Exercise 3 – Nanopore metagenomes

**Day 4**

Today we will have a go at a developing technology, namely metagenomics using nanopore sequencing. Unfortunately, this method requires substantial computational resources, and hence, neither our data nor our results will be particularly realistic because we have to run it on a laptop. Still, I believe it to be important that you are exposed to emerging technologies – you are more than likely to work with this method in the future, especially if you aim for a PhD in the field.

Today, you will download a metagenomic dataset, clean it up, assemble it and annotate it, and you will have to figure out many of the details as you go. Luckily, you should be ready at this point, and many of the commands you will use are even the same as before!

1. **Running commands:**

I now expect you to be “proficient” on the command line, i.e. that you can navigate between folders, move files and install software with conda. I use the term ‘proficient’ very loosely, because we all learn differently, so make sure to contact the teachers if you have any questions on the fundamentals.

## Get the data:

First we have to get the data. The data in question is metagenomic sequences from a student of mine, Kristina, who got if from a marine sediment in Denmark. I have subsampled it to 1/10 to make it possible to work with. Still, it is too big to fit on github, so I have put it on Zenodo, and you will have to download it from there.

Before you do, though, consider the project hygiene we discussed before. This deserves a dedicated folder – maybe “Day4” or “NanoMeta” or something you can remember. Within that folder, you should have a folder for this “raw” data – consider “input” or “raw” or similar.

Download with this command into the appropriate folder.

wget <https://zenodo.org/records/12617316/files/all.tenth.gz>

This file is a FASTQ-file, but is compressed into a **gz**-file to save space. Most programs will accept compressed files (technically by uncompressing them on the fly, analyzing them and then re-compressing the results).

## Prepare an environment for this analysis:

Now that we know that environments should usually be dedicated to specific purposes, we can start building one specifically for analyzing nanopore metagegenomes. All the packages we will use play well together, so we will stick with a single environment for most of this analysis. Make one and enter it (maybe ‘nanometa’ or similar would be a good name). As always, we use **conda**!

## Clean up the data

The data is raw nanopore data, and hence has **adaptors** and **low quality** reads. Both have to be removed. First, lets have a look at the raw data.

1. Install the program ‘seqkit’ and run the stats module on the data
   1. **seqkit stats all.tenth.gz**
2. We can technically keep all filed zipped all the way through our analysis, which is very normal due to the massive sizes these data sets can reach. It does, however, make everything a lot slower, so we will start by unzipping our file using the **gunzip** program

**? How many sequences do we have and how many bases?**

**? What is the longest sequence we have and what is the shortest? Do we have any use for the short ones?**

Next, we filter the sequences. Fundamentally, we only want the longest and highest quality ones. We certainly don’t want the short ones. For this, we will use **filtlong:**

1. Install filtlong and have a look.
2. The example in the github is made for single genomes and aims for too few bases for a metagenome (the ‘target\_bases’ argument), so we will skip that one, but stick with the minimum length (remove all reads below 1000bp) and the percentage of reads (90%)
3. Filtlong will print its output to the terminal, so we will have to capture the output into a new file with the ‘>’ operator.
4. Something like this will work:
   1. **filtlong --min\_length 1000 --keep\_percent 90 all.tenth.gz > all.tenth.filt.fastq**
5. **This might take a while!**

**? How many sequences do we now have and how many bases?**

**? What is the longest sequence we have and what is the shortest? Did this help?**

Now we remove the adaptors and barcodes. The adaptors are artificial sequences added to the ‘real’ sequences before we sequence, and are needed to guide the sequences to the nanopores and the barcodes are needed to keep track of different samples. After we are done sequencing, though, it is hugely important that we remove them – otherwise these artificial sequences will be incorporated into our assemblies. One could argue that we should remove the adaptors before filtering, but this will be faster and doesn’t really change the outcome.

We will use the program ‘**porechop’** for this. Porechop knows the usual adaptors, and will work out which ones are relevant by looking at the first couple of thousand of sequences.

1. Install porechop. I hope I don’t have to say conda again?
2. Porechop can do a bunch of fancy things, including separating sequences by their barcodes etc. We only have one sample/barcode in this case, so all we need are three flags, but we will use a couple more for speed
   1. ‘-i’ : the input file: This is the file you just filtered
   2. ‘-o’ : the output file. Call it what you will, but remember to **give a “.fastq” ending**!
   3. ‘-t’ : the number of cpus. I have 8 cpus on my machine, so I gave it 6
   4. ‘--no\_split’ : skip searching for adaptors within the reads
   5. ‘--check\_reads 1000’ : no reason to search for multiple adaptors here
3. See if you can make the command yourself with these 3 flags and arguments.
4. **This will take a long time!**

? How much did we lose?

? Should we do the porechopping before the filtlonging? Why/ why not?

## Assemble the data

Finally, we are ready to assemble our now clean and adaptor-free sequences into genomes. We will use the program **flye** for this, since it is developed specifically for nanopore-reads and has support for metagenomic sequences. We will include two key parameters here, namely the ‘--nano-hq’ and the ‘--meta’ flags. The first lets flye know that we have data from the 10.4 chips, meaning it should trust each sequence more than before, and the second lets flye know that some genomes have less coverage than others: If we had a single genome, each nucleotide position should be covered by the same number of reads to make biological sense. In the case of single genomes, flye is more likely to consider assemblies with low coverage as contamination, but for metagenomes, everything is conceptually contamination and flye needs to know that those minor things count as well.

1. Install flye as usual. In this case, make particular sure that you also use the **conda-forge**  repository for installation. You need **version 2.9** for this to work.
2. Run flye like this
   1. **flye --nano-hq all.tenth.filt.noBC.fastq --meta --threads 6 --out-dir flye**

This will crash almost immediately, because this operation requires at least 60 or so Gb of RAM, which none of you will have on your laptops. Not to worry, we will continue with already assembled data.

## Work with the data

First we accept that we simply cannot handle data of this size on a laptop. Next, we download a subset of the assembled data, specifically the 100 longest contigs from the full assembly.

**wget https://zenodo.org/records/12618582/files/top100.fasta**

Now you get to play! This data is fundamentally a collection of incomplete genomes, which can be analyzed just like you did on the single genomes:

* Can we annotate them?
* What BGCs do we have?
* What phages do we have?
* Can you predict the 16S genes in these genomes?
* What would be fun to look for?