ICME 13 - Rome Edition

Crash course on Nanopore sequencing for microbial ecology

Manuela Coci Luigi Gallucci Davide Corso

Last update: 2024-03-14

Contents

Chapter 1

Introduction

MicrobEco is a non-profit scientific organization dedicated to the advancement of scientific research and know-how in the field of microbiology and ecology. MICROBECO's mission is to promote the understanding and appreciation of microbes, to disseminate knowledge, strengthen collaboration, create and provide opportunities to learn about, discuss and challenge frontier issues in microbial ecology and to support the development of new technologies and applications that can benefit society. We connect scientists and we educate future generations to know and to do not fear microbes.

1.1 ICME 13 - Rome

The course "Crash course on Nanopore sequencing for microbial ecology" corresponds to the 13th edition of the ICME- International course in microbial ecology, which annually gathers selected students to receive theoretical and practical training on one or more techniques in microbial ecology. From 11 to 14 March 2024, 25 PhD students and early career researchers will have practical experience in sequencing using Nanopore technology (MinIon) and performing bioinformatic analysis of Nanopore sequencing data. As in the tradition of ICME, The entire course is designed to bring together beginners and experts, and to create a strong collaboration between colleagues that extends beyond the days of the course.

The course is held at the laboratory of CNR-IRSA Institute of Water Research Roma Montelibretti.

Official page: https://www.microbeco.org/icme13-roma-2024/

Chapter 2

Introduction to the command line

2.1 Set-up a terminal

MacOS/Linux: Launch terminal on your machine.

Windows users options: Windows Subsystem for Linux (WSL) –> It creates an Ubuntu terminal environment where you can code just like from a linux Ubuntu terminal. This is useful for the course as well as for practice working in bash. from ubuntu website and from the windows website

SSH client –> Windows: [MobaXterm](https://mobaxterm.mobatek.net/download-home-edition.html . This is a very basic ssh client, meaning, it will allow you to connect to the server and it will serve as a terminal for the course.

If you are already using Visual Studio, it needs one ssh extension plugin to serve as a ssh.

Git for windows –> I am not sure this can be used as a ssh but, in regards to this course, it is also useful to practice coding on the terminal.

Very last-minute resource –> launch this terminal emulator in a new window.

2.2 Working with the command line

Most of the activities of the bioinformatic section of this workshop will be done using the Unix command line (Unix shell).
It is therefore highly recommended to have at least a basic grasp of how to get around in the Unix shell.

We will now dedicate one hour or so to follow some basic to learn (or refresh) the basics of the Unix shell.

[!question] What is the UNIX SHELL? What is Bash?

> [!todo] The shell is a program that enables us to send commands to the computer and receive output. It is also referred to as the terminal or command line. Some computers include a default Unix Shell program.
>
> [!todo] The most popular Unix shell is Bash, Bash is a shell and a command language.

For a **Mac** computer running macOS Mojave or earlier releases, the default Unix Shell is Bash.

For a **Mac** computer running macOS Catalina or later releases, the default Unix Shell is Zsh. Your default shell is available via the Terminal program within your Utilities folder.

The default Unix Shell for **Linux** operating systems is usually Bash.

2.3 Playing around with basic UNIX commands

2.3.1 Some notes!

These commands:

```
mkdir unix_shell
cd unix_shell
```

…are commands you need to type in the shell. Each line is a command. Commands have to be typed in a single line, one at a time. After each command, hit "Enter" to execute it.

Things starting with a hashtag:

```
# This is a comment and is ignored by the shell
```

…are comments embedded in the code to give instructions to the user. Anything in a line starting with a `#` is ignored by the shell.

Different commands might expect different syntaxes and different types of arguments. Some times the order matters, some times it doesn't! The best way to check how to run a command is by taking a look at its manual with the command `man` or to the –help for a shorter version of it:

```
man mkdir

# You can scroll down by hitting the space bar
# To quit, hit "q"
```

```
mkdir -h

# did it work?
```

2.3.2 Creating and navigating directories

First let's see where we are:

```
pwd  # print working directory
```

Are there any files here? Let's list the contents of the folder:

```
ls

# or

ll
```

Let's now create a new folder called `unix_shell`. In addition to the command (`mkdir`), we are now passing a term (also known as an argument) which, in this case, is the name of the folder we want to create:

```
mkdir unix_shell
```

Has anything changed? How to list the contents of the folder again?

HINT (CLICK TO EXPAND)

> ls

And now let's enter the `unix_shell` folder:

```
cd unix_shell
```

Did it work? Where are we now?

HINT

> pwd

2.3.3 Creating a new file

Let's create a new file called `myfile.txt` by launching the text editor `nano`:

```
nano myfile.txt
```

Now inside the nano screen:

1. Write some text
2. Exit the "writing mode" with ctrl+x nano

3. To save the file, type **y** and hit "Enter"
4. Confirm the name of the file and hit "Enter"

List the contents of the folder. Can you see the file we have just created?

2.3.4 Copying, renaming, moving and deleting files

First let's create a new folder called `myfolder`. Do you remember how to do this?

HINT

> mkdir myfolder

And now let's make a copy of `myfile.txt`. Here, the command `cp` expects two arguments, and the order of these arguments matter. The first is the name of the file we want to copy, and the second is the name of the new file:

```
cp myfile.txt newfile.txt
```

List the contents of the folder. Do you see the new file there?

Now let's say we want to copy a file and put it inside a folder. In this case, we give the name of the folder as the second argument to `cp`:

```
cp myfile.txt myfolder

# while typing myfold.. try using the TAB to predict the name of the folder!

cp myfile.txt myfolder/  # it will recognise it is a directory and add the / at the en
```

List the contents of `myfolder`. Is `myfile.txt` there?

```
ls myfolder
```

We can also copy the file to another folder and give it a different name, like this:

```
cp myfile.txt myfolder/copy_of_myfile.txt
```

List the contents of `myfolder` again. Do you see two files there?

Instead of copying, we can move files around with the command `mv`:

```
mv newfile.txt myfolder
```

Let's list the contents of the folders. Where did `newfile.txt` go?

We can also use the command `mv` to rename files:

```
mv myfile.txt myfile_renamed.txt
```

List the contents of the folder again. What happened to `myfile.txt`?

Now, let's say we want to move things from inside `myfolder` to the current directory. Can you see what the dot (`.`) is doing in the command below? Let's try:

```
mv myfolder/newfile.txt .
```

Let's list the contents of the folders. The file `newfile.txt` was inside `myfolder` before, where is it now?

The same operation can be done in a different way. In the commands below, can you see what the two dots (`.`) are doing? Let's try:

```
# First we go inside the folder
cd myfolder

# Then we move the file one level up
mv myfile.txt ..

# And then we go back one level
cd ..
```

Let's list the contents of the folders. The file `myfile.txt` was inside `myfolder` before, where is it now?

To remove files :

```
rm newfile.txt
```

Let's list the contents of the folder. What happened to `newfile.txt`?

And now let's delete `myfolder`:

```
rm myfolder
```

It didn't work did it? An error message came up, what does it mean?

```
rm: cannot remove ‘myfolder’: Is a directory
```

To delete a folder we have to modify the command further by adding the flag (`-r`). Flags are used to pass additional options to the commands:

```
rm -r myfolder
```

Let's list the contents of the folder. What happened to `myfolder`?

> [!warning] **In Bash, If you remove the wrong file/directory, it is gone forever!! (no recycle bin!) aka BE CAREFUL!!**

Chapter 3

16s Analysis

In this tutorial we are going to see/use different commands and tools to perform the 16s analysis on long reads NanoPore. However, the practical analysis will cover only some of these steps as data has been prepared in advance due to their computation and time consuming limits.

This guide has been created with the purpose of a practical crush course and it is not intended as a complete reference but rather as a beginners pipeline to analyze NanoPore generated data.

3.1 Commands and features that we will use in our practice

3.1.1 `touch`

Create a new empty file.

Example: `touch prova.txt`

3.1.2 `echo`

The echo command is a built-in Linux feature that prints out arguments as the standard output. echo is commonly used *to display text strings or command results as messages*.

Example: `echo "Hello World"`

3.1.3 `cat`

Concatenate or print the contents of a file

Example: `cat prova.txt`

3.1.4 $

Append $ to the variable name to access the variable value.

Example:

```
var="Hello World"
echo $var
```

Output:

> Hello World

3.1.5 chmod

Change the permissions of a file or directory and make it executable.

- use the "chmod +x" command on a system file to give permission to all users to execute it.
- use the "chmod u+x" for made the file executable for your user.

Example: `chmod u+x s01_filtering.sh`

3.1.6 basename

It removes the path from a file string, providing only its filename and trailing suffix from given file names.

Example:

```
basename /path/to/filename.txt
```

Output:

> filename.txt

3.1.7 Create a shell scripts

Sometime you don't need to run a command at a time, we can pre-think, organize series of actions (a program) that you can then execute within Bash.

For example we can write a shell script that runs a series of commands and we can run the script from the terminal to execute all the steps that we have integrated in the script.

For example, lets assume that we want to visualize the first four reads from a FASTQ and redirect to a file. For this task we are going to create an empty file and write in it the following text:

```
#!/bin/bash

cat /SERVER/16s_data/mysample.FASTQ | head -n 4 > first_read.fastq
```

```
var="Hello World"
echo $var
```

Now we give the 'executable' permission to our script in order to be executed:

`chmod u+x myscript.sh`

Now we execute our script:

`./myscript.sh`

- The first row must be `#!/bin/bash` to allows the shell to interpret your code with bash
- The symbol `|` is the PIPE, it lets you connect actions: the output of a command is the input of the next command
- The command `head` allows to print a specified number of rows from an input.
- Sometime different actions cannot be linked with a PIPE, in this case we use to write the series of actions in each line, as we did in the last two rows of our scripts.

3.2 Package Manager

3.2.1 Conda

Conda is a powerful command line tool for package and environment management that runs on Windows, macOS, and Linux.

https://conda.io/projects/conda/en/latest/user-guide/getting-started.html

Within conda, you can create, export, list, remove, and update environments that have different versions of Python and/or packages installed in them. *Switching or moving between environments is called activating the environment.* You can also share an environment file.

3.2.2 Mamba (Recommended)

Mamba is a reimplementation of the conda package manager in C++, so it is faster and more convenient due to its faster dependencies solving.

https://mamba.readthedocs.io/en/latest/user_guide/mamba.html

3.2.2.1 Creating a conda/mamba env

To create an environment:

```
conda create -n myenv
```

To activate a created environment:

```
conda activate myenv
```

3.2.2.2 Installing tools

Most of the bioinformatics packages can be searched in Bioconda at this link https://bioconda.github.io, but also in `conda-forge` channel and then we can install them by executing

```
mamba install -c bioconda -c conda-forge nanofilt
```

For our practical analysis, conda environments have already been created.

The following environment were created:

- `mamba activate /home/irsa/miniconda3/envs/ONTpp`
- `mamba activate /home/irsa/miniconda3/envs/emu`

3.3 Base Calling

Base calling is the process of translating the electronic raw signal of the sequencer into bases, i.e., ATCG and converting the raw files (FAST5) to a FASTQ files (human-readable), which contains the nucleotide sequences of the reads.

Raw data are huge in terms of storage, and since basecalling is computationally and time demanding, the fastq files are already provided.

However, to perform this step we suggest to use one of the two following tools:

3.3.1 Guppy

https://community.nanoporetech.com/docs/prepare/library_prep_protocols/Guppy-protocol/v/gpb_2003_v1_revax_14dec2018/guppy-software-overview

Guppy usage:

```
guppy_basecaller \
  --num_callers 4 \
  --cpu_threads_per_caller 64 \
  --input_path \
  --save_path \
  --flowcell FLO-MIN106 \
  --kit SQK-RBK004s
```

As we can see from the command line this tool requires the flowcell model and eventually barcoding kit information.

3.3.2 Dorado

Recently released by Nanopore, Dorado is a high-performance, easy-to-use, open source basecaller for Oxford Nanopore reads, with the options of super, high and low, accuracy.

https://github.com/nanoporetech/dorado

First download the flowcell model kit (You need to know which flowcell was used):

```
dorado download --model dna_r10.4.1_e8.2_400bps_hac@v4.1.0
```

Dorado usage:

```
dorado basecaller \
  -b 36 \
  --device cpu \
  --emit-fastq
  mysample.FAST5
```

3.4 Filtering and Trimming

The fastq file containing the 16S sequence need to be filtered based on quality and/or read length, and optional trimmed after passing filter

3.4.1 NanoFilt

NanoFilt - filtering and trimming of long read sequencing data

https://github.com/wdecoster/nanofilt

Requirement

To execute this tool, you need to activate the conda environment

`mamba activate /home/irsa/miniconda3/envs/ONTpp`

STEP TIPS:

- Your input data are available in the following path: `/SERVER/16s_data/`
- Your task should be to use NanoFilt on each fastq file, using the following options:
  - `--length LENGTH` Filter on a minimum read length
  - `--maxlength MAXLENGTH` Filter on a maximum read length
  - `--quality QUALITY` Filter on a minimum average read quality score
- Create the folder for your output files (to use in your script)
- To perform this step you should create a script (Recommended name: `s01_nanofilt.sh`), and give it permission to be executed: `chmod u+x s01_nanofilt.sh`
- Execute your script as follow: `./s01_nanofilt.sh`

- Look at the results

[**SPOILER**] - Scripts that we will use

We create an empty file called `s01_nanofilt.sh`

```
touch s01_nanofilt.sh
```

We can write our actions in the scripts as follows:

```
#!/bin/bash

for i in /SERVER/16s_data/*fastq
do
    f=$(basename "$i" .fastq)
    echo "$f"
    echo "$i"
    cat $i | NanoFilt -q 9 -l 1200 --maxlength 1800 > /home/irsa/analisi_16s/output_s0
done
```

Create output directory for this script (Change `irsa` with your `utenteX` name)

```
mkdir -p /home/irsa/analisi_16s/output_s01/
```

Change its permission: `chmod u+x s01_nanofilt.sh`

Execute it: `./s01_nanofilt.sh`

3.5 Subsetting

We are going to reduce the number of reads in order to use less resources for our practical analysis. **It is important to note that this step is not part of a common pipeline**.

3.5.1 BBMAP Tools

BBMap - short read aligner for DNA/RNAseq, and other bioinformatic tools, including BBMap.

https://github.com/BioInfoTools/BBMap

From this step, we use a script provided by BBMAP tools collection, called: `reformat.sh`, which reformats reads to change ASCII quality encoding, interleaving, file format, or compression format.

Requirement

To execute this tool, you need to activate the conda environment:

```
mamba activate /home/irsa/miniconda3/envs/ONTpp
```

STEP TIPS:

- Your input data should be available from the output folder of the previous step
- Your task should be to use `reformat.sh` on **each output files obtained from the previous step**, using the following options:
  - `in=<file>` Input file
  - `out=<outfile>` Ouput file
  - `samplereadstarget=10000` Exact number of OUTPUT reads (or pairs) desired.
- Create the folder for your output files (to use in your script)
- To perform this step you should create a script (Recommended name: `s02_subsampling.sh`), and give it permission to be executed: `chmod u+x s02_subsampling.sh`
- Execute your script as follow: `./s02_subsampling.sh`
- Look at the results

[**SPOILER**] - Scripts that we will use

We create an empty file called `s02_subsampling.sh`

```
touch s02_subsampling.sh
```

We can write our actions in the scripts as follows:

```
#!/bin/bash

# Create output directory for this script
mkdir -p /home/irsa/analisi_16s/output_s02/

for i in /home/irsa/analisi_16s/output_s01/*-nf.fastq
do
    f=$(basename "$i" -nf.fastq)
    echo "$f"
    echo "$i"
    reformat.sh in="$i" out=/home/irsa/analisi_16s/output_s02/"$f"-ss-nf.fastq samplereadstarget=
done
```

Create output directory for this script (Change `irsa` with your `utenteX` name)

```
mkdir -p /home/irsa/analisi_16s/output_s02/
```

Change its permission: `chmod u+x s02_subsampling.sh`

3.6 Taxonomic Assignment

Last step, the sequences are compared to a reference database for taxonomic assignment and a relative abundance estimator for 16S genomic sequences.

3.6.1 EMU

Emu - species-level taxonomic abundance for full-length 16S reads.

This tool use a method optimized for error-prone full-length reads. However, it can be used for short-reads.

https://github.com/treangenlab/emu

To perform this annotation, we need a reference database that contains all the taxonomic information, which was already been downloaded in the following path: `/SERVER/emu_database`

Requirement

To execute this tool, you need to activate the conda environment:

```
mamba activate /home/irsa/miniconda3/envs/emu
```

STEP TIPS:

- Your input data should be available from the previous step
- Your task should be to use `emu abundance` on **each output files obtained from the previous step**, using the following options:
  - `--type map-ont` denote sequencer [short-read:sr, Pac-Bio:map-pb, ONT:map-ont]
  - `--keep-counts` include estimated read counts for each species in output
  - `--output-dir <output_dir>` directory for output results (to be created in advance)
  - `--output-basename <basename_files>` basename of all output files saved in output-dir; default utilizes basename from input file(s)
- Create the folder for your output files (to use in your script)
- To perform this step you should create a script (Recommended name: `s03_abundance.sh`), and give it permission to be executed: `chmod u+x s03_abundance.sh`
  - Specify the path of the EMU database with the following line as an action: `export EMU_DATABASE_DIR=/SERVER/emu_database`
  - Last line of your script, should be an action that performs the command `emu combine-outputs` to create a single table containing all Emu output relative abundances in a single directory. Note this function will select all the .tsv files in the provided directory that contain 'rel-abundance' in the filename. Use the following options in your `emu combine-outputs` command:
    - `--counts output` estimated counts rather than relative abundance percentage in combined table. Only includes Emu relative abundance outputs that already have 'estimated counts'
    - `tax_id` to get results for the most specific taxa level
- Execute your script as follow: `./s03_abundance.sh`
- Look at the results

[**SPOILER**] - Scripts that we will use

We create an empty file called `s03_abundance.sh`

`touch s03_abundance.sh`

We can write our actions in the scripts as follows:

```
#!/bin/bash

# mamba activate /home/irsa/miniconda3/envs/emu

export EMU_DATABASE_DIR=/SERVER/emu_database

for i in /home/irsa/analisi_16s/output_s02/*-ss-nf.fastq
do
    f=$(basename "$i" -ss-nf.fastq)
    echo "$f"
    echo "$i"
    emu abundance "$i" --type map-ont --output-basename "$f" --keep-counts --output-dir /home/irs
done


emu combine-outputs --counts /home/irsa/analisi_16s/output_s03/ tax_id
```

Create output directory for this script (Change `irsa` with your `utenteX` name)

`mkdir -p /home/irsa/analisi_16s/output_s03/`

Change its permission: `chmod u+x s03_abundance.sh`

Chapter 4

Metagenomics

The metagenomics sequence are produced using a shotgun approach in which each all the gene present inside the samples were sequenced. This sequencing not only extends taxonomic resolution to the species- or strain-level but also provides potential functional information

4.1 Reads Quality Check

After the basecalling, on the fastq file the "quality check" of the sequence can be performed using stats.sh, a tool inc luded in BBMap/BBTools

4.1.1 BBMap / `stats.sh`

`stats.sh` - Generates basic assembly statistics such as scaffold count, N50, L50, GC content, gap percent, etc. Works with fasta and fastq only (gzipped is fine).

https://github.com/BioInfoTools/BBMap/blob/master/sh/stats.sh

4.2 Assembly

One of the most important step for the metagenomic analysis is the Assembly, in which the reads were compared, aligned and overlapped in order to create longer sequenced called "CONTIGS".

4.2.1 Flye (Metaflye)

Setup Conda Env

```
mamba create -n flye -c conda-forge -c bioconda flye
```

Flye is a de novo assembler for single molecule sequencing reads using repeat graphs as core data structure. Compared to de Bruijn graphs (which require exact k-mer matches), repeat graphs are built using approximate sequence matches.

https://github.com/fenderglass/Flye

Options used

- `--nano-raw ONT` regular reads
- `--meta` enables the mode for metagenome/uneven coverage assembly
- `--out-dir <outputdir>` Output directory

4.3 Read mapping

To quantify the coverage, we map the original input reads to the contig

4.3.1 Minimap2

A versatile pairwise aligner for genomic and spliced nucleotide sequences, used for evaluate the coverage of the original reads on each previously created contig using pairwise alignment.

A report and two charts are generated with complementary information, showing a summary of the DNA-Seq Alignment results.

This page contains information about the reference genome sequences, the input FASTQ files, and a results overview.

The last section is divided into several subsections: globals, paired information, ACTG content, coverage, mapping quality, insert size, mismatches, and indels.

Minimap2 rates an alignment by the score of the max-scoring sub-segment, excluding introns, and marks the best alignment as primary in SAM.

Sequence Alignment Map (SAM) is a text-based format originally for storing biological sequences aligned to a reference sequence. Practically, SAM is a TAB-delimited text format consisting of a header section and an alignment section. Header lines start with '@', while alignment lines do not.

https://github.com/lh3/minimap2

Example command line

```
minimap2 -ax map-ont ref.fa ont-reads.fq > aln.sam
```

Options: - `map-ont` Align noisy long reads of ~10% error rate to a reference genome. This is the default mode

4.3.2 samtools

Samtools is a package for reading/writing/editing/indexing/viewing SAM/BAM/CRAM format.

A BAM file (*.bam) is a compressed binary version (BGZF format) of a SAM file that is used to represent aligned sequences. This file can be created starting from a SAM (using `samtools`) file in order to reduce its size.

```
samtools sort aln.sam -o aln.bam
```

4.4 Binning

In metagenomics, binning is the process of grouping reads or contigs and assigning them to individual genome, called "MAGs" (Metagenome Assembled Genomes).

4.4.1 SemiBin2

Setup Conda Env

```
mamba create -n semibin -c conda-forge -c bioconda semibin
```

SemiBin is a command line tool for metagenomic binning with semi-supervised siamese neural network using additional information from reference genomes and contigs themselves. It supports single sample, co-assembly, and multi-samples binning modes.

https://github.com/BigDataBiology/SemiBin

3 Options are available for this tool:

- `single_easy_bin`: Running with single-sample binning
- `multi_easy_bin`: Running with multi-sample binning
- `co-assembly`: samples are co-assembled first (as if the pool of samples were a single sample) and then bins are constructed from this pool of co-assembled contigs.

You will need the following inputs:

- A contig file (contig.fa in the example below)
- BAM file(s) from mapping short reads to the contigs, sorted (mapped_reads.sorted.bam in the example below)

The `single_easy_bin` command can be used to produce results in a single step

```
SemiBin2 single_easy_bin --sequencing-type=long_read --input-fasta assembly.fasta --input-bam alr
```

Alternatively, you can train a new model for that sample, by not passing in the `--environment` flag.

This is the fastest option and should work the best if you have metagenomes from one of our prebuilt habitats (alternatively, you can use the global "habitat" which combines all of them).

```
SemiBin2 single_easy_bin --sequencing-type=long_read --environment human_gut --input-fa
```

4.5 MAGs Quality Check

As well as the original reads and contigs, also MAGs can be checked to extract the ones that can be considered High Quality.

4.5.1 CheckM2

CheckM2 - Rapid assessment of genome bin quality using machine learning

https://github.com/chklovski/CheckM2

CheckM2 has universally trained machine learning models it applies regardless of taxonomic lineage to predict the completeness and contamination of genomic bins. Completeness is the percentage of the mapped genome that were covered by each mag. Contamination is the inclusion of foreign sequences on the mags

The strain heterogeneity (SH) index indicates the proportion of the contamination that appears to be from the same or similar strains (as determined with an AAI threshold).

In order to extract the completeness and contamination for each Mags, You will also need to download and install the external DIAMOND database that CheckM2 relies on for rapid annotation.

Setup Conda Env and install CheckM2 Database

```
mamba create -n checkm2 -c bioconda -c conda-forge checkm2
mamba activate checkm2

pip install CheckM2

mkdir -p /path/to/checkm2_database
checkm2 database --download --path /path/to/checkm2_database/
```

As we can see in the following command, the database path can also be set by using the environmental variable.

The main use of CheckM2 is to predict the completeness and contamination of metagenome-assembled genomes (MAGs)

```
export CHECKM2DB="/path/to/checkm2_database/"

checkm2 predict --threads 64 --input /path/output/output_bins/ --output-directory outpu
```

4.6 Taxonomic Classification

4.6.1 GTDB-Tk

4.6.1.1 Setup Conda Env

```
mamba create -n gtdbtk -c conda-forge -c bioconda gtdbtk=2.3.2
```

GTDB-tk - assigning taxonomic classifications

https://github.com/Ecogenomics/GTDBTk

GTDB-Tk is the software toolkit used for assigning objective taxonomic classifications to bacterial and archaeal genomes based on the Genome Database Taxonomy (GTDB). It is designed to work with recent advances that allow hundreds or thousands of metagenome-assembled genomes (MAGs) to be obtained directly from environmental samples.

GTDB-Tk requires an external database that needs to be downloaded and unarchived:

https://ecogenomics.github.io/GTDBTk/installing/index.html#gtdb-tk-reference-data

To perform the taxonomic classification, we use the workflow function `classify_wf` which consists (internally) of four steps: *ani_screen*, *identify*, *align*, and *classify*

```
gtdbtk classify_wf --genome_dir selected_genomes/ --out_dir classify_wf_out --extension fa --forc
```

Options used:

- `--genome_dir <directory>` directory containing genome files in FASTA format
- `--out_dir <directory>` directory to output files
- `--extension fa` extension of files to process, gz = gzipped
- `--force` continue processing if an error occurs on a single genome
- `--skip_ani_screen` Skip the ani_screening step to classify genomes using mash and FastANI
- `--pplacer_cpus 32` number of CPUs to use during pplacer placement

List of output files:

- summary.tsv: Classifications for bacterial and archaeal genomes (see the GTDB-Tk documentation for details). Here we will find:
  - *fastani_reference*: indicates the accession number of the reference genome (species) to which a user genome was assigned based on ANI and AF. ANI values are only calculated when a query genome is placed within a defined genus and are evaluated for all reference genomes in that genus.

  - *fastani_reference_radius*: indicates the species-specific ANI circumscription radius of the reference genomes used to determine if a query genome should be classified to the same species as the reference. https://ecogenomics.github.io/GTDBTk/files/summary.tsv.html
  - *fastani_af*: indicates the alignment fraction (AF) between the query and above reference genome.
  - *closest_placement_reference*: indicates the accession number of the reference genome when a genome is placed on a terminal branch.
  - *classification_method*: indicates the rule used to classify the genome.
- *classify.tree.gz*: Reference tree in Newick format containing query genomes placed with pplacer.
- *markers_summary.tsv*: A summary of unique, duplicated, and missing markers within the 120 bacterial marker set, or the 122 archaeal marker set for each submitted genome.
- *msa.fasta.gz*: FASTA file containing MSA of submitted and reference genomes.
- *filtered.tsv*: A list of genomes with an insufficient number of amino acids in MSA.
- *log*: Log files.
- *failed_genomes.tsv*: A list of genomes for which the GTDB-Tk analysis failed, e.g. because Prodigal could not detect any genes.
- *gtdbtk_summary.tsv*: A summary table of the GTDB-Tk classification results for all bins

The taxonomic classification of each bacterial and archaeal genome is contained in the `[prefix].[domain].summary.tsv` output files.

A strain identifier is used as a placeholder for the genus name when there is no existing genus name and no binomially named representative genome. This placeholder genus name is generally derived from the oldest representative genome within the lineage and formed from NCBI organism name or NCBI infraspecific/strain ID.

4.7 Functional Annotation

4.7.1 Anvi'o

For the final and detailed analysis of the MAGs (both from short and long reads) several specific tool were developed in the last years (i.e. is Anvi'o).

Anvi'o is a comprehensive platform that brings together many aspects of today's cutting-edge computational strategies of data-enabled microbiology, including genomics, metagenomics, metatranscriptomics, pangenomics, metapangenomics, phylogenomics, and microbial population genetics in an integrated and easy-to-use fashion through extensive interactive visualization capabilities.

https://anvio.org/

https://github.com/merenlab/anvio

Setup anvio and download databases

```
mamba create -y --name anvio-8 python=3.10
mamba activate anvio-8
mamba install -y -c conda-forge -c bioconda python=3.10 \
        sqlite prodigal idba mcl muscle=3.8.1551 famsa hmmer diamond \
        blast megahit spades bowtie2 bwa graphviz "samtools>=1.9" \
        trimal iqtree trnascan-se fasttree vmatch r-base r-tidyverse \
        r-optparse r-stringi r-magrittr bioconductor-qvalue meme ghostscript

mamba install -y -c bioconda fastani

curl -L https://github.com/merenlab/anvio/releases/download/v8/anvio-8.tar.gz \
        --output anvio-8.tar.gz

sudo apt install build-essential
pip install anvio-8.tar.gz


mkdir -p /SERVER/anvio-db/pfam /SERVER/anvio-db/kegg /SERVER/anvio-db/cazy /SERVER/anvio-db/scg

anvi-setup-kegg-data --mode KOfam --only-download --kegg-data-dir /SERVER/anvio-db/kegg/
anvi-setup-pfams --pfam-data-dir /SERVER/anvio-db/pfam/
anvi-setup-cazymes --cazyme-data-dir /SERVER/anvio-db/cazy/
anvi-setup-scg-taxonomy --scgs-taxonomy-data-dir /SERVER/anvio-db/scg/ --reset
```

the following workflow is useful for converting a bunch of genomes into an anvi'o-compatible format. It generates contigs databases from each input FASTA file, and subsequently runs a variety of annotation programs of your choice to populate these databases with some useful information for your downstream work (i.e. functions, single-copy-core genes, taxonomy, etc).

To start things going with this workflow, first ask anvi'o to give you a default workflow-config file for the contigs workflow:

```
anvi-run-workflow -w contigs --get-default-config contigs.json
```

Used options: - `-w contigs` select the different type of "workflow (i.e. contig, metagenome ...) - `--get-default-config contigs.json` generate a default config file (.json) for a given workflow

https://anvio.org/help/main/workflows/contigs/

Here we create a `file_list.tsv`

```
echo -e "name\tpath" > file_list.tsv && find path/to/selected_genomes -type f -exec sh -c 'echo -
```

If everything looks alright, you can run this workflow the following way from

the same folder of file_list.tsv:

```
anvi-run-workflow -w contigs -c contigs.json --additional-params --jobs 3
```

Finally, perform the annotation using one of the pre-loaded database

```
for i in 02_CONTIGS/*.db; # 02_CONTIGS viene generata da anvi-run-workflow
do
    echo "$i";
    anvi-run-cazymes -c $i --cazyme-data-dir path/to/CAZYdb/;
done
```

and export the annotation

```
for i in 02_CONTIGS/*.db; # 02_CONTIGS viene generata da anvi-run-workflow
do
    echo "$i";
    anvi-export-functions -c $i
done
```

Chapter 5

Our Metagenomic practice

In our practice we will cover the functional annotation on metagenomics data, both on assembly based and MAGs based. Due to resources and time limits we will try to run Prodigal, HMMer and Quast.

For the assembly-based analysis, we have the assembly fasta file located in the following path: `/SERVER/mg_data/mg/Assembly_metaflye/assembly.fasta`

5.1 ORF Prediction

Fast, reliable protein-coding gene prediction for prokaryotic genomes.

https://github.com/hyattpd/Prodigal

5.1.1 Prodigal

Setup the conda env

```
mamba create -n prodigal -c bioconda -c conda-forge prodigal
```

Tips:

- Input: Prodigal run using a fasta file, for example the one represented the assembly
- Output: You should obtains a GFF file and an Aminoacidic Fasta file of the predicted orfs

[**SPOILER**] - Scripts that we will use

We create an empty file called `s01_prodigal.sh`

```
touch s01_prodigal.sh
```

We can write our actions in the scripts as follows:

```
#!/bin/bash

assembly="/SERVER/mg_data/mg/Assembly_metaflye/assembly.fasta"


outfolder="output_s01"

mkdir -p $outfolder

prodigal -i ${assembly} \
    -o ${outfolder}/genes.gff \
    -a ${outfolder}/protein_translations.faa \
    -f gff \
    -p meta
```

Create output directory for this script (Change `irsa` with your `utenteX` name)

`mkdir -p /home/irsa/analisi_MG/output_s01/`

Change its permission: `chmod u+x s01_prodigal.sh`

Execute it: `./s01_prodigal.sh`

5.2 Hidden Markov Model

HMMER is a software package that provides tools for making probabilistic models of protein and DNA sequence domain families – called profile hidden Markov models, profile HMMs, or just profiles.

HMMER is used for searching sequence databases for sequence homologs, and for making sequence alignments. It implements methods using probabilistic models called profile hidden Markov models (profile HMMs).

https://github.com/EBI-Metagenomics/hmmer3

5.2.1 hmmer

Setup the conda env

```
mamba create -n hmmer -c bioconda -c conda-forge hmmer
```

5.2.1.1 Split PFAM files

Pfam is a comprehensive collection of protein domains and families, represented as multiple sequence alignments and as profile hidden Markov models.

Download the file `Pfam-A.hmm.gz` from the PFAM FTP Server: https://ftp.ebi.ac.uk/pub/databases/Pfam/current_release/

Step 0 De-Compress the file `Pfam-A.hmm.gz`. How will you do it? TIPS:

- Search on google

[**SPOILER**] - Scripts that we will use

```
gzip -d Pfam-A.hmm.gz
```

Step 1 Extract the name of the models in the PFAM file **(Advanced Task)**

[**SPOILER**] - Scripts that we will use

```
grep "^NAME" Pfam-A.hmm | awk '{print $2}' > all_names.txt
```

Step 2 Using the retrieved names, extract the corresponding models into multiple files.

Command to use: `hmmfetch` from the hmmer tools

[**SPOILER**] - Scripts that we will use

We create an empty file called `s02_splitPFAM.sh`

```
touch s02_splitPFAM.sh
```

We can write our actions in the scripts as follows:

```
#!/bin/bash

outfolder="output_s02"

while read name
do
    echo "$name"
    hmmfetch Pfam-A.hmm $name > "${outfolder}/$name.hmm"
done < all_names.txt
```

Create output directory for this script (Change `irsa` with your `utenteX` name)

```
mkdir -p /home/irsa/analisi_MG/output_s02/
```

Change its permission: `chmod u+x s02_splitPFAM.sh`

Execute it: `./s02_splitPFAM.sh`

As we can see, the run time of `hmmfetch` is very long. Try to execute the script only for 500 or 1000 extracted HMM names.

How can you do it?

5.2.1.2 `hmmsearch`

Now execute `hmmsearch` to the retrieved ORFs to annotate them based on the extracted models.

If you are working on the CNR system, use only 1 cpu.

[**SPOILER**] - Scripts that we will use

We create an empty file called `s03_hmmsearch.sh`

```
touch s03_hmmsearch.sh
```

We can write our actions in the scripts as follows:

```
#!/bin/bash


outfolder="output_s03"

while read name; do
    echo "$name"

    model_folder="${outfolder}/${name}"
    mkdir -p ${model_folder}

    model="output_s02/${name}.hmm"

    hmmsearch --tblout ${model_folder}/table.out -o ${model_folder}/align.out -E 0.000

done < 500_names.txt
```

Create output directory for this script (Change `irsa` with your `utenteX` name)

```
mkdir -p /home/irsa/analisi_MG/output_s03/
```

Change its permission: `chmod u+x s03_hmmsearch.sh`

Execute it: `./s03_hmmsearch.sh`

5.3 Assembly Quality Check

5.3.1 Quast

The QUAST package works both with and without reference genomes. However, it is much more informative if at least a close reference genome is provided along with the assemblies. The tool accepts multiple assemblies, thus is suitable for comparison.

https://github.com/ablab/quast

Setup the conda env

```
mamba create -n quast -c bioconda -c conda-forge quast
```

[**SPOILER**] - Scripts that we will use

We create an empty file called `s04_quast.sh`

```
touch s04_quast.sh
```

We can write our actions in the scripts as follows:

```
#!/bin/bash

outfolder="output_s04"


quast --labels flye --contig-thresholds 0,1000,10000,100000,1000000 --threads 2 -o ${outfolder} /
```

Create output directory for this script (Change `irsa` with your `utenteX` name)

```
mkdir -p /home/irsa/analisi_MG/output_s04/
```

Change its permission: `chmod u+x s04_quast.sh`

Execute it: `./s04_quast.sh`

What if you use `metaquast`?

[**SPOILER**] - Scripts that we will use

MetaQUAST the extension for metagenomic datasets, it evaluates and compares metagenome assemblies based on alignments to close references. It is based on QUAST genome quality assessment tool, but addresses features specific for metagenome datasets.

5.4 Web Tools Annotations

- KOFAM Koala https://www.genome.jp/tools/kofamkoala/
- EGGNOG http://eggnog-mapper.embl.de
- DBcan https://bcb.unl.edu/dbCAN2/blast.php

5.5 Functional Annotation - MAGs based

If you have finished, try to run the same analysis on each MAGs located in: `/SERVER/mg_data/mg/genomi_per_annotazione`