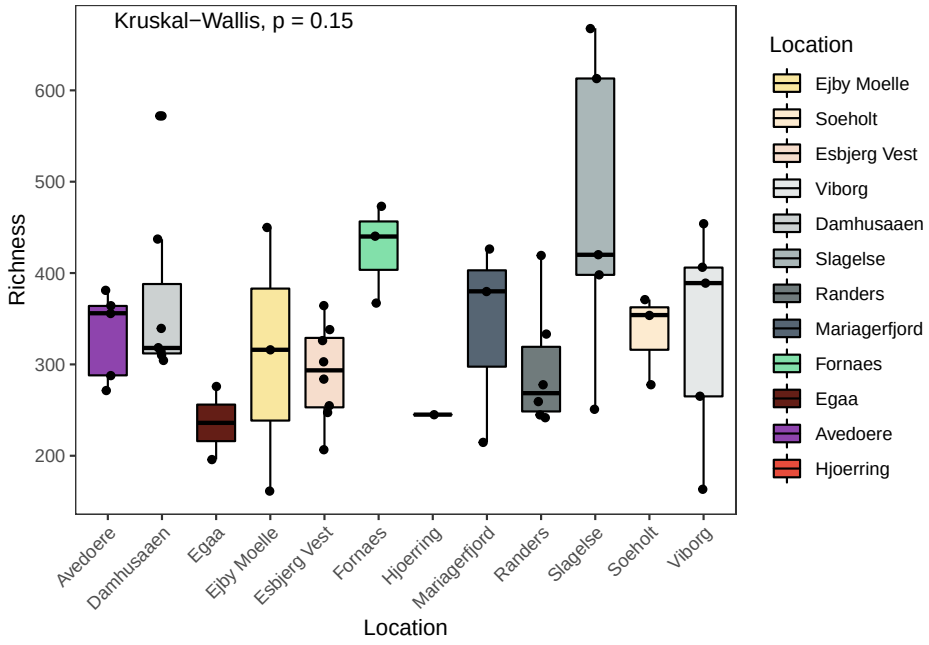
```
# You can check Hex codes with help of https://htmlcolorcodes.com/

cols <- c("#F9E79F", "#FDEBD0", "#F6DDCC",  "#E5E8E8", "#CCD1D1", "#AAB7B8",
          "#707B7C", "#566573","#82E0AA", "#641E16", "#8E44AD", "#E74C3C")

# you can also name each colour according to your treatment.
# This will help ggplot to associated the same colour for each treatment across
# different ggplots. For example, if we want to show different colour by location
# in this dataset:
names(cols)  <- unique((data.frame(sample_data(ps.flt)))$Location)
cols
```

```
##   Ejby Moelle       Soeholt  Esbjerg Vest        Viborg    Damhusaaen
##     "#F9E79F"     "#FDEBD0"     "#F6DDCC"     "#E5E8E8"     "#CCD1D1"
##      Slagelse       Randers Mariagerfjord       Fornaes          Egaa
##     "#AAB7B8"     "#707B7C"     "#566573"     "#82E0AA"     "#641E16"
##      Avedoere     Hjoerring
##     "#8E44AD"     "#E74C3C"
```

6.2.5 Richness

```
p1 <- data.frame(phyloseq::sample_data(ps.flt)) %>%
  ggpubr::ggboxplot(., x = "Location",
                    y = "Observed",
                    fill = "Location",
                    ylab = "Richness",
                    xlab = "Location",
                    palette = cols) +
    theme_bw() +
    theme( panel.grid.major = element_blank(), panel.grid.minor = element_blank() ) +
    geom_jitter(width = 0.1) +
   theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
```

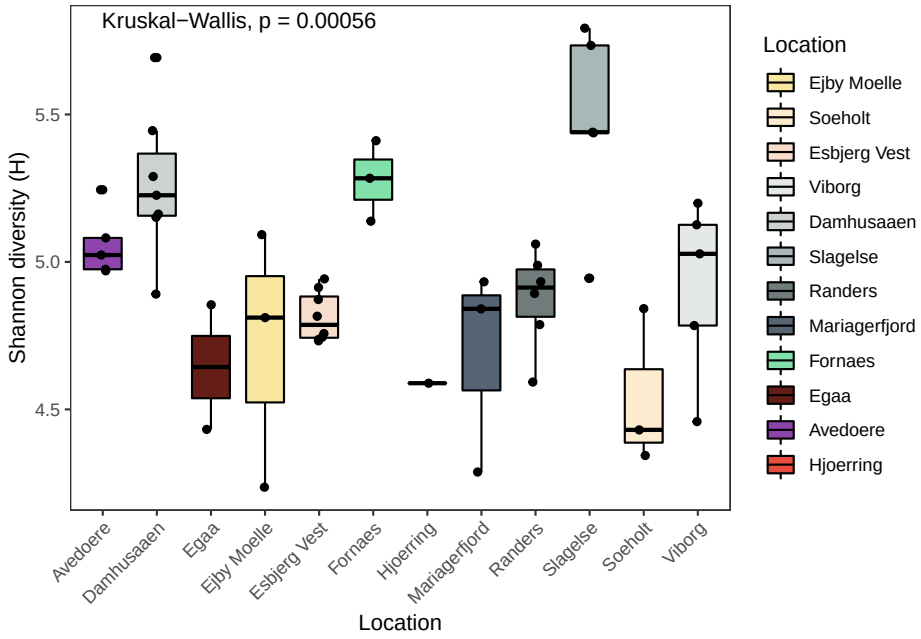
```
#    theme(axis.title.x=element_blank(),  # Options
#        axis.text.x=element_blank(),
#        axis.ticks.x=element_blank()) +
  ggpubr::stat_compare_means(method = "kruskal",label.x = 2)
p1
```



6.2.6 Shannon

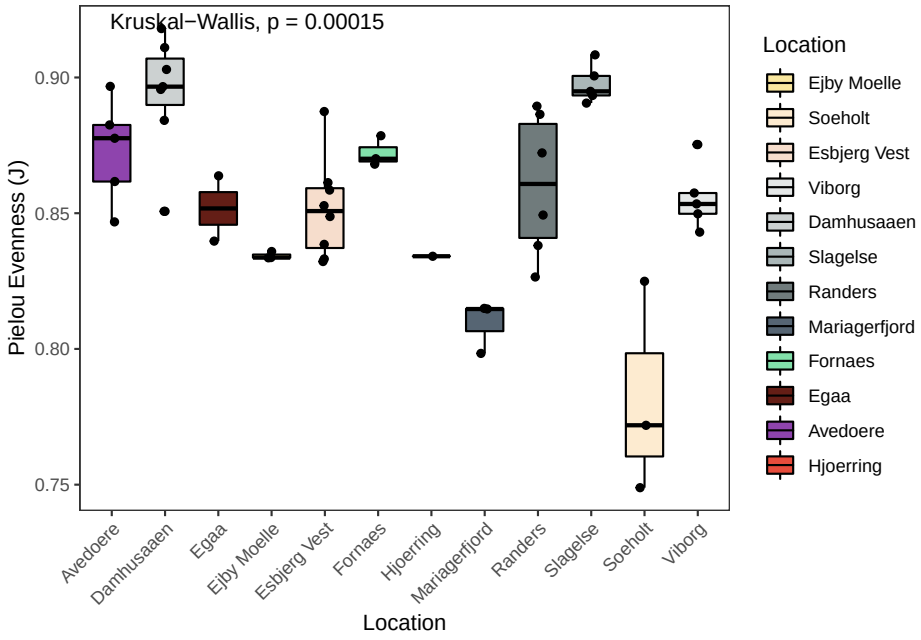
```
# Shannon
p2 <- data.frame(phyloseq::sample_data(ps.flt)) %>%
  ggpubr::ggboxplot(., x = "Location",
                    y = "Shannon",
                    fill = "Location",
                    ylab = "Shannon diversity (H)",
                    xlab = "Location",
                    palette = cols) +
    theme_bw() +
    theme( panel.grid.major = element_blank(), panel.grid.minor = element_blank() ) +
    geom_jitter(width = 0.1) +
   theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
#    theme(axis.title.x=element_blank(),  # Options
#        axis.text.x=element_blank(),
```

```
#        axis.ticks.x=element_blank()) +
  ggpubr::stat_compare_means(method = "kruskal",label.x = 2)
p2
```



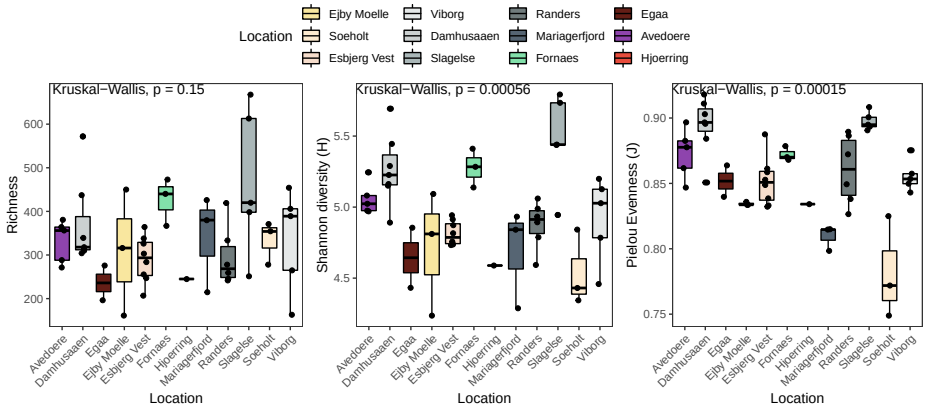
6.2.7 Pielou's evenness

```
# Pielou
p3 <- data.frame(phyloseq::sample_data(ps.flt)) %>%
  ggpubr::ggboxplot(., x = "Location",
                    y = "pielou_ev",
                    fill = "Location",
                    ylab = "Pielou Evenness (J)",
                    xlab = "Location",
                    palette = cols) +
    theme_bw() +
    theme( panel.grid.major = element_blank(), panel.grid.minor = element_blank() ) +
    geom_jitter(width = 0.1) +
   theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
#    theme(axis.title.x=element_blank(),  # Options
#        axis.text.x=element_blank(),
#        axis.ticks.x=element_blank()) +
  ggpubr::stat_compare_means(method = "kruskal",label.x = 2)
p3
```



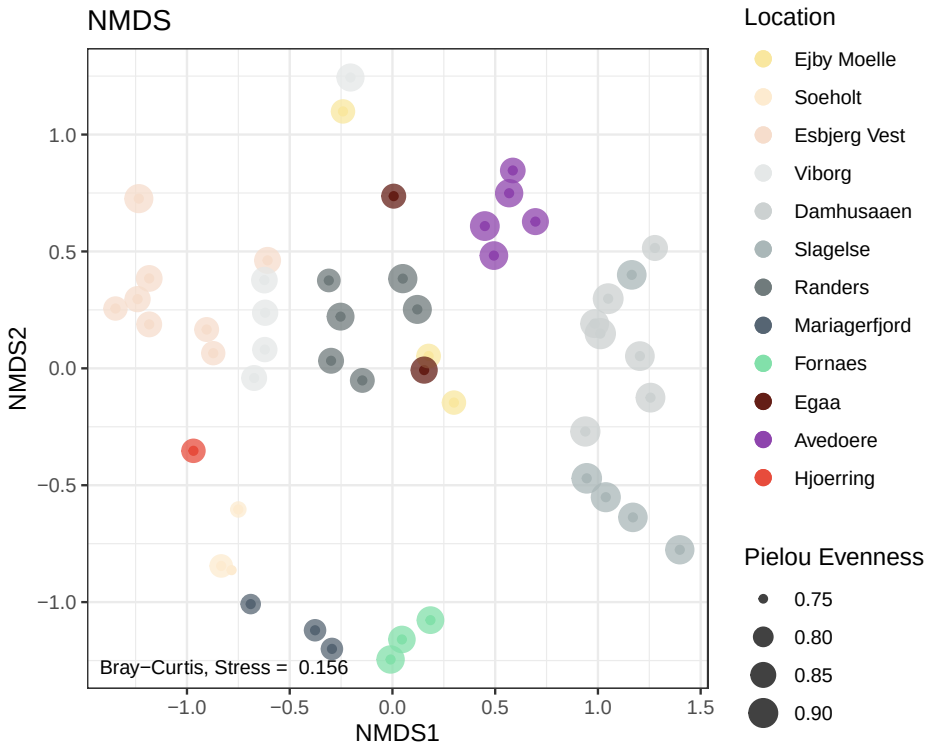
6.2.8 Combined plotting

```
#pdf(NULL)
g1 <- ggpubr::ggarrange(p1, p2, p3, common.legend = TRUE, nrow = 1, legend = 'top')
#x = dev.off()
g1
```



6.2.9 NMDS

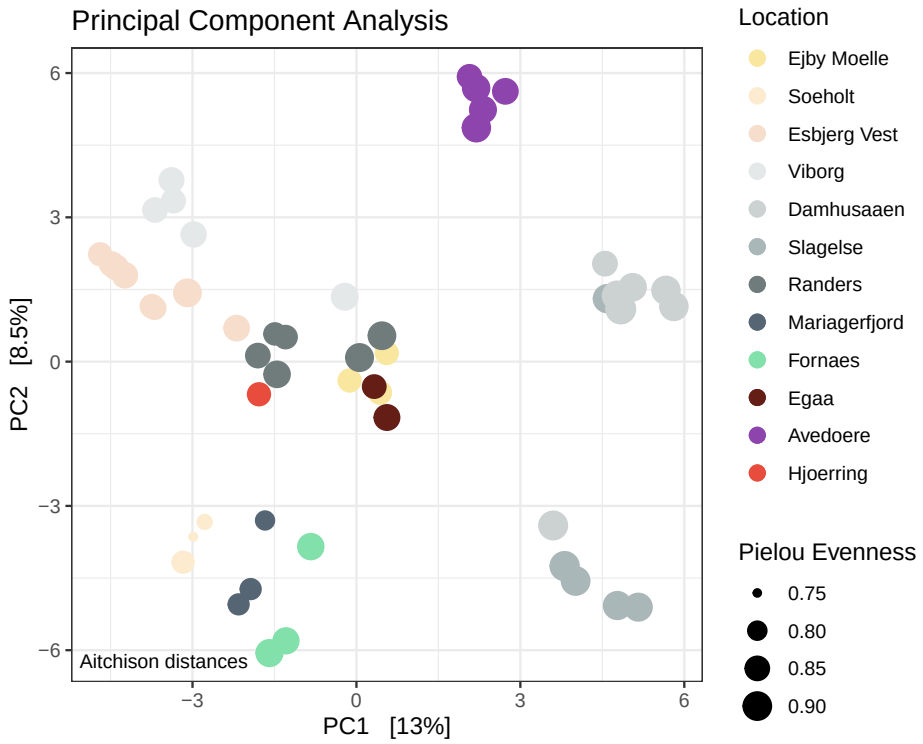
```
physeqNMDS <-  ps_rare  # We are using the rarefied abundances as explained in the intro

#ordination
ordination <- phyloseq::ordinate(physeqNMDS  , "NMDS", distance = 'bray')
# using Bray-Curtis dissimilarities here.

label <- round(ordination$stress,3) # stress label
#plotting ordination
NMDSplot <- phyloseq::plot_ordination(physeqNMDS,
                  ordination ,
                  color = "Location",
                  #shape = "Reactor",
                  # option to add shapes as data symbols
                  title = "NMDS") +
     geom_point(aes(size = pielou_ev), alpha = 0.75) +
# shape size in proportion to Pielou Evenness, also make transparent
     annotate(geom = 'text', label = paste("Bray-Curtis, Stress = ", label),
              x = -Inf, y = -Inf, hjust = -0.05, vjust = -1, size = 3)  +
     scale_color_manual(values = cols) +
# increase symbol size in legend and set order of legend items
     guides(color = guide_legend(override.aes = list(size = 3), order = 1),
            size = guide_legend(title ="Pielou Evenness") ) # change the legend title for `size`.

NMDSplot
```



6.2.10 PCA

```
physeqPCA <-  ps.flt  # just to keep it separate.

# normalisation and transform using the microbiome package
# I chose a centred log-ratio transform and a principal component analysis in a euclid
physeqPCA <- microbiome::transform(physeqPCA, "clr")

#ordination
ordination <- phyloseq::ordinate(physeqPCA , "RDA") # useing the centred-log transform

#plotting ordination
PCAplot <- phyloseq::plot_ordination(physeqPCA,
                                     ordination ,
                                     color = "Location",
                                     #shape = "Reactor",
                                    # option to add shapes as data symbols
                                     title = "Principal Component Analysis") +
     geom_point(aes(size = pielou_ev)) +    # # shape size in proportion to Pielou Eve
```

```
    annotate(geom = 'text', label = paste("Aitchison distances"),
             x = -Inf, y = -Inf, hjust = -0.05, vjust = -1, size = 3)  +
    scale_color_manual(values = cols) +
# increase symbol size in legend and set order of legend items
# change the legend title for `size`.
     guides(color = guide_legend(override.aes = list(size = 3), order = 1),
            size = guide_legend(title ="Pielou Evenness") )

PCAplot
```

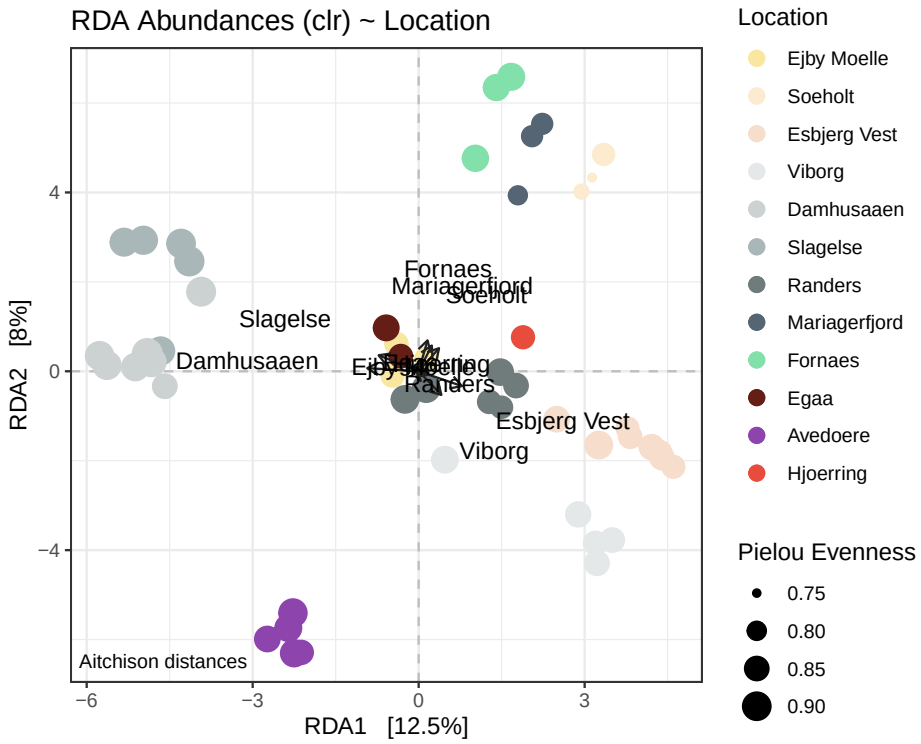


6.2.11 RDA

```
# PERCENT FILTERVALUE FOR ALL RDAs
pcnt <- 0.25
# FORMULA FOR ALL RDAs
form <- formula(~ Location)
```

```
# labels for arrows
# pos for arrows
xarw <- 5
yarw <- 5
# arrow length map
larw <- 1.5
# arrow length map for chromosol rda
larwch <- 1
# fixed width of x axis
xaxis = c(-6.75, 6.5)
# position of annotation
textposx = c(-4.4)
# arrow color
arrowcolor <- "grey13"


physeqPCA <- ps.flt # excluding Heat, H2O2, reference and blank

# normalisation and transform
physeqPCA <- microbiome::transform(physeqPCA, "clr")

#ordination
ordination <- phyloseq::ordinate(physeqPCA, "RDA", "bray", formula = form )

RDAplot <- phyloseq::plot_ordination(physeqPCA, ordination ,
                                color = "Location")  +
  geom_hline(yintercept=0, linetype="dashed", color = "gray") +
  geom_vline(xintercept=0, linetype="dashed", color = "gray") +    # shape size reflec
     annotate(geom = 'text', label = paste("Aitchison distances"),
              x = -Inf, y = -Inf, hjust = -0.05, vjust = -1, size = 3) +
  geom_point(aes(size = pielou_ev)) +        # shape size in proportion to Pielou Even
  #scale_colour_gradient(low = colourlow, high = colourhigh) +
  ggtitle("RDA Abundances (clr) ~ Location") +
  scale_color_manual(values = cols) +
# increase symbol size in legend and set order of legend items
# change the legend title for `size`.
     guides(color = guide_legend(override.aes = list(size = 3), order = 1),
            size = guide_legend(title ="Pielou Evenness") )



# Now add the environmental variables as arrows to either of these p1 or p2
arrowmat <- vegan::scores(ordination, display = "bp")
arrowmat <- data.frame(arrowmat)
#rownames(arrowmat) <- arrowlabel
```

```
# Add labels, make a data.frame
arrowdf <- data.frame(labels = rownames(arrowmat), arrowmat)
arrowdf$labels <- arrowdf$labels %>% gsub("Location", "", .) # remove the string "Location" from
# Define the arrow aesthetic mapping
arrow_map <- aes(xend = larw * RDA1,
                 yend = larw * RDA2,
                 x = 0,
                 y = 0,
                 shape = NULL,
                 color = NULL,
                 label = labels)

label_map <- aes(x = xarw * RDA1,
                 y = yarw * RDA2,
                 shape = NULL,
                 color = NULL,
                 label = labels)

arrowhead = arrow(length = unit(0.02, "npc"))

RDAplot <-RDAplot +
  geom_segment(
    mapping = arrow_map,
    size = .5,
    data = arrowdf,
    color = arrowcolor,
    arrow = arrowhead)  +
  geom_text(
   mapping = label_map,
  size = 4,
  data = arrowdf,
  show.legend = FALSE)


RDAplot
```



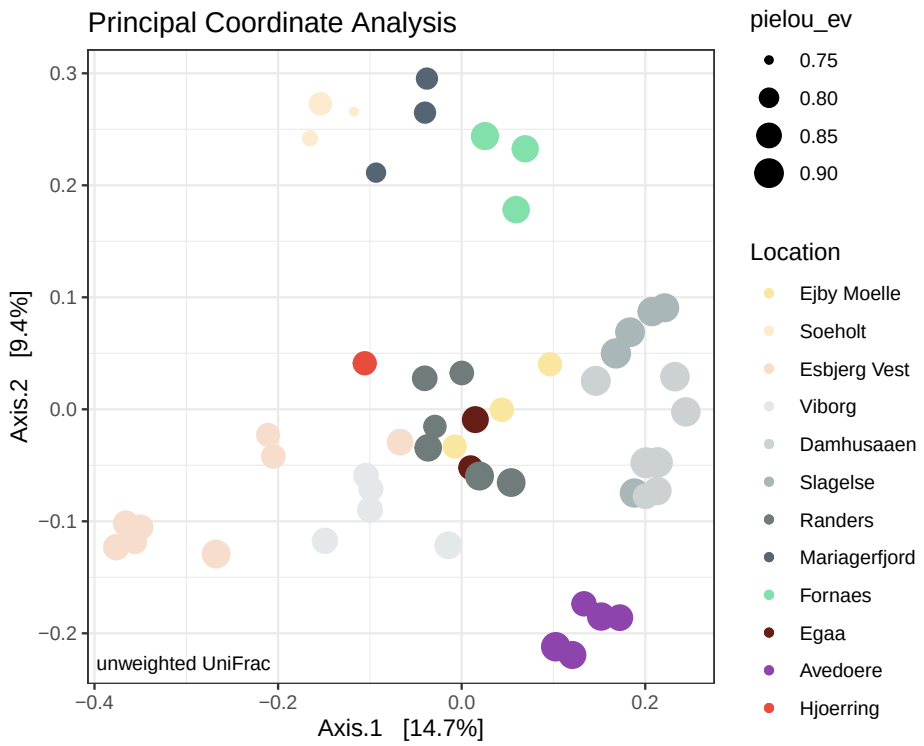

```
# save figure for publication
# ggsave("RDA.png", height=6, width=7.5, units='in', dpi=600)
#knitr::include_graphics("./RDA.png")
```

6.2.12 Unweighted uniFrac

UniFrac distances are derived from phylogenetic distances between ASVs, which means that a phylogenetic tree needs to be included into the phyloseq object.

```
physeqPcOA <-  ps.flt  # just to keep it separate.

#ordination
ordination <- phyloseq::ordinate(physeqPcOA  ,
                                 "PCoA",
                                 distance = 'unifrac') # using just presence and absen

#plotting ordination
```

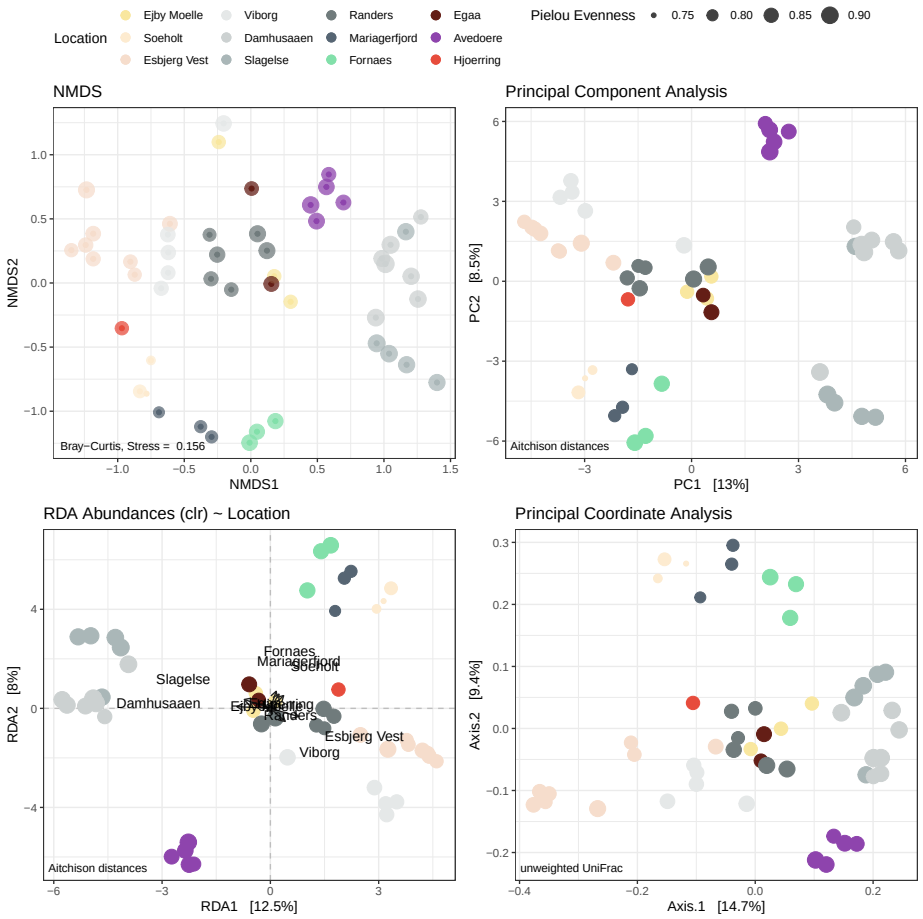
```
PCOAplot <- phyloseq::plot_ordination(physeqPcOA,
                                      ordination ,
                                      color = "Location",
                                      #shape = "Reactor",
                                      # option to add shapes to data symbols
                                      title = "Principal Coordinate Analysis") +
     geom_point(aes(size = pielou_ev)) +    # shape size reflective of Pielou
     annotate(geom = 'text', label = paste("unweighted UniFrac"),
              x = -Inf, y = -Inf, hjust = -0.05, vjust = -1, size = 3)  +
     scale_color_manual(values = cols)

PCOAplot
```



6.2.13 Combined plotting

```
#pdf(NULL)
g2 <- ggpubr::ggarrange(NMDSplot, PCAplot, RDAplot, PCOAplot,
                        common.legend = TRUE, nrow = 2, ncol = 2, legend = 'top')
```

```
#x = dev.off()
g2
```



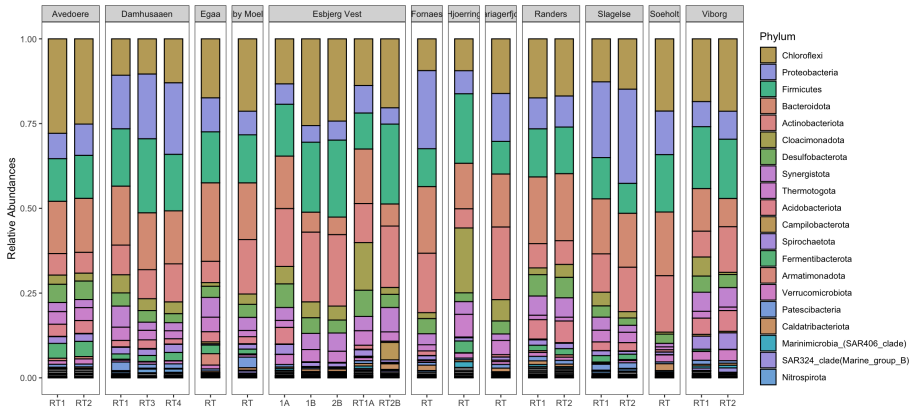
6.2.14 Save plots to png and pdf

```
# either as png or pdf
ggsave("alphadiversity.png", plot = (g1), height=4, width=10, units='in', dpi=600)
ggsave("alphadiversity.pdf", plot = (g1), height=4, width=10, units='in', dpi=600)

ggsave("betadiversity.png", plot = (g2), height=10, width=10, units='in', dpi=600)
ggsave("betadiversity.pdf", plot = (g2), height=10, width=10, units='in', dpi=600)
```

Note: This code is an amalgamation from various sources. Apart from putting it together into a pipeline I do not take credit for it.

Chapter 7

Plotting taxonomy



7.1 Introduction

In this chapter you will learn how to plot a custom taxa barplot to compare relative abundances between different sets of samples or treatments.

7.1.1 Required files or objects

- A `phyloseq` object; either read in from a pre-saved `.rds` file or created as described in chapter 5 under "Import qiime-files and create a phyloseq object".

- If you dont have your own data you can download the pre-saved `.rds` object from Chapter 5 and follow the below steps: ps_ProjectX_2022July.

This object is based on publicly available data from anaerobic sludge of Danish wastewater treatment plants.

7.2 Workflow

7.2.1 Packages

```
library(phyloseq)
library(ggpubr)       # a handy helper package for ggplots
library(tidyverse)
theme_set(theme_bw())  # setting the theme for ggplots
library(microbiome)
```

7.2.2 Load phyloseq object

Do this if you saved a phyloseq object in .rds format before. Otherwise create a phyloseq object as described in chapter 5 under "Import qiime-files and create a phyloseq object".

```
# reading in a previously saved phyloseq object
ps <- readRDS('ps_ProjectX_2022July')

#ps (not run) to get an overview of number of taxa and samples contained in the phylos

# output
# phyloseq-class experiment-level object
# otu_table()   OTU Table:         [ 4218 taxa and 51 samples ]
# sample_data() Sample Data:       [ 51 samples by 15 sample variables ]
# tax_table()   Taxonomy Table:    [ 4218 taxa by 7 taxonomic ranks ]
# phy_tree()    Phylogenetic Tree: [ 4218 tips and 4217 internal nodes ]
```

7.2.3 Check metadata contained in the phyloseq object.

Now lets check which columns in the metadata contain factors. Factors enable you to determine the order of categorical data, such as treatments, in the ggplots. Learn more about factors here: https://swcarpentry.github.io/r-novice-inflammation/12-supp-factors/index.html.

In this case, the columns `Reactor` and `Location` contain factors.

```
# Use the 'structure' (str) command to get an overview of your data columns
# Learn what are characters, integers or factors,
str(data.frame(sample_data(ps)))
```

```
## 'data.frame':    51 obs. of  15 variables:
##  $ Assay.Type     : chr  "AMPLICON" "AMPLICON" "AMPLICON" "AMPLICON" ...
##  $ Bases          : int  13254234 24196788 15758554 20048406 37634632 43555904 35107436 138881
##  $ BioProject     : chr  "PRJNA645373" "PRJNA645373" "PRJNA645373" "PRJNA645373" ...
##  $ Bytes          : int  9235913 16404070 10758456 13009806 24207581 27963491 22585685 9388862
##  $ Organism       : chr  "anaerobic digester metagenome" "anaerobic digester metagenome" "anae
##  $ collection_date: chr  "30/11/16" "9/3/16" "17/3/16" "5/12/16" ...
##  $ loc            : chr  "Ejby Moelle" "Soeholt" "Esbjerg Vest" "Viborg" ...
##  $ Reactor        : Factor w/ 10 levels "1A","1B","2B",..: 4 4 6 7 7 5 5 5 4 4 ...
##  $ Type           : chr  NA NA NA "DS" ...
##  $ Purpose        : chr  "reactorfoampotentials" "reactorfoampotentials" "reactorfoampotential
##  $ Experiment     : chr  "SRX8715180" "SRX8715169" "SRX8715158" "SRX8715151" ...
##  $ lat_lon        : chr  "55.398077 N 10.415041 E" "56.175302 N 9.582588 E" "55.488235 N 8.430
##  $ ReleaseDate    : chr  "2021-01-01T00:00:00Z" "2021-01-01T00:00:00Z" "2021-01-01T00:00:00Z"
##  $ Sample.Name    : chr  "16SAMP-17236" "16SAMP-17231" "16SAMP-17233" "MQ170915-14" ...
##  $ Location       : Factor w/ 12 levels "Avedoere","Damhusaaen",..: 4 11 5 12 12 12 12 2 11 11
```

7.2.4 Colors

Before you create ggplots to compare taxonomic compositions between samples and treatments it is helpful to create a named color vector that will apply to all your plots and ensures that each taxonomic name is a specific color across all your plots.

There are different packages that help to create a names color vector. Here we use the package `colorspace`.

A handy color cheat sheet is available here: https://www.nceas.ucsb.edu/sites/default/files/2020-04/colorPaletteCheatsheet.pdf

First create a vector of all possible taxonomic names that could be included in your ggplot. For example, if you plan to plot on phylum-level then all phyla names are extracted from the phyloseq object and then those names are given a color code. The colors are defined as hex color codes.

```
# You can check Hex codes with help of https://htmlcolorcodes.com/
# https://www.nceas.ucsb.edu/sites/default/files/2020-04/colorPaletteCheatsheet.pdf

# For all possible phyla  names
phylum_data <-  microbiome::aggregate_taxa(ps, "Phylum") # aggregate data to phylum level
# the pacakage 'microbiome' has some handy helper functions such as `aggregate`.
```

```
names <- rownames(phylum_data@tax_table) # extract a vector of phylum names using `row

# Package colorspace
library(colorspace)
cols <- colorspace::rainbow_hcl(length(names)) # this creates x number of hex codes. x
#  in this case there are 48 phyla, hence we get 48 hex codes.
names(cols) <- names  # name each hex code with phyla names.

# Option to use the package  viridis
#library(viridis)
# cols <- viridis::turbo(length(names), direction = -1)
# names(cols) <- names

# Option to use the package RColorBrewer
# library(RColorBrewer)
# display.brewer.all(length(names))  # check colour themes available in the RColorBrew
# cols <- brewer.pal(length(names), 'RdYlBu')
# cols <- rainbow_hcl(length(names))
# names(cols) <- names

cols # check content of the color vector
```

```
##                Acetothermia              Acidobacteriota
##                   "#E495A5"                    "#E3979D"
##             Actinobacteriota               Armatimonadota
##                   "#E19995"                    "#DE9B8D"
##                Bacteroidota            Bdellovibrionota
##                   "#DB9D85"                    "#D79F7E"
##           Caldatribacteriota               Caldisericota
##                   "#D2A277"                    "#CDA471"
##             Campilobacterota                 Chloroflexi
##                   "#C7A76C"                    "#C1A968"
##                    CK-2C2-2               Cloacimonadota
##                   "#BAAC65"                    "#B3AE64"
##         Coprothermobacterota               Cyanobacteria
##                   "#ABB065"                    "#A2B367"
##                Deinococcota                Dependentiae
##                   "#99B56B"                    "#90B66F"
##             Desulfobacterota             Elusimicrobiota
##                   "#86B875"                    "#7CBA7C"
##                     FCPU426          Fermentibacterota
##                   "#72BB83"                    "#67BC8A"
##               Fibrobacterota                  Firmicutes
##                   "#5CBD92"                    "#51BE9A"
##               Fusobacteriota             Gemmatimonadota
```

```
##                      "#47BEA2"                      "#3FBEA9"
##                Hydrogenedentes                   Hydrothermae
##                      "#39BEB1"                      "#37BDB8"
##               Latescibacterota                Margulisbacteria
##                      "#3BBCBF"                      "#42BBC6"
## Marinimicrobia_(SAR406_clade)                         MBNT15
##                      "#4CB9CC"                      "#57B8D1"
##              Methylomirabilota                   Myxococcota
##                      "#64B5D6"                      "#70B3DA"
##                   Nitrospirota                 Patescibacteria
##                      "#7DB0DD"                      "#8AADE0"
##                Planctomycetota                 Proteobacteria
##                      "#96AAE1"                      "#A1A7E2"
##  SAR324_clade(Marine_group_B)                  Spirochaetota
##                      "#ACA4E2"                      "#B5A1E1"
##                     Sumerlaeota                        Sva0485
##                      "#BE9EDF"                      "#C69BDC"
##                   Synergistota                  Thermotogota
##                      "#CD99D8"                      "#D497D4"
##              Verrucomicrobiota                         WPS-2
##                      "#D995CF"                      "#DD94C9"
##                             WS1                            WS4
##                      "#E093C3"                      "#E293BC"
##                   Zixibacteria                        Unknown
##                      "#E493B4"                      "#E494AD"
```

7.2.5 Pre-filtering

Some arbitrary quality filtering of individual ASVs. For example, here we choose to remove ASVs that have very few reads (i.e. less than 5 counts). Use the `prune_taxa()` function from phyloseq to filter taxa or the `prune_samples()` function to filter selected samples.

```
ps_flt  <-  prune_taxa(taxa_sums(ps) >= 5, ps) #minimum reads per feature
#ps_flt
# phyloseq-class experiment-level object
# otu_table()   OTU Table:         [ 3911 taxa and 51 samples ]
# sample_data() Sample Data:       [ 51 samples by 15 sample variables ]
# tax_table()   Taxonomy Table:    [ 3911 taxa by 7 taxonomic ranks ]
# phy_tree()    Phylogenetic Tree: [ 3911 tips and 3910 internal nodes ]

# Use `prune_samples()` If you need to filter data to a subset of samples.
# E.g. choose samples of reactor 1A only
ps_flt_1A  <-  prune_samples(sample_data(ps_flt)$Reactor == "1A", ps_flt)
```

```
# Use `prune_taxa` to remove any ASVs/taxa that are now zero
# (i.e. are not present in reactor 1A).
# In other words we keep only those ASVs/Taxa that are present in the filtered samples

ps_flt_1A  <- prune_taxa(taxa_sums(ps_flt_1A) > 0, ps_flt_1A)
#phyloseq-class experiment-level object
#otu_table()   OTU Table:         [ 420 taxa and 2 samples ]
#sample_data() Sample Data:       [ 2 samples by 15 sample variables ]
#tax_table()   Taxonomy Table:    [ 420 taxa by 7 taxonomic ranks ]
#phy_tree()    Phylogenetic Tree: [ 420 tips and 419 internal nodes ]
```

7.2.6 Analysis and plotting

Create stacked barplot comparing relative abundances of the normalised phyloseq object.

First abundances are aggregated to phylum level and then transformed into relative abundances. This is followed by the creation of a long-form abundance table that includes metadata with help of the very handy `psmelt` function from the phyloseq package. The function `psmelt` takes a phyloseq object and extracts abundances and related metadata into a dataframe for plotting.

Finally, the abundances of this long abundance table are summarised into the factors that you want to compare. In this example, abundances of different Locations are compared, hence the column `Location` will be summarised and plotted.

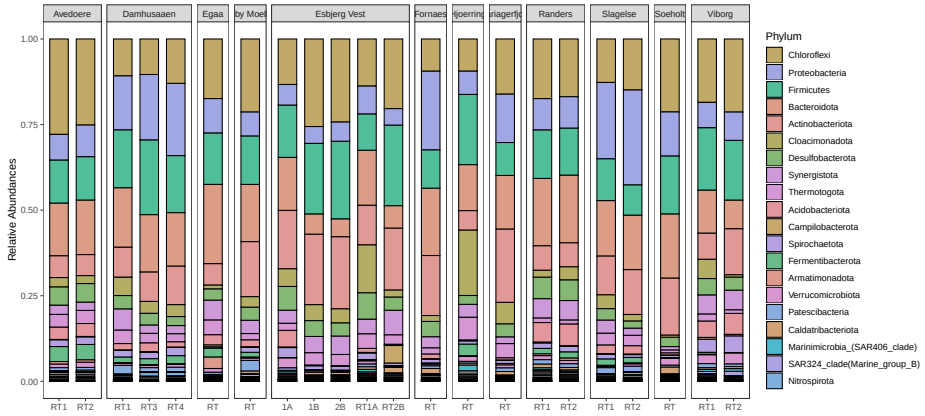
```
# Phylum plot
# First, create a long abundance data table
taxa_data <-  microbiome::aggregate_taxa(ps_flt, "Phylum") %>%
  # aggregate abundances to phylum-level
  microbiome::transform(., "compositional") %>%  # transform abundance to relative abu
  phyloseq::psmelt(.)  # extract long abundance table with metadata for plotting.

# Note: The `.` replaces the phyloseq object.
# This is because the functions are piped into a sequence by using the `%>%` operator.
# The `%>%` operator is a Tidyverse function and is very helpful in putting
# various different operations and changes to the data into one sequence.
# We could have put the below operations into the same above sequence too.
# There is no difference in the outcome.

# Summarise abundances by columns of your choice containing factors.
# E.g. compute the mean abundances based on the columns `Location`, `Reactor` and `Phy
taxa_data <- taxa_data %>%
    dplyr::group_by(Location, Reactor, Phylum) %>%  # group by Location and Phylum to
```

```
    dplyr::summarise(Meanabu = mean(Abundance)) %>%  # Create a column called `Meanabu`
    dplyr::filter(Meanabu > 0) %>%                   # filter out rows that have 0 abundance
    dplyr::arrange(desc(Meanabu)) # sort the abundances

# Now, create a vector with phylum names that will determine the order of
# phylum names in the stacked barcharts.
order <- taxa_data$Phylum %>% unique()

# Phylum names need to be factors for plotting.
# This is the final dataframe for plotting
taxa_data$Phylum <- factor(taxa_data$Phylum, levels = order) #
# Check out the first few rows
head(taxa_data)
```

```
## # A tibble: 6 x 4
## # Groups:   Location, Reactor [6]
##   Location     Reactor Phylum         Meanabu
##   <fct>        <fct>   <fct>            <dbl>
## 1 Avedoere     RT1     Chloroflexi      0.279
## 2 Slagelse     RT2     Proteobacteria   0.278
## 3 Esbjerg Vest 1B      Chloroflexi      0.256
## 4 Avedoere     RT2     Chloroflexi      0.251
## 5 Esbjerg Vest 2B      Chloroflexi      0.243
## 6 Esbjerg Vest RT2B    Firmicutes       0.236
```

```
# Finally its time to create a ggplot.
taxaplot <- taxa_data %>%
    ggbarplot( x= "Reactor",
               y = "Meanabu",
               fill = "Phylum",
               position = position_stack(),
               legend = "right",
               ylab = "Relative Abundances") +
            # Tell ggplot to use the `cols` vector created earlier
            # only the first 20 colors are shown in legend, hence the `order[1:20]`
    scale_fill_manual(values = cols,
                      limits = order[1:20]) +
    facet_grid(cols = vars(Location),     # create facets by factors in column `Location`.
               scales = "free_x",
               space = "free_x") +
    guides(fill = guide_legend(byrow = FALSE)) +
    rotate_x_text(angle = 45) +           # rotate the x-axis labels
    theme_bw() +                          # makes ggplots look better
    theme(axis.title.x=element_blank(),   # remove the background grid
          panel.grid.major = element_blank(),
```

```
          panel.grid.minor = element_blank())

taxaplot
```



7.2.7 Prevalance table

A prevalence table may assist in getting an overview of the abundances. The mean prevalence is the average number of times that the taxon appears across all samples.

```
prevalancedf = apply(X = phyloseq::otu_table(ps_flt),
                     MARGIN = 1,
                     FUN = function(x){sum(x > 0)})

prevalancedf = data.frame(Prevalence = prevalancedf,
                     TotalAbundance = phyloseq::taxa_sums(ps_flt),
                     phyloseq::tax_table(ps_flt))

prevalancedf <- plyr::ddply(prevalancedf, "Phylum", function(df1){
                  data.frame(mean_prevalence=mean(df1$Prevalence),
                  total_abundance=sum(df1$TotalAbundance, na.rm = T),
                  stringsAsFactors = F)
                    })

# print table from highest to lowest abundance
prevalancedf %>%
  arrange(desc(total_abundance))
```

```
##                          Phylum mean_prevalence total_abundance
## 1                    Chloroflexi        5.124069          228197
```

```
## 2                    Bacteroidota        4.688679          201921
## 3                      Firmicutes        4.453435          197683
## 4                  Actinobacteriota        5.231343          165740
## 5                    Proteobacteria        4.249147          161756
## 6                  Desulfobacterota        5.425743           57146
## 7                      Synergistota        6.595506           51225
## 8                    Cloacimonadota        6.500000           45294
## 9                   Acidobacteriota        3.953488           35304
## 10                     Thermotogota       10.190476           33489
## 11                   Spirochaetota        4.578431           25015
## 12               Verrucomicrobiota        3.382022           16598
## 13               Fermentibacterota       35.000000           14812
## 14                 Patescibacteria        3.669173           14399
## 15              Caldatribacteriota        5.500000            8932
## 16                 Planctomycetota        2.573171            5455
## 17                  Armatimonadota        2.933333            4443
## 18 Marinimicrobia_(SAR406_clade)       20.000000            3818
## 19                             <NA>        4.052632            3511
## 20                 Hydrogenedentes        3.950000            3411
## 21                    Nitrospirota        8.750000            2853
## 22                 Campilobacterota        4.187500            2710
## 23  SAR324_clade(Marine_group_B)        4.000000            2616
## 24                              WS1        3.625000            2559
## 25                  Fibrobacterota        2.826087            1938
## 26                     Myxococcota        3.176471            1353
## 27                Bdellovibrionota        3.600000            1111
## 28                   Caldisericota        3.818182            1067
## 29                           WPS-2        2.666667             897
## 30                    Sumerlaeota        3.888889             683
## 31                   Cyanobacteria        9.000000             647
## 32                  Fusobacteriota        2.888889             565
## 33           Coprothermobacterota        2.800000             511
## 34                Gemmatimonadota        2.875000             300
## 35                Elusimicrobiota        2.833333             225
## 36                        CK-2C2-2        4.000000             162
## 37              Methylomirabilota        4.000000             159
## 38                Latescibacterota        2.333333              86
## 39                    Dependentiae        2.500000              78
## 40                    Zixibacteria        5.000000              74
## 41                         Sva0485        2.333333              66
## 42                    Acetothermia        2.000000              59
## 43                             WS4        1.500000              39
## 44                    Hydrothermae        1.000000              26
## 45                    Deinococcota        1.000000              11
```

7.2.8 Krona charts

On macOS or Linux, a more interactive way to explore the various levels of the taxonomy in your data is to use Krona charts.

All you need is KronaTools and the R package `psadd`.

To install KronaTools simply download and unzip the repository of https://github.com/marbl/Krona (The green "Code" button) and execute the `install.pl` file as described in the Installation Wiki at https://github.com/marbl/Krona/wiki/Installing. Once installed you can run the following:

```
# install a package with custom function for the phyloseq package
# remotes::install_github("cpauvert/psadd")
library(psadd)
# Load ps files
# Use the data-set that we filtered earlier 'ps_flt'
# Separate the output by the metadata column "Location". We will get a krona plot for

# not run
# plot_krona(ps_flt,"krona-file", "Location",trim=T) # Function plot_krona from the ps
# This creates an 'html' file named 'krona-file, which you can open in any web browser
```

Explore the Krona plot in your web browser: krona-file

Chapter 8

Phylofactor analysis

8.1 Introduction

In this chapter you will learn how to use one of the main functions from the package phylofactor, as well as analyse and visualise the results.

Phylofactor is a package developed by Alex Washburne, a mathematical biologist. It is a compositional analysis tool that 'iteratively identifies the most important clades driving variation in the data through their associations with independent variables' (Washburne et al. (2017)). For me, it is a great way to assess clade-specific treatment effects or environmental associations. In combination with other differential abundance analysis methods it helped me to gain confidence in biological interpretations of microbial community associations to environmental metadata or treatments. I also use it to visualise selected identified clades on a phylogenetic tree.

The package enables the use of different statistical tests, such as twoSample comparisons (i.e. t-test, Wilcox text, Fisher exact test) or regression modelling of abundances, transformed into isometric log ratios (ILRs) - sometimes called balances - to study treatment effects or environmental associations. It does this for aggregated abundances on edges of a phylogenetic tree, hence includes associations of abundances of whole clades of ASVs.

Check out the papers Phylofactorization: a graph-partitioning algorithm to identify phylogenetic scales of ecological data (Washburne et al. (2019)) and Phylogenetic factorization of compositional data yields lineage-level associations in microbiome datasets (Washburne et al. (2017)) to learn more about the algorithms used.

You will need a phylogenic tree for this analysis. For example, in chapter 4 we have built a phylogenetic tree from ASVs using the insertion tree method with

qiime2. This tree was then imported into R and included into a phyloseq object as shown in chapter 5.

For this tutorial we will be using the same phyloseq object and apply the `PhyloFactor` function of the Phylofactor package. This function is used to assess associations of relative abundances with selected metadata, assuming a normal distribution (ILRs are normally distributed). It is basically finding variation in ILRs or balances (which are two groups of abundances delineated by an edge in the tree) that respond to treatments or environmental data.

8.1.1 Prerequisites and required files

- Phylofactor tutorial read
- Phylofactor package installed
- A `phyloseq` object that includes a phylogenetic tree; either read in from a pre-saved `.rds` file or created as described in chapter 5 under "Import qiime-files and create a phyloseq object".
- If you dont have your own data you can download the pre-saved `.rds` object from Chapter 5 and follow the below steps: ps_ProjectX_2022July. This object was created from publicly available data from anaerobic sludge of Danish wastewater treatment plants.

8.1.2 Custom functions

There are several functions available in phylofactor to review the model outputs and to review which taxa were associated with groups of ASVs/clades that are part of significant edges in each factor.

I also created some custom functions that make it a little easier (for me). They are available to download. I am no coder so the code in the functions may offend you but they worked :). If you want to use these extra functions you can download them into your working directory and load them into your R environment using the `dget` function (e.g. `phyfacsummary <- dget("./custom-functions/phyfacsummary.R")`). I usually keep custom functions in a separate folder, here named 'custom-functions'.

If you want to learn what the functions do, you can also copy the code out of the function and have a go at running the code yourself.

Custom functions. Download and add to your working directory:

- phyfacsummary

- addILRtophyloseq
- ILR_plotfunction
- mytreefunctionwithbarplots
- myprevalencetablefunction

8.2 Workflow

8.2.1 Load packages and custom functions

```
library(phyloseq)
library(ggpubr)       # a handy helper package for ggplots
library(tidyverse)
theme_set(theme_bw())  # setting the theme for ggplots
library(colorspace)
library(ggtreeExtra)
library(ggtree)

## phyfacsummary function
# Function to summarise all phylofactor factors in a meaningful way into a dataframe
# p adjusted method = Bonferroni
# Enter the phylofactor file and the phyloseq object to get a dataframe with model results,
# abundances and taxa info
# the third function input is '4' or '5',  4 is used when the phylofactor
# model setting was set to 'F', if 'var' use 5
phyfacsummary <- dget("./custom-functions/phyfacsummary.R")

## addILRtophyloseq function
# Function to take a phylofactor file and extract the mean sample ILRs and
# add those to the phyloseq sample data those can then be used for plotting graphs
# together with treatment data.
addILRtophyloseq <- dget("./custom-functions/addILRtophyloseq.R")

## ILR_plotfunction
ILR_plotfunction <- dget("./custom-functions/ILR_plotfunction.R")

## mytreefunctionwithbarplots
# Tree functions combining phylum colors with grey
# phylofactor highlights.
# Requires a colour vector 'treecols'
# Requires an order vector 'ordervec' that includes the names of
```

```
# all phyla to be included into the legend in that order (example shown below).
mytreefunctionwithbarplots <- dget("./custom-functions/mytreefunctionwithbarplots.R")

## Prevalence table function
# get overview of abundances, mean prevalence is the mean 'appearance' of
# ASVs of the taxon across all samples
prevalencedf <- dget("./custom-functions/myprevalencetablefunction.R")
```

8.2.2 Load phyloseq object

Do this if you saved a phyloseq object in .rds format before. Otherwise create a phyloseq object as described in chapter 5 under "Import qiime-files and create a phyloseq object".

```
# reading in a previously saved phyloseq object
ps <- readRDS('ps_ProjectX_2022July')

#ps (not run) to get an overview of number of taxa and samples contained in the phylos

# output
# phyloseq-class experiment-level object
# otu_table()   OTU Table:         [ 4218 taxa and 51 samples ]
# sample_data() Sample Data:       [ 51 samples by 15 sample variables ]
# tax_table()   Taxonomy Table:    [ 4218 taxa by 7 taxonomic ranks ]
# phy_tree()    Phylogenetic Tree: [ 4218 tips and 4217 internal nodes ]
```

8.2.3 Pre-filtering

Some quality filtering of individual ASVs. This is up to the analyst and aims of the project.

Here it was choosen to remove ASVs with few reads and those ASVs that could not be assigned a Phylum. Use the `prune_taxa()` function from phyloseq to filter taxa or the `prune_samples()` function to filter selected samples.

The ASV abundances are then agglomerated to Genus level. This makes things a little easier to handle computationally and interpret visually at the end.

```
# Filter any phyla that have not been classified, create a new phyloseq object called p
ps.phylo = subset_taxa(ps, !is.na(Phylum) & !Phylum %in% c("", "uncharacterized"))
ps.phylo <- prune_taxa(taxa_sums(ps.phylo)  > 10, ps.phylo)

# ps.phylo
# phyloseq-class experiment-level object
```

```
# otu_table()   OTU Table:         [ 3325 taxa and 51 samples ]
# sample_data() Sample Data:       [ 51 samples by 15 sample variables ]
# tax_table()   Taxonomy Table:    [ 3325 taxa by 7 taxonomic ranks ]
# phy_tree()    Phylogenetic Tree: [ 3325 tips and 3324 internal nodes ]

# Option to agglomerate abundances to a certain taxa level
# (not necessary; increases speed of running PhyloFactor for this tutorial)
ps.phylo <- phyloseq::tax_glom(ps.phylo, taxrank = "Genus")
# ps.phylo
# phyloseq-class experiment-level object
# otu_table()   OTU Table:         [ 525 taxa and 51 samples ]
# sample_data() Sample Data:       [ 51 samples by 15 sample variables ]
# tax_table()   Taxonomy Table:    [ 525 taxa by 7 taxonomic ranks ]
# phy_tree()    Phylogenetic Tree: [ 525 tips and 524 internal nodes ]
```

8.2.4 Running PhyloFactor

For this example, the explanatory variable `Location` is used to regress the response (abundances). This has to be a factor or numeric so check your explanatory beforehand.

```
# create/extract metadata object from phyloseq object
meta.df <- data.frame(sample_data(ps.phylo))
# Extract rownames (Sample IDs)
rnames <- rownames(meta.df)
# Explanatory variable for model
X <- meta.df$Location
# check explanatory
summary(X)
```

```
##      Avedoere    Damhusaaen          Egaa   Ejby Moelle  Esbjerg Vest
##             5             7             2             3             8
##       Fornaes     Hjoerring Mariagerfjord       Randers      Slagelse
##             3             1             3             6             5
##       Soeholt        Viborg
##             3             5
```

```
# name the vector data with sample IDs
names(X) <- rnames
# create an ASV table for phylofactor
Data = as(otu_table(ps.phylo), "matrix")
Data = as.data.frame(t(Data))
Data <- Data[match(rnames,row.names(Data)),]
```

```
Data <- as.data.frame(t(Data))
# create tree object from phyloseq object for phylofactor
tree <- phy_tree(ps.phylo)
# unroot the tree, this is critical for phylofactor to work
tree_ur <- ape::unroot(tree)
# check if unrooted
ape::is.rooted(tree_ur) # FALSE IS GOOD
```

```
## [1] FALSE
```

```
# check that all in order
all(rownames(Data) == row.names(tax_table(ps.phylo)))
```

```
## [1] TRUE
```

```
all(tree_ur$tip.label == row.names(tax_table(ps.phylo)))
```

```
## [1] TRUE
```

```
# Run phylofactor
PF <-  phylofactor::PhyloFactor(Data,tree_ur,X,
        frmla=Data~X,     # Data (ASV abundances) as a response to X (Location)
        ncores = 2,       # number of cores
        stop.early = T,
        transform.fcn=log,     # log-ratio transform
        KS.Pthreshold = 0.01, # Kolmogorov Smirnov significance level
        nfactors = 5, # the first 5 phylofactors only - in interest of time
        choice = "var")
```

```
##  1 factor completed in 0.0257 minutes.    Estimated time of completion: 2022-09-27 1
```

```
# saveRDS(PF,'pf-example.rds')  # save Phylofactor object to .rds file
rm(tree, tree_ur, Data, X, meta.df) #remove unnecessary objects from R environment

# The KS threshold determines the significance at which the variation
# (in response to the explanatory) of one edge balance is different enough to the rema
# to become a factor. If you have a better definition please let me know
# (on the github discussion page ideally (see Chapter 1)).
# If no value is provided to nfactors - the algorithm will continue until
# no edge are found anymore.
```

8.2.5 Reviewing output

Simply run the phylofactor output object (here `PF`) to get an overview of the results. You can also access individual data outputs using the `$` operator, e.g. by running `PF$factors` you get a full list of all factors.

```
# Overview
PF # showing the output of the first 10 factors
```

```
##        phylofactor object from function PhyloFactor
##        --------------------------------------------
## Method                    : glm
## Choice                    : var
## Formula                   : Data ~ X
## Number of species         : 525
## Number of factors         : 5
## Frac Explained Variance   : 0.0458
## Largest non-remainder bin : 393
## Number of singletons      : 4
## Paraphyletic Remainder    : 130 species
##
## ------------------------------------------------------------
## Factor Table:
##                                  Group1                          Group2    ExpVar
## Factor 1                            tip 524 member Monophyletic clade 0.0103300
## Factor 2                            tip 523 member Paraphyletic clade 0.0099537
## Factor 3 393 member Paraphyletic clade 130 member Monophyletic clade 0.0086014
## Factor 4                            tip 392 member Paraphyletic clade 0.0085960
## Factor 5                            tip 391 member Paraphyletic clade 0.0082704
##                F     Pr(>F)
## Factor 1 66.413 0.0000e+00
## Factor 2 18.379 3.8480e-12
## Factor 3 13.627 3.5753e-10
## Factor 4 16.545 1.9611e-11
## Factor 5 24.943 2.7978e-14
```

```
# Factors only
# PF$factors
#                                  Group1                          Group2      ExpVar        F
# Factor 1                            tip 524 member Monophyletic clade 0.010329782 66.41304 0.000
# Factor 2                            tip 523 member Paraphyletic clade 0.009953684 18.37912 3.848
# Factor 3 393 member Paraphyletic clade 130 member Monophyletic clade 0.008601447 13.62745 3.575
# Factor 4                            tip 392 member Paraphyletic clade 0.008595997 16.54491 1.961
# Factor 5                            tip 391 member Paraphyletic clade 0.008270358 24.94341 2.797
```

In this case there are five factors. Factor 1,2, 4 and 5 are only one genus each. Factor 3 contains 393 genera in group 1.

Group 1 contains the ASVs/taxa with abundances that are different to the remainder (Group 2).

This can be interpreted as following: For example, the abundances of the genus represented by factor 1 are significantly different among the locations of wastewater treatment plants. Factor 1 also explains most of the variation and has the highest F value, indicating its significance.

The next factors, factor 2, factor 3 etc, were selected by the algorithm from the remainder edges (excluding factor 1, and then excluding factor 2 etc..).

From this output we cannot tell which taxa are involved and in which locations the taxa have low or higher abundances, i.e. how different they are (+ or -) among the locations.

To look at the ASVs/taxa we require other functions and also create a taxonomy object:

```
## Taxa summaries
# To create a taxa summary a taxonomy dataframe is needed first,
# whose first column contains the species in the phylogeny and whose
# second column contains a semicolon-delimited taxonomy.
# Here, we extract that out of the phyloseq object as following.
taxonomy <- data.frame(tax_table(ps.phylo)) %>%
  unite(Taxon, sep=";", c("Kingdom","Phylum","Class","Order","Family","Genus","Species"
  rownames_to_column("Feature.ID")

# running pf.taxa function from phylofactor
phylofactor::pf.taxa(PF, taxonomy, factor = 1)
```

```
## $group1
## [1] "Bacteria;Acidobacteriota;Thermoanaerobaculia;Thermoanaerobaculales;Thermoanaero
##
## $group2
##  [1] "Bacteria;Proteobacteria"
##  [2] "Bacteria;Cyanobacteria"
##  [3] "Bacteria;Campilobacterota"
##  [4] "Bacteria;WPS-2"
##  [5] "Bacteria;Gemmatimonadota"
##  [6] "Bacteria;Nitrospirota"
##  [7] "Bacteria;Actinobacteriota"
##  [8] "Bacteria;Desulfobacterota"
##  [9] "Bacteria;Sva0485"
## [10] "Bacteria;Myxococcota"
## [11] "Bacteria;Bdellovibrionota"
```

```
## [12] "Bacteria;SAR324_clade(Marine_group_B)"
## [13] "Bacteria;Dependentiae"
## [14] "Bacteria;Chloroflexi"
## [15] "Bacteria;Fusobacteriota"
## [16] "Bacteria;Firmicutes"
## [17] "Bacteria;Bacteroidota"
## [18] "Bacteria;Synergistota"
## [19] "Bacteria;Verrucomicrobiota"
## [20] "Bacteria;Spirochaetota"
## [21] "Bacteria;Cloacimonadota"
## [22] "Bacteria;Patescibacteria"
## [23] "Bacteria;Elusimicrobiota"
## [24] "Bacteria;Planctomycetota"
## [25] "Bacteria;Fibrobacterota"
## [26] "Bacteria;WS4"
## [27] "Bacteria;WS1"
## [28] "Bacteria;Armatimonadota"
## [29] "Bacteria;Methylomirabilota"
## [30] "Bacteria;Acetothermia"
## [31] "Bacteria;Coprothermobacterota"
## [32] "Bacteria;Thermotogota"
## [33] "Bacteria;Deinococcota"
## [34] "Bacteria;Hydrothermae"
## [35] "Bacteria;Acidobacteriota;Aminicenantia"
## [36] "Bacteria;Caldisericota"
## [37] "Bacteria;Caldatribacteriota"
## [38] "Bacteria;Sumerlaeota"
## [39] "Bacteria;Fermentibacterota"
## [40] "Bacteria;Hydrogenedentes"
## [41] "Bacteria;Latescibacterota"
## [42] "Bacteria;CK-2C2-2"
## [43] "Bacteria;Zixibacteria"
## [44] "Bacteria;Acidobacteriota;c5LKS83"
## [45] "Bacteria;Acidobacteriota;Thermoanaerobaculia;Thermoanaerobaculales;Thermoanaerobaculacea
## [46] "Bacteria;Acidobacteriota;Holophagae"
## [47] "Bacteria;Acidobacteriota;Blastocatellia"
## [48] "Bacteria;Acidobacteriota;Acidobacteriae"
## [49] "Bacteria;Acidobacteriota;Vicinamibacteria"
## [50] "Bacteria;Acidobacteriota;Subgroup_21"
## [51] "Bacteria;Acidobacteriota;Subgroup_18"
## [52] "Bacteria;Marinimicrobia_(SAR406_clade)"
```

This output shows that the genus *Thermoanaerobaculum* (Phylum Acidobacteriota) was represented by factor 1 (Group 1) and was significantly different to the remainder (Group 2) between Locations.

```
# create a summary for a factor
s <- summary(PF,taxonomy,factor=3)
s
```

```
##        phylofactor object from function PhyloFactor
##        --------------------------------------------
## Method                    : glm
## Algorithm                 : contrast_basis
## Choice                    : var
## Formula                   : Data ~ X
## Factor                    : 3
## Edges Considered          : 1043
##
##                                  Group1                          Group2
## Factor 3 393 member Paraphyletic clade 130 member Monophyletic clade
##               ExpVar        F       Pr(>F)
## Factor 3 0.008601447 13.62745 3.575326e-10
## ========================================================================
## Taxon Tables:
## Group1:
##                   Taxon nSpecies signal
## 1 Bacteria;Bacteroidota       57     NA
## 2   Bacteria;Firmicutes       89     NA
## 3  Bacteria;Chloroflexi       34     NA
##                      .................  37 rows omitted .................
## ------------------------------------------------------------------------
## Group2:
##                        Taxon nSpecies signal
## 1   Bacteria;Proteobacteria      123     NA
## 2 Bacteria;Campilobacterota        4     NA
## 3            Bacteria;WPS-2        1     NA
##                      .................  1 row omitted .................
```

```
# List all taxa for each group
s$taxa.split
```

```
## $group1
##  [1] "Bacteria;Gemmatimonadota"
##  [2] "Bacteria;Nitrospirota"
##  [3] "Bacteria;Actinobacteriota"
##  [4] "Bacteria;Desulfobacterota"
##  [5] "Bacteria;Sva0485"
##  [6] "Bacteria;Myxococcota"
##  [7] "Bacteria;Bdellovibrionota"
```

```
##  [8] "Bacteria;SAR324_clade~Marine_group_B~"
##  [9] "Bacteria;Dependentiae"
## [10] "Bacteria;Chloroflexi"
## [11] "Bacteria;Fusobacteriota"
## [12] "Bacteria;Firmicutes"
## [13] "Bacteria;Bacteroidota"
## [14] "Bacteria;Synergistota"
## [15] "Bacteria;Verrucomicrobiota"
## [16] "Bacteria;Spirochaetota"
## [17] "Bacteria;Cloacimonadota"
## [18] "Bacteria;Patescibacteria"
## [19] "Bacteria;Elusimicrobiota"
## [20] "Bacteria;Planctomycetota"
## [21] "Bacteria;Fibrobacterota"
## [22] "Bacteria;WS4"
## [23] "Bacteria;WS1"
## [24] "Bacteria;Armatimonadota"
## [25] "Bacteria;Methylomirabilota"
## [26] "Bacteria;Acetothermia"
## [27] "Bacteria;Coprothermobacterota"
## [28] "Bacteria;Thermotogota"
## [29] "Bacteria;Deinococcota"
## [30] "Bacteria;Hydrothermae"
## [31] "Bacteria;Acidobacteriota"
## [32] "Bacteria;Caldisericota"
## [33] "Bacteria;Caldatribacteriota"
## [34] "Bacteria;Sumerlaeota"
## [35] "Bacteria;Fermentibacterota"
## [36] "Bacteria;Hydrogenedentes"
## [37] "Bacteria;Latescibacterota"
## [38] "Bacteria;CK-2C2-2"
## [39] "Bacteria;Zixibacteria"
## [40] "Bacteria;Marinimicrobia_~SAR406_clade~"
##
## $group2
## [1] "Bacteria;Proteobacteria"   "Bacteria;Cyanobacteria"
## [3] "Bacteria;Campilobacterota" "Bacteria;WPS-2"
```

The default for `summary`, provided a taxonomy object, is to find the shortest-unique-prefix labels for each species in each group. For example, there are no Proteobacteria or Firmicutes in Group 1.

The summary objects contains other useful data, including all ASVs IDs or the ILRs for each group, which may become handy if you wanted to do some custom filtering and figures of your taxonomy.

There are other summary tools available in the phylofactor package. Check

them out in the phylofactor tutorial.

I created a custom function `phyfacsummary` that helped me to create a dataframe containing all ASVs, including the factors and model results. It includes the fitted values for each explanatory category (in this case the different digester locations), showing how abundances differ between different locations.

```
## Summary tables
# Following a custom function to provide a dataframe of all ASVs and their phylofactor
# It requires as input the phylofactor and phyloseq objects.
# It needs to be exactly the same pholoseq object (contain the ASVs and taxa)
# that was used prior to running PhyloFactor.
# The number (5 or 4) indicates which `choice` ('var' or 'F') was provided
# to the PhyloFactor function. Either 'var' (5) or 'F' (4). Just my poor coding here.

PF_summary <- phyfacsummary(PF, ps.phylo, 5)

head(PF_summary)
```

```
## # A tibble: 6 x 24
##   OTU   Kingdom Phylum Class Order Family Genus Species Phylo~1 `Pr(>F)` (Inte~2
##   <chr> <chr>   <chr>  <chr> <chr> <chr>  <chr> <chr>     <int>    <dbl>   <dbl>
## 1 59c6~ Bacter~ Acido~ Ther~ Ther~ Therm~ Ther~ NA            1 0          3.85
## 2 2a4a~ Bacter~ Chlor~ Anae~ Anae~ Anaer~ ADur~ NA            2 3.85e-12  -0.523
## 3 ea17~ Bacter~ Gemma~ Gemm~ Gemm~ Gemma~ uncu~ NA            3 3.58e-10   3.63
## 4 b3ca~ Bacter~ Nitro~ Nitr~ Nitr~ Nitro~ Nitr~ NA            3 3.58e-10   3.63
## 5 085b~ Bacter~ Actin~ Ther~ Gaie~ uncul~ uncu~ NA            3 3.58e-10   3.63
## 6 e990~ Bacter~ Actin~ Ther~ Soli~ Solir~ Cone~ NA            3 3.58e-10   3.63
## # ... with 13 more variables: XDamhusaaen <dbl>, XEgaa <dbl>,
## #   `XEjby Moelle` <dbl>, `XEsbjerg Vest` <dbl>, XFornaes <dbl>,
## #   XHjoerring <dbl>, XMariagerfjord <dbl>, XRanders <dbl>, XSlagelse <dbl>,
## #   XSoeholt <dbl>, XViborg <dbl>, `Pr(>F)adj.` <dbl>, Abundance <dbl>, and
## #   abbreviated variable names 1: Phylofactor, 2: `(Intercept)`
```

Check out the dataframe in your R environment.

8.2.6 Visualising factors

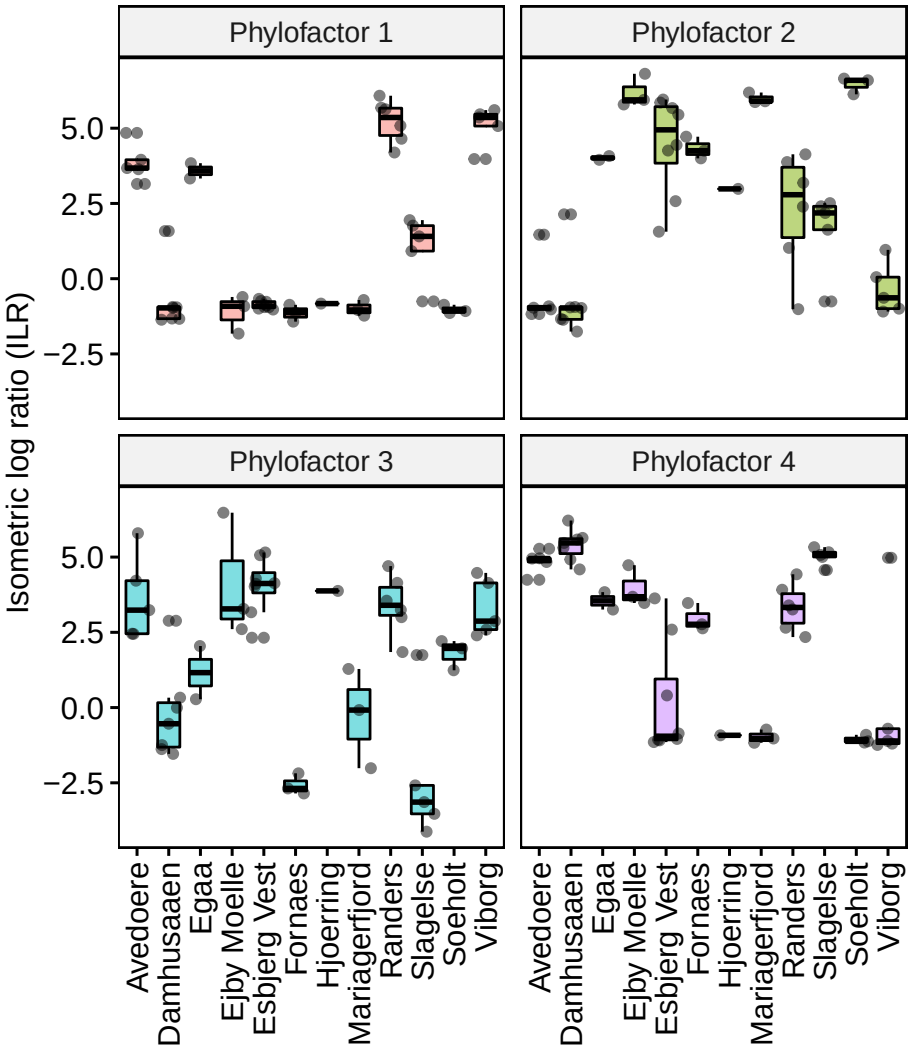
8.2.6.1 ILR boxplots

If the explanatory variable is a factor you can compare the ILRs using boxplots. Customise this to your needs.

```
## ILR boxplots
# If needed add another element to this plot. The boxplots of ILRs.
# Use the custom addILRtophyloseq function and create a new phyloseq object with
# added ILR results in the sample_data (for plotting)
ps.ILR <- addILRtophyloseq(ps.phylo, PF)
# Quick Check that they have 'arrived'.
head(data.frame(sample_data(ps.ILR)))
```

```
##          OTU_ID Assay.Type    Bases  BioProject    Bytes
## sa1 SRR12204258   AMPLICON 13254234 PRJNA645373  9235913
## sa2 SRR12204269   AMPLICON 24196788 PRJNA645373 16404070
## sa3 SRR12204280   AMPLICON 15758554 PRJNA645373 10758456
## sa4 SRR12204287   AMPLICON 20048406 PRJNA645373 13009806
## sa5 SRR12204288   AMPLICON 37634632 PRJNA645373 24207581
## sa6 SRR12204289   AMPLICON 43555904 PRJNA645373 27963491
##                          Organism collection_date          loc Reactor Type
## sa1 anaerobic digester metagenome        30/11/16  Ejby Moelle      RT <NA>
## sa2 anaerobic digester metagenome          9/3/16      Soeholt      RT <NA>
## sa3 anaerobic digester metagenome         17/3/16 Esbjerg Vest    RT1A <NA>
## sa4 anaerobic digester metagenome         5/12/16       Viborg     RT2   DS
## sa5 anaerobic digester metagenome         5/12/16       Viborg     RT2   BL
## sa6 anaerobic digester metagenome         5/12/16       Viborg     RT1   DS
##                   Purpose Experiment                 lat_lon
## sa1 reactorfoampotentials SRX8715180 55.398077 N 10.415041 E
## sa2 reactorfoampotentials SRX8715169  56.175302 N 9.582588 E
## sa3 reactorfoampotentials SRX8715158  55.488235 N 8.430655 E
## sa4  foamnofoamcomparison SRX8715151 56.425367 N 9.4527943 E
## sa5  foamnofoamcomparison SRX8715150 56.425367 N 9.4527943 E
## sa6  foamnofoamcomparison SRX8715149 56.425367 N 9.4527943 E
##              ReleaseDate  Sample.Name     Location Phylofactor_1 Phylofactor_2
## sa1 2021-01-01T00:00:00Z 16SAMP-17236  Ejby Moelle    -0.6074442    5.94063684
## sa2 2021-01-01T00:00:00Z 16SAMP-17231      Soeholt    -1.1404128    6.12600491
## sa3 2021-01-01T00:00:00Z 16SAMP-17233 Esbjerg Vest    -0.7420321    1.56133586
## sa4 2021-01-01T00:00:00Z  MQ170915-14       Viborg     5.0713421   -0.63550964
## sa5 2021-01-01T00:00:00Z  MQ170915-13       Viborg     5.4619244   -1.09156044
## sa6 2021-01-01T00:00:00Z  MQ170915-12       Viborg     5.6013748    0.05169486
##     Phylofactor_3 Phylofactor_4 Phylofactor_5
## sa1      3.278522     3.6816103     3.7423730
## sa2      1.238992    -1.1624380    -1.1654072
## sa3      3.166636     3.6278741     0.6330391
## sa4      2.590512    -0.7021602    -0.7039537
## sa5      4.140900    -1.1982694    -1.2013301
## sa6      2.871972    -1.2413042    -1.2444748
```

```
# Notice the Phylofactor columns

## Create plotdata
# Some vectors to filter and change the appearance of the plot
factorvector <- c("Phylofactor_1","Phylofactor_2",
                  "Phylofactor_3","Phylofactor_4")
labelvec = c("Phylofactor 1","Phylofactor 2",
             "Phylofactor 3","Phylofactor 4")
# rearrange the sample data for plotting
plotdata <- data.frame(sample_data(ps.ILR)) %>%
  pivot_longer(cols = starts_with("Phylofactor"),
    names_to = "ILR")
# create factors of the ILR column
plotdata$ILR <- factor(plotdata$ILR, levels = plotdata[1:PF$nfactors, ]$ILR)
# Select specific factor vectors for plotting using the factorvector to filter
plotdata_sub <- plotdata %>%
  filter(ILR %in% factorvector)
# create factors of the ILR column
plotdata_sub$ILR <- factor(plotdata_sub$ILR, levels = plotdata_sub[1:length(unique(plot
# create ggplot object
pilr <- plotdata_sub %>%
  ggboxplot(x = "Location", y = "value",
    combine = TRUE,
    facet.by = "ILR",
    fill = "ILR",
    ncol = 2,
    alpha = 0.5,
    ylab = ("Isometric log ratio (ILR)"),
    panel.labs.font = list(size = 11),
    panel.labs = list(ILR = labelvec)) +
    geom_point(position = "jitter", alpha = 0.5) +  # add jitter
    theme(legend.position = "none") +               # do not show legend
    theme(axis.text.x = element_text(angle = 90, hjust = 1,
    vjust = 0.5), axis.title.x = element_blank(), axis.text = element_text(size = 12))

# Choose custom colors with + scale_fill_manual(values = yourcolvector)
pilr
```



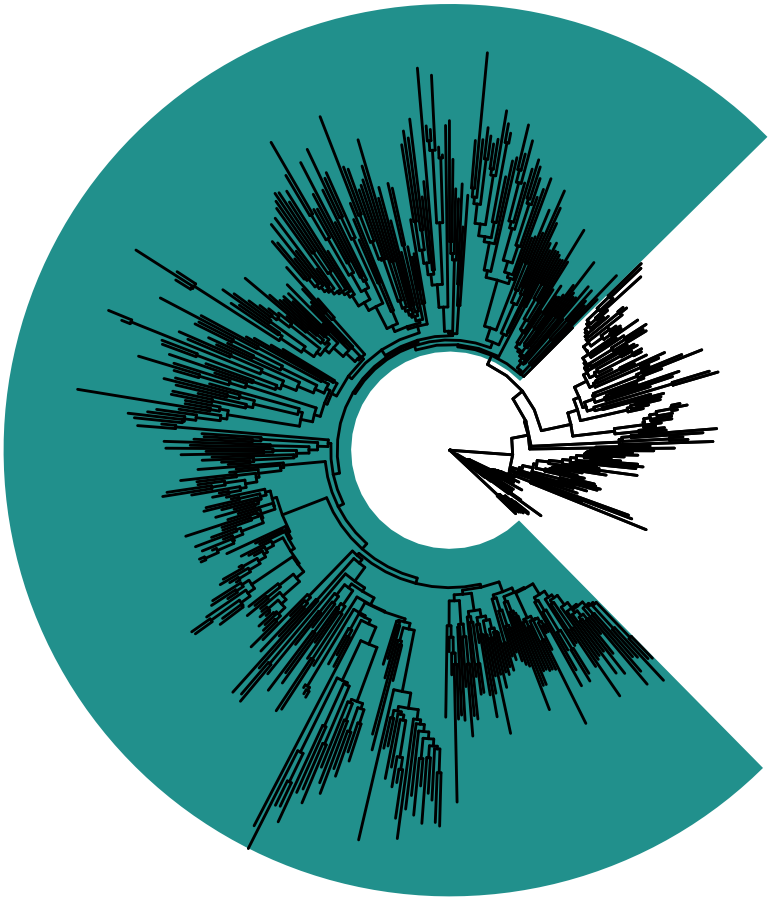
This visualises the taxa relative abundances (represented by the ILR or balances) of the different 'Locations'. For example, the relative abundances of factor 1 (genus *Thermoanaerobaculum*) are highest in Randers and Viborg. It also appears to be the factor with the clearest differentiation between locations, as the ILRs of other factors are more variable with wider boxplots and more outliers.

In the next part we will visualise how the taxa relate to other clades in the phylogenetic tree.

8.2.6.2 Basic tree with factors highlighted

Here we are using one of the functions from the phylofactor package. It is only one of several and I encourage you to check out the phylofactor tutorial to explore other visualisation, including heatmaps etc..

```
gtree <- phylofactor::pf.tree(PF)
ggtree::rotate_tree(gtree$ggplot,-45)
```



This highlights that factor 3 represents the biggest group of various genera in this case. The green colour represents Group1 and the uncolored part the remainder group (Group 2 or also called 'bin'). From the results of the summary function shown earlier in this Workflow, we can check which taxa are in each of the groups. For example, the uncolored group (Group 2) contains Proteobacteria,Cyanobacteria and Campilobacterota.

The other factors, which represent only 1 genus are not visible in this figure because the pixels are simply too small.

I think representing it as a tree also highlights how one can interpret compositional changes in two ways. For example, one interpretation for Group 1 of factor 3, (green coloured) is that is decreased in the locations Formaes and Slagelse (as indicated in the boxplots). Another interpretation is that Group 2 of factor 3 has increased instead. This is important to consider.

8.2.6.3 Coloured tree with factors highlighted in grey

Maybe some find it useful to identify the phyla directly in the tree and also get a sense of the abundances of individual tips (which are individual genera here). Do do that we can colour edges of the tree to represent their phylum.

First, prepare a color vector for the tree. There are different ways to do this and often I revert to add colours manually. Here we extract all phyla names from the taxonomy data and create a named color vector for each phylum.

```
## Create color vector
# Get a dataframe of phyla, and while we are at it we also count how often they appear
taxa.df.phylum <- phyloseq::tax_table(ps.phylo)@.Data %>% as.data.frame %>%
    rownames_to_column("OTUID") %>%
    dplyr::select(OTUID, Phylum) %>%
    column_to_rownames("OTUID") %>%
    count(Phylum) %>%
    arrange(desc(n))
# set seed differently will change the random sampling of the colours
set.seed(1234)
# randomly sample colours from this colour scheme
treecols <- sample(colorspace::rainbow_hcl(nrow(unique(taxa.df.phylum))))
# Other color options
# treecols <- sample(viridis::turbo(nrow(unique(taxa.df.phylum)), direction = -1))
# treecols <- viridis::rainbow(nrow(unique(taxa.df.phylum)), direction = -1)
names(treecols) <- unique(taxa.df.phylum$Phylum)
rm(taxa.df.phylum)
treecols
```

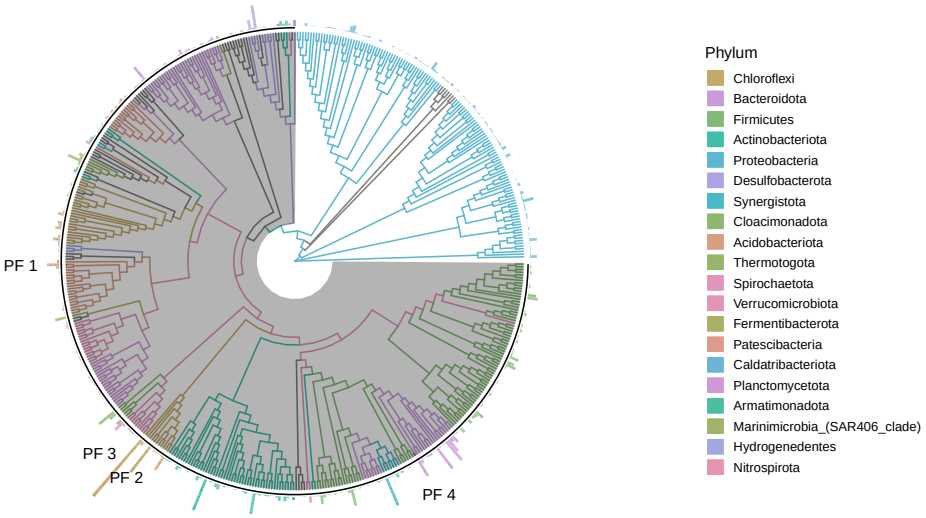
```
## Get list of the abundant phyla - using the custom function I made available above
df <- prevalencedf(ps.phylo, Phylum)
# Choose the 20 most abundant Phyla for the tree
# This vector will determine the length and order of the phyla shown in the legend
# change this to your needs.
ordervec <- df$Phylum[c(1:20)]

## Plot the tree
# setting the variables
layout = "circular"
```

```
branch.length = "none"
metadatacolumn <- 'Location'
offset.text = 5
pwidth = 0.05
factorvector = c(1,2,3,4,5)
# 'Copy' tree object from the phylofactor object into the phyloseq object for plotting
phyloseq::phy_tree(ps.phylo) <- PF$tree
# create long dataframe of metadata and select the columns OTU and Abundance
# then filter only unique rows
melt_simple <- psmelt(ps.phylo) %>%
  dplyr::select(OTU, Abundance) %>%
  unique()
# Create a phyla list to include into the tree for colouring
taxa.df.phylum <- phyloseq::tax_table(ps.phylo)@.Data %>%
  as.data.frame %>%
  rownames_to_column("OTUID") %>%
  dplyr::select(OTUID, Phylum) %>%
  column_to_rownames("OTUID")
phyla.list <- split(rownames(taxa.df.phylum),
                    taxa.df.phylum$Phylum,
                    drop = TRUE)
tree.df <- phyloseq::phy_tree(PF$tree)
ggtree_gps <- ggtree::groupOTU(tree.df, phyla.list, "Phylum")
# gg plotting
p <- ggtree::ggtree(ggtree_gps, aes(color = Phylum), layout = layout,
        branch.length = "none") +
        theme(legend.position = "right") +
        scale_colour_manual(values = treecols, limits = ordervec,
        breaks = ordervec,
        guide = guide_legend(override.aes = list(size = 5),
        keywidth = 0.2, keyheight = 0.2, order = 1))
# add barplots on the outside, representing relative abundances
p <- p + ggnewscale::new_scale_fill() +
         geom_fruit(data = melt_simple,
          geom = geom_bar,
          mapping = aes(y = OTU, x = Abundance, group = label,
          fill = Phylum), pwidth = 0.38, orientation = "y",
          stat = "identity", show.legend = FALSE) +
          scale_fill_manual(values = treecols,
          limits = ordervec,
          breaks = ordervec)
## add phylofactor highlights in grey
factor.map = data.frame(Phylofactor = 1:PF$nfactors, group = rep(1, PF$nfactors))
if (is.null(factorvector)) {
  factor.map = data.frame(Phylofactor = 1:PF$nfactors,
```

```
                          group = rep(1, PF$nfactors))
  } else {
  factor.map <- factor.map %>%
    dplyr::filter(Phylofactor %in% factorvector)
  }
m <- nrow(factor.map)
n = ape::Ntip(PF$tree)
method = "factors"
nd <- NULL
for (i in 1:m) {
  if (method == "factors") {
    grp <- PF$tree$tip.label[PF$groups[[factor.map[i,1]]][[factor.map[i, 2]]]]
  }
  else {
    grp <- PF$tree$tip.label[GroupList[[i]]]
  }
  grp <- intersect(grp, PF$tree$tip.label)
  if (length(grp) > 0 & any(setdiff(PF$tree$tip.label,grp) %in% PF$tree$tip.label)) {
    if (length(grp) > 1) {
      nd <- c(nd, tidytree::MRCA(PF$tree, grp))
    }
    else {
      nd <- c(nd, match(grp, PF$tree$tip.label))
    }
  }
}
df <- data.frame(factor.map, node = nd)
df$Phylofactor <- factor(df$Phylofactor, levels = df$Phylofactor)
p <- p + ggtree::geom_hilight(data = df, aes(node = node),
          fill = "black", alpha = 0.3)
# add a text label to identify the number and location of the phylofactor
  for (i in factorvector) {
  p <- p + geom_cladelabel(node =
                 (df %>% dplyr::filter(Phylofactor ==  i))$node, label = paste("PF",
                 (df %>% dplyr::filter(Phylofactor == i))$Phylofactor),
                 offset.text = offset.text,
                 hjust = "center")
  }
rm(taxa.df.phylum, df)
```

All the above steps can also be called with the custom function `mytreefunctionwithbarplots` which I provided in the intro. This makes for neater code in your probably already busy R studio editor. The following output is identical to the above:

```
# requires the same ordervector as created above
df <- prevalencedf(ps.phylo, Phylum)
# Choose the 20 most abundant Phyla for the tree
# This vector will determine the length and order of the phyla shown in the legend
# change this to your needs.
ordervec <- df$Phylum[c(1:20)]
p <- mytreefunctionwithbarplots(PF, ps.phylo,
         layout = "circular",
         branch.length = "none",
         factorvector =c(1,2,3,4),
         offset.text = 5,
         pwidth = 0.05)
p
```



The locations of phylofactor (PF) 1, 2, 4 and 5 are now indicated by a text label. If you zoom into the image you may also notice a grey shading on the tip of the relevant genus.

The text label for PF 3 is aligned with the outside 'centre' of the grey shaded circle representing Group 1 of this factor.

The outside bars indicate abundance of the relevant genus.

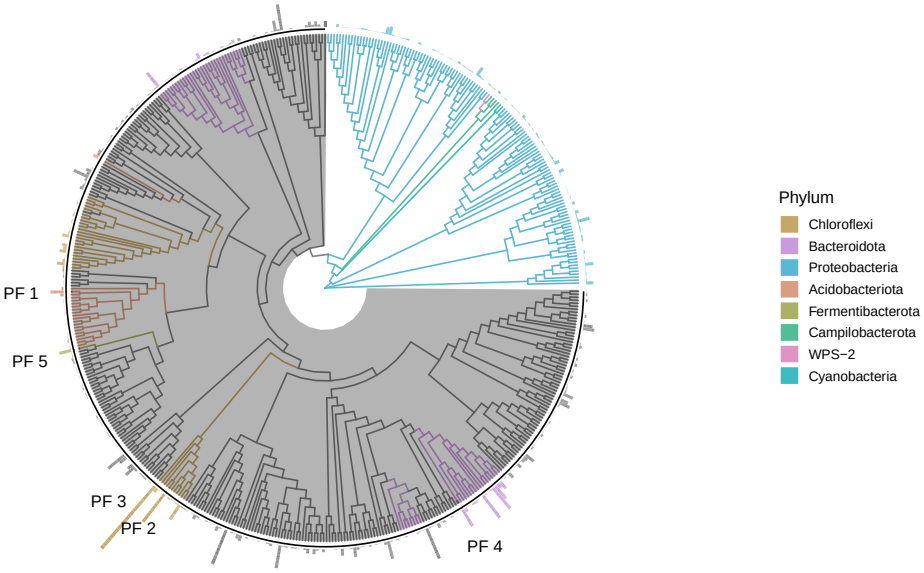
There are too many colours here and it is difficult to differentiate the different phyla. One solution for that is to create custom colours and order the legend in order of appearance of the phyla (using the `ordervec` vector). Alternatively, you can just add colour for selected phyla that you want to highlight (again using the `ordervec` vector). For example, from the summary results above we may want to highlight only the phyla in Group 2 of factor 3 (the uncoloured

remainder bin from Factor 3) and perhaps include the phyla from PF 1, 2, 4 and 5.

```
# check which phyla to highlight in tree
s$taxa.split$group2 # this is the summary for factor 3 create above
```

```
## [1] "Bacteria;Proteobacteria"   "Bacteria;Cyanobacteria"
## [3] "Bacteria;Campilobacterota" "Bacteria;WPS-2"
```

```
# check which taxa are involved for other factors
# summary(PF,taxonomy,factor=1) # Acidobacteriota;Thermoanaerobaculum
# summary(PF,taxonomy,factor=2) # Chloroflexi;Anaerolineaceae;ADurb.Bin120
# summary(PF,taxonomy,factor=4) # Bacteroidota;Rikenellaceae;DMER64
# summary(PF,taxonomy,factor=5) # Fermentibacterota

# create ordervec
df <- prevalencedf(ps.phylo, Phylum)
ordervec <- (df %>%
  dplyr::filter(Phylum %in% c("Proteobacteria", "Cyanobacteria", "Campilobacterota", "WPS-2",
                              "Acidobacteriota","Chloroflexi", "Bacteroidota", "Fermentibacterota
# plot tree
p <- mytreefunctionwithbarplots(PF, ps.phylo,
         layout = "circular",
         branch.length = "none",
         factorvector =c(1,2,3,4,5),
         offset.text = 5,
         pwidth = 0.05)
p
```



Now its a little easier to identify the who is who. This is just another way to explore your phylogeny and how they relate to the phylofactors.

Bibliography

Bharti, R. and Grimm, D. G. (2021). Current challenges and best-practice protocols for microbiome analysis. *Briefings in bioinformatics*, 22(1):178–193.

Cameron, E. S., Schmidt, P. J., Tremblay, B. J.-M., Emelko, M. B., and Müller, K. M. (2021). Enhancing diversity analysis by repeatedly rarefying next generation sequencing data describing microbial communities. *Scientific reports*, 11(1):1–13.

Gloor, G. B., Macklaim, J. M., Pawlowsky-Glahn, V., and Egozcue, J. J. (2017). Microbiome datasets are compositional: And this is not optional. *Frontiers in Microbiology*, 8(NOV).

Hugerth, L. W. and Andersson, A. F. (2017). Analysing microbial community composition through amplicon sequencing: From sampling to hypothesis testing.

McKnight, D. T., Huerlimann, R., Bower, D. S., Schwarzkopf, L., Alford, R. A., and Zenger, K. R. (2019). Methods for normalizing microbiome data: An ecological perspective. *Methods in Ecology and Evolution*.

McMurdie, P. J. and Holmes, S. (2014). Waste not, want not: why rarefying microbiome data is inadmissible. *PLoS computational biology*, 10(4):e1003531.

Washburne, A. D., Silverman, J. D., Leff, J. W., Bennett, D. J., Darcy, J. L., Mukherjee, S., Fierer, N., and David, L. A. (2017). Phylogenetic factorization of compositional data yields lineage-level associations in microbiome datasets. *PeerJ*, 5:2969.

Washburne, A. D., Silverman, J. D., Morton, J. T., Becker, D. J., Crowley, D., Mukherjee, S., David, L. A., and Plowright, R. K. (2019). Phylofactorization: a graph partitioning algorithm to identify phylogenetic scales of ecological data. *Ecological Monographs*, 89:e01353.