Workshop

Characterisation of soil and groundwater microbiology: sampling, analytical methods, interpretation and applications for agriculture management

14-15 December 2017

Sheffield, United Kingdom

The aim of this 2-day event is to explore specific microbiology techniques in the context of soil and groundwater microbiology for sustainable agriculture management. The workshop will focus on DNA sample extraction methods from complex substrates, cell counts on planktonic and attached communities, primer design for 16S rRNA sequencing, illumina sequencing, bioinformatics analysis, as well as data wrangling and presentation using open source software.

The sessions will be presented by experienced training facilitators and experts from the INSPIRATION Innovative Training Network (www.inspirationitn.eu) and the University of Sheffield. The programme includes theoretical sessions and laboratory-based demonstrations, with the objective of consolidating participant understanding and skills development. Please contact Stephen Rolfe (s.rolfe@sheffield.ac.uk) or Gabriella Kakonyi (g.kakonyi@sheffield.ac.uk) for further information.

The workshop is open to all Inspiration early-stage researchers. This is in accordance with Grant Agreement 675120 between the Inspiration Beneficiaries and the European Commission.

Schedule

Wednesday, 13 December 2017

18.00 Icebreaker and dinner in local restaurant

Day 1 : Thursday, 14 December 2017

9.00 – 9.30 Arrival/registration

9.30 – 10.30 Sequencing strategies (DNA/RNA/stable isotopes) (seminar)

10.30 – 11.30 DNA extraction and quantification methods (laboratory demonstration)

11.30 – 13.00 Lunch

13.00 – 14.00 Primer design for 16S rRNA sequencing (lab session)

14.00 – 15.00 Handling large datasets efficiently – Unix commands/Virtual boxes/ (lab session)

15.00 – 17.00 Processing pipelines for 16S rRNA Illumina sequences
Usearch and Qiime (lab session)
Filtering, chimera detection

19.00 – 21.00 Networking and dinner at local restaurant

Day 2 : Friday, 15 December 2017

9.30 – 10.30 Strategies for analysing 16S rRNA sequences (an introduction to Phyloseq) (seminar)

10.30 – 11.30 BIOM files and data input (lab session)

11.30 – 13.00 Lunch

13.00 – 16.00 DESEQ2, Phyloseq and R (lab session)
Data input and filtering, ordination methods, differential abundance analysis and display

Location

Department of Animal and Plant Sciences, Alfred Denny Building, University of Sheffield
Western Bank, Sheffield S10 2TN, UK

INSPIRATION

A short course in the analysis of 16S rRNA gene fragments

Steve Rolfe

University of Sheffield

Dec 2017

Contents

1 Handling data in Unix

Golden rule:

Only ever work on a copy of your data. If you make a mistake, then you can always start again.

Document the important steps of the analysis – including default values – as these may change in future versions of the programs

Get used to using the command line interface (CLI). Graphical interfaces are OK but do not work well when handling hundreds of files.

1.1 Basic unix commands

Unix commands are CASE SENSITIVE ('More' and 'more' are not the same)

`ls`	`ls` `ls -l`	Lists files in current directory List in long format
`ll`	`ll`	List in long format

`cd`	`cd tempdir`
	`cd ..`
	`cd ~`
		Change directory to tempdir
		Move up one directory
		Move to your 'home' directory

`mkdir`	`mkdir data`	Make a directory called data
`rmdir`	`rmdir data`	Remove directory (must be empty)
`cp`	`cp file1 data` `cp file1 file1.bak`	Copy file1 into directory 'data' Create a copy of file1 called file1.bak
`rm`	`rm file1.bak` `rm *.tmp`	Remove or delete file Remove all files with the extension *.tmp (rm *.* is very dangerous!)
`mv`	`mv file1.txt file2.txt`	Move or rename files (works for directories)
`more`	`more index.html`	Look at file, one page at a time
`man`	`man ls`	Online manual (help) about command (but Google is your friend!)
`cat`	`cat file1.txt`	Look at the contents (catalog) of a file
`head`	`head file1.txt`	Look at the top of file1.txt
`tail`	`tail file1.txt`	Look at the end of file1.txt
`$PWD`		Display current directory
`find exec`		See section 1.4

Tips

If you add & to the end of a command, it will run in the background. That's not needed for simple commands but can be useful if they are going to take a while to run.

Automatic filename expansion and wildcards

When typing a file name, press <Tab> and the filename will be completed (as far as possible).

* is a wildcard (e.g. *.txt) ? is a single character (e.g. file?.txt)

Other commands will be introduced as we go along – but these are the basics to get you started.

1.2 Compressed file formats

Data are often supplied in the ZIP format (highly compressed) to allow more rapid file transfers.

Data for this course are in the compressed file 'inspiration.tar.gz'

Unix uses lots of different compression methods which can get very complicated. Rather than explain everything, use Google to find out the best route when faced with something new.

To unpack this file (and the associated subdirectories) use:

```
tar -xvzf inspiration.tar.gz
```

Tar is a very flexible program – this will unzip all of the directories using zip format.
We can then look inside the directory

```
cd INSPIRATION_course/
ll
cd Sample_11_W1MP1/
ll

11_W1MP1_TCGACGTC-ACTGCATA_L001_R1.fastq.gz
11_W1MP1_TCGACGTC-ACTGCATA_L001_R1.fastq.gz.md5.gz
11_W1MP1_TCGACGTC-ACTGCATA_L001_R2.fastq.gz
11_W1MP1_TCGACGTC-ACTGCATA_L001_R2.fastq.gz.md5.gz
```

These files are still zipped. The fastq files are the ones we want – the md5 files are checks for file transfers.

Google 'md5sum' this if you want to know more

Go back to the main INSPIRATION_course directory

```
gunzip -r *
```

This runs the gunzip command recursively (i.e. steps down through each directory) on all files (*)

If you inspect the directory now you can see the files are unzipped.

1.3 Looking at the contents of a file

Move to the Sample_11 folder (remember the trick with the Tab key to avoid lots of typing)

Use `cat` to look at the first fastq file

```
cat 11_W1MP1_TCGACGTC-ACTGCATA_L001_R1.fastq
```

HELP!!!!!!!!!! It goes on forever!!!!!! Press CTRL-C to cancel this command.

These files are BIG – use `ll` to see how big!

This is completely impractical. We can look at the top of the file using `head`

```
head 11_W1MP1_TCGACGTC-ACTGCATA_L001_R1.fastq

@HWI-M01242:134:000000000-B35LL:1:1101:14285:1420 1:N:0:TCGACGTCACTGCATA
CCTACGGGGGGCAGCAGTGGGGAATTTTGGACAATGGGCGCAAGCCTGATCCAGCCATGCCGCGTGCAGGATGAAGGCCTTCGGGTTGTAAACT
GCTTTTGTACGGTACGAAAAGACTCTTTCTAATAAAGTGGGTCCATGACGGTACCGTAAGAATAAGCACCGGCTAACTACGTGCCAGCAGCCGC
GGTTATACGTAGGGTGCTAGCGTTAATCGGTATTTCTGGGCGTATAGCGTGCGCAGGCGGTTT
+
AAAAAA111ADACGFCCGGBCC//BBGH2/1B10B1100/<<///<?<0?1<1<?<0?1<1-@@@-..0.<<//00/.:;GG...;C-
;000;;09CFFF0F#####################################################################################
#####################################################################
```

There are 4 lines of text here.

The first line (@HWI) is created by the Illumina sequencer and has a bunch of information about where on the machine it was run (that you can safely ignore).

Then comes the sequence
Then a +
Then codes that reflect the quality of each sequence. (We'll come back to that later).

1.4 Moving files around efficiently – the find/exec combination

We need all of the fastq files in the same subdirectory. We could use mv commands to do this one-by-one.

Go to the INSPIRATION-course subdirectory

Make a directory called fastq

```
mkdir fastq

mv Sample_11_W1MP1/*.fastq fastq/
```

You could repeat this for all of the files in each subdirectory. But actually this is a cut down set of files (there were hundreds in the original data set) so moving these one at a time would take hours and be prone to error.

Therefore, let the computer do the work! You will use this concept a lot so it's worth learning.

We can enter this command (from the main course directory) and do it all in one go.

```
find . -name "*.fastq" -exec mv {} $PWD/fastq/ \;
```

There will be a warning as we have already moved a couple of the files

```
find . -name "*.fastq"
```

This 'finds' in the current directory and all of the subdirectories (.) files that are named (-name) anything (*) ending with .fastq

If you run the find part of the command on its own you will see a list of all the fastq files (that are now in the subdirectory fastq/)

We then execute (exec) the move (mv) command using these filenames { } sending them to the current directory ($PWD) and subdirectory /fastq/. The final ' \; ' is needed to say that's the end of the command.

If you enter $PWD on its own you see the current directory.

```
ll fastq/*
```

All of the fastq files are in the fastq folder.

This is a very powerful command but you need to be careful! Use the find command without the exec bit to check that it's working on the files you want.

Running a program to look at the fastq file

There are too many files and they are too big to inspect everything, but we should look at some before proceeding to a full analysis.

1.5 FASTQC – fastq quality check

FASTQC is a nice program – it runs on multiple platforms
https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

Move to your home directory

```
cd ~
```

Use wget to get the program.

```
wget https://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.5.zip

ll *.zip

unzip fastqc_v0.11.5.zip
```

This creates a subdirectory called FastQC

```
cd FastQC

cat INSTALL.txt
```

Some instructions…

Unix won't let you run a program without explicitly saying that you want to. This is controlled by the access rights (the string of rwxrwxrwx next to the filename) – read, write, execute for different users.

```
chmod 755 fastqc
```

This allows us to run the program

```
./fastqc
```

Runs it.
(This is a pain as we have to be in the correct subdirectory to run it. We can make a permanent link with:

```
sudo ln -s  ~/FastQC/fastqc /usr/local/bin/fastqc
```

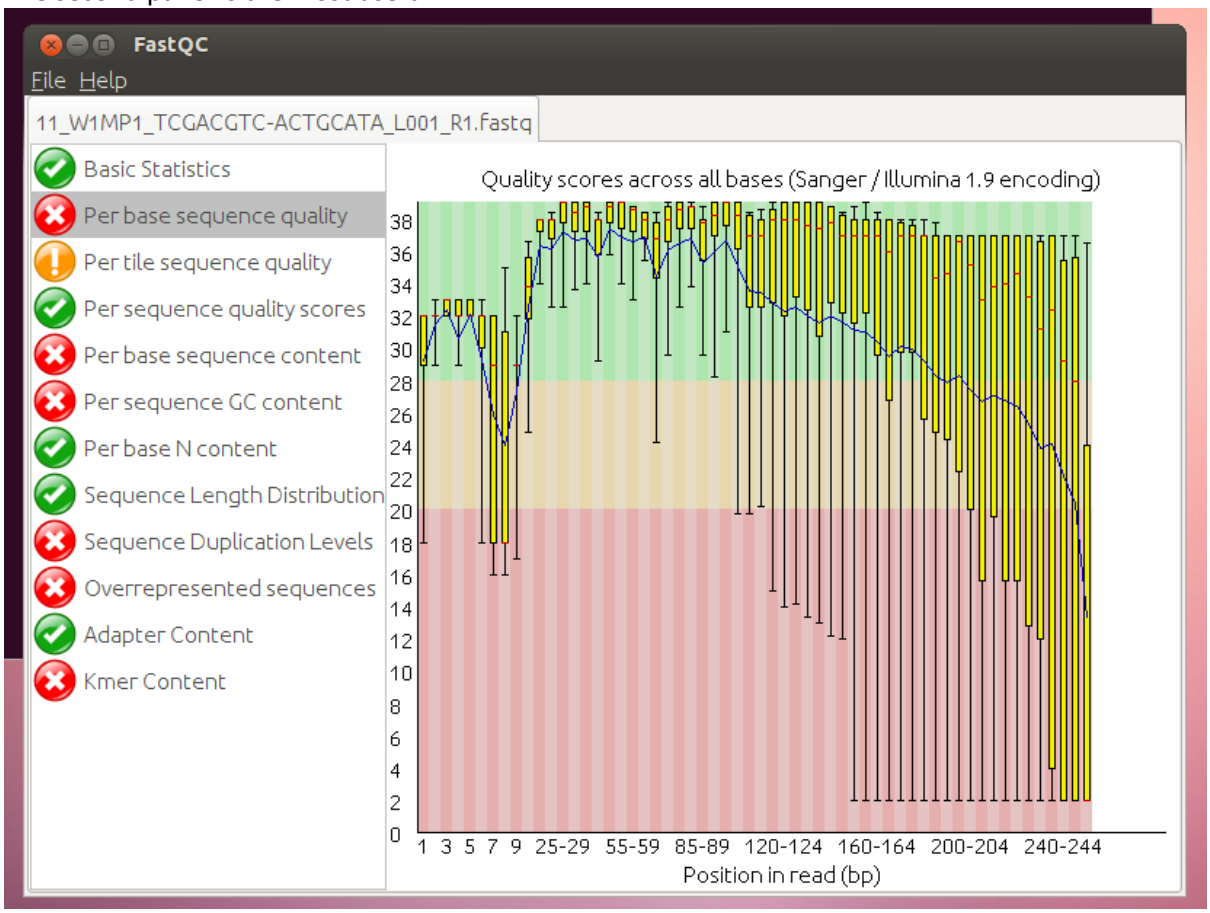
The password is qiime

Now fastqc will run every time (see Appendix 1: Installing software for more info)

```
fastqc
```

We now get a graphical interface and can inspect the FastQ files.
Navigate to the fastq directory and open one of the fastq files

The second panel is the most useful



We can see that we have 250 bp of sequence – quality is good until somewhere around the 200 bp mark and then drops off.

1.6 Quality Scores

Values are expressed as 10x log error rate

i.e.

1 error in 10	=	10
1 error in 100	=	20
1 error in 1000	=	30

What this means for your data depends very much on what you want to do with it.

2 Processing 16S rRNA sequences

Work flow up until this point:

- Extract DNA
- Amplify 16S rRNA gene fragments
- Add 'index' primers to allow samples to be identified (demultiplexing)
- Send for sequencing
- Demultiplexing of sequences based on the index primers
- Fastq files

2.1 A word on demultiplexing

Depending on your sequencing supplier, data may be provided in different ways. Sometimes it's a single big file and you need to separate out your samples, other times this will be done for you.

If your data have not been demultiplexed then programmes such as Qiime can do this for you. There are problems with very complex data sets where index primers might have been re-used but the PCR amplification primers differ. I have written a programme to cope with these, but it's best to avoid the problem in the first place.

2.2 Installing the necessary software

This analysis 'pipeline' uses a mix of programs. Originally it used the package 'Qiime' (pronounced Chime) for most steps, but we are increasingly moving to other, better routes, especially for downstream analysis.

The main problem is that once you are in a pipeline it can be hard to get out! So there's a balance between easy to use and flexibility.

One of these issues is illustrated with the interaction between the Qiime pipeline (which acts as a wrapper around lots of other programs) and a fast search program called usearch. Usearch keeps getting updated but Qiime wants older versions, so you end up handling lots of different versions. Just to complicate matters, usearch is commercial software. An open source version called vsearch is available (Rognes T, Flouri T, Nichols B, Quince C, Mahé F. VSEARCH: a versatile open source tool for metagenomics. Hrbek T, ed. PeerJ. 2016;4:e2584. doi:10.7717/peerj.2584.) but that's yet another version…..

If you have really big data sets then a PC will struggle – so you will need to swap between iceberg and your PC. Some analyses might take days so there's really no choice.

A detailed set of instructions is provided in Appendix 1: Installing software

2.3 Installing databases

What is your data? 16S, 18S, ITS?

The analysis needs to know what sort of data have been sequenced. 16S and 18S rRNA sequences are phylogenetically useful whereas ITS sequences generally are not. Therefore, the analysis may or may not perform phylogenetic analysis based on the sort of sequence you have.

To assign a taxonomy to a sequence you compare it with a database.

There's an excellent summary of some of the problems with taxonomy assignment at

http://drive5.com/usearch/manual/taxonomy_validation.html

It's well worth reading this to be wary against over-classifying. Usearch recommend RDP. However, I can't find the RDP taxonomy files in a format compatible with Qiime, so we'll use this for chimera checking (as recommended by nearly everyone) and then something else for the taxonomies.

2.4 Comparisons with databases

Qiime supplies a number of databases. There is a resource page at http://qiime.org/home_static/dataFiles.html

The 16S rRNA databases are Greengenes and Silva. These are kept up-to-date. There is quite a lot of debate about the quality of these (http://drive5.com/usearch/manual/cmd_uchime2_ref.html and other pages on usearch as to why that author in particular really dislikes the larger databases).

An alternative and smaller database but one based on purely cultured organisms is RDP.

https://sourceforge.net/projects/rdp-classifier/files/RDP_Classifier_TrainingData/

This has files called RDPClassifier_16S_trainsetNo15_rawtrainingdata.zip for 16S rRNA genes.

Fungalits_UNITE_trainingdata_date.zip for ITS.

For 18S analysis then the Silva database contains both 16S and 18S (bacteria, Archaea and eukaryotes).

You can see what and where things are installed within Qiime using:

```
print_qiime_config.py

pick_otus_reference_seqs_fp:  /usr/local/lib/python2.7/dist-
packages/qiime_default_reference/gg_13_8_otus/rep_set/97_otus.fasta

assign_taxonomy_reference_seqs_fp:  /usr/local/lib/python2.7/dist-
packages/qiime_default_reference/gg_13_8_otus/rep_set/97_otus.fasta

assign_taxonomy_id_to_taxonomy_fp:  /usr/local/lib/python2.7/dist-
packages/qiime_default_reference/gg_13_8_otus/taxonomy/97_otu_taxonomy.txt
```

So by default Qiime will use Greengenes (gg). That's what we will use for this training package.

If you want to use something else, then you will need to download it and refer to it at appropriate times. It is always a good idea to know what files you are using as, when you come to write it up, you need to say what databases, default settings etc were used. Many Qiime scripts take a parameter file that lets you over-ride the default values – it's probably safer to use this rather than rely on the defaults which might change in a future release.

2.5 Download RDP (for 16S)

The RDP gold database is much smaller and recommended by usearch.

It can be found at http://www.drive5.com/usearch/manual/utax_downloads.html

Download the rdp training set and save it on your Shared Folder

With some analyses we hit problems doing this. Errors popped up at the time OTU tables were created. This has been reported with Qiime 1.9 on some forums but no solution has appeared. In this case use the default greengenes or Silva.

The downloaded file is in RDPClassifier_16S_trainsetNo15_rawtrainingdata/trainset15_092015.fa which is horrible to remember. We'll rename it something more memorable.

In your Shared_Folder create a folder rdp

```
mkdir rdp
cp RDPClassifier_16S_trainsetNo15_rawtrainingdata/trainset15_092015.fa
rdp/rdp_gold.fa
```

2.6 Getting to know your data

It is really important that you understand your data set. Obviously you will have a better understanding when you have prepared it yourself, but be critical of each step of the process.

You have been provided with a cut-down data set of microbial sequences produced by Juan Mujica.

The project

Ground water was collected from a contaminated aquifer, 30 metres below ground level. This is a pollution plume from an old coal tar distillation site and is rich in phenols and other phenolics. Water from the field was introduced into the microcosm and incubated at 10 ^{o}C. Samples were taken at intervals (W = week X = number of weeks e.g. W26 is week 26) from the planktonic phase (P) or attached phase (A). There were 3 replicate measurements (1,2,3).

11_W1MP1_TCGACGTC-ACTGCATA_L001_R1.fastq

Sample 11

Week 1

Microcosm

Planktonic replicate 1

Index primer

Lane 1 (from the Illumina machine)

R1 (read 1 = forward, read 2 = reverse)

.fastq It's a fastq file.

We have 36 files which are weeks 1,4 and 26 x attached/planktonic x 3 replicates x forward/reverse reads

These were amplified using the following primers using the indices listed in the filenames. These indices have already been removed by the demultiplexing process but the amplification primers are still there.

Illumina-341F (5`- TCGTCGGCAGCGTCAGATGTGTATAAGAGACAGCCTACGGGNGGCWGCAG -3`)
Illumina-805R (5`- GTCTCGTGGGCTCGGAGATGTGTATAAGAGACAGGACTACHVGGGTATCTAATCC

The underlined region is the 16S primer, the rest is needed by the Illumina sequencer.

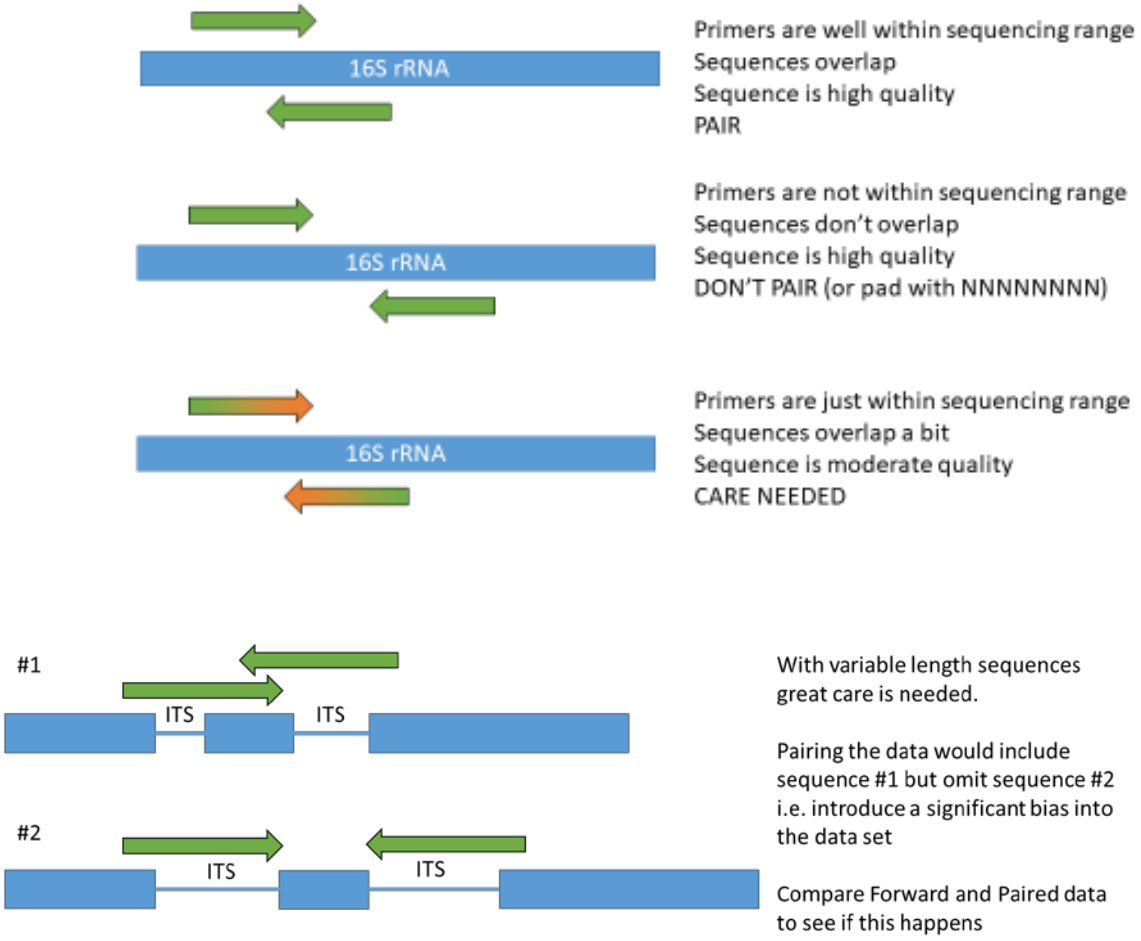
2.7 Important decisions

Most Illumina sequencing machines produce paired reads. However, it might not always be appropriate to analyse these reads as paired. This is a very important decision and depends upon:

- The primers used for amplification
- The length of the sequences obtained
- The quality of the sequence obtained
- The expected variation in the sequences expected

If you have any doubt it's worth analysing the forward reads on their own and comparing them with the paired reads (for technical reasons the read quality of the reverse reads is sometimes lower).

If they produce similar results, then you can use the paired reads (and the longer reads give better phylogenetic information). However, inappropriate use of paired reads can lead to the generation of artefacts in the data that can completely confound your experiments.



Quality checks

It's a good idea to get a feel for the quality of your data. Use FastQC as described in FASTQC – fastq quality check. Further quality checks and filtering are done in the analysis pipeline.

2.8 Usearch quality checks

You need to start the Qiime machine and then navigate to the fastq folder.

Usearch has a useful quality check

```
usearch100 -fastq_eestats2 53_W26MA1_ATGCGCAG-CGTCTAAT_L001_R1.fastq -
output stats.txt
```

```
cat stats.txt

316528 reads, max len 251, avg 251.0

Length         MaxEE 0.50         MaxEE 1.00         MaxEE 2.00
------   ----------------   ----------------   ----------------
    50     307895( 97.3%)     316080( 99.9%)     316299( 99.9%)
   100     270905( 85.6%)     300633( 95.0%)     314544( 99.4%)
   150     162278( 51.3%)     195972( 61.9%)     225708( 71.3%)
   200     135387( 42.8%)     165175( 52.2%)     195863( 61.9%)
   250      77409( 24.5%)     103687( 32.8%)     132542( 41.9%)
```

The norm is to use a maxEE value of 1 (an error of 1 in 1000 – an alternative to the Q30 score)

We can see that only 32.8% of our sequences pass this at full length – but many more pass at shorter read lengths or with higher error rates.

We actually have paired reads and so we can use the best quality sequences from each read. But we need to be careful that we don't introduce biases.

2.9 Merging paired data with usearch

```
usearch100 -fastq_mergepairs 11_W1MP1_TCGACGTC-ACTGCATA_L001_R1.fastq -
fastqout W1MP1.pair.fastq
```

This merges the forward read (R1) with the reverse read (looks for the same filename with R2) and creates a new file called W1MP1.pair.fastq

```
00:01 90Mb    100.0% 67.4% merged

Totals:
    102863  Pairs (102.9k)
     69305  Merged (69.3k, 67.38%)
…and so on
```

We could do this laboriously for each sample but we can used find/exec to do the work for us.

```
find . –name "???_*R1.fastq" -exec usearch10 -fastq_mergepairs {} -fastqout
{}.pair \;
```

We should delete the one we did manually

```
rm W1MP1.fastq
```

We now filter for quality and discarding short reads.

The primers are 341F and 805R so we should have 464 bp of sequence – but that might vary between organisms. So we'll require a minimum of 400 bp

Do it with one sequence first to check the parameters

```
usearch10 -fastq_filter 11_W1MP1_TCGACGTC-ACTGCATA_L001_R1.fastq.pair -
fastaout test.fasta -fastq_maxee 1 -fastq_minlen 400
```

```
usearch v10.0.240_i86linux32, 4.0Gb RAM (11.2Gb total), 3 cores
(C) Copyright 2013-17 Robert C. Edgar, all rights reserved.
http://drive5.com/usearch

License: s.rolfe@sheffield.ac.uk

00:01 22Mb    100.0% Filtering, 65.6% passed
     69305  Reads (69.3k)
     18584  Discarded reads with expected errs > 1.00
     45432  Filtered reads (45.4k, 65.6%)
```

We've kept 65% of our reads – so that's reasonable. Don't be afraid to throw out poor quality data. Better to analyse fewer sequences of good quality than lots of rubbish.

Get rid of all the existing fasta files

```
rm *.fasta

find . –name "*.pair" -exec usearch10 -fastq_filter {} -fastaout {}.fasta -
fastq_maxee 1 -fastq_minlen 400 \;
```

Move all the fasta files to a subdirectory called fasta

```
mkdir ../fasta

mv *.fasta ../fasta/
```

Qiime doesn't like underscores in the filename so we'll get rid of them

```
rename s/_//g *.fasta
```

2.10 Create a mapping file

The analysis program needs to know who's in what file. We do this using a mapping file.

It contains columns of data (separated with tabs) with the following format

```
#SampleID   unique sample IG
BarcodeSequence   This is used for separating out data – TGAC has done this
so we put a tab in place
LinkerPrimerSequence    The sequence used for Illumina sequencing
InputFilename     The fasta filename
Description something meaningful
```

The filenames are a pain to type so we will use the list command to grab them into a file

```
ls -1 >files.csv
```

We then open this file in Excel (or Libre Office Calc to stay in Unix)

You can add other columns after this if you like to add more information (known as metadata)

It is a bit tricky to get this right so you've been supplied with the file 'map.txt' that contains this info.

```
#SampleID        BarcodeSequence  LinkerPrimerSequence      InputFilename    Description      Week
        Status
W1MP1            CCTACGGGNGGCWGCAG11W1MP1TCGACGTC-ACTGCATAL001R1.fastq.pair.fasta    1P      1       P
W1MP2            CCTACGGGNGGCWGCAG12W1MP2TGCAGCTA-ACTGCATAL001R1.fastq.pair.fasta    1P      1       P
W1MP3            CCTACGGGNGGCWGCAG13W1MP3CGATCAGT-ACTGCATAL001R1.fastq.pair.fasta    1P      1       P
W4MP1            CCTACGGGNGGCWGCAG20W4MP1ATGCGCAG-ACTGCATAL001R1.fastq.pair.fasta    4P      4       P
W4MP2            CCTACGGGNGGCWGCAG21W4MP2TACGCTGC-ACTGCATAL001R1.fastq.pair.fasta    4P      4       P
W4MP3            CCTACGGGNGGCWGCAG22W4MP3CGGAGCCT-ACTGCATAL001R1.fastq.pair.fasta    4P      4       P
W1MA1            CCTACGGGNGGCWGCAG37W1MA1TCGACGTC-TCTCTCCGL001R1.fastq.pair.fasta    1A      1       A
W1MA2            CCTACGGGNGGCWGCAG38W1MA2TGCAGCTA-TCTCTCCGL001R1.fastq.pair.fasta    1A      1       A
W1MA3            CCTACGGGNGGCWGCAG39W1MA3CGATCAGT-TCTCTCCGL001R1.fastq.pair.fasta    1A      1       A
W4MA1            CCTACGGGNGGCWGCAG41W4MA1ACTGAGCG-TCTCTCCGL001R1.fastq.pair.fasta    4A      4       A
W4MA2            CCTACGGGNGGCWGCAG42W4MA2TAGCGCTC-TCTCTCCGL001R1.fastq.pair.fasta    4A      4       A
W4MA3            CCTACGGGNGGCWGCAG43W4MA3ATGCGCAG-TCTCTCCGL001R1.fastq.pair.fasta    4A      4       A
W26MA1           CCTACGGGNGGCWGCAG53W26MA1ATGCGCAG-CGTCTAATL001R1.fastq.pair.fasta   26A     26      A
W26MA2           CCTACGGGNGGCWGCAG54W26MA2TACGCTGC-CGTCTAATL001R1.fastq.pair.fasta   26A     26      A
W26MA3           CCTACGGGNGGCWGCAG55W26MA3CGGAGCCT-CGTCTAATL001R1.fastq.pair.fasta   26A     26      A
W26MP1           CCTACGGGNGGCWGCAG65W26MP1CGATCAGT-GTAAGGAGL001R1.fastq.pair.fasta   26P     26      P
W26MP2           CCTACGGGNGGCWGCAG66W26MP2CCTAAGAC-GTAAGGAGL001R1.fastq.pair.fasta   26P     26      P
W26MP3           CCTACGGGNGGCWGCAG67W26MP3ACTGAGCG-GTAAGGAGL001R1.fastq.pair.fasta   26P
```

We know need to convert the sequences to Qiime format (it's still fasta but it changes the names a bit)

```
cd ..

add_qiime_labels.py -m fasta/map.txt -i fasta/ -c InputFilename -o fout/
```

New sequences are in fout/combined_seqs.fna

```
head fout/combined_seqs.fna
```

2.11 Removing primers

The TGAC data has the barcoding sequences removed but the amplicon primer sequences are still in place.

THESE SHOULD BE REMOVED AS THEY ARE NOT RELIABLE SEQUENCE.

This shows up in the FASTQC **Per Base sequence content** option

You can see the variability in the first few basepairs (which are the primers) is much less than the rest. These (and the reverse primers) need to be trimmed off. We do this after merging the sequences.

If the data are paired (as these are) we use

```
usearch10 -fastx_truncate fout/combined_seqs.fna -stripleft 17 -stripright
21 -fastaout fout/combined_np.fna
```

If not paired then don't add the stripright.

Look at the sequences using the 'head' command and make sure that they match your primers.

2.12 Remove chimeras

Chimeras are a common problem in PCR reactions. They form if DNA is fragmented or some extensions don't go to full length.

Remove chimeras by comparing with rdp_gold.fa (needs to be downloaded). This is a compute intensive step and can take some time.

```
identify_chimeric_seqs.py -i fout/combined_np.fna -m usearch61 -o chimeras/
-r ~/Desktop/Shared_Folder/rdp/rdp_gold.fa
```

Looking in chimeras/ we see identify_chimeric_seqs.log

```
ref_non_chimeras  889458
ref_chimeras      42388
denovo_chimeras   12722
denovo_non_chimeras     919124
```

The algorithm uses two approaches to identify chimeras. The first is to compare sequences with a reference database and look for sharp transitions in sequence similarity. The *de novo* approach looks within the data set and sees if new sequences appear which are composed of existing sequences. Both approaches have limitations.

About 10% are chimeras which is to be expected. These are listed in chimeras.txt.

Remove these from combined_seqs.fna

```
count_seqs.py -i fout/combined_np.fna

931846  : fout/combined_np.fna (Sequence lengths (mean +/- std): 420.4176
+/- 10.2715)
931846  : Total

filter_fasta.py -f fout/combined_np.fna -o fout/combined_seqs_nc.fna -s
chimeras/chimeras.txt -n

count_seqs.py -i fout/combined_seqs_nc.fna

923228  : fout/combined_seqs_nc.fna (Sequence lengths (mean +/- std):
420.3966 +/- 10.2815)
923228  : Total
```

We've removed the chimeras

3 Pick OTUs – for 16S rRNA

Now we can pick OTUs – we can choose the database. The default is to use Greengenes at 97% similarity but we'll control it here so that you can see how to change the database and the similarity settings. We choose 97% as that is a good balance between grouping similar sequences (maybe with some small sequencing errors or very closely related organisms) and identifying truly different organisms.

You need to think carefully about the consequences of this. If you group OTUs at 97% similarity then you can't meaningfully talk about differences between sequences that might by 98% or 99% similar – you've lumped them all together. If you use too high a value, then small errors and sequence differences will inflate estimations of microbial diversity.

We control the database and the similarity as these are linked (you need to select a database at a given similarity and match your sequences to it at the same value).

We will use the usearch program (the older version 6.1) but run it within Qiime.

Qiime has lots of alternatives for identification – see this link for details.

http://qiime.org/tutorials/otu_picking.html

We will use open reference OTU picking.

- Cluster the sequences at 97% similarity against the reference database
- Any that don't hit are clustered against each other
- Generate a representative sequence for each OTU (pick the most common in the cluster)
- Assign taxonomy using the reference database
- Create a phylogenetic tree

Create a parameter file (param.txt) with the following entries

```
pick_otus:similarity 0.97

pick_otus:refseqs /usr/local/lib/python2.7/dist-
packages/qiime_default_reference/gg_13_8_otus/rep_set/97_otus.fasta
```

Then we do the picking.

```
pick_open_reference_otus.py -i $PWD/fout/combined_seqs_nc.fna -o
$PWD/usearch_otu_gg/ -m usearch61 –p param.txt
```

This is a slow step. When it's done we have a biom file that contains our sequences.

3.1 BIOM tables

```
biom summarize-table -i
usearch_otu_gg/otu_table_mc2_w_tax_no_pynast_failures.biom

Num samples: 18
Num observations: 3716
Total count: 922067
Table density (fraction of non-zero values): 0.211

Counts/sample summary:
 Min: 14768.0
 Max: 121292.0
 Median: 47900.000
 Mean: 51225.944
 Std. dev.: 25924.576
 Sample Metadata Categories: None provided
 Observation Metadata Categories: taxonomy

Counts/sample detail:
 W1MA1: 14768.0
 W4MP1: 16767.0
 W4MA1: 28545.0
 W4MP2: 29217.0
 W1MA2: 31164.0
 W26MA2: 39795.0
 W4MP3: 39877.0
 W1MP1: 44743.0
 W26MP3: 45432.0
 W1MP3: 50368.0
 W1MP2: 50553.0
 W4MA3: 51160.0
 W26MP1: 59560.0
```

```
W4MA2: 65080.0
W1MA3: 66056.0
W26MA3: 71102.0
W26MA1: 96588.0
W26MP2: 121292.0
```

We can see that we have a minimum of 14768 sequences in 18 samples. The taxonomy is present as taxonomy metadata but the sample metadata gets lost in the process so we add it back in.

Biom files also come in lots of 'flavours'. Our downstream analysis will be performed in phyloseq so we need to use a format that is compatible with this.

Qiime also throws out sequences it only sees once (mc2 means minimum count 2). Experience tells us that the singletons tend to be sequencing errors. We also only keep sequences that align at some level with the 16S rRNA database – other DNA may be contaminants.

Be aware that if you are interested in the 'rare' biosphere then this route will NOT be appropriate. However, if you want to know the dominant organisms in a system then this is fine.

3.2 Finding out what your sequences really are!

The analysis clusters your sequences and picks a representative set. It does this on the basis of clustering by similarity (at 97%) and then picking the most common sequence.

97% is rather arbitrary works quite well. But it means there's no point looking at your taxonomy assignments and believing you can tell apart organisms which are closer than 97% similar.

In the usearch_otu_gg output files you will find a couple of useful files.

final_otu_map.txt contains a 'map' that links your sequences with the representative sequences

Here's the top of such a file.

```
4479946     W4MA2_804877      W4MA2_829666      W4MA2_794769
      W26MA1_13240      W4MA2_804878
4371191     W1MA3_519884      W4MA2_810671      W26MA2_922335
      W4MP2_773156      W26MA1_1645 W1MP2_116956      W4MA3_465722
```

If I look in rep_set.fna I can find

```
>4479946 W4MA2_804877
TGGGGAATTTTGGACAATGGGCGCAAGCCTGATCCAGCCATGCCGCGTGTGTGATGAAGGCCTTCGGGTTGTAAA
GCACTTTCGGACGGAACGAAATCGCGCGGGCGAATATCCCGCGTGGATGACGGTACCGTAAGAAGAAGCACCGGC
TAACTACGTGCCAGCAGCCGCGGTAATACGTAGGGTGCGAGCGTTAATCGGAATTACTGGGCGTAAAGGGTGCGC
AGGCGGCTCCGCAAGTCAGACGTGAAATCCCCGGGCTTAACTTGGGAATGGCGTTTGGAACTATGGAGCTCGAGT
GTGGCAGAGGGAGGTGGAATTCCACGTGTAGCGGTGAAATGCGTAGATATGTGGAGGAACACCGATGGCGAAGGC
AGCCTCCTGGGCTAACACTGACGCTCATGCACGAAAGCGTGGGGAGCAAACA
```

This is the representative sequence used for taxonomic assignment.

It's actually sequence W4MA2_804877

We could use online databases to look further into this. But we won't – as you have nearly a million sequences to deal with.

The file index.html has information about the files that have been produced.

3.3 Qiime or phyloseq?

Qiime has lots of analysis programs available but I prefer to do the analysis in phyloseq (in R) as it is much more flexible and allows full access to the wide range of R statistics. This seems to be the way that things are heading.

Getting your BIOM file ready for R

The input into the phyloseq script is a biom file produced by Qiime. It needs to have a Greengenes taxonomy assigned and samples info added. This should not be rarefied or filtered (other than removing singletons) as we'll do that in phyloseq. It's easier that way as phyloseq will deal with keeping track of all of the info needed.

If you've followed the standard analysis then you need the files
otu_table_mc2_w_tax_no_pynast_failures.biom

rep_set.fna

rep_set.tre

map.txt

Copy these to a new subdirectory called phyloseq. Rename the biom file as otu_table.biom to keep things manageable. Also copy your map.txt file to this directory.

If you are in the directory where your original biom table is stored, these would be the commands.

```
mkdir ../phyloseq
cp otu_table_mc2_w_tax_no_pynast_failures.biom ../phyloseq/otu_table.biom
cp rep_set.fna ../phyloseq
cp rep_set.tre ../phyloseq
cd ..
cp ../fasta/map.txt phyloseq/
cd phyloseq
```

If we look at the biom file we can see it has taxonomy but no sample metadata. We need to add it.

```
biom summarize-table -i otu_table.biom
biom add-metadata -i otu_table.biom -o otu_table_md.biom --sample-metadata-
fp map.txt
biom summarize-table -i otu_table_md.biom
```

The biom files needs to be in json format.

```
biom convert -i otu_table_md.biom -o json.biom --table-type="OTU table" --
to-json --header-key taxonomy
biom summarize-table -i json.biom
```

Now we can shift to R

4 R and RStudio

R will run on lots of platforms but personally I find it easier to use Windows.

You need a current version of R and RStudio

https://www.rstudio.com/products/rstudio/download/

The free Desktop version is all you will need.

This tutorial can't make you proficient in R but it will get you started and allow you to analyse your data.

4.1 Loading libraries

This script requires properly formatted biom files as an input

First time you need to install the libraries. You only need to do this once so I've commented these out.

```
#source("https://bioconductor.org/biocLite.R")
#biocLite("phyloseq")
rm(list=ls())
library(phyloseq)
library(DESeq2)
library(ggplot2)
library(Biostrings)
library(ape)
library(cowplot)
library(dplyr)
library(vegan)
library(mvabund)
library(venneuler)
library("RColorBrewer")
library("gplots")
library(reshape2)

#Phyloseq is nice but lacks some functions - here's some additional code t
o help

#install.packages("devtools")
#library(devtools)
#install_github("MadsAlbertsen/ampvis2")
library(ampvis2)
```

4.2 Reading in your data

Before going any further, use the Session > Set Working Directory > Choose Destination to select your phyloseq folder

First we read in the necessary files and tweak them a bit to make phyloseq happy Don't worry too much about the technical details here!

```
setwd("D:/Google Drive/Inspiration course/phyloseq")

myFile="json.biom"
treeFile="rep_set.tre"
refseqFile="rep_set.fna"

myBiom=import_biom(myFile,treefilename=treeFile,parseFunction = parse_taxo
nomy_greengenes)
```

```
#the reference sequences have extra information beyond the OTU name.
#so read in the sequence and the names
bs=readDNAStringSet(refseqFile)
#use some code to get rid of everthing after the space
names(bs) <- gsub("\\s.+$", "", names(bs))
#now merge into the biom data
myBiom=merge_phyloseq(myBiom,bs)

#The phylogenetic tree is 'unrooted' which means that results may differ e
ach time we run phylsoeq
#The recommendation is to root it

phy_tree(myBiom)<-root(phy_tree(myBiom),sample(taxa_names(myBiom),1),resol
ve.root=TRUE)

#One of the problems with the Greengene taxonomies is that it is incomplet
e.
#We'll pad things out with sp.
#get rid of the NAs in the tax table

t<-tax_table(myBiom)
t<-as.matrix(data.frame(t))

for (row in 1:nrow(t)) {
  tname=""
  for (col in 1:ncol(t)){
    if (is.na(t[row,col]))
    {t[row,col]=paste(tname,"sp")}
      else {(tname=t[row,col])}
  }
}
tax_table(myBiom)<-t
```

4.3 Inspecting the biom object

Now we can inspect the myBiom object. Phyloseq has a number of functions that allow you to extract elements from the phyloseq object

```
myBiom
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 3716 taxa and 18 samples ]
## sample_data() Sample Data:       [ 18 samples by 6 sample variables ]
## tax_table()   Taxonomy Table:    [ 3716 taxa by 8 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 3716 tips and 3700 internal nodes ]
## refseq()      DNAStringSet:      [ 3716 reference sequences ]
```

```
ntaxa(myBiom)
```

```
## [1] 3716
```

```
nsamples(myBiom)
```

```
## [1] 18
```

```
sample_names(myBiom)
```

```
##  [1] "W4MA2"  "W26MA1" "W1MA3"  "W26MA2" "W4MP2"  "W1MP2"  "W4MA3"
##  [8] "W1MA2"  "W26MP2" "W26MP1" "W4MA1"  "W1MP3"  "W26MA3" "W1MP1"
## [15] "W4MP3"  "W1MA1"  "W26MP3" "W4MP1"
```

```
rank_names(myBiom)
```

```
## [1] "Kingdom" "Phylum"  "Class"   "Order"   "Family"  "Genus"   "Specie
s"
## [8] "Rank1"
```

```
sample_variables(myBiom)
```

```
## [1] "Status"               "Week"                 "LinkerPrimerSequence
"
## [4] "Description"          "BarcodeSequence"      "InputFilename"
```

```
sample_sums(myBiom)
```

```
##  W4MA2 W26MA1  W1MA3 W26MA2  W4MP2  W1MP2  W4MA3  W1MA2 W26MP2 W26MP1
##  65080  96588  66056  39795  29217  50553  51160  31164 121292  59560
##  W4MA1  W1MP3 W26MA3  W1MP1  W4MP3  W1MA1 W26MP3  W4MP1
##  28545  50368  71102  44743  39877  14768  45432  16767
```

We can see that the data are loaded, that we have 18 samples, 3716 unique OTUs and a set of OTU counts.

Obviously we have different sampling depths. Some analyses require sampling depths to be equal. Others (such as DESQ2) require the raw data. So we'll create a rarefied biom file with equal numbers of counts. Obviously, this can't be higher than the minimum so you have to choose whether this is OK, or whether to exclude certain samples.

4.4 Filtering and transforming the data

We will also filter out rare sequences. For DESEQ2 analyses these just get in the way - but obviously for richness analysis we should keep them.

There are lots of OTUs that are only in one sample or have very few counts. Leaving these in makes the downstream analysis slow but, more importantly, messes up the fitting of models. There's so much noise that the models are swamped. Therefore we remove all the OTUs where we have few counts and also those that only appear in a few samples.

```
#lots of OTUs
ntaxa(myBiom)
```

```
## [1] 3716
```

```
#require a minimum total count of 30 reads
myFiltBiom<-filter_taxa(myBiom,function(x) sum(x)>30,TRUE)
ntaxa(myFiltBiom)
```

```
## [1] 1087
```

```
#filter again so that we must see a sample at least 3 times in 20% of the
samples
```

```
myFiltBiom<-filter_taxa(myFiltBiom,function(x) sum(x>3)>(0.2*length(x)),TR
UE)
ntaxa(myFiltBiom)

## [1] 571
```

We should transform to an even sampling depth for ordinations This has an element of randomness to it (which ones do we pick?) so we'll control this with the set.seed command. This sets the 'seed' used in the random number generator. The number doesn't matter - just that we set it to something.

```
set.seed(3)
myEvenBiom=rarefy_even_depth(myFiltBiom)

## You set `rngseed` to FALSE. Make sure you've set & recorded
##  the random seed of your session for reproducibility.
## See `?set.seed`

## ...

sample_sums(myEvenBiom)

##  W4MA2 W26MA1  W1MA3 W26MA2  W4MP2  W1MP2  W4MA3  W1MA2 W26MP2 W26MP1
##  12591  12591  12591  12591  12591  12591  12591  12591  12591  12591
##  W4MA1  W1MP3 W26MA3  W1MP1  W4MP3  W1MA1 W26MP3  W4MP1
##  12591  12591  12591  12591  12591  12591  12591  12591

#For displaying graphs it's useful to have relative abundances (e.g. %)
mynormBiom=transform_sample_counts(myFiltBiom,function(x) 100*x/sum(x))
sample_sums(mynormBiom)

##  W4MA2 W26MA1  W1MA3 W26MA2  W4MP2  W1MP2  W4MA3  W1MA2 W26MP2 W26MP1
##    100    100    100    100    100    100    100    100    100    100
##  W4MA1  W1MP3 W26MA3  W1MP1  W4MP3  W1MA1 W26MP3  W4MP1
##    100    100    100    100    100    100    100    100
```

4.5 Ampvis – a useful additional package

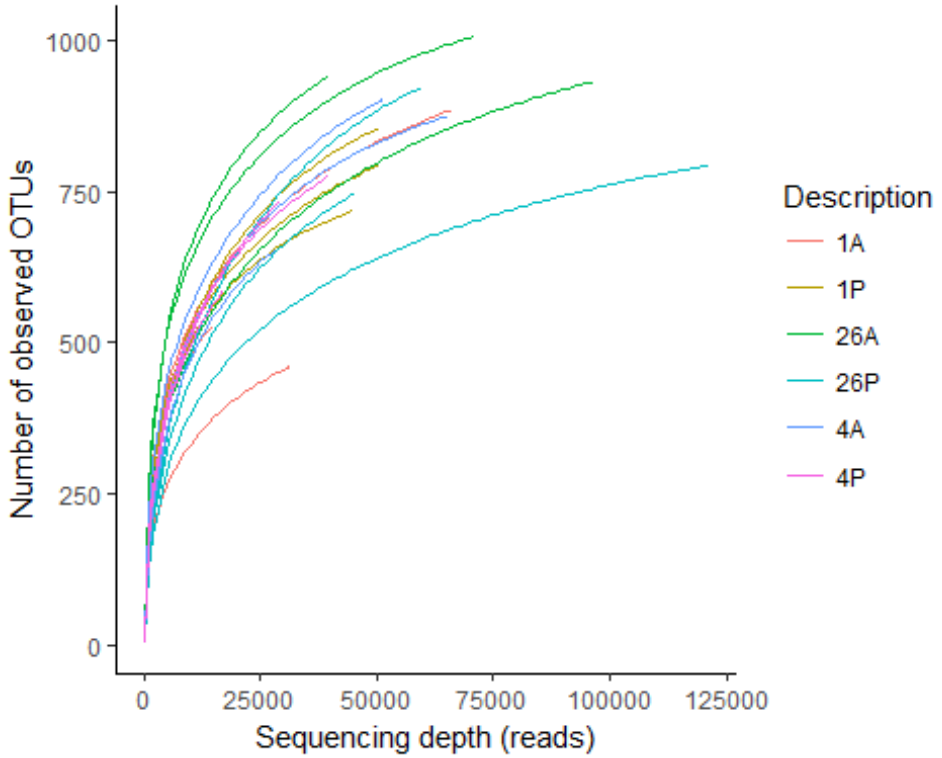
Rarefaction

Look at the sequencing depth to see whether we have done enough

ampvis makes this easy but we need to re-load the data for this package

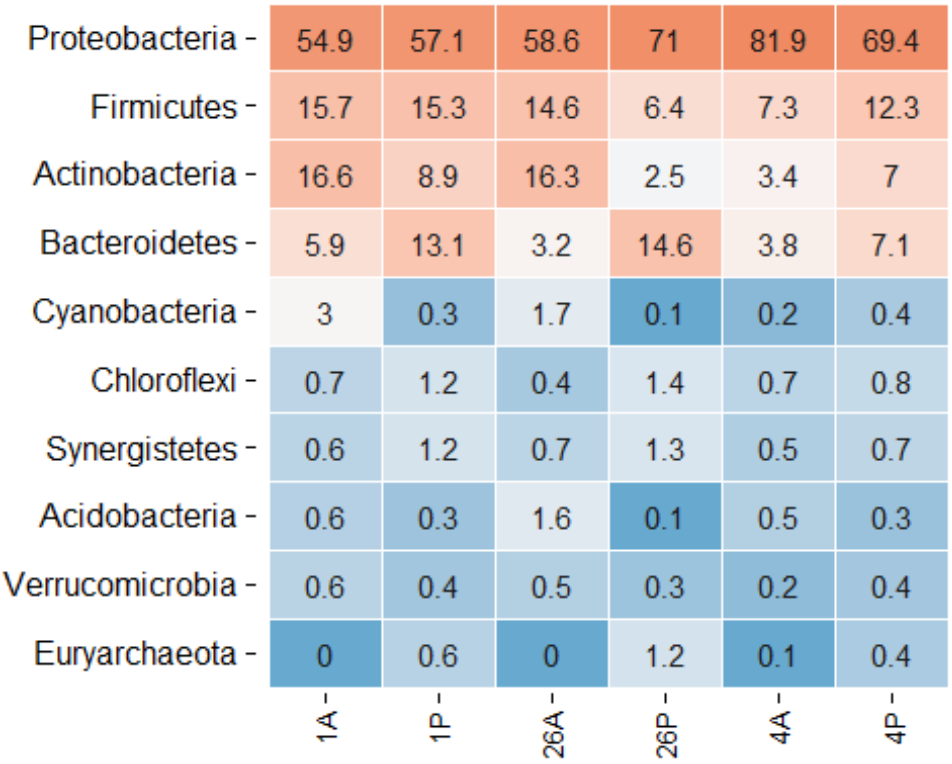
```
myotutable<-data.frame(otu_table(myBiom))
myotutable$OTU<-rownames(myotutable)
mytaxtable<-data.frame(tax_table(myBiom))
mytaxtable$OTU<-row.names(mytaxtable)
mytaxtable$Rank1<-NULL
myotutable<-merge(myotutable,mytaxtable,by="OTU")

mymetadata<-data.frame(sample_data(myBiom))
mymetadata<-data.frame(Samples=rownames(mymetadata),Description=mymetadata
$Description,Status=mymetadata$Status,Week=mymetadata$Week)
ampvis_data <- amp_load(otutable = myotutable,
              metadata = mymetadata)
```

```
amp_rarecurve(ampvis_data,color="Description")
```

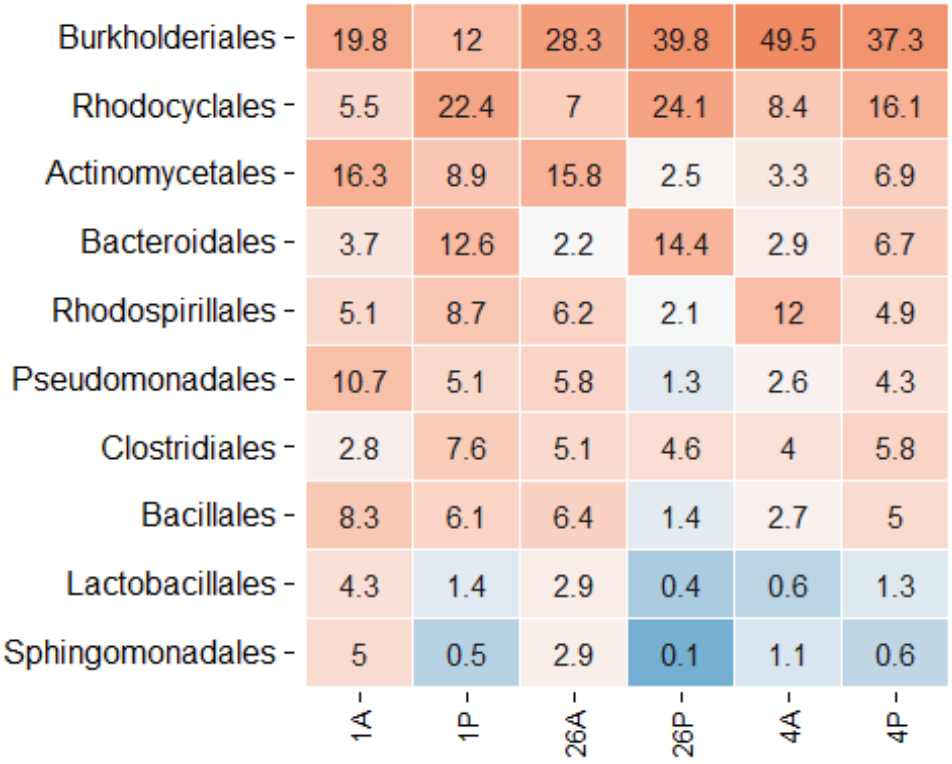


Heatmaps and box plots

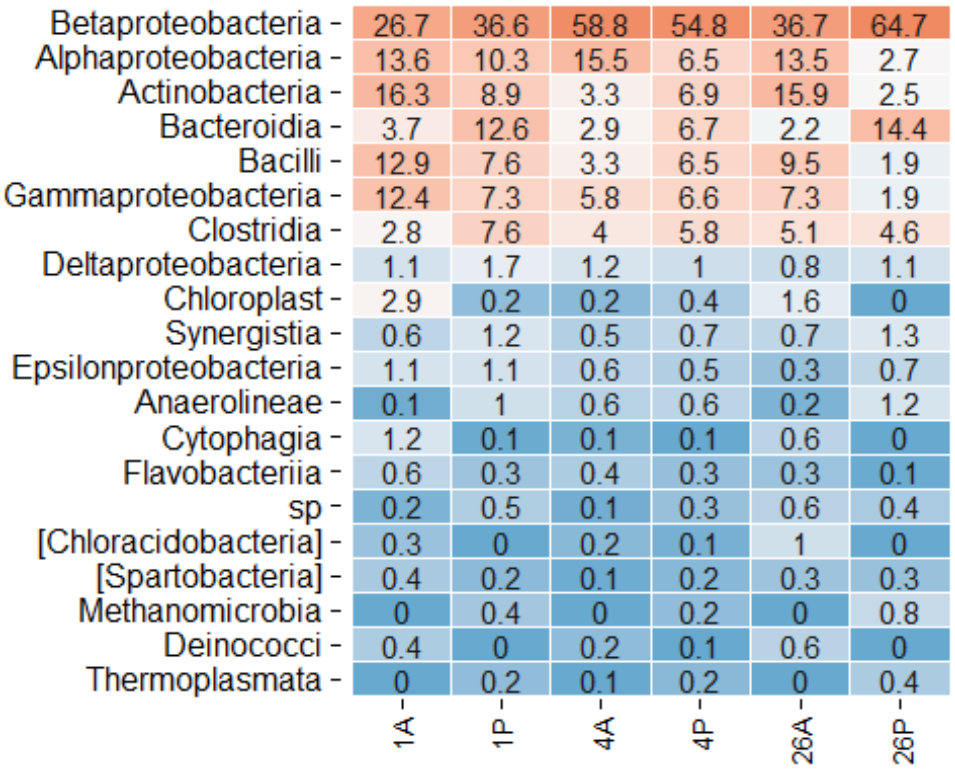
```
amp_heatmap(ampvis_data,group_by="Description")
```



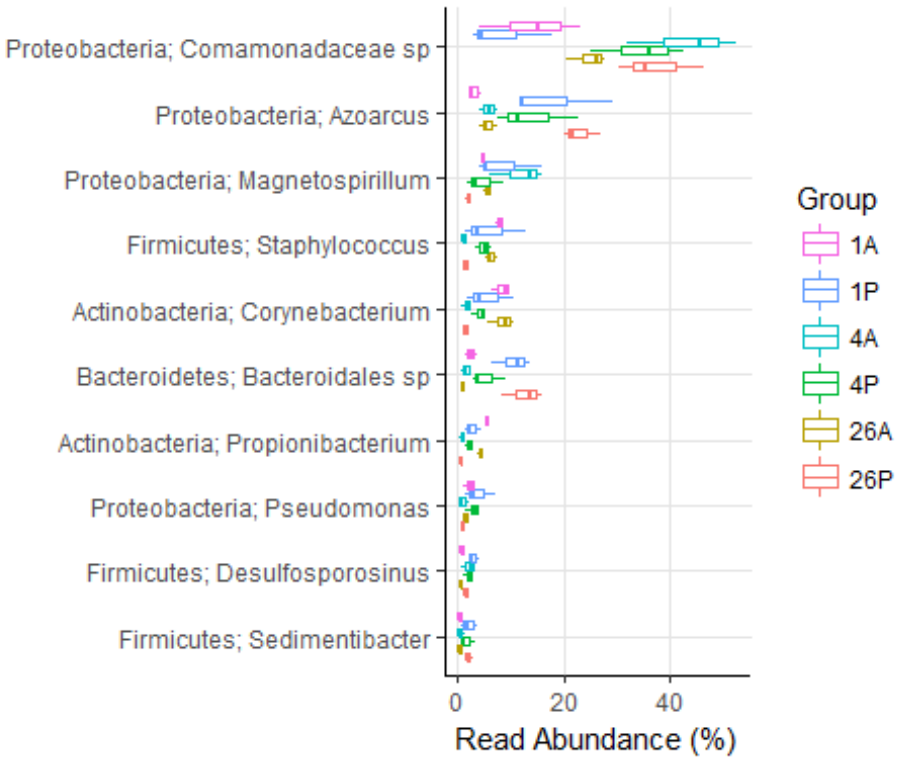
```
amp_heatmap(ampvis_data,group_by="Description",tax_aggregate = "Order")
```




```
amp_heatmap(ampvis_data,group_by="Description",tax_aggregate = "Class",tax
_show = 20,order_x_by = c("1A","1P","4A","4P","26A","26P"))
```




```
amp_boxplot(ampvis_data,group_by="Description",tax_show=10,tax_add="Phylum
",order_group = c("1A","1P","4A","4P","26A","26P"))
```

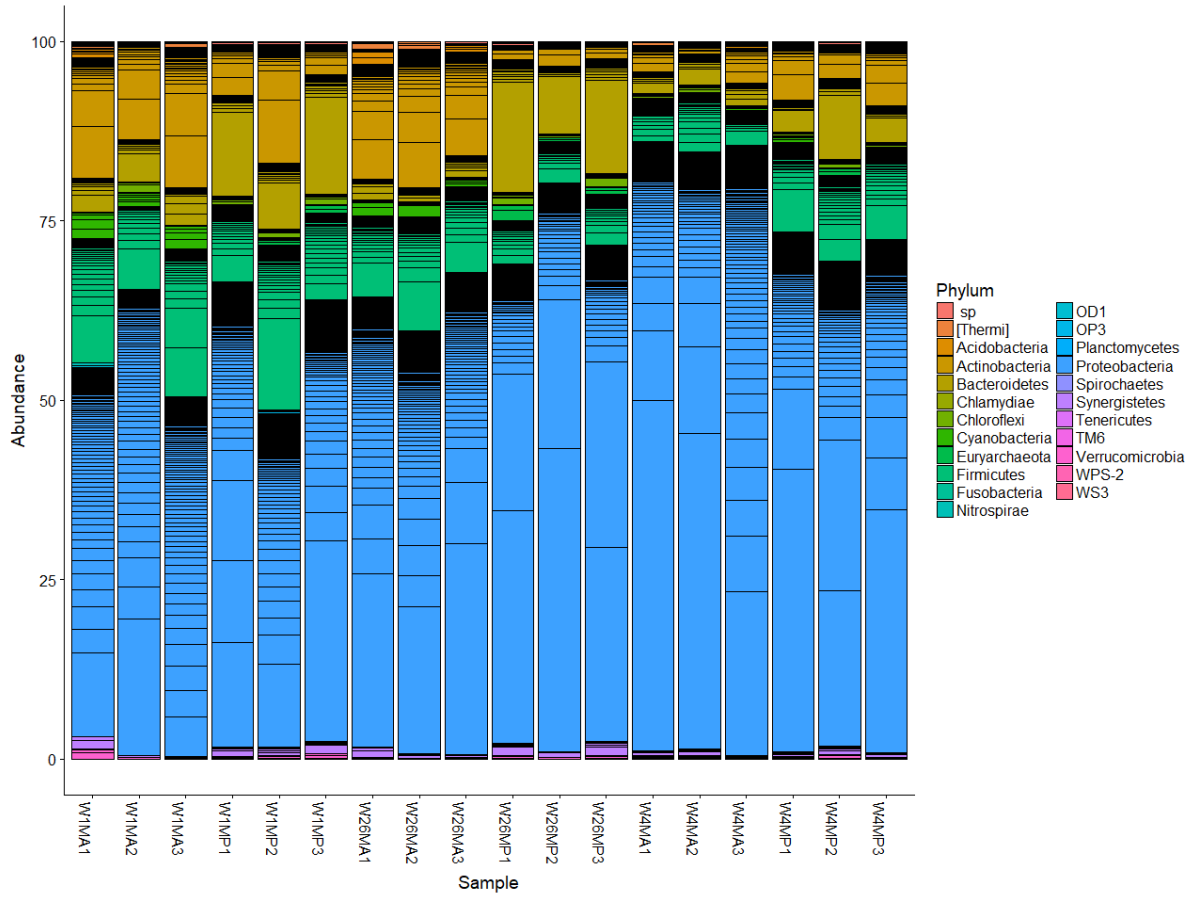


We can see that 20k good counts would capture most of the data

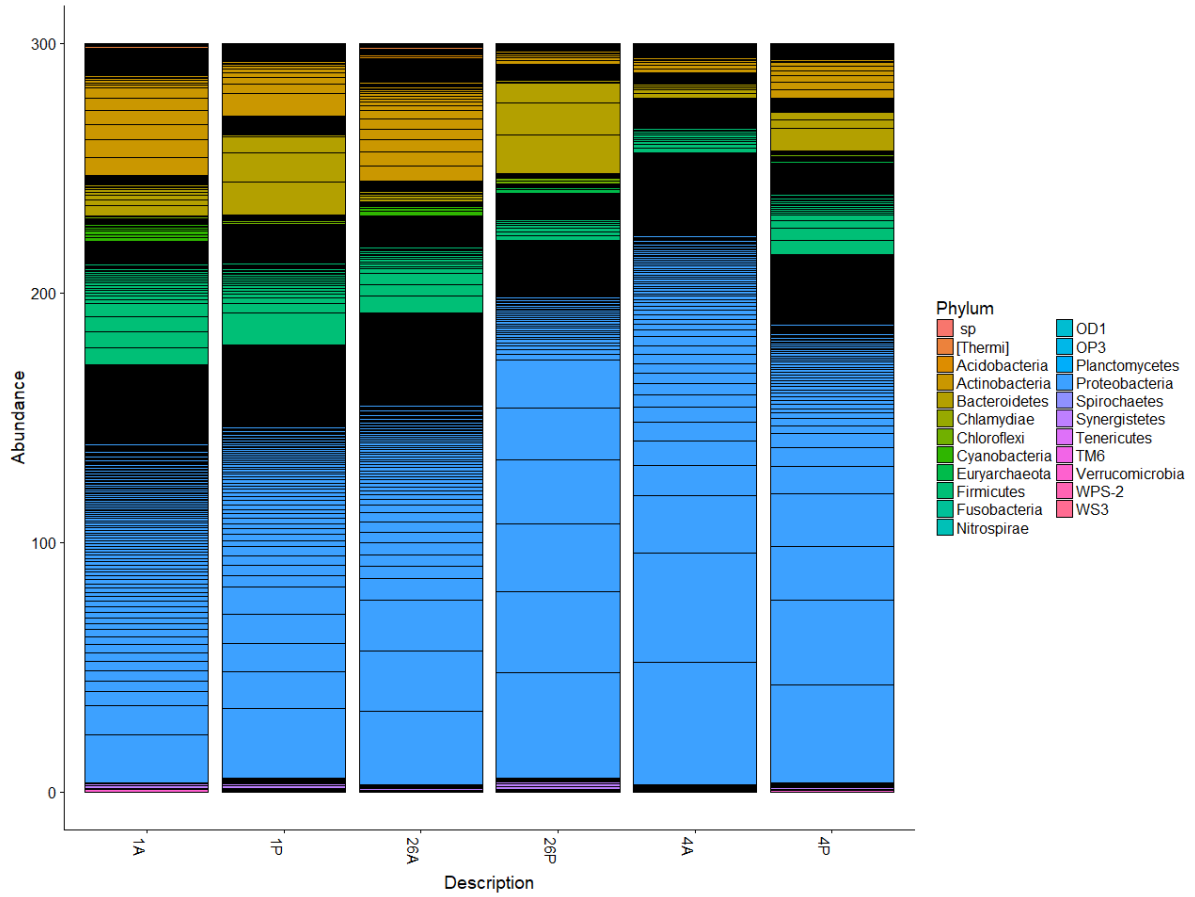
4.6 Displaying OTUs in phyloseq

Display the normalised OTUs

```
#Plot all of the samples
plot_bar(mynormBiom,fill="Phylum")
```



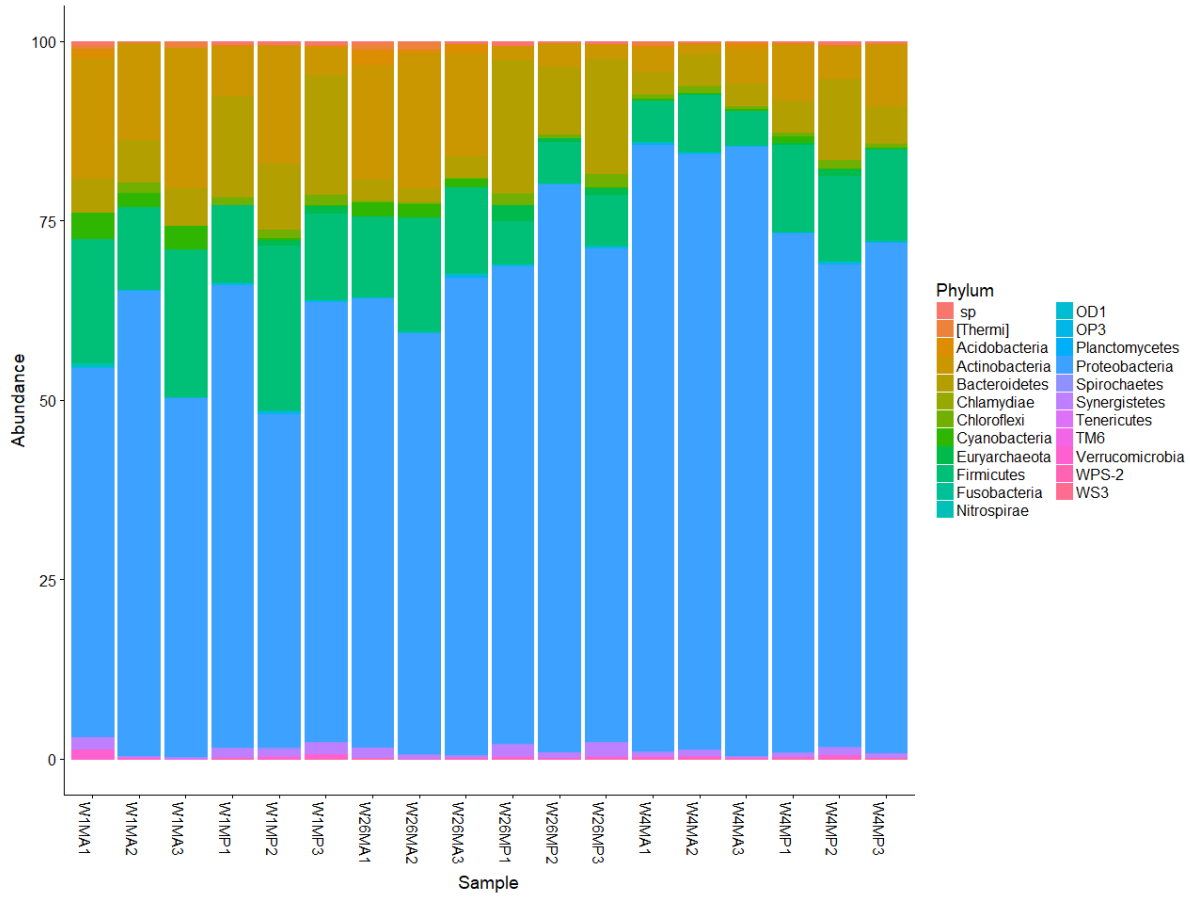

```
#Group them by Description
plot_bar(mynormBiom,x="Description",fill="Phylum")
```



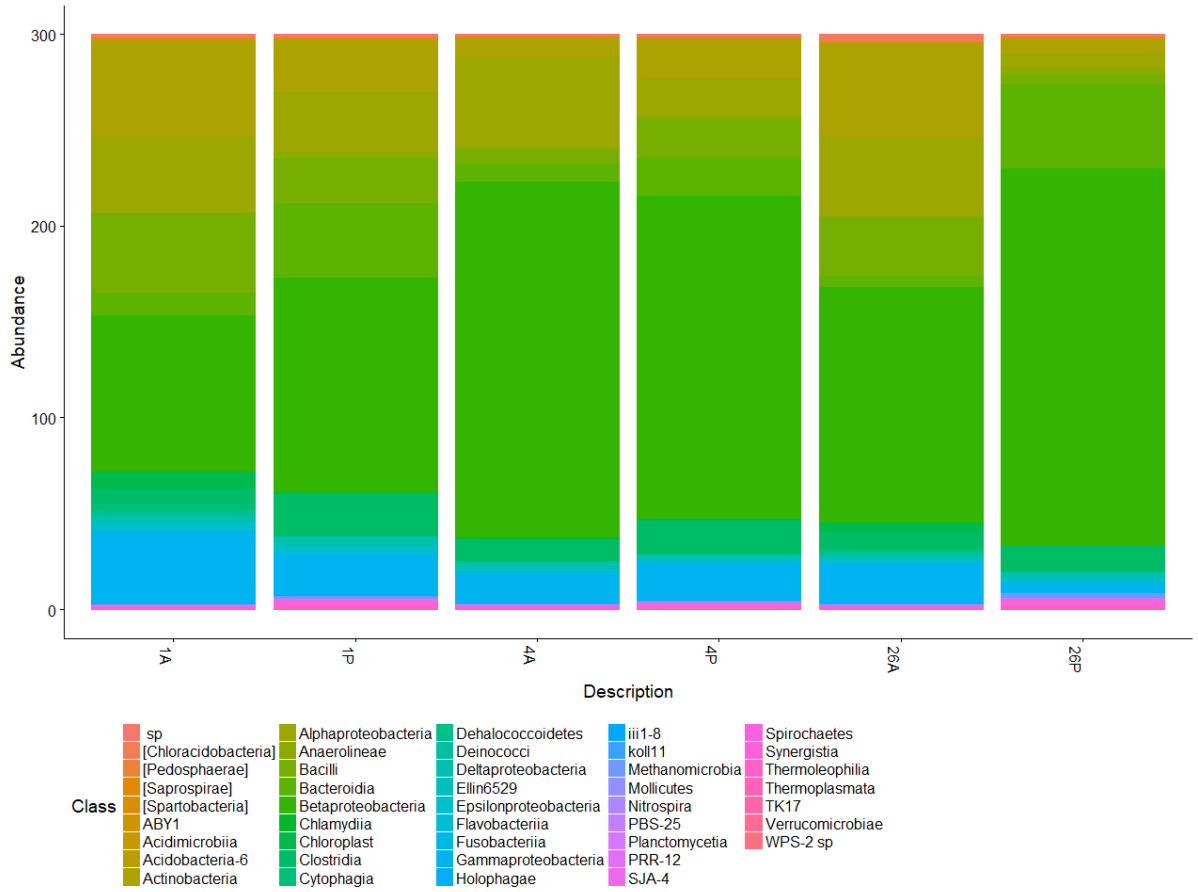
We can see that there are a number of problems with this plot. Even at the Phylum level, there are so many OTUs that they merge together. The samples are in the wrong order.

phyloseq produces a ggplot object (Google it) so we can modify the plot as much as we want. R uses factors to group things together of a similar type (e.g. samples) By controlling the factors and their order, you control the way things are plotted and analysed.

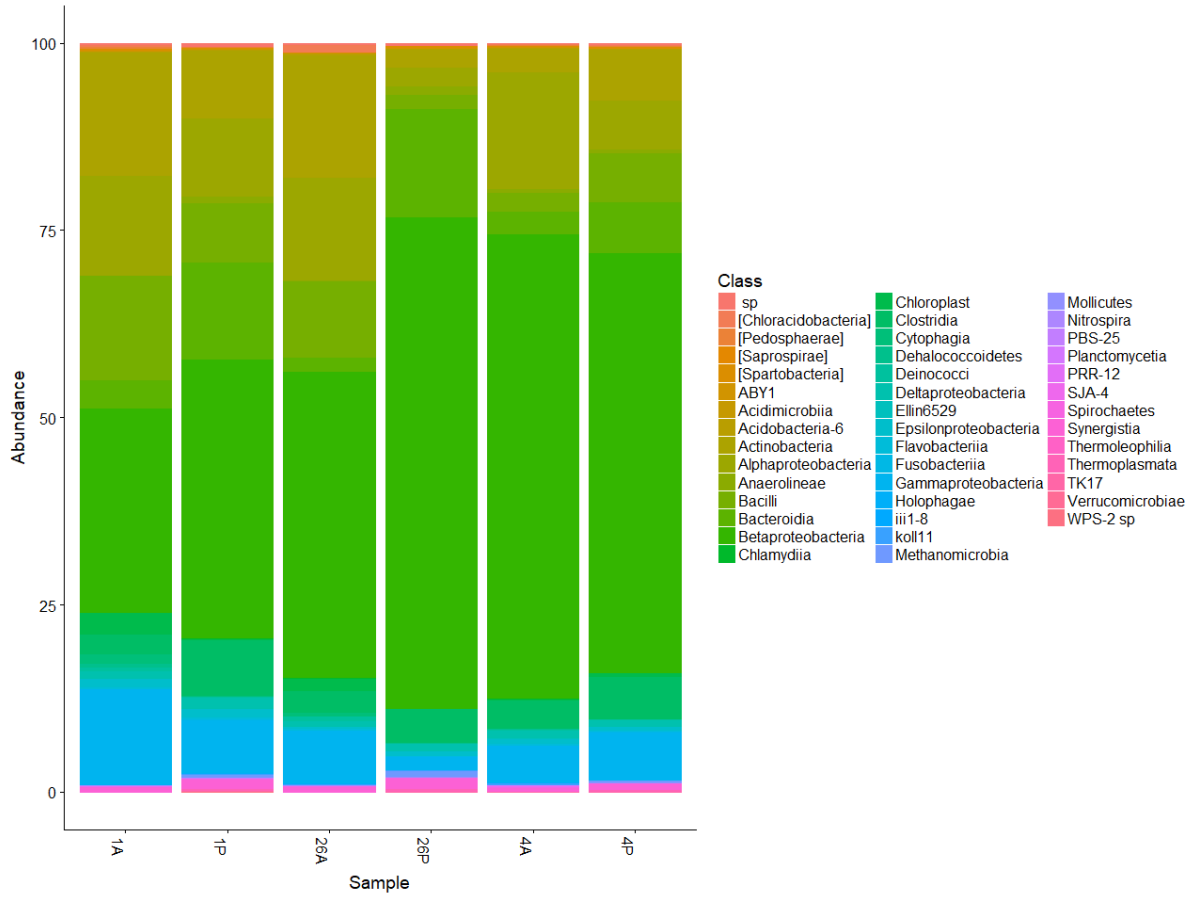
```
#fix the order
s<-data.frame(sample_data(mynormBiom))
s$Status<-factor(s$Status,levels = c("A","P"))
s$Week<-factor(s$Week,levels=c("1","4","26"))
s$Description<-factor(s$Description,levels=c("1A","1P","4A","4P","26A","26
P"))
sample_data(mynormBiom)<-s

#might as well fix it for the other biom files as well
sample_data(myEvenBiom)<-s
sample_data(myFiltBiom)<-s

#Plot all of the samples
p_phylum<-plot_bar(mynormBiom,fill="Phylum")
p_phylum<-p_phylum+ geom_bar(aes(color=Phylum, fill=Phylum), stat="identit
y", position="stack")
p_phylum
```




```
p_class<-plot_bar(mynormBiom,fill="Class",x="Description")
p_class<-p_class+ geom_bar(aes(color=Class, fill=Class), stat="identity",
position="stack")+theme(legend.position = "bottom")
p_class
```

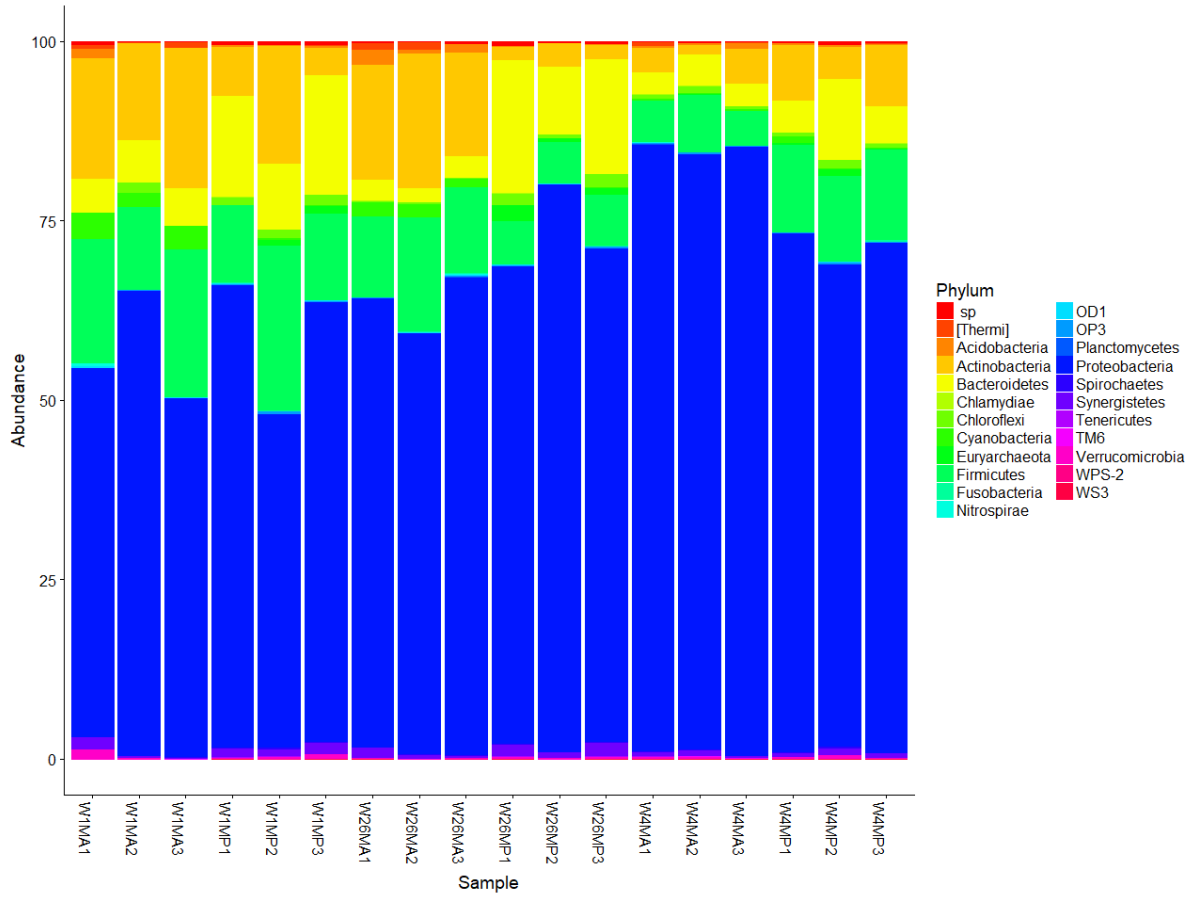



```
#The samples have been added together which is a nuisance
#Merge the samples first and then plot them
myMergeBiom<-merge_samples(mynormBiom,group = "Description")

## Warning in asMethod(object): NAs introduced by coercion

## Warning in asMethod(object): NAs introduced by coercion

otus<-otu_table(myMergeBiom)/3
otu_table(myMergeBiom)<-otus
p_class<-plot_bar(myMergeBiom,fill="Class")
p_class<-p_class+ geom_bar(aes(color=Class, fill=Class), stat="identity",
position="stack")
p_class
```



The colors aren't that great so we can control them. Google R and colors to see how else you might deal with this.

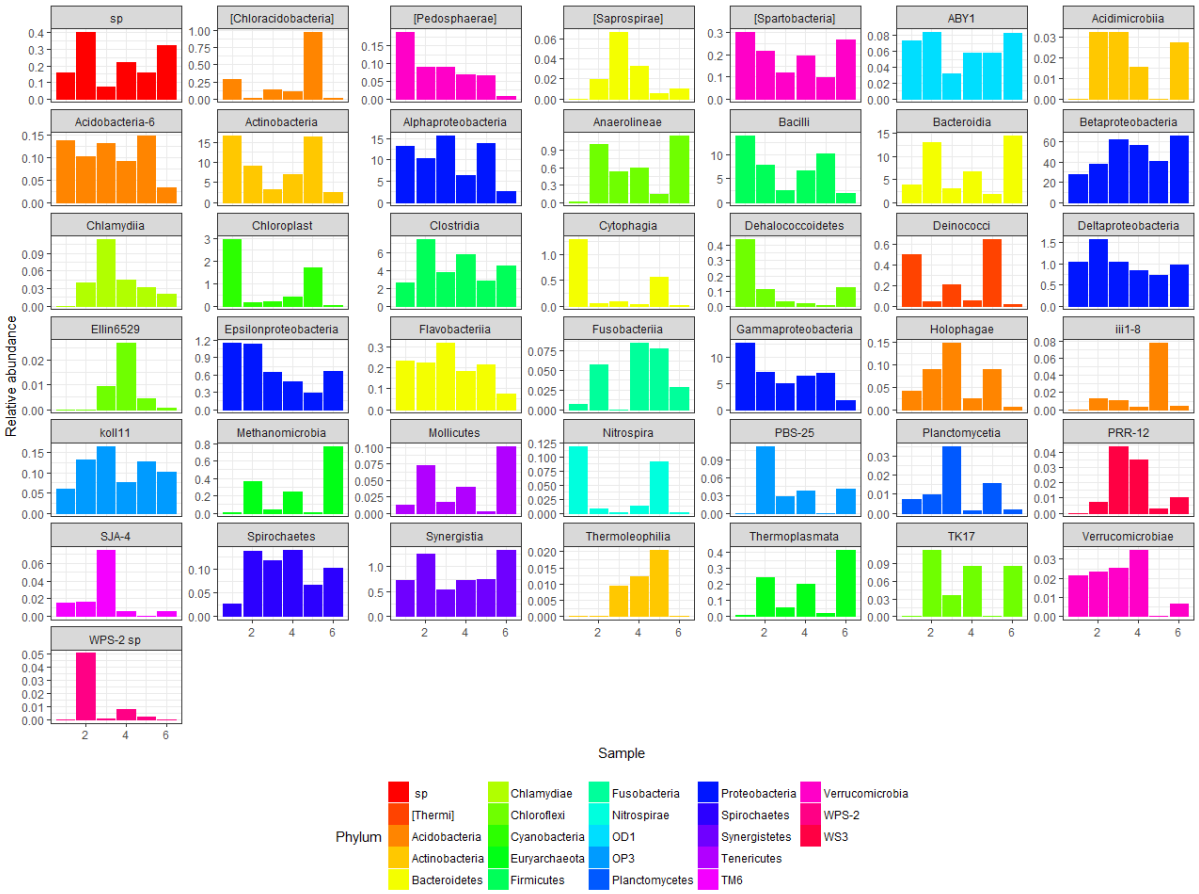
```
#How many colors do we need?
n<-length(levels(p_phylum$data$Phylum))
phylum_colors <- rainbow(n, s = 1, v = 1, start = 0, end = max(1, n - 1)/n
, alpha = 1)
p_phylum<-p_phylum+scale_fill_manual(values=phylum_colors)+scale_color_man
ual(values=phylum_colors)
p_phylum
```



4.7 Facets and plotting data by different meta data

Facets allow us to separate things out

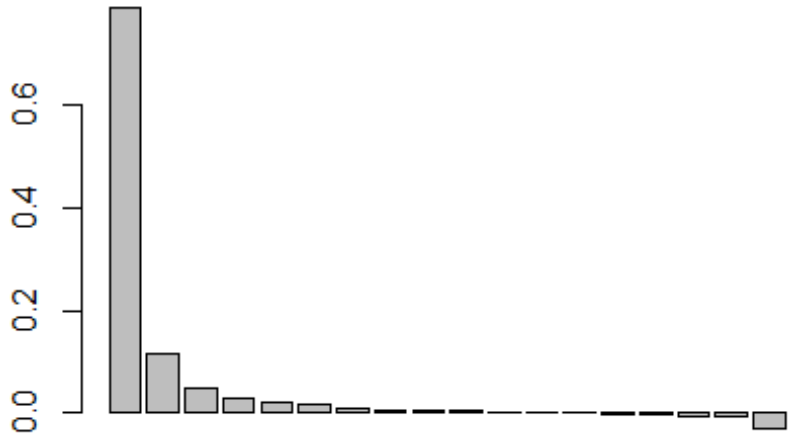
```
p_pretty<-plot_bar(myMergeBiom,fill="Phylum",x="Description")+
  geom_bar(aes(color=Phylum, fill=Phylum), stat="identity", position="stac
k")+
  scale_fill_manual(values=phylum_colors)+scale_color_manual(values=phylum
_colors)+
  theme_bw()+
  xlab("Sample")+ylab("Relative abundance")+
  facet_wrap(~Class,scales="free_y")+
  theme(legend.position = "bottom")
p_pretty
```



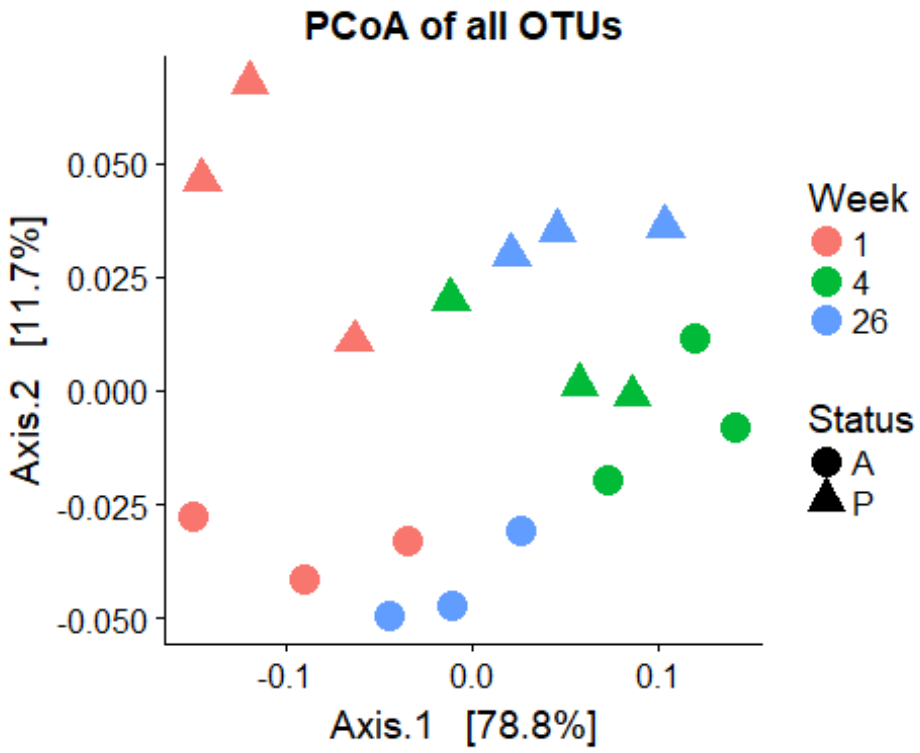
4.8 Ordinations

Now let's do some oridinations to see how the samples compare

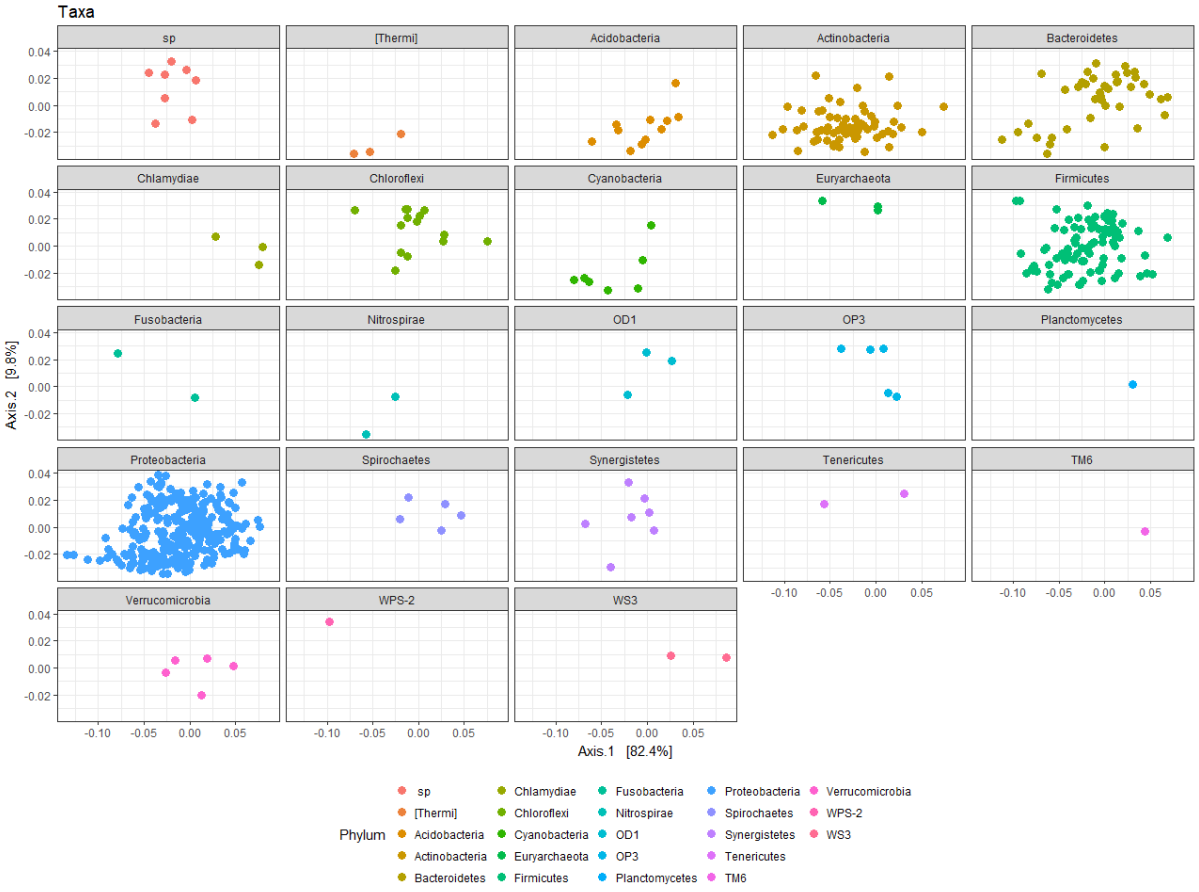
```
#Ordinate using principal component analysis, using the weighted Unifrac a
pproach
myEvenBiom.ord <- ordinate(myEvenBiom, "PCoA", "Unifrac",weighted=TRUE)

## Warning in matrix(tree$edge[order(tree$edge[, 1]), ][, 2], byrow = TRUE
, :
## data length [1133] is not a sub-multiple or multiple of the number of r
ows
## [567]

#Look at the important components - 2 seem to be enought
barplot(myEvenBiom.ord$values$Relative_eig)
```



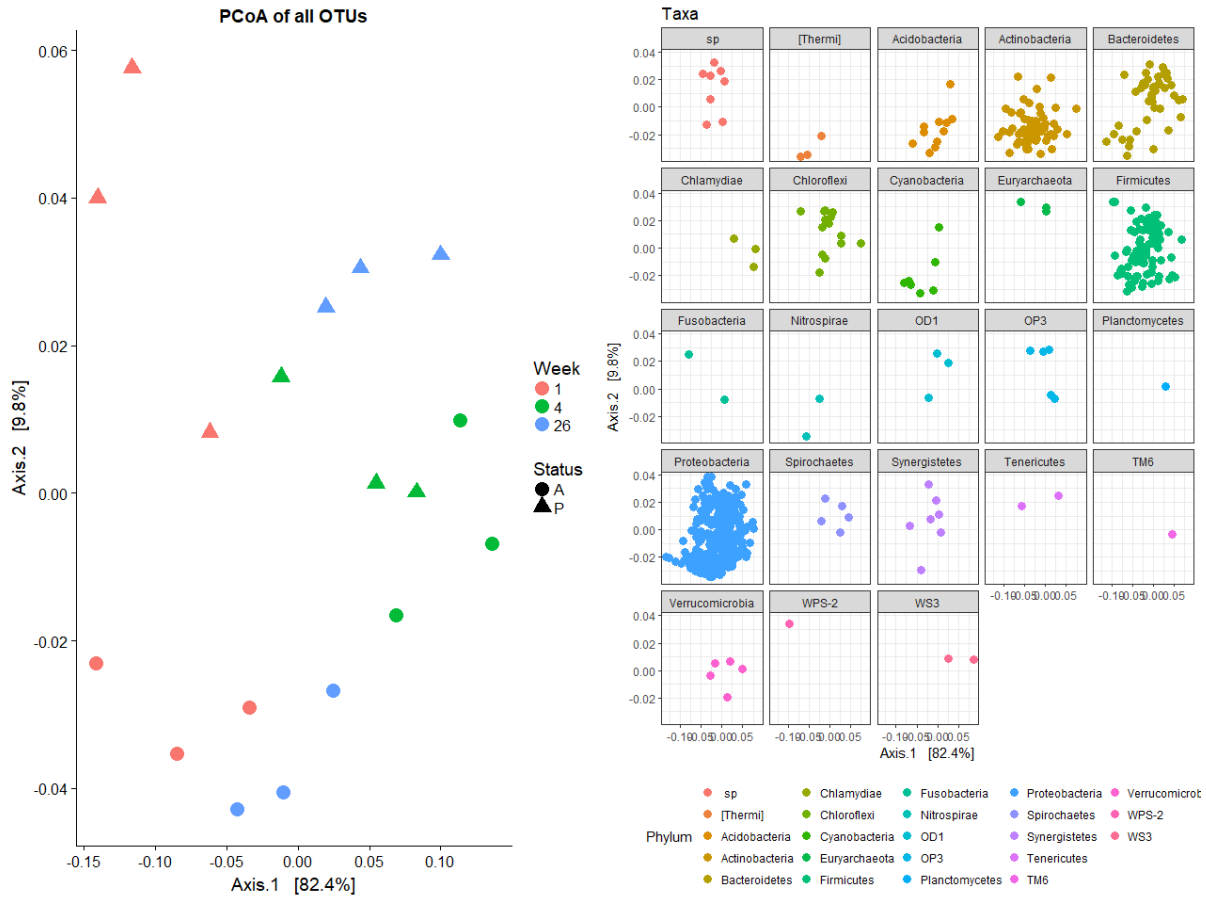

```
#Plot it an colour by week and status
p_uni=plot_ordination(myEvenBiom,myEvenBiom.ord,type="samples",color="Week
",shape="Status",title="PCoA of all OTUs")
p_uni<-p_uni+geom_point(size=5)
p_uni
```



```
#We can also see the OTU that drive this ordination
p_uni_taxa=plot_ordination(myEvenBiom,myEvenBiom.ord,type="taxa",color="Phylum",title="Taxa")
p_uni_taxa<-p_uni_taxa+geom_point(size=3)+theme_bw()+theme(legend.position = "bottom")
p_uni_taxa<-p_uni_taxa+facet_wrap(~Phylum)
p_uni_taxa
```




```
plot_grid(p_uni,p_uni_taxa)
```



We can see that some samples separate away from each other. Also, these two components explain most of the variation in the data.

There are many different sorts of ordination methods
https://joey711.github.io/phyloseq/plot_ordination-examples.html

4.9 Diversity indices - calculations and statistics

We can look at some of the diversity indices in the samples. These 'within' sample measures are known as alpha diversity measures.

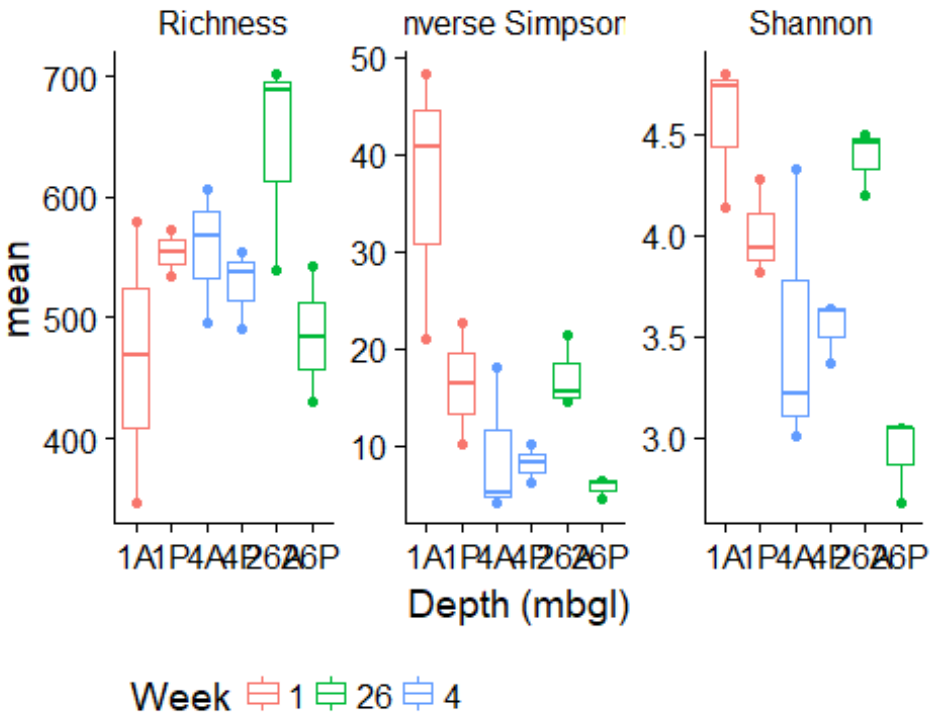
Look at the Richness, Inverse Simpson (Evenness) and Shannon We will use a proper statistical analysis to see what differs. We must use the raw data but for sensible comparisons we need to rarefy to the same depth. To avoid bias we do this a hundred times and take the average.

We use the package dplyr to help handle the data.

```
#minimum sequence count
min_lib<-min(sample_sums(myBiom))
#number of samples
nsamp<-nsamples(myBiom)

trials=100
richness <- matrix(nrow = nsamp, ncol = trials)
row.names(richness) <- sample_names(myBiom)
```

```
evenness <- matrix(nrow = nsamp, ncol = trials)
row.names(evenness) <- sample_names(myBiom)

shannon <- matrix(nrow = nsamp, ncol = trials)
row.names(shannon) <- sample_names(myBiom)

# It is always important to set a seed when you subsample so your result i
s replicable
set.seed(3)
#pb<-txtProgressBar(min=0,max=trials,style=3)
for (i in 1:trials) {
  # Subsample
  r <- rarefy_even_depth(myBiom, sample.size = min_lib, verbose = FALSE, r
eplace = TRUE)

  # Calculate richness
  rich <- as.numeric(as.matrix(estimate_richness(r, measures = "Observed")
))
  richness[ ,i] <- rich

  # Calculate evenness
  even <- as.numeric(as.matrix(estimate_richness(r, measures = "InvSimpson
")))
  evenness[ ,i] <- even

  shan<-as.numeric(as.matrix(estimate_richness(r,measures="Shannon")))
  shannon[,i]<-shan
#  setTxtProgressBar(pb, i)
#  Sys.sleep(0.1)
}
#close(pb)
#These commented out values would display a progress bar
#we know manipulate tghe results to get the average values
SampleID <- row.names(richness)
mean <- apply(richness, 1, mean)
sd <- apply(richness, 1, sd)
measure <- rep("Richness", nsamp)
rich_stats <- data.frame(SampleID, mean, sd, measure)

# Create a new dataframe to hold the means and standard deviations of even
ness estimates
SampleID <- row.names(evenness)
mean <- apply(evenness, 1, mean)
sd <- apply(evenness, 1, sd)
measure <- rep("Inverse Simpson", nsamp)
even_stats <- data.frame(SampleID, mean, sd, measure)

SampleID <- row.names(shannon)
mean <- apply(shannon, 1, mean)
sd <- apply(shannon, 1, sd)
measure <- rep("Shannon", nsamp)
shannon_stats <- data.frame(SampleID, mean, sd, measure)

alpha <- rbind(rich_stats, even_stats,shannon_stats)
```

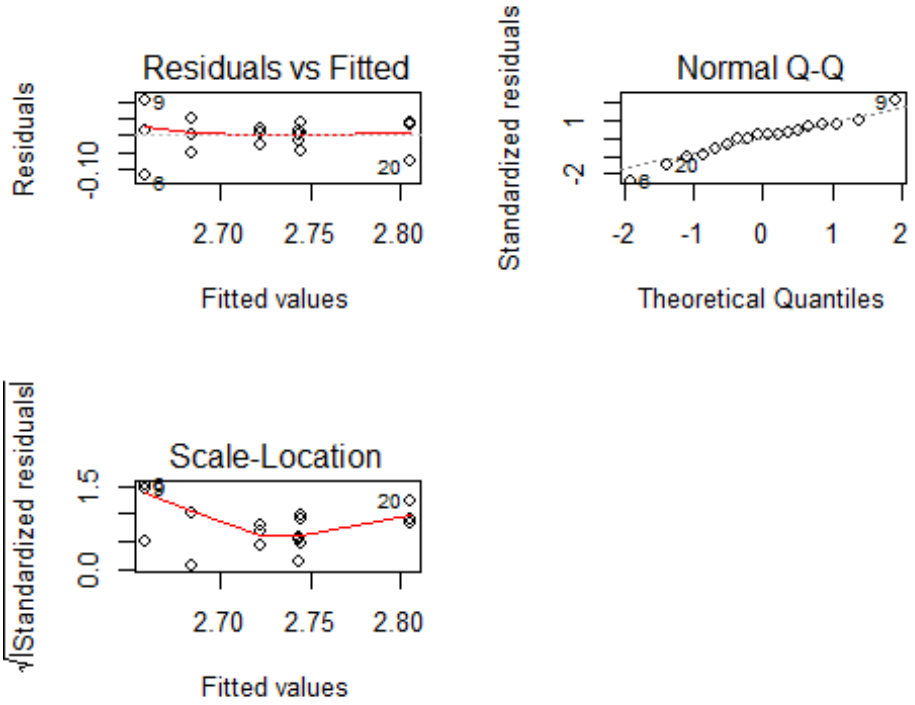
```
rawSamples<-data.frame(sample_data(myBiom))
rawSamples$SampleID<-row.names(rawSamples)
alphadiv <- merge(alpha, rawSamples, by = "SampleID")

#we write the results to a file in case you want to plot them somehow else
write.csv(alphadiv,file="alphadiv.csv")

#get the values
alphaRichness<-alphadiv[which(alphadiv$measure=="Richness"),]
alphaEveness<-alphadiv[which(alphadiv$measure=="Inverse Simpson"),]
alphaShannon<-alphadiv[which(alphadiv$measure=="Shannon"),]

#log them for statistical analysis
alphaRichness$logmean<-log10(alphaRichness$mean)
alphaEveness$logmean<-log10(alphaEveness$mean)
alphaShannon$logmean<-log10(alphaShannon$mean)

#control the order of the plot
alphadiv$Description<-factor(alphadiv$Description,levels=c("1A","1P","4A",
"4P","26A","26P"))

statplot<-ggplot(data=alphadiv,aes(x=(Description),y=mean,color=Week,group
=Description))+geom_point()+facet_wrap(~measure,scales="free_y")+xlab("Dep
th (mbgl)")+geom_boxplot()+theme(legend.position="bottom")+theme(strip.bac
kground=element_rect(fill="white"))
statplot
```



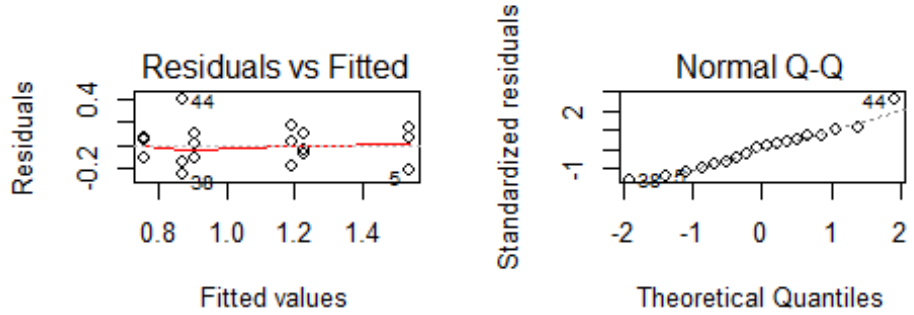

```
#ggsave allows us to save plots
ggsave(plot=statplot,file="measures.png",units="cm",width=14.5,height=14.5
)
```

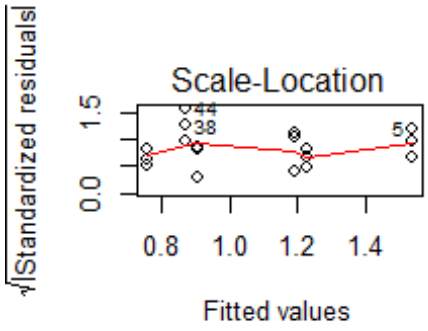
These graphs show that there might be something happening e.g. Values seem to decrease in the planktonic samples over time We can formally test these using ANOVAs

```
fitRichness <- aov(logmean ~ Description, data=alphaRichness)
par(mfrow=c(2,2))
plot(fitRichness)

## hat values (leverages) are all = 0.3333333
##  and there are no factor predictors; no plot no. 5
```

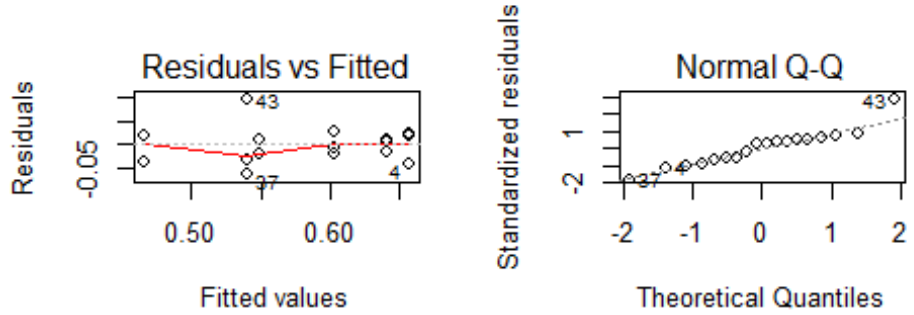


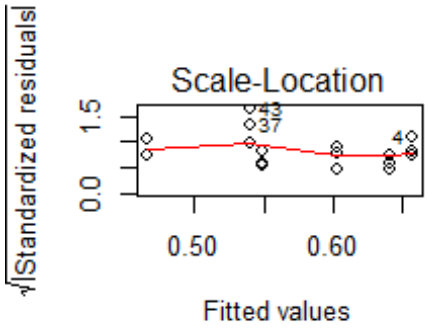

```
summary(fitRichness)

##             Df  Sum Sq  Mean Sq F value Pr(>F)
## Description  5 0.03962 0.007924   2.159  0.128
## Residuals   12 0.04405 0.003671

fitEveness <- aov(logmean ~ Description, data=alphaEveness)
plot(fitEveness)

## hat values (leverages) are all = 0.3333333
##  and there are no factor predictors; no plot no. 5
```






```
summary(fitEveness)

##             Df Sum Sq Mean Sq F value  Pr(>F)
## Description  5 1.2788 0.25575   7.333 0.00231 **
## Residuals   12 0.4185 0.03488
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

fitShannon <- aov(logmean ~ Description, data=alphaShannon)
plot(fitShannon)

## hat values (leverages) are all = 0.3333333
##  and there are no factor predictors; no plot no. 5
```






```
summary(fitShannon)
```

```
##             Df  Sum Sq  Mean Sq F value  Pr(>F)
## Description  5 0.07756 0.015513   8.741 0.00108 **
## Residuals   12 0.02130 0.001775
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We have an effect of sample type on Eveness and Shannon. We can use aTukey test to see what's different

```
TukeyHSD(fitEveness)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = logmean ~ Description, data = alphaEveness)
##
## $Description
##                 diff        lwr         upr     p adj
## 1P-1A   -0.34491696 -0.8571079  0.16727400 0.2799235
## 26A-1A  -0.31042990 -0.8226209  0.20176107 0.3781473
## 26P-1A  -0.78328994 -1.2954809 -0.27109898 0.0025970
## 4A-1A   -0.66966251 -1.1818535 -0.15747154 0.0087865
## 4P-1A   -0.63496711 -1.1471581 -0.12277614 0.0128615
## 26A-1P   0.03448707 -0.4777039  0.54667803 0.9998944
## 26P-1P  -0.43837298 -0.9505639  0.07381799 0.1108668
## 4A-1P   -0.32474554 -0.8369365  0.18744542 0.3349287
## 4P-1P   -0.29005015 -0.8022411  0.22214082 0.4451801
## 26P-26A -0.47286005 -0.9850510  0.03933092 0.0767513
## 4A-26A  -0.35923261 -0.8714236  0.15295836 0.2451493
```

```
## 4P-26A  -0.32453721 -0.8367282  0.18765375 0.3355332
## 4A-26P   0.11362744 -0.3985635  0.62581840 0.9717990
## 4P-26P   0.14832283 -0.3638681  0.66051380 0.9181535
## 4P-4A    0.03469540 -0.4774956  0.54688637 0.9998912

TukeyHSD(fitShannon)

##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = logmean ~ Description, data = alphaShannon)
##
## $Description
##                 diff         lwr          upr     p adj
## 1P-1A   -0.054714506 -0.17024977  0.060820759 0.6187892
## 26A-1A  -0.016072175 -0.13160744  0.099463090 0.9965037
## 26P-1A  -0.192077829 -0.30761309 -0.076542564 0.0012841
## 4A-1A   -0.116962399 -0.23249766 -0.001427134 0.0466407
## 4P-1A   -0.108508122 -0.22404339  0.007027143 0.0702539
## 26A-1P   0.038642331 -0.07689293  0.154177596 0.8626628
## 26P-1P  -0.137363323 -0.25289859 -0.021828058 0.0171422
## 4A-1P   -0.062247894 -0.17778316  0.053287372 0.4947680
## 4P-1P   -0.053793617 -0.16932888  0.061741649 0.6341810
## 26P-26A -0.176005654 -0.29154092 -0.060470389 0.0026806
## 4A-26A  -0.100890224 -0.21642549  0.014645041 0.1009289
## 4P-26A  -0.092435947 -0.20797121  0.023099318 0.1491164
## 4A-26P   0.075115430 -0.04041984  0.190650695 0.3115675
## 4P-26P   0.083569707 -0.03196556  0.199104972 0.2202717
## 4P-4A    0.008454277 -0.10708099  0.123989542 0.9998411
```

We can get a bit more sophisticated with interaction terms

```
fitEveness<-aov(logmean ~Status*Week,data=alphaEveness)
summary(fitEveness)

##             Df Sum Sq Mean Sq F value  Pr(>F)
## Status       1 0.3066  0.3066   8.791 0.01181 *
## Week         2 0.7631  0.3816  10.940 0.00197 **
## Status:Week  2 0.2090  0.1045   2.997 0.08798 .
## Residuals   12 0.4185  0.0349
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

fitShannon<-aov(logmean ~Status*Week,data=alphaShannon)
summary(fitShannon)

##             Df  Sum Sq  Mean Sq F value  Pr(>F)
## Status       1 0.02470 0.024701  13.919 0.00287 **
## Week         2 0.02650 0.013250   7.466 0.00782 **
## Status:Week  2 0.02636 0.013182   7.428 0.00796 **
## Residuals   12 0.02130 0.001775
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

TukeyHSD(fitEveness)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = logmean ~ Status * Week, data = alphaEveness)
##
## $Status
##           diff        lwr         upr    p adj
## P-A -0.2610272 -0.4528461 -0.06920829 0.011812
##
## $Week
##            diff        lwr         upr     p adj
## 26-1 -0.3744014 -0.6620623 -0.08674058 0.0118725
## 4-1  -0.4798563 -0.7675172 -0.19219547 0.0021120
## 4-26 -0.1054549 -0.3931157  0.18220597 0.6038522
##
## $`Status:Week`
##                   diff        lwr         upr     p adj
## P:1-A:1   -0.34491696 -0.8571079  0.16727400 0.2799235
## A:26-A:1  -0.31042990 -0.8226209  0.20176107 0.3781473
## P:26-A:1  -0.78328994 -1.2954809 -0.27109898 0.0025970
## A:4-A:1   -0.66966251 -1.1818535 -0.15747154 0.0087865
## P:4-A:1   -0.63496711 -1.1471581 -0.12277614 0.0128615
## A:26-P:1   0.03448707 -0.4777039  0.54667803 0.9998944
## P:26-P:1  -0.43837298 -0.9505639  0.07381799 0.1108668
## A:4-P:1   -0.32474554 -0.8369365  0.18744542 0.3349287
## P:4-P:1   -0.29005015 -0.8022411  0.22214082 0.4451801
## P:26-A:26 -0.47286005 -0.9850510  0.03933092 0.0767513
## A:4-A:26  -0.35923261 -0.8714236  0.15295836 0.2451493
## P:4-A:26  -0.32453721 -0.8367282  0.18765375 0.3355332
## A:4-P:26   0.11362744 -0.3985635  0.62581840 0.9717990
## P:4-P:26   0.14832283 -0.3638681  0.66051380 0.9181535
## P:4-A:4    0.03469540 -0.4774956  0.54688637 0.9998912
```

```
TukeyHSD(fitShannon)
```

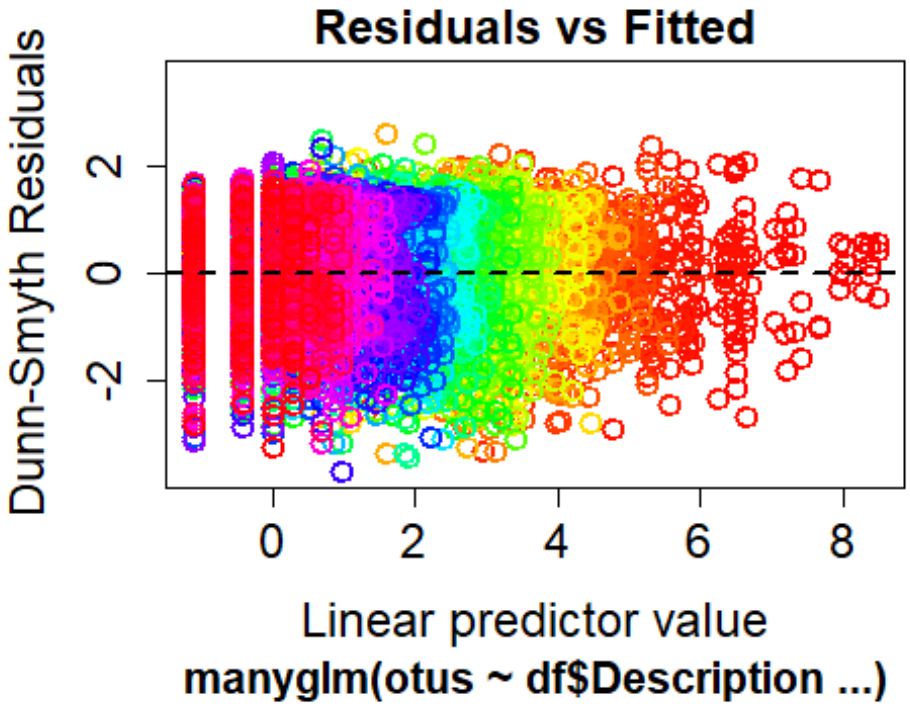
```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = logmean ~ Status * Week, data = alphaShannon)
##
## $Status
##            diff        lwr        upr     p adj
## P-A -0.07408863 -0.1173574 -0.0308199 0.0028699
##
## $Week
##              diff         lwr         upr     p adj
## 26-1 -0.076717749 -0.14160561 -0.01182989 0.0210247
## 4-1  -0.085378008 -0.15026586 -0.02049015 0.0110900
## 4-26 -0.008660259 -0.07354811  0.05622760 0.9328844
##
## $`Status:Week`
##                    diff         lwr          upr     p adj
## P:1-A:1   -0.054714506 -0.17024977  0.060820759 0.6187892
## A:26-A:1  -0.016072175 -0.13160744  0.099463090 0.9965037
```

```
## P:26-A:1  -0.192077829 -0.30761309 -0.076542564 0.0012841
## A:4-A:1   -0.116962399 -0.23249766 -0.001427134 0.0466407
## P:4-A:1   -0.108508122 -0.22404339  0.007027143 0.0702539
## A:26-P:1   0.038642331 -0.07689293  0.154177596 0.8626628
## P:26-P:1  -0.137363323 -0.25289859 -0.021828058 0.0171422
## A:4-P:1   -0.062247894 -0.17778316  0.053287372 0.4947680
## P:4-P:1   -0.053793617 -0.16932888  0.061741649 0.6341810
## P:26-A:26 -0.176005654 -0.29154092 -0.060470389 0.0026806
## A:4-A:26  -0.100890224 -0.21642549  0.014645041 0.1009289
## P:4-A:26  -0.092435947 -0.20797121  0.023099318 0.1491164
## A:4-P:26   0.075115430 -0.04041984  0.190650695 0.3115675
## P:4-P:26   0.083569707 -0.03196556  0.199104972 0.2202717
## P:4-A:4    0.008454277 -0.10708099  0.123989542 0.9998411
```

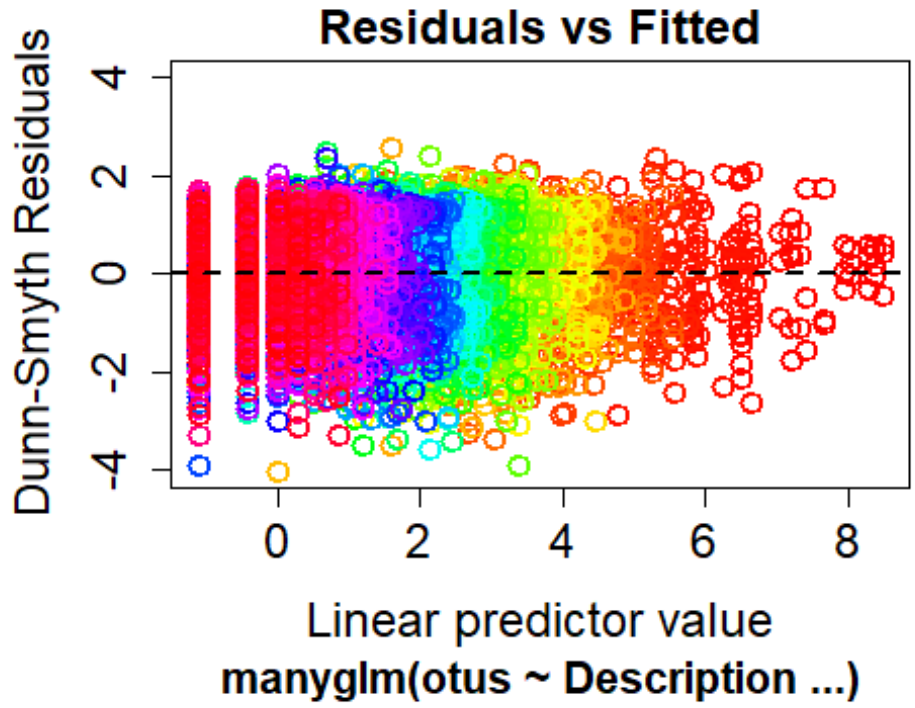
4.10 Testing which samples differ - mvabund

Are the samples different?

```
#Get the data from the even Biom
df<-data.frame(sample_data(myEvenBiom))

#get the otu_table
o<-as.data.frame(otu_table(myEvenBiom))
#transpose it
o<-t(o)


#convert to a mvabund object
otus<-mvabund(o)

#use a GLM by Description to see if any of the samples differ
#Inspection showed that the OTUs followed a negative binomial distribution
as expected
#Description has the inoculum as the first element - so this is the compar
ison made
mod1 <- manyglm(otus ~ df$Description, family="negative_bionomial")

plot(mod1)
```




```
#This takes a long time to run!
myBoot=99
#set this to 999 for a publication

#Create a list of the levels we want to test

df$Description<-factor(df$Description)
#The first factor in the list is what everything else is compared against
levels(df$Description)

## [1] "1A"  "1P"  "4A"  "4P"  "26A" "26P"

#We can change this with relevel
#To see what differs from 1A
mod <- manyglm(otus ~ Description,data = df, family = 'negative.binomial',
test="LR")
plot(mod)
```




```
summary(mod,nBoot=myBoot,test="LR")

##
## Test statistics:
##                 LR value Pr(>LR)
## (Intercept)        10309    0.01 **
## Description1P       2493    0.01 **
## Description4A       2173    0.01 **
## Description4P       2181    0.01 **
## Description26A      1419    0.01 **
## Description26P      2985    0.01 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Likelihood Ratio statistic:   7240, p-value: 0.01
## Arguments:
##  Test statistics calculated assuming response assumed to be uncorrelate
d
##  P-value calculated using 99 resampling iterations via pit.trap resampl
ing (to account for correlation in testing).

#To see what differs from 4A

df$Description<-relevel(df$Description,ref="4A")
mod <- manyglm(otus ~ Description,data = df, family = 'negative.binomial',
test="LR")
summary(mod,nBoot=myBoot,test="LR")

##
## Test statistics:
##                 LR value Pr(>LR)
```

```
## (Intercept)        8222    0.01 **
## Description1A      2173    0.01 **
## Description1P      1705    0.01 **
## Description4P      1396    0.01 **
## Description26A     1753    0.01 **
## Description26P     1746    0.01 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Likelihood Ratio statistic:   7240, p-value: 0.01
## Arguments:
##  Test statistics calculated assuming response assumed to be uncorrelate
d
##  P-value calculated using 99 resampling iterations via pit.trap resampl
ing (to account for correlation in testing).

#We can also use interaction terms
#remember that comparisons are made with the first factor
levels(df$Status)

## [1] "A" "P"

levels(df$Week)

## [1] "1"  "4"  "26"

mod <- manyglm(otus ~ Week*Status,data = df, family = 'negative.binomial',
test="LR")
summary(mod,nBoot=myBoot,test="LR")

##
## Test statistics:
##                LR value Pr(>LR)
## (Intercept)       10309    0.01 **
## Week4              2173    0.01 **
## Week26             1419    0.01 **
## StatusP            2493    0.01 **
## Week4:StatusP      1759    0.01 **
## Week26:StatusP     1372    0.01 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Likelihood Ratio statistic:   7240, p-value: 0.01
## Arguments:
##  Test statistics calculated assuming response assumed to be uncorrelate
d
##  P-value calculated using 99 resampling iterations via pit.trap resampl
ing (to account for correlation in testing).

#so these are differences between week1 Attached and everything else
```

4.11 Testing which OTUs differ – DESEQ2

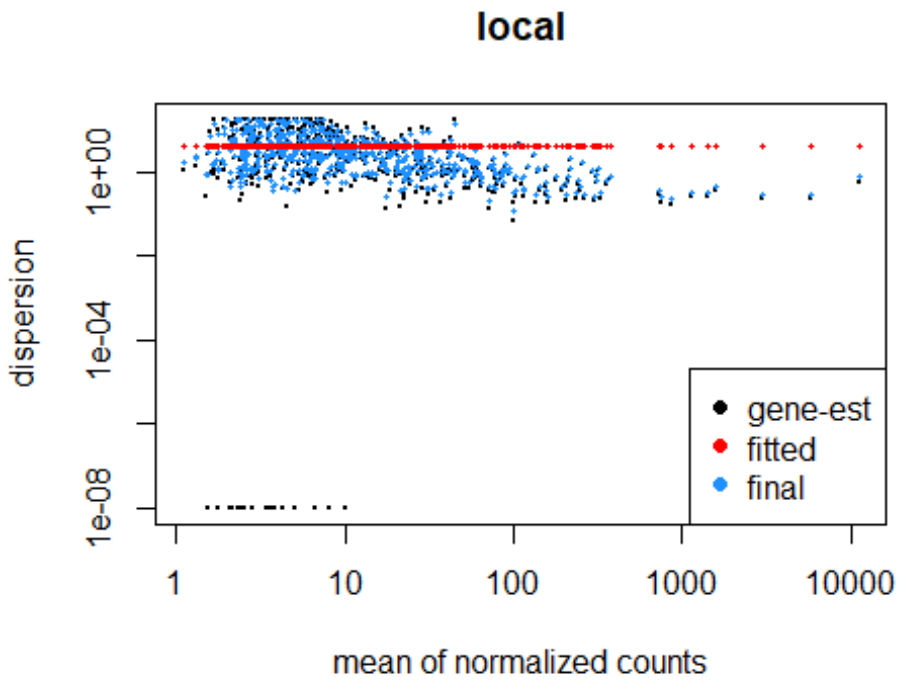
Model choice

Now we use DESeq2 to test for differential OTUs. DESEQ2 fits a negative bionomial distribution to the data. This assumption needs to be checked. There are 3 options for the fit - mean, parametric and local

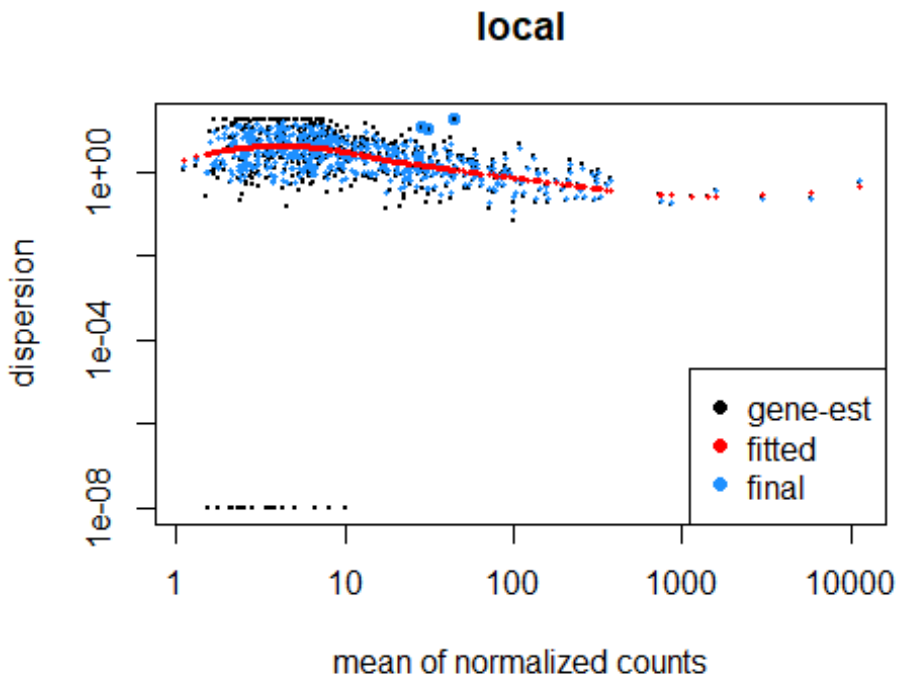
The default (and preferred) method is to use parametric but the local fit may be better (you need to check!)

The fits are done on the filtered data. It's important to remove the noise form the analyses and these OTUs will never be identified as significant anyway.

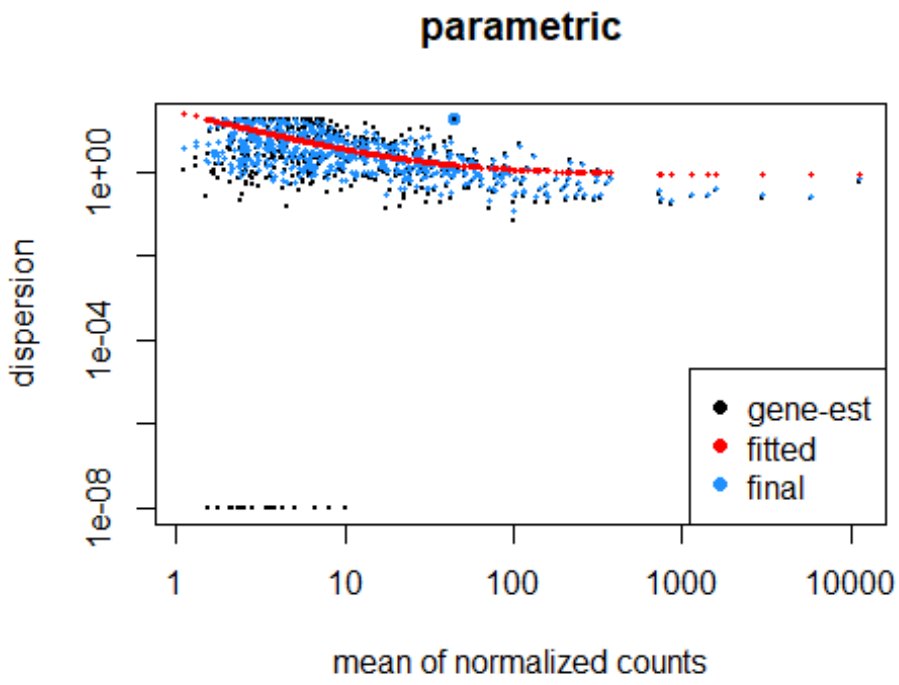
```
diagdds=phyloseq_to_deseq2(myFiltBiom, ~ Description)

## converting counts to integer mode

#we can do local or parametric fits
diagdds_m=DESeq(diagdds,test="Wald",fitType = "mean")

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

plotDispEsts(diagdds_m,main="local")
```




```
diagdds_l=DESeq(diagdds,test="Wald",fitType = "local")
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
plotDispEsts(diagdds_l,main="local")
```




```
diagdds_p=DESeq(diagdds,test="Wald",fitType = "parametric")
## estimating size factors
## estimating dispersions
## gene-wise dispersion estimates
## mean-dispersion relationship
## final dispersion estimates
## fitting model and testing
plotDispEsts(diagdds_p,main="parametric")
```




```
#The parametric fit is fine

#do some comparisons
#Get the results table
res1AP<-results(diagdds_p,contrast=c("Description","1A","1P"))
res4AP<-results(diagdds_p,contrast=c("Description","4A","4P"))
res26AP<-results(diagdds_p,contrast=c("Description","26A","26P"))

#Filter out the significant changes (either up or down)

alpha=0.05
sigtab1AP=res1AP[which(res1AP$padj <= alpha),]
sigtab4AP=res4AP[which(res4AP$padj <= alpha),]
sigtab26AP=res26AP[which(res26AP$padj <= alpha),]

head(sigtab1AP)
```

```
## log2 fold change (MLE): Description 1A vs 1P
## Wald test p-value: Description 1A vs 1P
## DataFrame with 6 rows and 6 columns
##                                baseMean log2FoldChange     lfcSE      st
at
##                               <numeric>      <numeric> <numeric> <numeri
c>
## 4349320                     154.950049       2.793814 0.8436635  3.3115
26
## New.CleanUp.ReferenceOTU750  31.383753      -3.508433 0.8793100 -3.9899
84
## 80896                         7.780459      -7.166346 2.2118629 -3.2399
59
## 735246                        9.125276      -6.121664 2.4073558 -2.5429
```

```
00
## New.CleanUp.ReferenceOTU35   10.386352      -6.217405 2.1248042 -2.9261
07
## 2707                         20.308183      -7.434515 1.7906566 -4.1518
37
##                                   pvalue         padj
##                                 <numeric>    <numeric>
## 4349320                      9.278858e-04 0.0080297811
## New.CleanUp.ReferenceOTU750 6.607763e-05 0.0008889756
## 80896                        1.195467e-03 0.0101501938
## 735246                       1.099368e-02 0.0448499810
## New.CleanUp.ReferenceOTU35  3.432323e-03 0.0203244462
## 2707                         3.298168e-05 0.0005496946
```

```
head(sigtab4AP)
```

```
## log2 fold change (MLE): Description 4A vs 4P
## Wald test p-value: Description 4A vs 4P
## DataFrame with 6 rows and 6 columns
##                             baseMean log2FoldChange     lfcSE      stat
##                            <numeric>      <numeric> <numeric> <numeric>
## 581069                     14.756077      21.605820  3.500347  6.172480
## 4437014                    28.853324      -4.047068  1.120854 -3.610701
## 4365877                     7.475851      16.564589  3.959348  4.183665
## New.CleanUp.ReferenceOTU1 49.923410      -6.538862  1.945712 -3.360653
## 1941616                    11.077551      -6.116913  1.695111 -3.608561
## 4473295                    12.539384      -7.047256  2.128255 -3.311283
##                                  pvalue         padj
##                                <numeric>    <numeric>
## 581069                     6.722701e-10 1.811768e-07
## 4437014                    3.053710e-04 1.843976e-02
## 4365877                    2.868463e-05 3.092203e-03
## New.CleanUp.ReferenceOTU1 7.775847e-04 3.810165e-02
## 1941616                    3.078996e-04 1.843976e-02
## 4473295                    9.286925e-04 4.120808e-02
```

```
head(sigtab26AP)
```

```
## log2 fold change (MLE): Description 26A vs 26P
## Wald test p-value: Description 26A vs 26P
## DataFrame with 6 rows and 6 columns
##                               baseMean log2FoldChange     lfcSE      st
at
##                              <numeric>      <numeric> <numeric> <numeri
c>
## 4349320                    154.950049       3.704055 0.8713673  4.2508
54
## 1119551                     17.446182      -2.415547 0.6933411 -3.4839
23
## 841724                      30.623907      -2.153943 0.7575784 -2.8431
95
## 205245                       4.313501      -5.585475 2.2980491 -2.4305
29
## 4456104                     40.617879      -2.205401 0.9048413 -2.4373
35
```

```
## New.CleanUp.ReferenceOTU602   9.777483      -6.955918 2.2531862 -3.0871
47
##                                    pvalue         padj
##                                   <numeric>    <numeric>
## 4349320                       2.129569e-05 0.0002389849
## 1119551                       4.941225e-04 0.0033237501
## 841724                        4.466367e-03 0.0206928031
## 205245                        1.507679e-02 0.0491211629
## 4456104                       1.479598e-02 0.0488363912
## New.CleanUp.ReferenceOTU602 2.020874e-03 0.0110928392
```

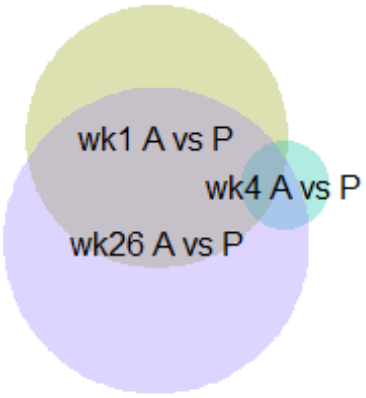
```
#Merge in the taxonmic data

sigtab1AP = cbind(as(sigtab1AP, "data.frame"), as(tax_table(myBiom)[rownam
es(sigtab1AP), ], "matrix"))
sigtab4AP = cbind(as(sigtab4AP, "data.frame"), as(tax_table(myBiom)[rownam
es(sigtab4AP), ], "matrix"))
sigtab26AP = cbind(as(sigtab26AP, "data.frame"), as(tax_table(myBiom)[rown
ames(sigtab26AP), ], "matrix"))

#get the significant OTU names and combine into a unique list
otu1AP<-rownames(sigtab1AP)
otu4AP<-rownames(sigtab4AP)
otu26AP<-rownames(sigtab26AP)


otu_AP<-c(otu1AP,otu4AP,otu26AP)
otu_AP<-unique(otu_AP)

#We have this many OTUs
length(otu_AP)
```

```
## [1] 203
```

Looking at the results

We can display the data in lots of different ways - this is just a taster

```
#Prepare for a Venn diagram
df1<-data.frame(OTU=otu1AP,comp="wk1 A vs P")
df2<-data.frame(OTU=otu4AP,comp="wk4 A vs P")
df3<-data.frame(OTU=otu26AP,comp="wk26 A vs P")


venn_df<-rbind(df1,df2,df3)
plot(venneuler(venn_df))
```



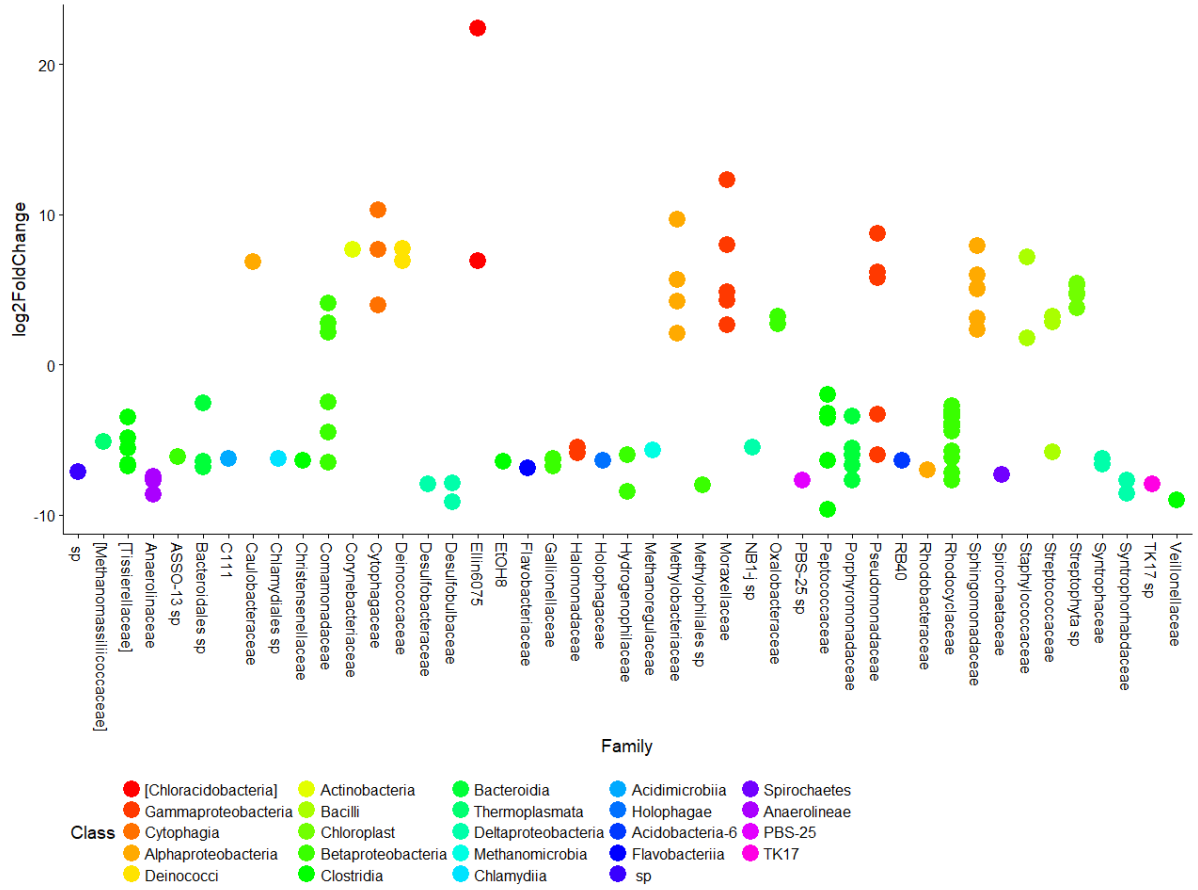
DESEQ2 uses variance stabilised values (essentially it corrects for the increasing noise at low counts) We can use these corrected values in our plots

```
#copy the biom object
vstBiom<-myBiom
#get the variance stabilised counts
diagvst = getVarianceStabilizedData(diagdds_p)
otu_table(vstBiom)<-otu_table(diagvst,taxa_are_rows=TRUE)
#we want to only show significant OTUS
vstTrim=prune_taxa(otu_AP,vstBiom)
```

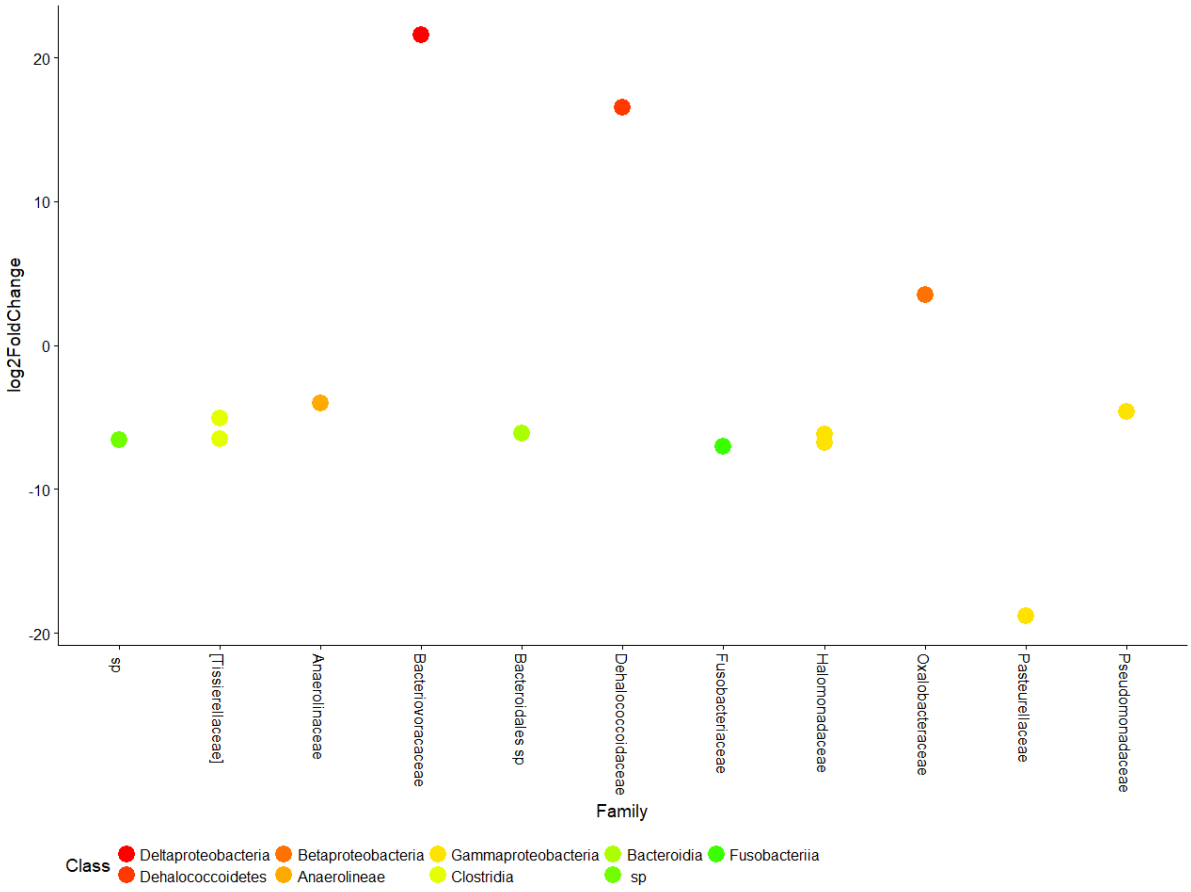
We can plot the fold changes

```
myPlot<-function(sigtab){
  x=tapply(sigtab$log2FoldChange,sigtab$Class,function(x) max(x))
  x=sort(x,TRUE)
  sigtab$Class=factor(as.character(sigtab$Class),levels=names(x))
  x = tapply(sigtab$log2FoldChange, sigtab$Family, function(x) max(x))
  x = sort(x, TRUE)
  g<-ggplot(sigtab, aes(x=Family, y=log2FoldChange, color=Class)) + geom_p
oint(size=6) +   theme(axis.text.x = element_text(angle = -90, hjust = 0,
vjust=0.5))
  return(g)
}

#How many colors do we need?
n<-length(levels(sigtab26AP$Class))
class_colors <- rainbow(n, s = 1, v = 1, start = 0, end = max(1, n - 1)/n,
alpha = 1)

g1AP<-myPlot(sigtab1AP)
```

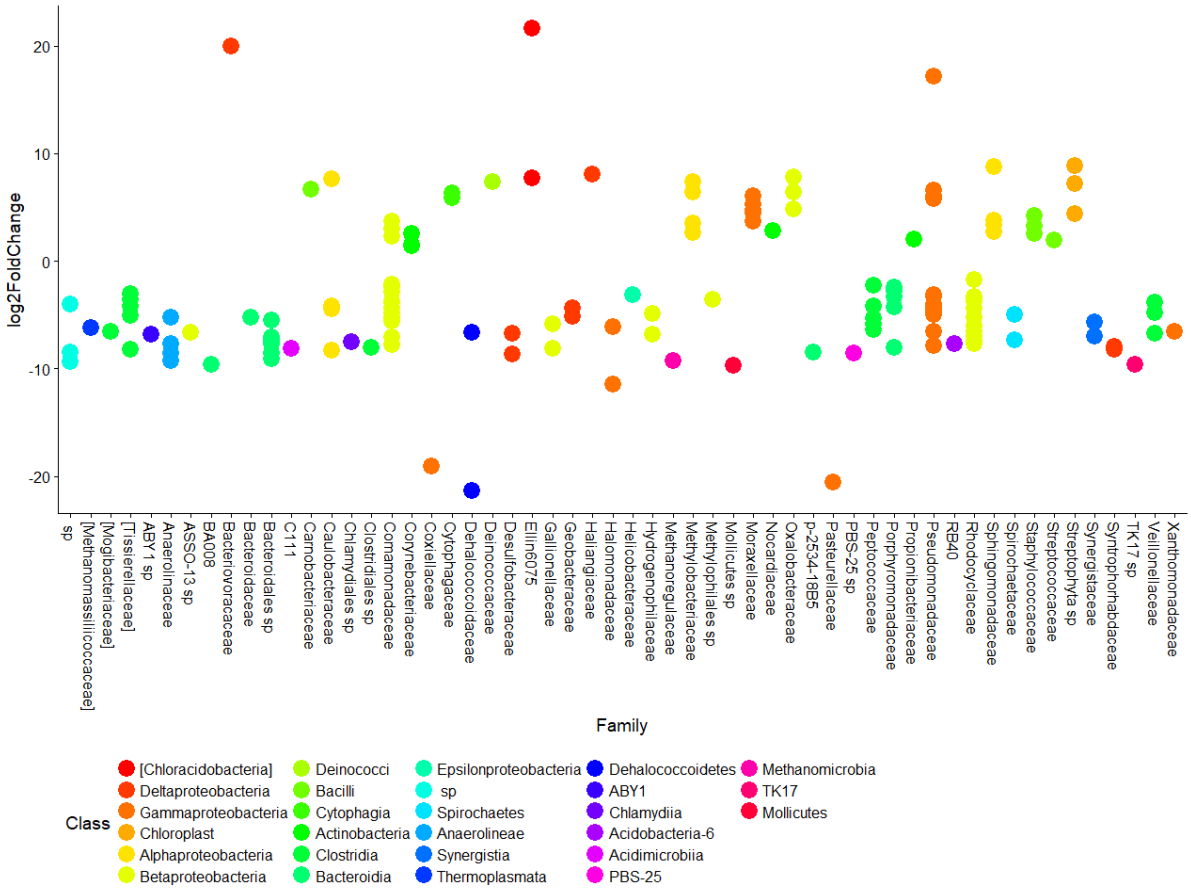
```
g1AP<-g1AP+scale_fill_manual(values=class_colors)+scale_color_manual(value
s=class_colors)+theme(legend.position = "bottom")
g1AP
```




```
g4AP<-myPlot(sigtab4AP)
g4AP<-g4AP+scale_fill_manual(values=class_colors)+scale_color_manual(value
s=class_colors)+theme(legend.position = "bottom")
g4AP
```



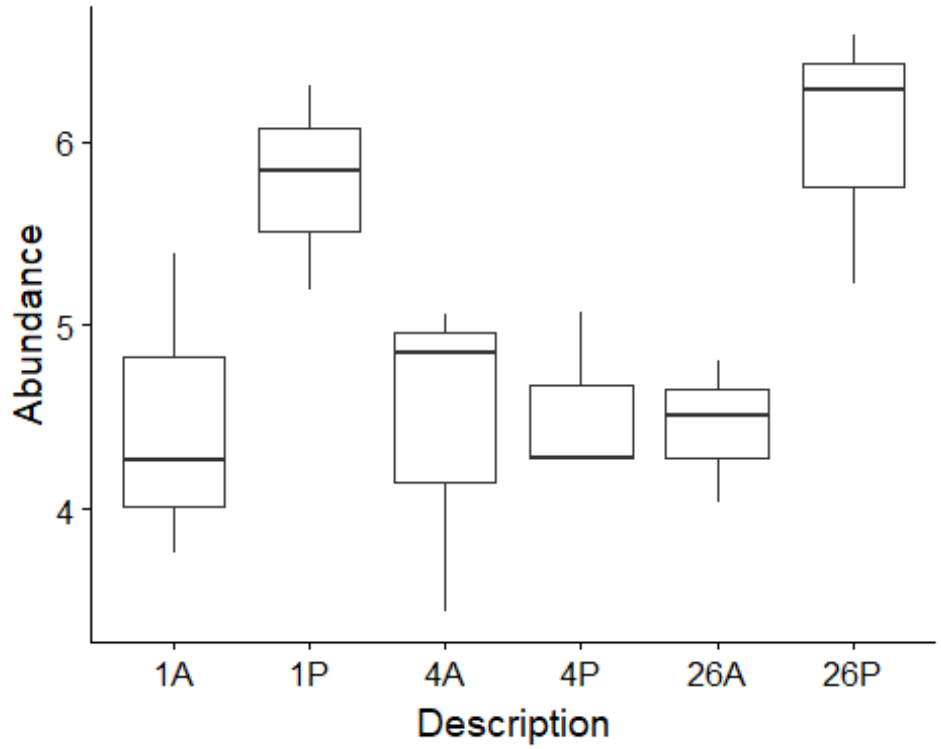

```
g26AP<-myPlot(sigtab26AP)
g26AP<-g26AP+scale_fill_manual(values=class_colors)+scale_color_manual(val
ues=class_colors)+theme(legend.position = "bottom")
g26AP
```



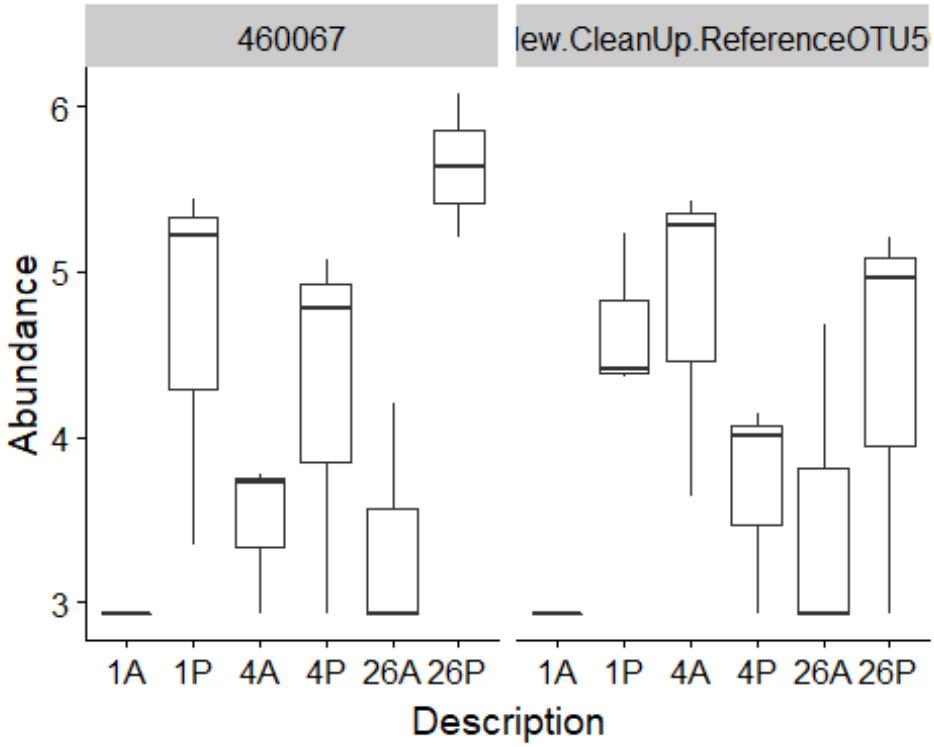
If we want to look at a single OTU then we use the variance stabilised values

```
#get the data
my_data<-data.frame(otu_table(vstTrim))
my_taxa<-data.frame(tax_table(vstTrim))
my_sample_data<-data.frame(sample_data(vstTrim))

my_data$OTU<-rownames(my_data)
my_taxa$OTU<-rownames(my_taxa)
my_sample_data$Sample<-rownames(my_sample_data)

#make the OTU data long
my_plot_data<-melt(my_data,id.vars="OTU")
colnames(my_plot_data)<-c("OTU","Sample","Abundance")

#merge in the taxonomy
my_plot_data<-merge(my_plot_data,my_taxa, by="OTU")
#merge in the sample data
my_plot_data<-merge(my_plot_data,my_sample_data, by="Sample")

my_plot_data$InputFilename<-NULL
my_plot_data$BarcodeSequence<-NULL
my_plot_data$LinkerPrimerSequence<-NULL

my_plot_data$Description<-factor(my_plot_data$Description,levels=c("1A","1
P","4A","4P","26A","26P"))
```

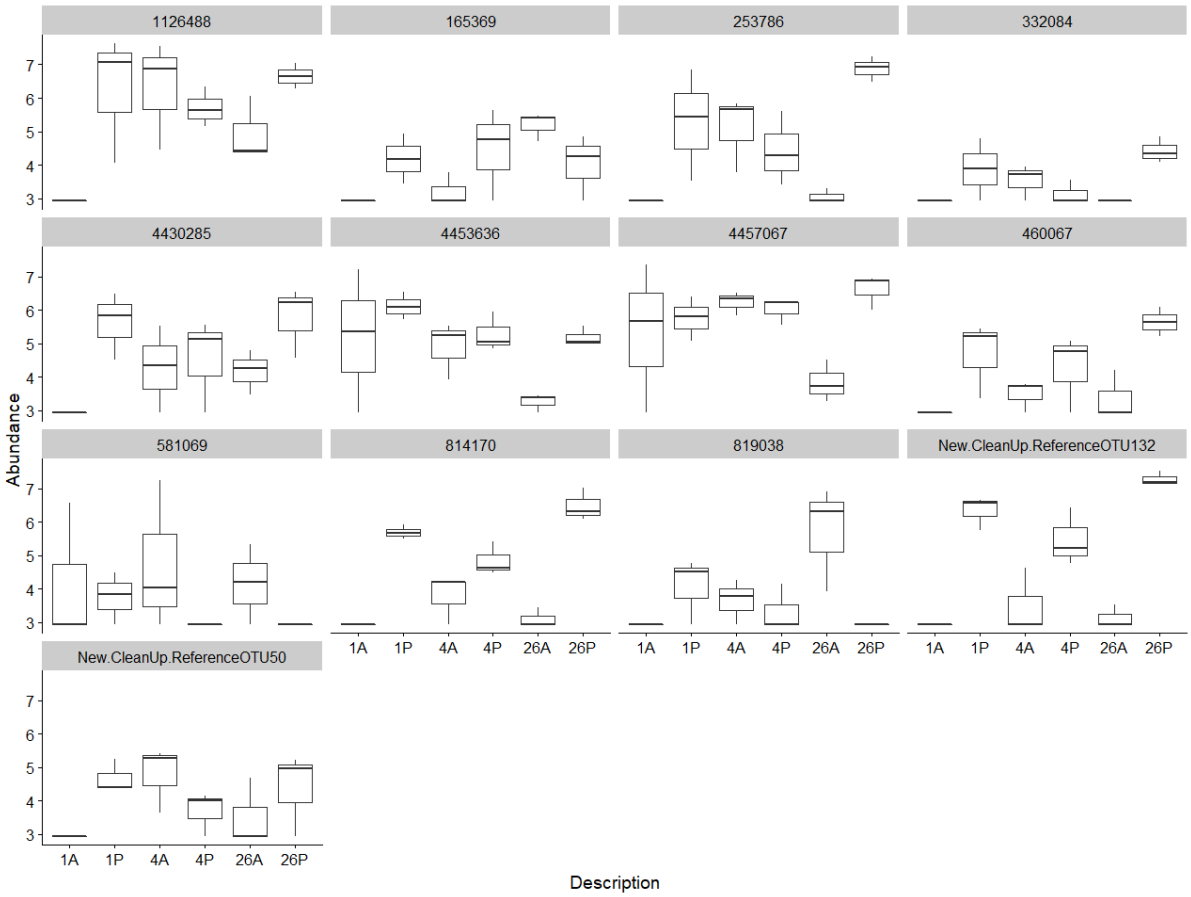
```
#The data are now nicely organised

#Select an OTU to plot
sel_otu<-c("4312424")

pdata<-my_plot_data[which(my_plot_data$OTU==sel_otu),]
ggplot(data=pdata,aes(x=Description,y=Abundance))+geom_boxplot()
```




```
#Show several OTUs

sel_family="Syntrophaceae"
pdata<-my_plot_data[which(my_plot_data$Family==sel_family),]
ggplot(data=pdata,aes(x=Description,y=Abundance))+geom_boxplot()+facet_wra
p(~OTU)
```




```
sel_class="Deltaproteobacteria"
pdata<-my_plot_data[which(my_plot_data$Class==sel_class),]
ggplot(data=pdata,aes(x=Description,y=Abundance))+geom_boxplot()+facet_wra
p(~OTU)
```


1126488
165369
253786
332084
4430285
4453636
4457067
460067
581069
814170
819038
New.CleanUp.ReferenceOTU132
New.CleanUp.ReferenceOTU50
Abundance
Description
1A
1P
4A
4P
26A
26P

5 Appendix 1: Installing software

5.1 Unzipping files with 7-zip

Most compression and decompression of files will be done in Unix but it's useful to do some things in Windows.

The free 7-zip program is useful for this.

http://www.7-zip.org/download.html

5.2 Install Qiime

Download VirtualBox for your system (needs to be on a machine with 8GB or more of memory) with lots of free disc space (50GB)

https://www.virtualbox.org/wiki/Downloads

also download the extension pack

Download the Qiime disc image - do this at work – it's a big file!

http://qiime.org/install/virtual_box.html

Unzip this to a convenient location (e.g. C:\) using a program like 7-Zip (it will take a time!)

Start Oracle VM Box

Select New

Name: Qiime 1.9 (or whatever version)

Type: Linux

Version: Ubuntu 64

Set memory to the maximum (top of green bar)

Use an existing hard disc – select your unzipped Qiime disc image (ends with .vdi)

Now find the extension pack download and double-click it.

On the VMBox settings

General > Advanced > Shared clipboard >Bidirectional

System > processor > CPUs (set to top of green bar – typically 2 on a laptop, 4 on a desktop)

Press the big green Start arrow – Ubuntu should start up

On the Desktop you will see a folder called Shared_Folder. This is empty.

We'll set it up so that it maps onto a folder of the same name in Windows. That way you can easily shift files between systems. You can also use some Windows programs to look at your files.

This next section is tricky and doesn't always seem to work smoothly!

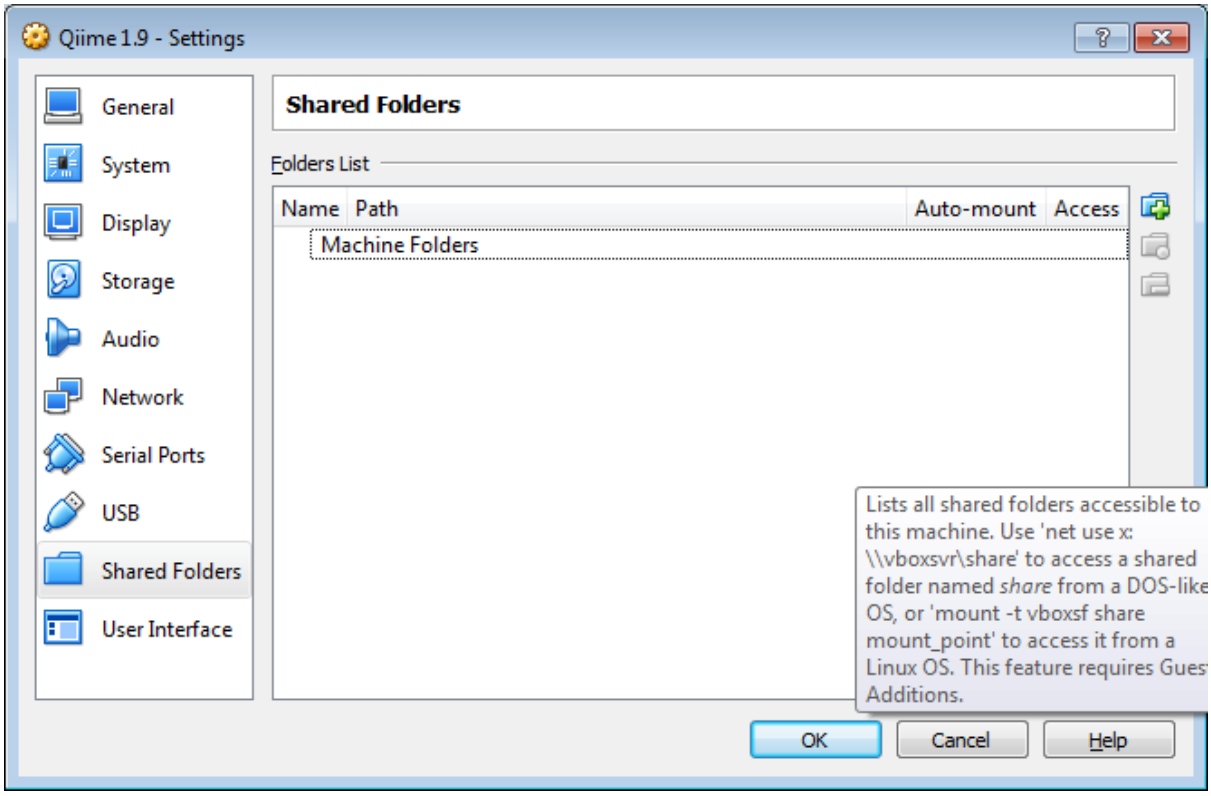
Create a folder on C:\ called Shared_Folder

Create a meaningful subdirectory and put some files in it (useful to see that the share is working).

Close down the Ubuntu virtual machine (use Shutdown – don't save the machine state as the files are huge).

Go back to Settings and Shared Folders

Click the small directory symbol with a green + on it.



From Folder path select your new Windows directory called Shared_Folder (select Other…) and press OK.

The folder name will be set to Shared_Folder

Select Auto-mount (but **not** Read-only!)

Press OK

You know need to tell Linux that the folder on the Desktop is the same as the folder on your C:\ drive. (If you open the Ubuntu Shared_Folder drive you will see that it's empty still).

The keyboard defaults to US English

Click the big red cog wheel on the left of the screen – Keyboard Layout

+ English (UK) Add

Select US English and click -

Now click Devices

Install Guest Additions CD Image…

Select the default

You will need to enter a password – it's qiime

(This is the system 'root' password)

Re-start the system

It is entirely possible that the Shared_Folder on the desktop will now be linked with your files. (Best check).

If it isn't then you need to start a command line and enter the following

sudo mount -t vboxsf -o uid=1000,gid=1000 Shared_Folder ~/Desktop/Shared_Folder

This should work – if it doesn't then ask me (or start Googling!)

Shared folders are shown with a really horrible colour scheme - Blue text on a green background.

Again, Google will show you a huge amount of frustration about this.

It can be fixed!

Enter gedit ~/.bashrc

This is a text editor

At the end of your .bashrc file enter

LS_COLORS='di=1:fi=0:ln=31:pi=5:so=5:bd=5:cd=5:or=31:mi=0:ex=35:*.rpm=90'
export LS_COLORS

Save the file, exit gedit

Type

source ~./bashrc

If you get an error it might be that you are using the wrong ' character (Word has a habit of altering them to smart quotes!) – just type ' directly in gedit

Now your directories should be legible.

5.3 Usearch

We will also use usearch. An older version integrates with Qiime but the new version doesn't (yet). We'll use both!

Download the most recent version of usearch from

http://www.drive5.com/usearch/download.html

This document was written with version 10.0

Also download v6.1.544 (beta)

The files are Unix executables and have very long names. Copy them to your Shared_Folder.

Rename your version 10.0 as usearch100

```
mv your_long_v10_filename usearch100
```

```
mv your_long_v6_filename usearch61
```

Now we'll move these to somewhere where we can run them

```
sudo mkdir /usr/bin/usearch_v10

sudo mkdir /usr/bin/usearch_v6

sudo mv usearch61 /usr/bin/usearch_v6/

sudo mv usearch100 /usr/bin/usearch_v10/
```

Now we set a symbolic link so that Unix thinks the executable files are in /usr/bin

```
sudo ln -s /usr/bin/usearch_v10/usearch100 /usr/bin/usearch100

sudo ln -s /usr/bin/usearch_v6/usearch61 /usr/bin/usearch61
```

If you enter usearch61 then you should see

```
(C) Copyright 2010-12 Robert C. Edgar, all rights reserved.
http://drive5.com/usearch

Licensed to: s.rolfe@sheffield.ac.uk


For documentation, please visit:

  http://drive5.com/usearch
```

and usearch10

```
usearch v10.0.240_i86linux32, 4.0Gb RAM (11.2Gb total), 3 cores
(C) Copyright 2013-17 Robert C. Edgar, all rights reserved.
http://drive5.com/usearch
```

5.4 FastQC

FastQC is a really useful utility to look at files.

Download FASTQC

http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

It's a java program so the executable is the same for Windows and Unix.

I find it easier to use under Windows but it can be used as a command line under Unix which is useful for looking at lots of files. Follow the instructions on the site – they are quite straightforward.

5.5 Notepad++

If you use Windows to edit files then the programs can mess up the file format. Also Notepad struggles with big files.

Notepad++ is a nice editor that runs under Windows and allows you to edit files without hassle.

https://notepad-plus-plus.org/

Attendance sheet

No.	Name	WS signature
1	Amoah-Antwi, Collins	
2	Henry Nicholls	
3	Rolfe, Stephen	
4	Saputra, Bastian	
5	Soria, Rosa	