Microbiome data manipulation and visualization in R

WestGrid Summer School Notes

Kim Dill-McFarland

version June 14, 2018

Introduction

In this workshop, we will demonstrate how to utilize R on the cluster, including

- loading R
- installing packages
- running an Rscript
- saving Rscript outputs

As well as how to begin to explore microbiome data in RStudio on your personal machine with the packages

- tidyverse (https://www.tidyverse.org/)
- phyloseq (https://www.bioconductor.org/packages/release/bioc/html/phyloseq.html)

Data used in this workshop are from an on-going time series in Sannich Inlet (http://www.oceannetworks.ca/introduction-saanich-inlet), British Columbia, Canada. If you would like a brief introduction to the system, please read Hallam SJ *et al*. 2017. Sci Data 4: 170158 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5663219/) “Monitoring microbial responses to ocean deoxygenation in a model oxygen minimum zone”. You can also check out this short video (https://drive.google.com/file/d/1s6V263-Vj2KQ6rExOBaYn-UDOERyQVGZ/view?usp=sharing) showing how the sampling was done!

More information on the sequencing data can be found in Hawley *et al*. 2017. Sci Data 4: 170160 (https://www.nature.com/articles/sdata2017160) and details on sequence processing for this workshop are given in the mothur pipeline (https://github.com/EDUCE-UBC/workshops/blob/master/microbiome_summer_school/data/mothur_pipeline.html) in the workshop materials. Details on the geochemical data are also available in Torres-Beltrán *et al*. 2017. Sci Data 4: 170159 (https://www.nature.com/articles/sdata2017159).

All data and materials for this workshop are available at https://github.com/EDUCE-UBC/workshops/tree/master/microbiome_summer_school (https://github.com/EDUCE-UBC/workshops/tree/master/microbiome_summer_school).

R on a cluster

Set-up

We will be working on cedar (https://docs.computecanada.ca/wiki/Cedar) (Compute Canada) so please log-in.

```
ssh -Y username@cedar.computecanada.ca
```

And create a directory for this workshop in your scratch (working) directory.

```
mkdir ~/scratch/microbiome
```

Data

In another terminal, not logged into cedar, copy the `dist.mat.RData` distance matrix from wherever it is downloaded on your computer to `~/scratch/microbiome/` on cedar.

```
scp ~/GitHub/workshops/microbiome_summer_school/data/dist.mat.RData username@ce
dar.computecanada.ca:~/scratch/microbiome/
```

Running R

R is already installed and you can checkout what versions are on the cluster with the following. *Note that R is listed as lowercase "r" on Compute Canada resources but the actual program name is uppercase "R".*

```
module spider r
```

```
r:
----------------------------------------------
    Description:
      R is a free software environment for statistical computing and
      graphics.

     Versions:
        r/3.3.3
        r/3.4.0
        r/3.4.3
        r/3.5.0
     Other possible modules matches:
        admixture  amber  annovar  arcs  armadillo  arpack-ng  arrow  ...

----------------------------------------------
  To find other possible module matches execute:

      $ module -r spider '.*r.*'

----------------------------------------------
  For detailed information about a specific "r" module (including how to load t
he modules) use the module's full name.
  For example:

     $ module spider r/3.5.0
----------------------------------------------
```

We want to use the most recent version of R available (though anything 3.4+ will work). Load the default version with

```
module load r
```

We will also load gcc since many R packages are compiled using the Gnu family.

```
module load gcc
```

Now that R is loaded, we can run it in interactive mode by calling the program.

```
R
```

```
R version 3.5.0 (2018-04-23) -- "Joy in Playing"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```

Installing packages

R itself does not contain all of the functions we would like to use to analyze microbiome data. Many functions are added to R in packages that must be installed separately.

We can see what packages are already installed with

```
installed.packages()
```

```
           Package
base       "base"
boot       "boot"
class      "class"
cluster    "cluster"
codetools  "codetools"
compiler   "compiler"
datasets   "datasets"
foreign    "foreign"
graphics   "graphics"
grDevices  "grDevices"
grid       "grid"
KernSmooth "KernSmooth"
lattice    "lattice"
MASS       "MASS"
Matrix     "Matrix"
methods    "methods"
mgcv       "mgcv"
nlme       "nlme"
nnet       "nnet"
parallel   "parallel"
rpart      "rpart"
spatial    "spatial"
splines    "splines"
stats      "stats"
stats4     "stats4"
survival   "survival"
tcltk      "tcltk"
tools      "tools"
utils      "utils"
```

Unless you've previously used R on cedar, these are the basic packages that come with base R and RStudio install.

It is best practices to install additional packages natively to your account so that you are able to add and update them as needed (without needing server admin privileges).

We will install 1 package for this workshop.

ape

The Analysis of Phylogenetics and Evolution package, or `ape`, contains functions related to the calculation, visualization, and analysis of phylogenetic trees. `ape` is a CRAN (https://cran.r-project.org/) hosted package; any CRAN package can be installed similarly without need to call a specific source as CRAN is the default.

```
install.packages("ape")
```

Since you cannot write to the packages directory on cedar, R with ask you if you would like to create a personal library for packages.

```
Warning in install.packages("ape") :
  'lib = "/cvmfs/soft.computecanada.ca/easybuild/software/2017/avx2/Compiler/gc
c5.4/r/3.5.0/lib64/R/library"' is not writable
Would you like to use a personal library instead? (yes/No/cancel)

Would you like to create a personal library
‘~/R/x86_64-pc-linux-gnu-library/3.5’
to install packages into? (yes/No/cancel)
```

Answer `yes` to both questions and then select a CRAN server near your current geographical location.

```
Selection: 64
also installing the dependency ‘Rcpp’

trying URL 'https://ftp.osuosl.org/pub/cran/src/contrib/Rcpp_0.12.17.tar.gz'

...

The downloaded source packages are in
    ‘/tmp/RtmpR0jpS9/downloaded_packages’
```

If we exit R and list within our personal library, we can see our successfully installed packages.

```
quit()
```

```
ls ~/R/x86_64-pc-linux-gnu-library/3.5
```

```
ape  Rcpp
```

You will see that `ape` as well as its dependency package `Rcpp` have been installed.

Running an R script

While you can run R in interactive mode, it is more efficient to create an R script. R scripts are simply text documents listing the functions you would like to run in R and saved under `.R` .

To calculate a neighbor-joining tree of our representative sequences, we will use the `bionj` function from the `ape` package. Then we can save the resulting tree data as `.RData` with `save.image` , which comes as a function in base R.

Let’s create an Rscript; any text editor will do.

```
cd scratch/microbiome/

nano
```

Enter the following into the document.

```
# Set directory to current project
setwd("~/scratch/microbiome")

# Load relevant packages
library(ape)

# Read in distance matrix
load("dist.mat.RData")

# Calculate neighbor-joining tree
NJtree = bionj(dist.mat)

# Save workspace to an RData object
save.image(file="NJtree.RData")
```

As you exit, save the script as `NJtree.R` . Then, you can use `NJtree.R` within a normal bash script.

Next, create the following in a text editor and save as `submit.sh`

```
#!/bin/bash
#SBATCH --account=wgssubc-wa_cpu
#SBATCH --reservation=wgssubc-wr_cpu
#SBATCH --ntasks=5               # number of MPI processes
#SBATCH --mem-per-cpu=2048M      # memory; default unit is megabytes
#SBATCH --time=0-00:30           # time (DD-HH:MM)

module load r

mpirun -np 1 R CMD BATCH NJtree.R
```

This account and reservation are specific to this workshop. If you were to run an Rscript job in future, you would use your username (if you have allocation) or your PIs account if you are working within their specified project.

Finally we submit the job

```
sbatch submit.sh
```

We can watch it's progress in the queue.

```
squeue -u username
```

Or view the `.Rout` file to see how far along into the Rscript the job it. This is also very helpful in de-bugging scripts as it will end at a failed step if one occurs.

```
cat NJtree.Rout
```

Once the job is complete (~20 min), you can copy the `NJtree.RData` file to your computer or use the one provided with the materials for this workshop.

RStudio on a personal machine

While some steps, like calculating trees, need to be performed on the cluster, many microbiome data sets are sufficiently small to run on your own machine once the raw sequences have been processed.

This is best done using the R GUI RStudio. This interface contains the R console (similar to what you saw on the cluster) as well as additional data, package, and graphics features.

Here, we will begin to explore the Saanich Inlet data using the `tidyverse` and `phyloseq` packages.

Installing packages

As with the cluster, packages need to be installed on your machine. This is done with similar code to what we used on cedar. However, now we will install `tidyverse` and `phyloseq`, instead of `ape`. The `tidyverse` is on CRAN so it is simply installed like so.

```
install.packages("tidyverse")
```

In contrast, `phyloseq` is stored on Bioconductor (https://www.bioconductor.org/) and must be installed by first specifying the source.

```
source('http://bioconductor.org/biocLite.R')

biocLite("phyloseq")
```

We then load the packages into our current R session. *Loading must be done every time you open R/RStudio whereas the install is one time.*

```
library(tidyverse)
```

```
## ── Attaching packages ──────────────────── tidyverse 1.2.1 ──
```

```
## ✔ ggplot2 2.2.1     ✔ purrr   0.2.4
## ✔ tibble  1.4.2     ✔ dplyr   0.7.4
## ✔ tidyr   0.8.0     ✔ stringr 1.3.0
## ✔ readr   1.1.1     ✔ forcats 0.3.0
```

```
## ── Conflicts ─────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(phyloseq)
```

Data

In order to facilitate our later use of `phyloseq`, we need the following data from the workshop repo (https://github.com/EDUCE-UBC/workshops/tree/master/microbiome_summer_school).

- OTU table (`.shared` from mothur (https://www.mothur.org/wiki/Main_Page))
- Taxonomy (`.taxonomy` from mothur (https://www.mothur.org/wiki/Main_Page))

- Sample metadata (`.txt`)
- Neighbor joining tree (`.RData` from our calculation on cedar)

Tables can be read in with `read.table` , which is a generic import function from base R. We specify that our tables are tab delimited (`sep`) as well as have column names (`header`) and row names in the first column (`row.names`). *Remember to change to file path to where you have the data downloaded on your machine.*

```
OTU = read.table("data/Saanich.final.opti_mcc.unique_list.shared", sep="\t", he
ader=TRUE, row.names=2)

taxonomy = read.table("data/Saanich.final.opti_mcc.unique_list.0.03.cons.taxono
my", sep="\t", header=TRUE, row.names=1)

metadata = read.table("data/Saanich.metadata.txt", sep="\t", header=TRUE, row.n
ames=1)
```

`.RData` files, *i.e. our tree*, are imported with `load` .

```
load("data/NJtree.RData")
```

If we look at the data from mothur (OTU and taxonomy), they are not perfectly setup so we need to manipulate them a little with the `tidyverse` before we can use `phyloseq` .

Introduction to the tidyverse

The tidyverse (https://www.tidyverse.org/) is actually a group of packages for data science. All packages share an underlying design philosophy, grammar, and data structures.

To clean-up our mothur-processed data, we will use

- `select` a subset of variables (columns) (from `dplyr` package)
- `separate` separate 1 column into multiple (from `tidyr` package)

However, there are many more functions like

- `filter` out a subset of observations (rows)
- `arrange` the observations by sorting a variable in ascending or descending order
- `mutate` all values of a variable (apply a transformation)
- `*_join` two data frames into a single data frame
- etc…

Compared to base R, tidyverse code runs much faster. It is also much more readable because all operations are based on *verbs* (select, filter, mutate…) rather than base R's more difficult to read indexing system (brackets, parentheses…).

Each verb works similarly:

- input data frame in the first argument
- other arguments can refer to variables as if they were local objects
- output is another data frame

```
verb(data, argument1, argument2...)
```

Data cleaning

Now let's use the `tidyverse` to clean-up the mothur-formatted tables.

OTU table

We need to

- Remove extra columns "label" and "numOtus"

which is done with `select` and `-` before the variables we would like to remove (as opposed to those we would like to keep, which would not have the `-`).

```
OTU.clean = select(OTU, -label, -numOtus)
```

Taxonomy table

We need to

- Remove extra column "Size"
- Separate taxonomy into 1 column per level
- Name taxonomy columns by level

In this case, we want to string multiple tidyverse verbs together. We can do so by connecting them with a pipe `%>%`. This takes the output of the first verb and puts it in as the data used by the second verb.

```
taxonomy.clean = select(taxonomy, -Size) %>%
  separate(Taxonomy, c("Domain", "Phylum", "Class", "Order", "Family", "Genus",
 "Species"), sep=";")
```

This begins to show you the utility of the `tidyverse` in data cleaning as you can string together many steps easily and in a relatively readable fashion.

Introduction to phyloseq

Now that we have properly formatted data, we can use it in `phyloseq`.

The `phyloseq` package is specifically designed for the analysis of microbiome data. In particular, it is used to

- import data processed in other software (QIIME, mothur, RDP)
- assess alpha- and beta-diversity
    - Calculation
    - Visualization
    - Multiple testing methods specific to high-throughput amplicon sequencing data

`phyloseq` also follows a syntax similar to that of the tidyverse, which means it fits quite well in our pipeline!

phyloseq object

`phyloseq` has very specific formatting so we need to use its functions to tell it exactly what sort of data each of our data frames is. Note that our tree does not need to be formatted as it is already in a format recognized by phyloseq.

```
OTU.clean.physeq = otu_table(as.matrix(OTU.clean), taxa_are_rows=FALSE)

tax.clean.physeq = tax_table(as.matrix(taxonomy.clean))

metadata.physeq = sample_data(metadata)
```

Then we merge all the pieces into 1 phyloseq object.

```
saanich = phyloseq(OTU.clean.physeq, tax.clean.physeq, metadata.physeq, NJtree)
```

```
saanich
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 4368 taxa and 7 samples ]
## sample_data() Sample Data:       [ 7 samples by 11 sample variables ]
## tax_table()   Taxonomy Table:    [ 4368 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 4368 tips and 4366 internal nodes ]
```

We see that we have out 7 depths with 4,368 OTUs identified and 11 geochemical metadata variables. Note that merging the tables into a `phyloseq` object automatically removed any depths from the metadata that do not have corresponding sequence data.

Tree plots

Let's first plot the tree we so painstakingly calculated on the cluster.

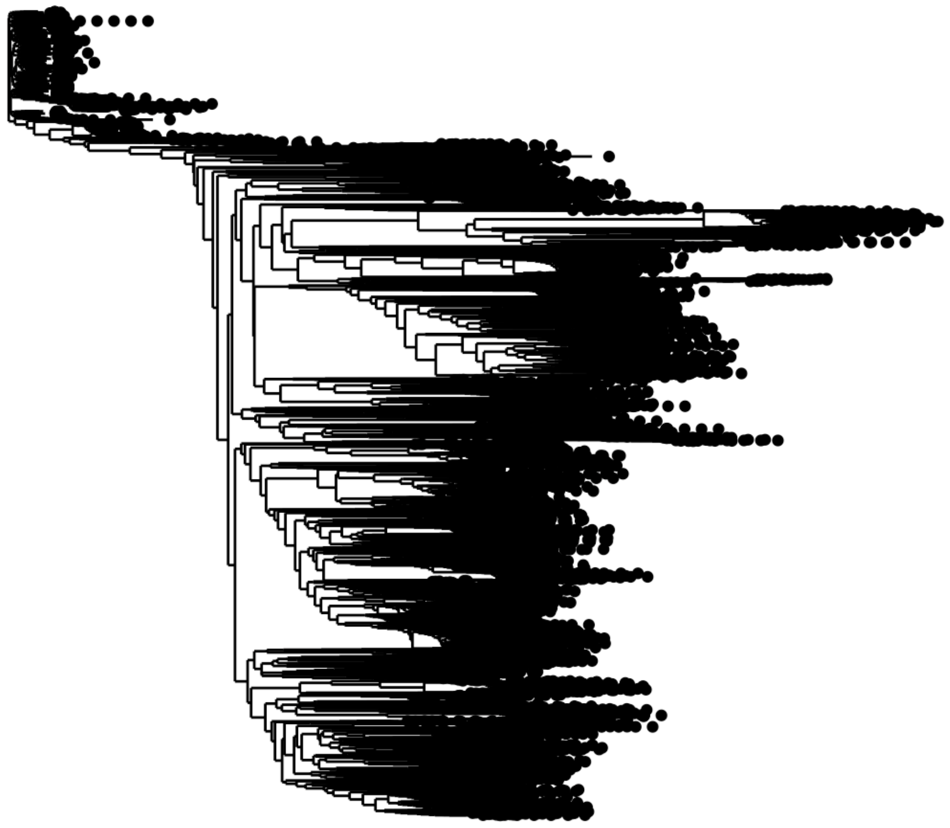
`phyloseq` plots mirror the syntax and logic of ggplot from the `tidyverse` in that they both have:

- *aesthetics*: map variables to visual attributes (e.g., position)
- *geoms*: graphical representation of data (points, lines, etc.)
- *stats*: statistical transformations to get from data to points in the plot(binning, summarizing, smoothing)
- *scales*: control *how* to map a variable to an aesthetic
- *facets*: juxtapose mini-plots of data subsets, split by variable(s)
- *guides*: axes, legend, etc. reflect the variables and their values

The idea is to independently specify and combine the blocks (with +) to create the plot you want.

We start simply with the `plot_tree` function.

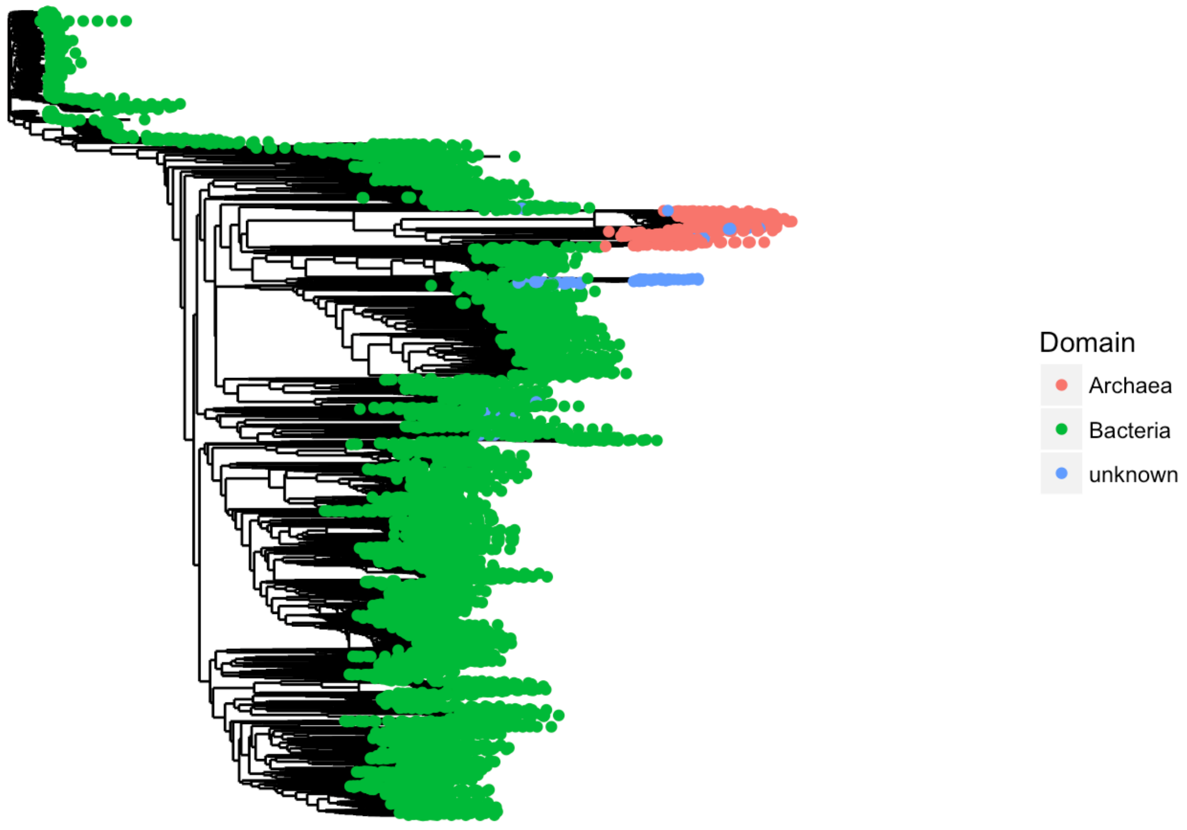
```
plot_tree(saanich)
```



Now this is not very informative as it has all 4000+ OTUs and is generally unreadable. The beauty of the `phyloseq` object, however, is that it contains everything - OTUs, taxonomy, metadata, and the tree - so we can easily add a color aesthetic from other data pieces onto the tree.

First, we'll color by domain to see how the tree did at separating out sequences at this high taxonomic level.

```
plot_tree(saanich, color="Domain")
```



As expected, the Archaea cluster together along one branch while the more diverse Bacteria take up the rest of the tree.

However, this is a lot of data to try to visualize. We can subset to a taxon of interest using `subset_taxa` so that we may more easily see results and patterns. Let's look at the Genus *Nitrospina* as it plays an important role in the nitrogen cycle converting nitrite (NO_2) to nitrate (NO_3).

```
nitrospina = subset_taxa(saanich, Genus == "Nitrospina")
```

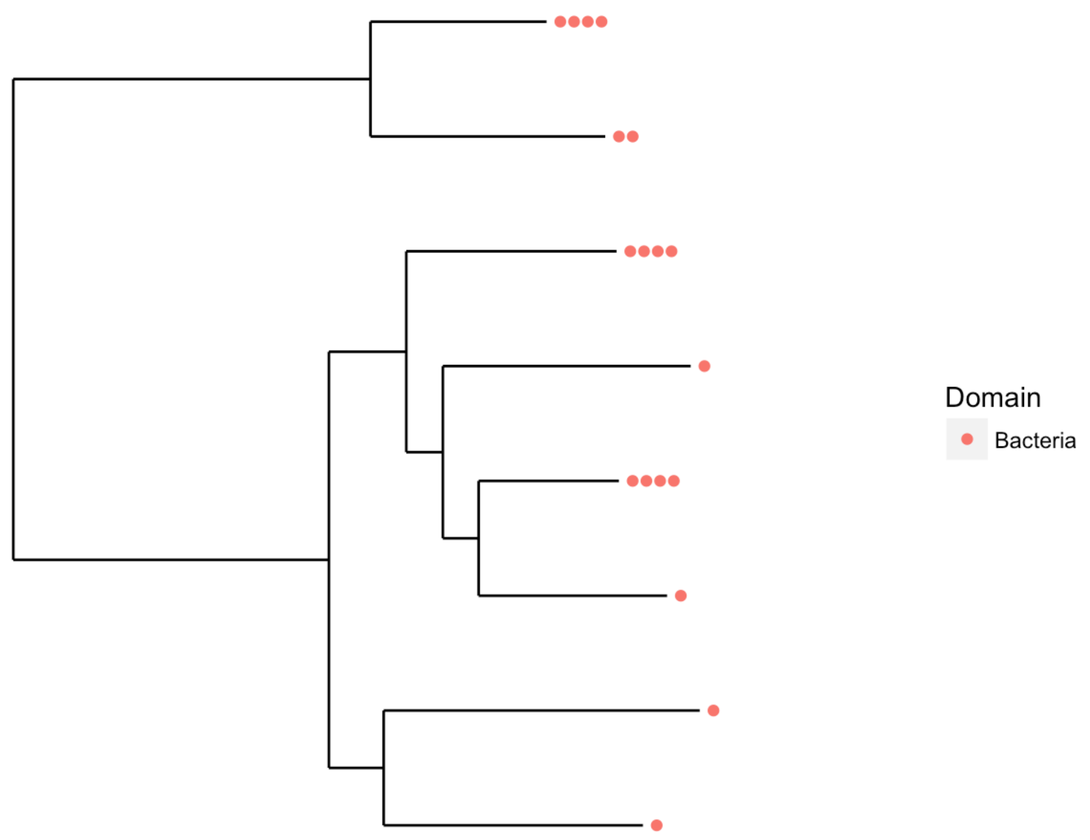
We see that this has reduced the data set to just the 8 OTUs classified as *Nitrospina*.

```
nitrospina
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 8 taxa and 7 samples ]
## sample_data() Sample Data:       [ 7 samples by 11 sample variables ]
## tax_table()   Taxonomy Table:    [ 8 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 8 tips and 7 internal nodes ]
```

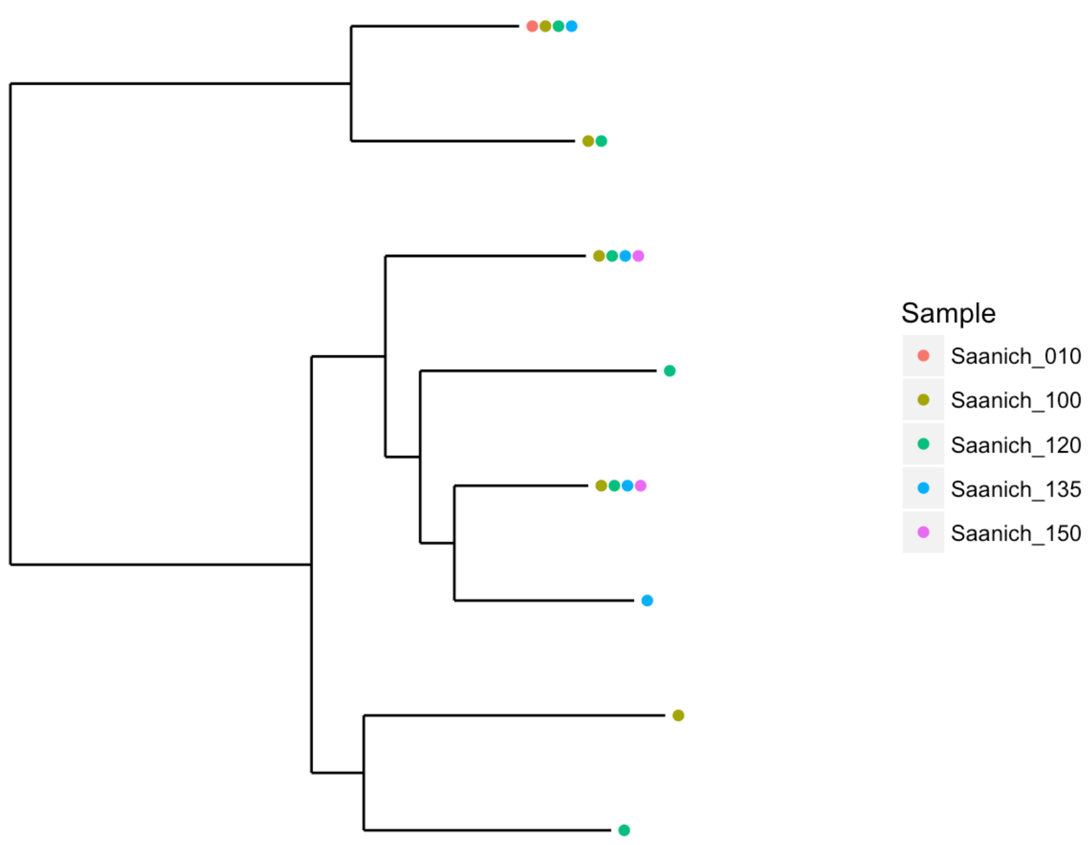
Now our tree is quite manageable.

```
plot_tree(nitrospina, color="Domain")
```



We can now start to plot more informative metadata onto the tree as color, shape, size, etc. For example, we can color by the sample name to see in which sample(s) each OTU occurs.

```
plot_tree(nitrospina, color="Sample")
```

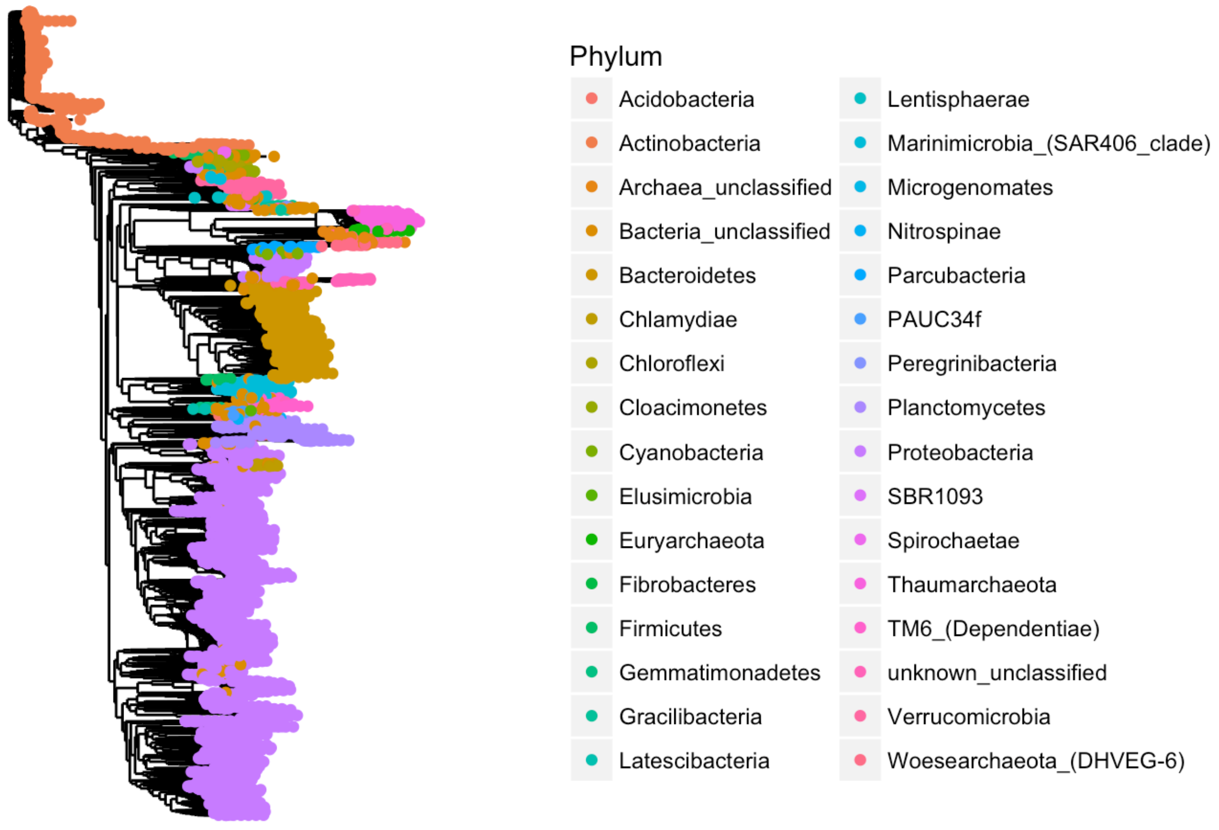


You can learn more about the parameters available for `plot_tree` (or any function) using the help `?` function.

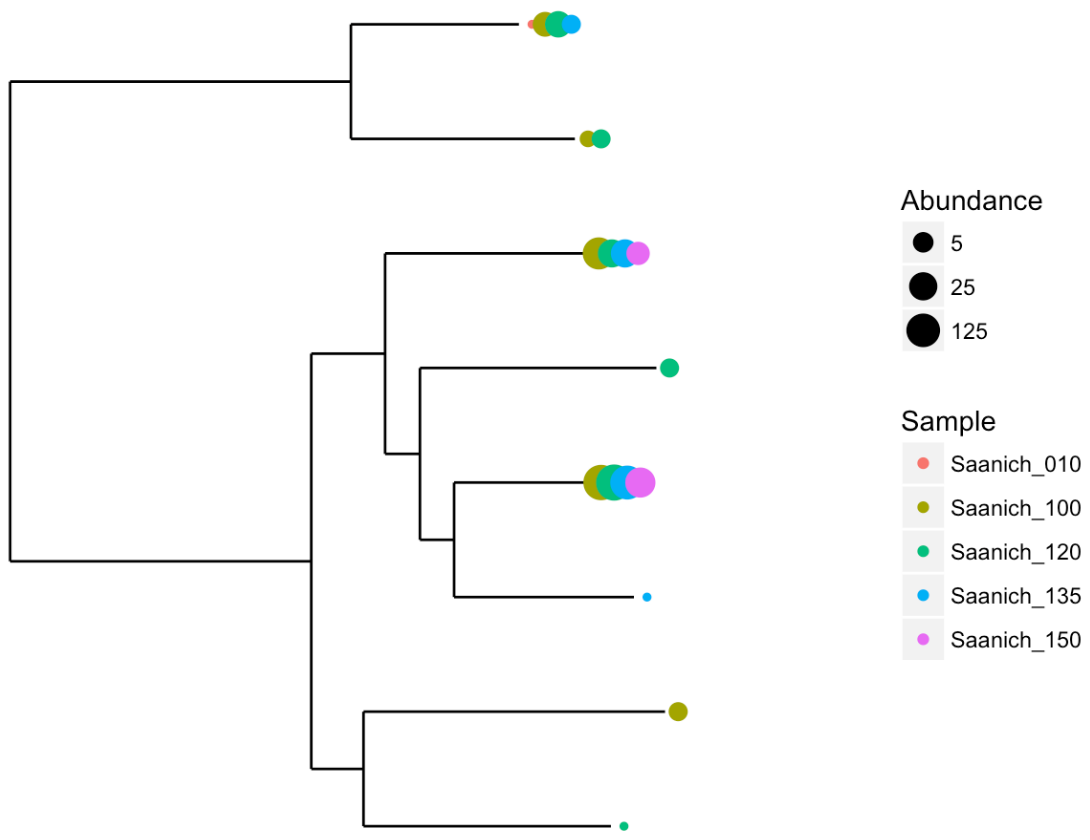
```
?plot_tree
```

Exercise 1

1. Change the color in the large (sannich) tree to Phylum taxonomy.



2. Add "Abundance" data in the form of `size` to the Nitrospina tree.

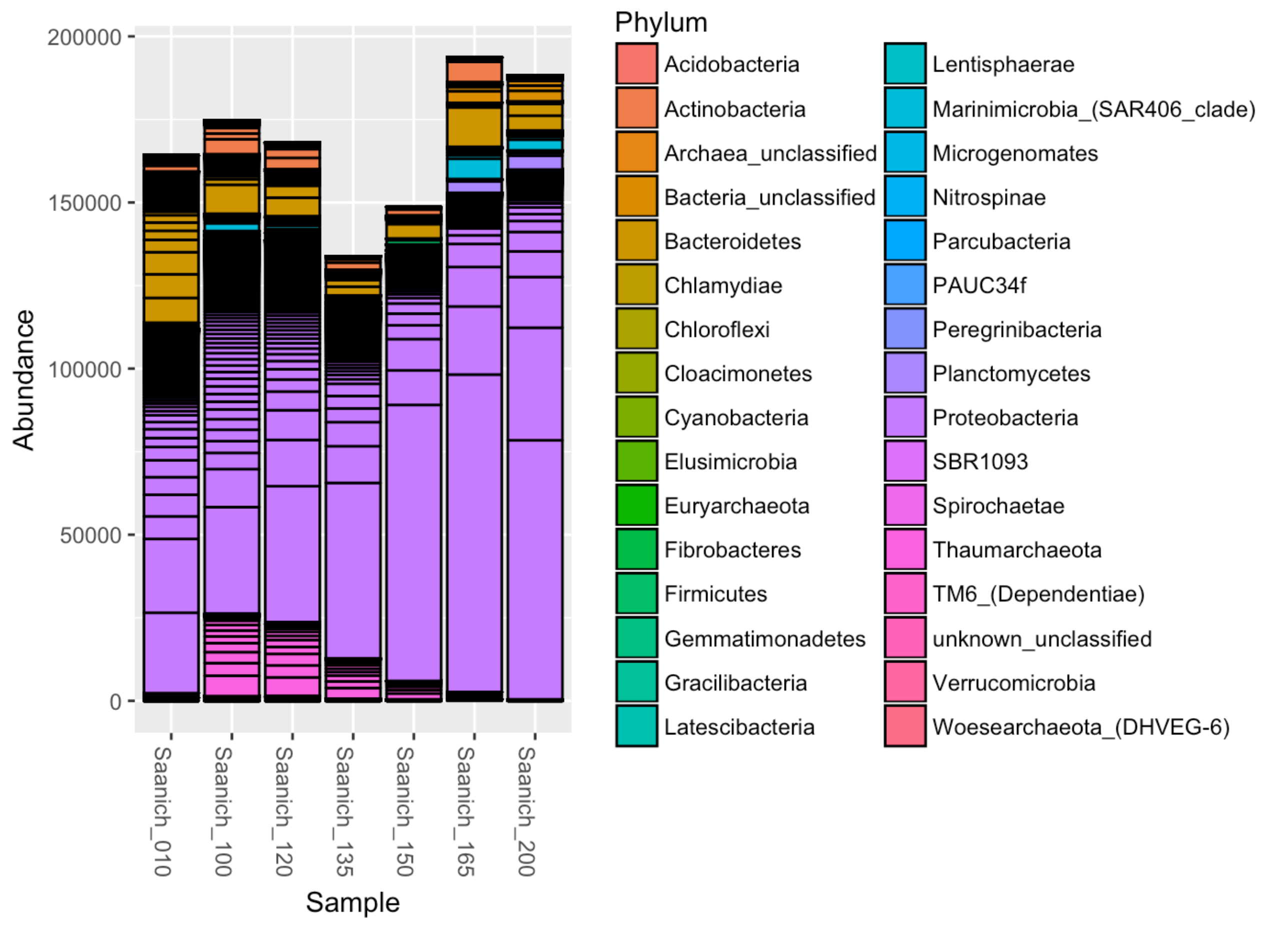


Taxonomy plots

Now let's leverage our data to begin to answer some of our earlier questions like "How do microbial communities change with depth or nutrient availability?"

With `plot_bar`, we can create stacked taxonomy plots to see overall trends in microbial communities.

```
plot_bar(saanich, fill="Phylum")
```

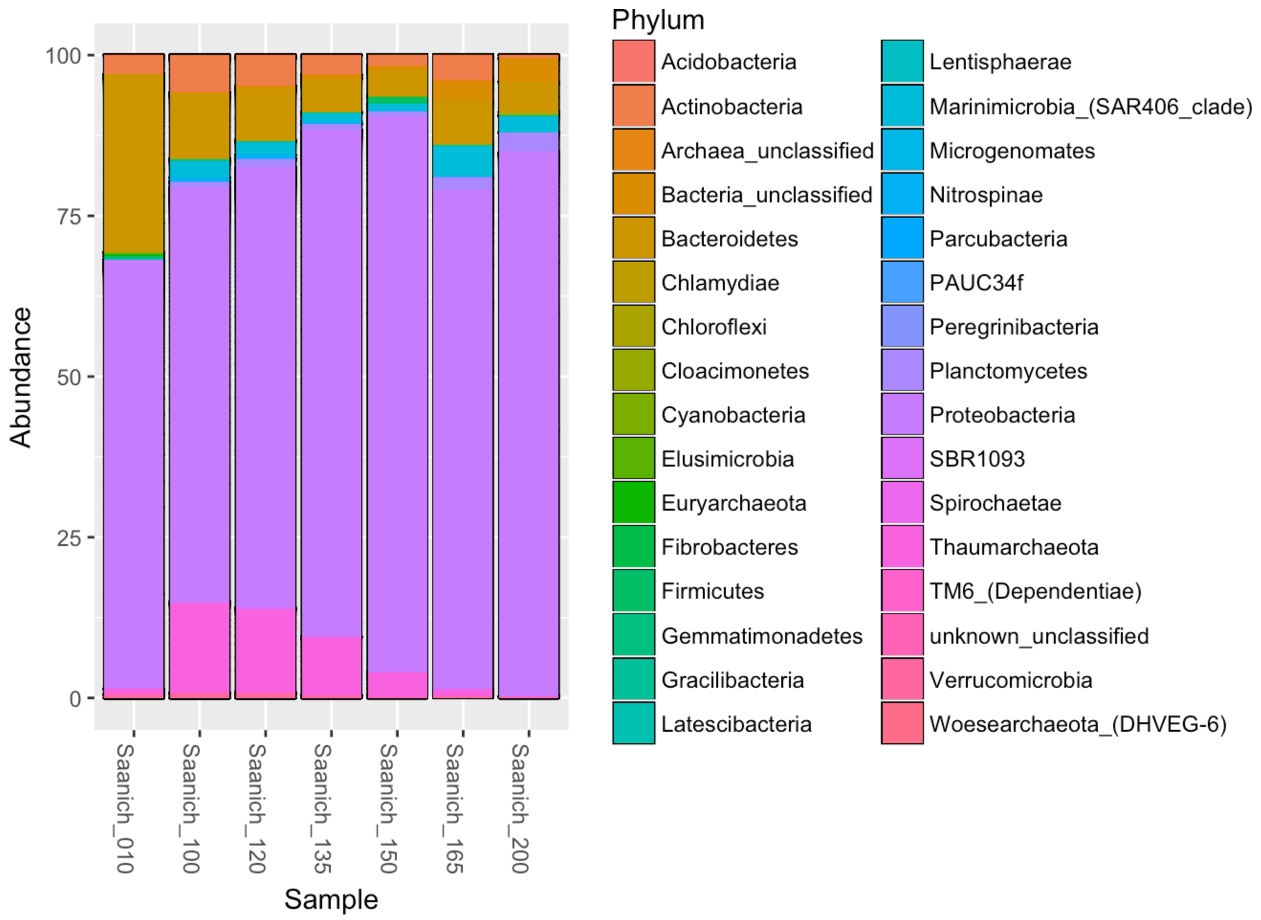


However, we do not have equal numbers of sequences per sample (a common issue). So we can call on another `phyloseq` function, namely `transform_sample_counts`, to convert our raw sequence counts into percent relative abundance.

```
saanich_percent = transform_sample_counts(saanich, function(x) 100 * x/sum(x))
```

Plotting the transformed data and adding a `geom` to pretty up the bars by removing all the lines between individual OTUs, we can start to see how the communities change with depth.

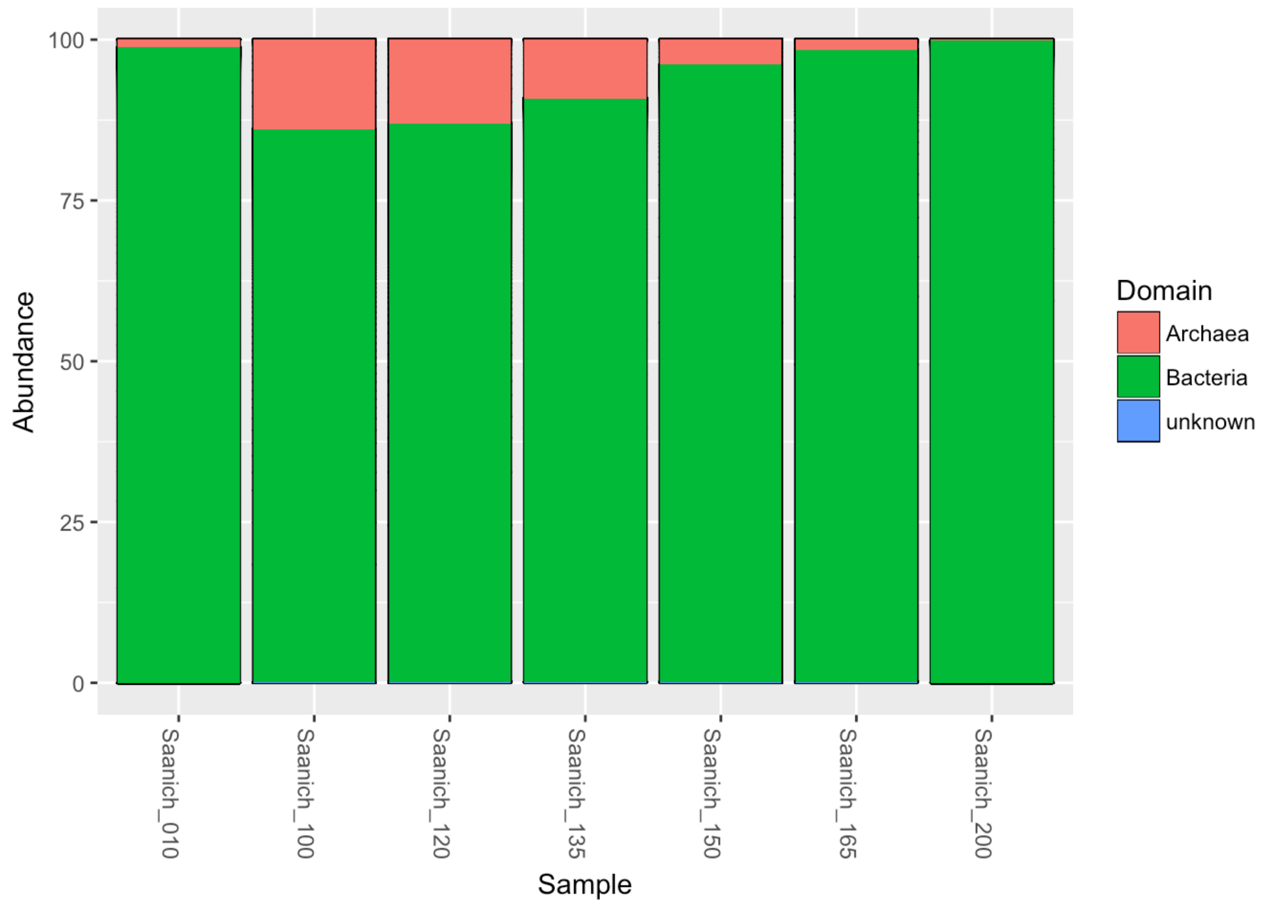
```
plot_bar(saanich_percent, fill="Phylum") +
  geom_bar(aes(fill=Phylum), stat="identity")
```



Again there are many more parameters and options, so checkout `?plot_bar` !

Exercise 2

1. Change the color to Domain level taxnomy in the bar chart. *Hint*: You will need to change something in both geoms.



Correlating metadata

Finally, we'll delve into a more specific microbial question by sub-setting our data to 1 bacterial genus and looking to see if it correlates with any of our geochemical measurements. This is the same `subset_taxa` function we used with the tree, only now we want to use our percent relative abundance data (`saanich_percent`) to correct for unequal sequence counts across samples.

First, we subset the data, still working in `phyloseq` .

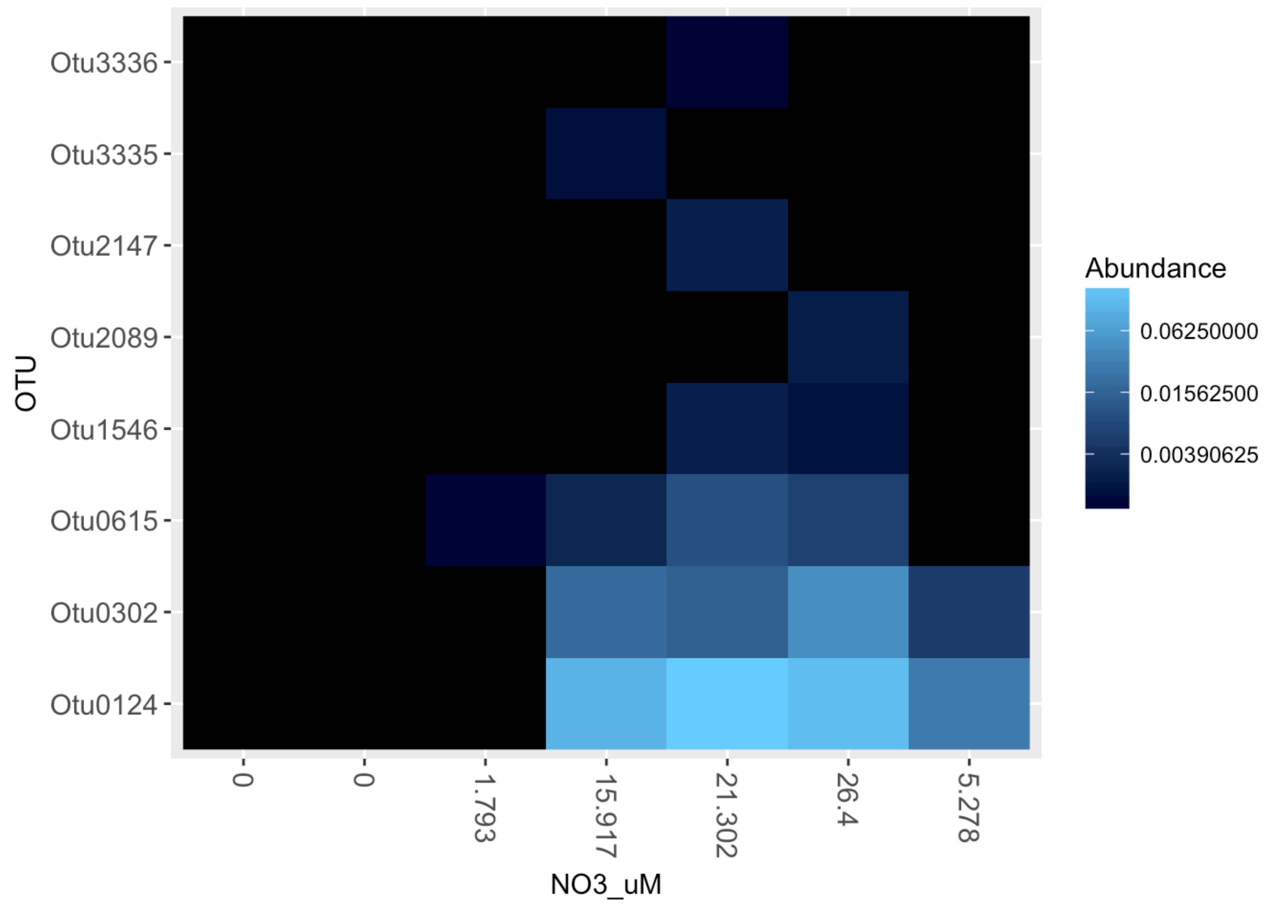
```
nitrospina_percent = subset_taxa(saanich_percent, Genus=="Nitrospina")

nitrospina_percent
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 8 taxa and 7 samples ]
## sample_data() Sample Data:       [ 7 samples by 11 sample variables ]
## tax_table()   Taxonomy Table:    [ 8 taxa by 7 taxonomic ranks ]
## phy_tree()    Phylogenetic Tree: [ 8 tips and 7 internal nodes ]
```

Then, we can plot a heatmap against a variable from our metadata to look for trends. Given *Nitrospina*'s role in the nitrogen cycle, we will explore nitrate (NO_3).

```
plot_heatmap(nitrospina_percent, method=NULL, sample.label="NO3_uM", sample.ord
er="NO3_uM")
```

```
## Warning: Transformation introduced infinite values in discrete y-axis
```

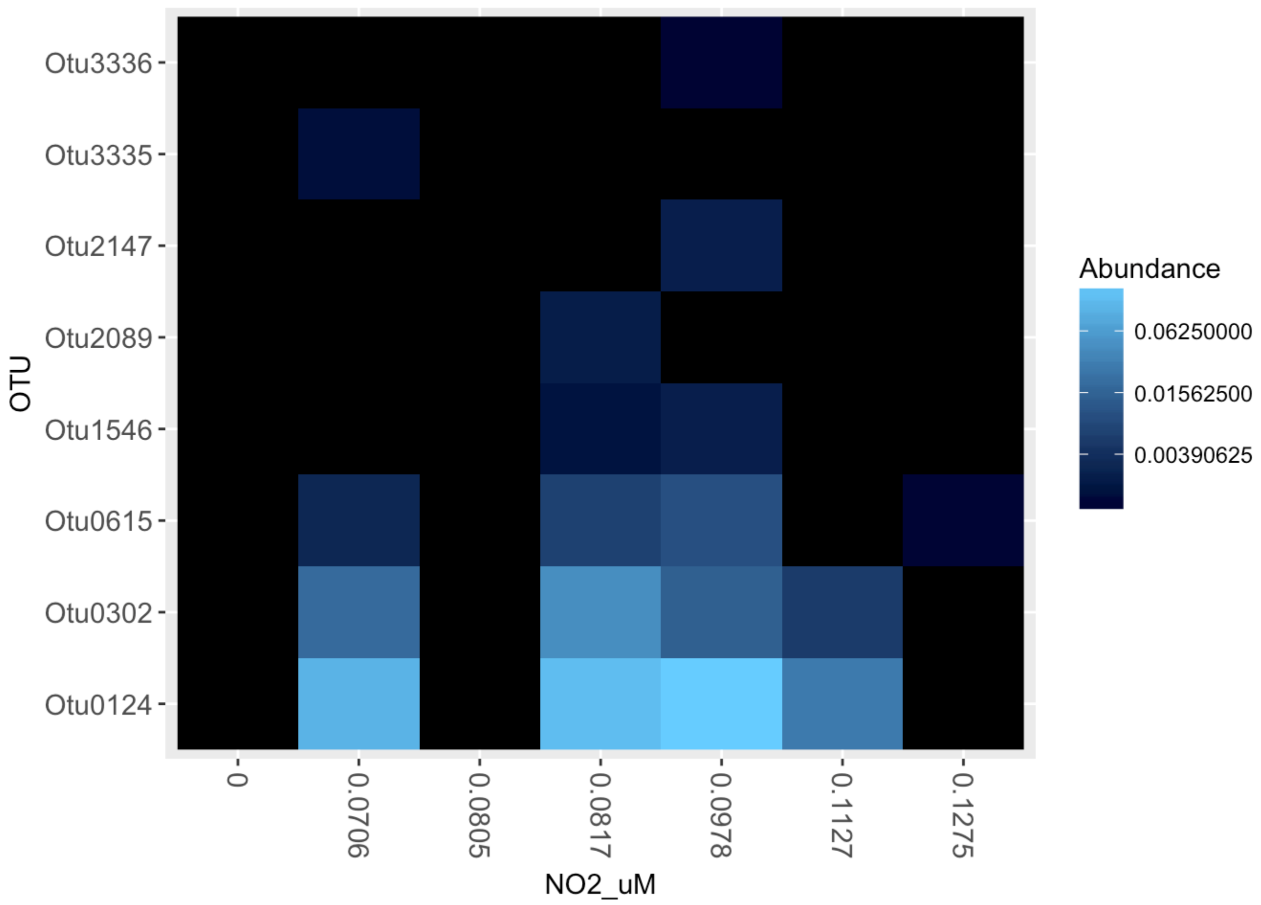


As expected, *Nitrospina* OTUs tend to be more highly abundant when NO_3 is abundant, likely due in part to this genus of bacteria creating NO_3 as part of its metabolism.

Exercise 3

1. Create a heatmap plotting *Nitrospina* against NO_2.

```
## Warning: Transformation introduced infinite values in discrete y-axis
```



2. Choose another genus of interest to you and
    a. subset `saanich_percent` to your chosen genus
    b. create a tree of this genus showing its distribution among the samples with shape, color, and/or size.
    c. create a heatmap ploting the genus abundance against a metadata variable it may be correlated with.

Concluding remarks

This has been a brief introduction to a number of tools of use in microbiome sequence data analysis and is by no means exhaustive. Hopefully, you've learned enough to begin to explore your own data!

The best way to learn R is to use it so please use the resources below to continue exploring the `tidyverse` and `phyloseq`. You may also be interested in more extensive R workshops given by ECOSCOPE (http://ecoscope.ubc.ca/events/).

Happy coding!

More resources

In this workshop, we have gone over just some of the functions and graphics within the tidyverse and phyloseq. Below are some resources for further learning and practice!

- R cheatsheets (https://www.rstudio.com/resources/cheatsheets/) also available in RStudio under Help > Cheatsheets

- Introduction to dplyr (https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html)
- dplyr tutorial (https://rpubs.com/justmarkham/dplyr-tutorial)
- dplyr video tutorial (https://www.r-bloggers.com/hands-on-dplyr-tutorial-for-faster-data-manipulation-in-r/)
- More functions in dplyr and tidyr (https://rpubs.com/bradleyboehmke/data_wrangling)

- ggplot tutorial 1 (http://r-statistics.co/Complete-Ggplot2-Tutorial-Part1-With-R-Code.html)
- ggplot tutorial 2 (https://rpubs.com/g_jw/ggplot2_tutorial)
- ggplot tutorial 3 (http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html#the_1_faq)

- phyloseq tutorials (https://joey711.github.io/phyloseq/)