Welcome to the sequence analysis section of the practical course for the module Functional Microbiome Research, 2022! What follows is a living document that may change as the course progresses. Please be tolerant of errors therein, and let me know when you find them. I will try to keep the most current version of this document available on the course website and/or the github repository.

Introduction:

Modern microbial ecology projects typically rely at least in part on culture-free sequencing of DNA or RNA extracted from complex substrates such as soil, water, or animal/plant tissue. These microbial surveys use two broadly defined methods: **metabarcoding** (also known as "amplicon sequencing") and **metagenomics**. In this sequence analysis section of the module, we will examine both methods, using some popular software packages. In order to do this, we must also learn about using remote computing resources, and how to work in a Linux command line environment.

Throughout the course, you have been given several theoretical lectures on bioinformatic methods, including some explanations of fundamental algorithms. You have also been given numerous examples of microbiomes in nature. Finally, you are creating data from a microbiome right now, during the course of this practical course. In this section of the module, you will bring your new knowledge and your data together, to conduct a metabarcoding study on your soil samples, and a rudimentary metagenomic analysis of two public data sets.

We will cover three large topics, each of which will have a script associated with it that we as a class will generate together:

1. Scientific computing and Linux – script here
2. Metagenomic methods – script here
3. Metabarcoding methods – script here

Each night I will update the scripts to structure the discussion for the next day, and we will run through them in class together. You will have to adapt them to your own computing environment. Together they will become a record of our activity as a class, as we work through the bioinformatic pipelines. The living versions will be kept in the github repository links above, and we can return to former versions if necessary using the magic of git.

Topic 1. Scientific computing

Many algorithms that we will use require large reference databases and examine large portions of experimental data in real-time. Modern studies generate sequence data that can also be quite large in size, sometimes multiple terabytes in size. We can safely say that bioinformatic analyses are very "memory hungry", sometimes requiring hundreds of gb of RAM to conduct more intensive calculations and alignments. Our data will be relatively modest in size, but even our datasets and databases will sometimes require enough memory to overwhelm the (random-access) memory resources of standard home or office computers.

Scientists address these issues of "big data" in two ways:

(1) Scientists typically minimize the non-essential parts of the programs that they write to implement their algorithms. This means that you will see very few Graphical User Interfaces (GUIs) used for bioinformatic pipelines, though some high-quality bioinformatic GUIs ones do exist. Instead, software for bioinformatics are usually called from text-based environments called **command-line interfaces** (also called "terminals" or "shells"). Because they are so minimal, these programs are also more flexible - they can be wrapped into other programs, including your own simple programs (**scripts**) that you write. Our first sessions will focus on learning BASH to use these programs, probably the most popular command-line interface for scientific computing.

(2) Scientists do quite a lot of remote computing, using computing clusters and other shared computing resources to conduct memory-intensive operations on their data. In order to conduct modern research, universities, governments, and corporations have invested in shared computing centers with the hardware necessary for this kind of research. For most of these very large computing resources, Linux/Unix-like operating systems are the most efficient and universal work environment. In our case, we will be using Ubuntu Linux virtual machines in the cloud computing resources of the German Network for Bioinformatics Infrastructure (de.NBI), who have kindly offered their support for this course.

Getting linux working locally

The first step to learning about scientific computing is to get a working Linux environment onto your personal machine. **Please attempt the following installation of Linux before our first day**. This step of getting Linux working on your computer can be complicated, but it is essential for everything else that we do! If it fails for you, we understand, and we will spend much of our first session debugging this step and the following so that everyone can get to their computing resources.

Mac users:

You may not know it, but you have been running something very close to Linux every time you opened your computer! So leave your computer at peace, and just open the terminal application: use finder to go to the /Applications/Utilities folder, and double click "terminal".

PC users, Windows version >=10 :

Windows has an in-box solution for creating a Linux environment, called **W**indows **S**ubsystem for **L**inux (WSL). If your windows version is 10 or newer, you probably have this available in some form on your computer, but you must activate it. The exact process for activating WSL will depend on your windows version.
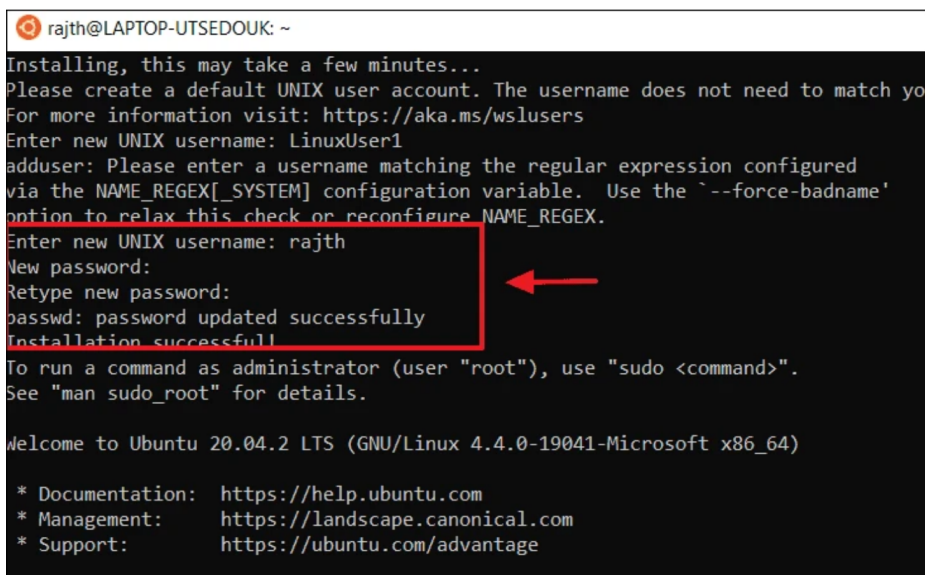
In addition to the official documentation, I found the following website to be very helpful for installation on the PC which I tested, using Windows 10:

https://allthings.how/how-to-use-linux-terminal-in-windows-10/

This site may be helpful for newer versions of windows:
https://www.altisconsulting.com/uk/insights/installing-ubuntu-bash-for-windows-10-wsl2-setup/

Once you have activated WSL on your PC, and have installed Ubuntu, test it out by searching for "Ubuntu" in your start menu. Running this should bring up a terminal, and ask you to create a username and password, something that looks something like:

PC users, Windows version <10 :

Finally, if you are running a really old version of Windows, let me know. We will find another solution.

Getting around a Linux Environment:

Basic shell commands and symbols

These are the universal BASH commands we will be using to get around and to manipulate our files, in addition to specialized bioinformatic programs. We will now try out each in our local BASH shells, so you can get accustomed to them.

pwd – print the current working directory

ls – list the contents of the current directory

cd – change directory

cp – copy a file

mkdir – make a new directory

echo – print something out to the display

cat – concatenate two or more files

rm [-r] – remove a file. Essentially deletes a file forever. Be careful!!

sudo – execute another command that requires root privileges

apt – access Ubuntu's native software package management system, for updating and upgrading software.

top – opens up a real time display that shows you the "busiest" processes ongoing in your machine.

man – print the reference manual for a utility, command, or program, if available. If not, try the --help flag for any given program without a manual.

head – show top several lines from a text file

tail – show last several lines from a text file

less – interactively open a text file for reading

Important symbols for BASH environment

We will also play with the following special characters:

~ home directory (try with cd)

. current directory (try with cd). Also filenames that start with "." are "hidden".

.. parent directory (try with cd)

> direct the output of a process to a file

$ variable expansion

| pipe, connects the output from one command to another

; command separator

/ directory separator (compare to Windows!)

**** escape a character, so that it is no longer a "special character"

& send a process to run in the background

= variable assignment

* wildcard for multiple characters (try with ls)

? wildcard for single character (try with ls)

comment (place it before a command and see what happens)

Advanced programs

These are more complicated utilities that we will be needing quite frequently:

scp – secure copy. We will use this to move files between our remote server (de.NBI) and our local server.

ssh – secure shell. This opens an encrypted terminal, usually on a remote machine, so you can use that computer as if it were right in front of you.

wget - a file downloader, using standard file transfer protocols.

conda – a software package and environment program that is used to handle the complex installation environments necessary for our kind of work. Conda should activate immediately when you log in to your de.NBI virtual machine. We will learn about the intricacies of Conda as we go.

We will use numerous other programs and features of BASH. I will try to explain them as we use them. Use all the above to explore the terminal, and ask me questions as you wander. Be independent: remember that your computing environment will not look exactly like mine, you will have your own directory names, etc. Part of scientific computing is learning to adapt other's scripts to your own setup!

Finding Windows and Ubuntu filetree systems

In your exploration, you may notice that the directories we have been perusing don't look anything like the folders you see when you look around with the windows file explorer. That's because Windows and Ubuntu Linux do not want to mix their filesystems. You are probably currently wandering around the inside of the Linux filesystem tree. For windows users, now that you have learned how to get around in a linux filesystem, we need to find your old windows file system tree. For most WSL users, while in Linux, it should be visible here:

cd /mnt/c/Users/

From there, can you find your way to some files and folders that you recognize from your Windows environment?

For extra credit, can you reverse this? As in, can you find your Ubuntu file system with the Windows file Explorer program? (it's tricky, [see here](#)).

Note – be very careful when using Windows-based programs to edit anything that you want to keep in your Ubuntu environment. Modifying - and sometimes just reading - a Linux-created file with Windows can sometimes corrupt the file for further use by Linux tools. The opposite is usually fine, Linux respects the common file formats of Windows OS. However, to be safe, we will tend to use the following one-way information flow:

| de.NBI virtual machine | → | Local linux environment | → | Windows | → | Browsers, Excel, other GUIs… |
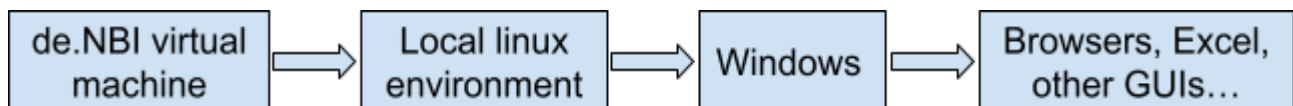
Remote computing

Now that your local Linux environment is working, let's use it to talk to your de.NBI virtual machine. We will use SSH and SCP to communicate with our de.NBI machines. To do this, you need to create an SSH keypair.

SSH Key generation

Secure login systems usually rely on an asymmetric, public/private keypair scheme. In such systems, you as a user create both a public key that you can give to servers, and also a complementary private key that only you control. When logging into a server that has your public key, you can prove your identity by providing the only key in the world that "fits" the public key, your very own private key.

We need you to generate your keyset, and following this we need to add your public key to your de.NBI virtual machine. To do this, in your local terminal generate a key pair, we'll use ssh-keygen:

```
ssh-keygen -t rsa -f <nameOfYourKeyHere>
```

-t rsa tells the algorithm which type of key generation algorithm to use.
-f is the name you want to give your key file

You will probably be asked to generate a password associated with your key. Keep this password, you will need it often!

This process will generate two keys. The public key should have a ".pub" extension, and can be shared freely. The private key will not have a file extension – it should be exactly as you named it in the command above. If you are not sure, look at the file with **less** or **cat** or **head**, and it should say -----BEGIN OPENSSH PRIVATE KEY----- in the first line.

Put both keys in your .ssh folder, which should be in your home directory (use "**cd ~**" or just "**cd**" to go to your home directory). If the .ssh folder doesn't exist, make it. Guard the private key with your life, and send me a copy of the public one in an email, with your name clearly in the email somewhere.

Logging into your de.NBI virtual machine

Once we have placed your public key into your assigned de.NBI machine, it should be ready to accept your login.

To get the login for your machine, you will use SSH. The correct SSH login command with be different for each of you, because your virtual machine has its own unique IP address, and port number through which it is accessed. To get your login command exactly right, log into the de.NBI cloud (https://cloud.denbi.de/portal). Using the left panel to navigate, look at Overviews --> Instances. The information to log in to your machine is revealed by clicking on "How to Connect", revealing an SSH command customized to your virtual machine:
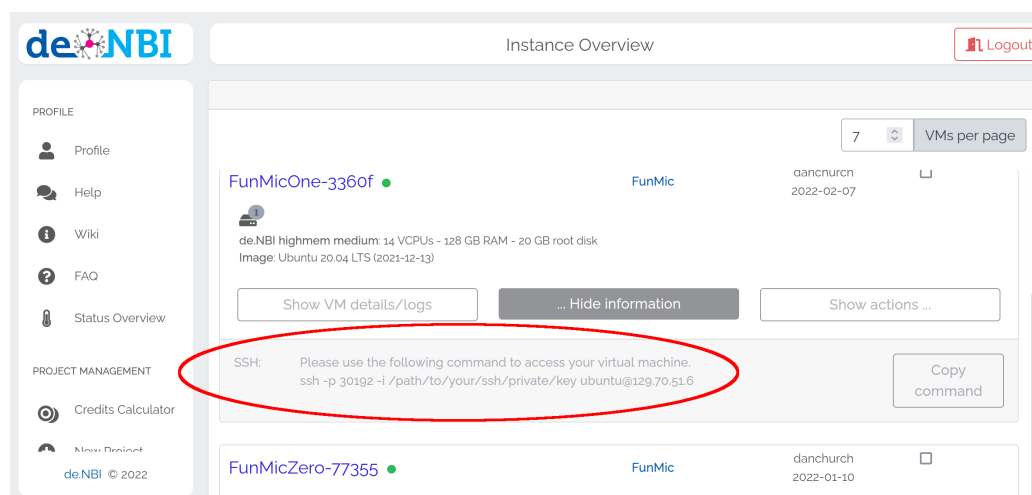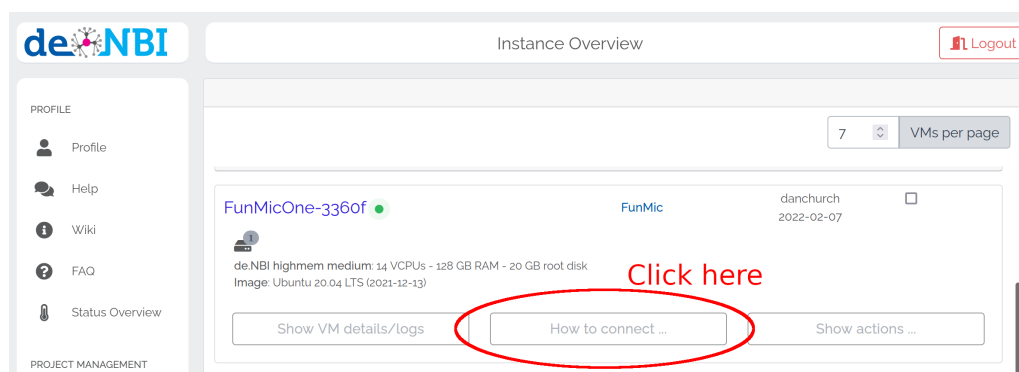
In my case, this tells me to use the following command:

> **ssh -p 30192 -i </path/to/your/ssh/private/key> ubuntu@129.70.51.6**

-p is the port number. Use only this port to interact with on your virtual machine.
-i is where you have stored your private ssh key. If you followed the instructions above, this will be in your .ssh folder in your home directory.

For me, the full command therefore looks like this:

> **ssh -p 30192 -i /home/daniel/.ssh/privatekey ubuntu@129.70.51.6**

If all keys are in place, this should allow you to log into your de.NBI virtual machine. Go ahead and explore it in just the same way you explored your local Linux filetree above, using your new knowledge of BASH commands!

File transfer using SCP:

Think of SCP as a long-distance version of the BASH command "cp", which we learned about above. Both of these commands copy a file from location A to location B. SCP requires a lot of the same information as SSH to do its job (port number, ip address).

To upload a filed called **localFile.txt** that is on the current directory on my office computer, to the directory **farAwayDirectory/** on my de.NBI machine, I do the following:

> **scp -r -i <...> -P <...> localFile.txt ubuntu@129.70.51.6:/farAwayDirectory/**

> **-i** is the path to my private key.
> **-P** is the port on the de.NBI machine (note the **capital -P**. ssh uses a lowercase **-p** flag).
> **-r** (recursive) is not necessary with single files, but it is essential if you want to download an entire folder.

The reverse is also possible. To download **farAwayFile.txt** from my de.NBI virtual machine, to the current directory (**.**) in my office computer, I do the following:

> **scp -i <….> -P <...> ubuntu@129.70.51.6:/farAwayDirectory/farAwayFile.txt .**

Note the dot at the end of the command, indicating the current directory on your local computer.

Now try some file transfers between your local computer and your de.NBI virtual machine!

Topic 2: Metagenomic methods

When direct sequencing of environmental DNA and RNA became possible, scientists developed two broadly-defined DNA-based methods for exploring the diversity in a given biological sample: metagenomic and metabarcoding (="amplicon sequencing"). We've talked at length about both in lecture, so we won't dive into intricacies here. Briefly, however, metagenomics involves randomly fragmenting all of the DNA available in a sample, and sequencing as many of these DNA fragments as possible, hopefully without bias. This renders millions of reads that can be considered a statistical sample of the genetic material of all the organisms present in the biological sample, i.e. the metagenome of the sample.
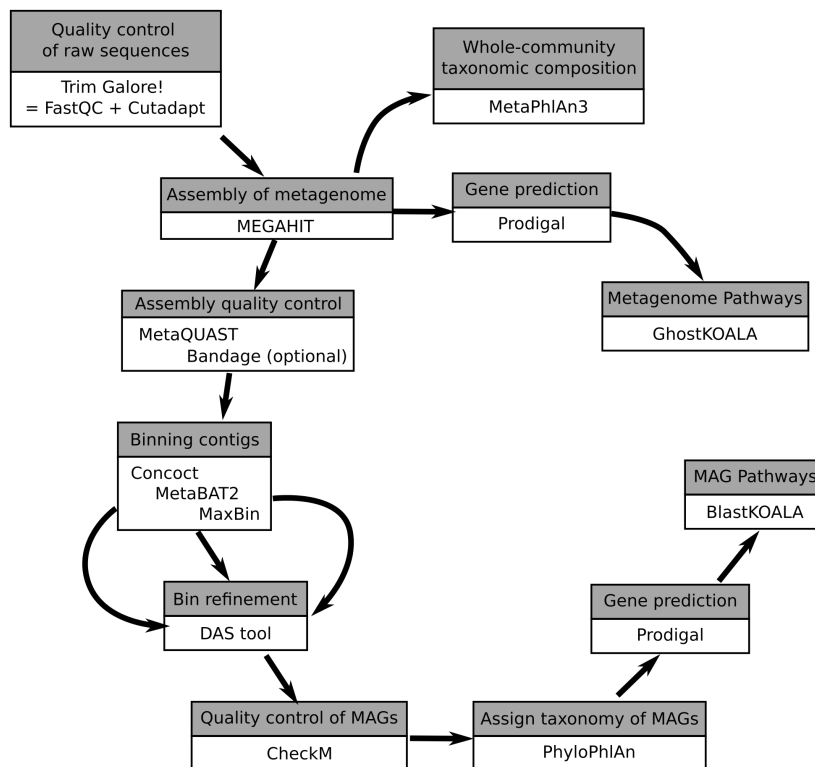
This sequenced sample of the metagenome can be handled in two ways...

Option 1 is to treat a metagenome like a single genome, and observe the metabolic pathways that are present in this "super" genome. We often don't need to know exactly which organisms are responsible for the presence of a metabolic pathway in which we are interested, and often numerous organisms are responsible anyway, due to horizontal gene flow and other forms of ecological redundancy. From this method we can quickly mine our biological samples (soil, water, plant tissue, etc) for interesting genes and pathways such as antibiotic resistance efflux pumps, etc.

Option 2 is to try to recover individual genomes from the soup of our metagenome reads. This approach can give you some very-fine-grain information about the taxonomy and function of the most abundant microbes in your sample. Quality **metagenome-assembled-genomes (MAGs)** are also rapidly increasing our ability to resolve microbial dark matter (see for example Nayfach et al. (2021)). However, this approach requires quite deep sequencing, meaning fewer samples per sequencer run, and more computational steps and computing resources needed. Additionally, very few genomes are typically recovered, especially in very microbially diverse samples or in unusual substrates, and especially without sufficient sequencing depth to allow for de-novo methods of assembly.

Both approaches can yield interesting information about a microbiome, so we will explore both. We will first explore our MAG-creation pipeline with a well-known publicly-available mock community dataset, MBARC-26. We will then repeat with a "wild" microbiome, using publicly-available data from a study of wastewater treatment influent and effluent, by Chu et al. (2018).

The exact steps of our analysis will be in the code that we produce each day. Here is an overall schematic of the approach, with the particular software packages we will use.

Quality control
of raw sequences

Trim Galore!
= FastQC + Cutadapt

Whole-community
taxonomic composition

MetaPhlAn3

Assembly of metagenome

MEGAHIT

Gene prediction

Prodigal

Assembly quality control

MetaQUAST
Bandage (optional)

Metagenome Pathways

GhostKOALA

Binning contigs

Concoct
MetaBAT2
MaxBin

Bin refinement

DAS tool

MAG Pathways

BlastKOALA

Gene prediction

Prodigal

Quality control of MAGs

CheckM

Assign taxonomy of MAGs

PhyloPhlAn

Topic 3: Metabarcoding ("amplicon sequencing") methods

In contrast to metagenomics, metabarcoding uses selective PCR or bioinformatic methods to target only a particular loci of the genome for all of a group of organisms in a biological sample. If taxonomy of microbes (e.g. prokaryotes and fungi) is of interest, ribosomal genes are typically targeted. These genes are useful barcodes because they are theoretically present in all life, in multiple copies, and differences in their sequences can often predict evolutionary relationships (taxonomy). By targeting these genes, we can draw a general picture of who is present in our sample, and perhaps even relative abundances (careful!!!!). We also begin to model changes in the microbial community structure by ecological predictors such as pH or moisture, if we have this data. In the case of prokaryotes, the 16s small subunit of the rRNA is commonly used. For your soil study with Dr. Lüders, you will be using primers that target the variable region 4 (V4).

Generally, metabarcoding studies do not require the same level of deep-sequencing that metagenome studies require. Many more samples can sequence in a single sequencer run, sometimes as few as several thousand reads is sufficient to saturate diversity curves for a single sample. As such, they are often a good "first step" to understanding your study system.

Other genes are possible, if your question is not taxonomy but function. For example, *Nif* genes can be used as a marker for nitrogen fixation, or secondary metabolic gene clusters can be sampled using different backbone synthase genes.

As above, the exact steps of our analysis will be in the code that we produce each day. Here is an overall schematic of the approach, many of which we will execute within the Qiime2 pipeline:

Quality control of raw sequences
Trim Galore! = FastQC + Cutadapt

Denoise sequencer error
DADA2

Build phylogenetic tree
MAFFT

Cluster or Agglomerate OTUs (optional)
Qiime2 (VSEARCH) or Phyloseq

Calculate diversity and dissimilarity
QIIME2 or VEGAN (R)

Build ecological models
vegan