MinION bioinformatics practical: download software & assemble a bacterial genome

In this course, we will use miniasm, which is a very quick assembler – but the assemblies will be less accurate as we do not include an error correction step. It's a good way to check you're heading in the right direction in your research, but you should consider switching to more accurate assemblers as your project progresses. When & which assembler will depend on your specific research question.

We will also install some software from the version control website Github. This software is open access and free to use. Github can be a great resource for finding and installing bioinformatics software, but as always, do take care that what you are installing is genuine, safe to use, and suitable for your operating system and computer specifications

Getting started:

Open a terminal and check you are in the home directory

In [ ]:
```
pwd
```

Make a new directory called longread (all one word) and cd into it. (You can do this in two steps or you can chain them together using &&

In [ ]:
```
mkdir longread && cd longread
```

Installing software

You will need to install a dependency package. Type in the following:

sudo apt-get install libz-dev

If prompted for a password, use "manager"

Open a new tab in a browser window and search for (Google) miniasm

Click on top hit – github.com/lh3/miniasm

Find the text below on the webpage, copy it and paste it into your terminal window (You may need to use ctrl+ins and shift+ins instead of ctrl+c and ctrl+v)

git clone https://github.com/lh3/miniasm && (cd miniasm && make)

minimap2 should already be installed as part of an earlier session on the course - if it is not, do the same for

git clone https://github.com/lh3/minimap2 && (cd minimap2 && make)

Making an assembly

Check you are working in the longread folder and you have three subfolders, minimap2, miniasm.

You've been given some long read Oxford Nanopore data in your VM. Check that you have this data available to you. This should be found in the directory ONT_data

In [ ]:
```
ls ~/ONT_data
```

You are now ready to run your assembly – it will probably take 10-15 minutes. First, use minimap2 to map the reads to each other, creating overlaps. Don't forget, you will need to either copy (cp) the reads to the directory you are working in, or, a better way is to give the path to where your reads are stored (this will use a lot less of your computer's storage space)then use miniasm to assemble the overlapped reads

In [ ]:
```
minimap2/minimap2 -x ava-ont /your/path/barcode1.fastq /your/path/barcode1.fastq > barcode1.paf
```

In [ ]:
```
miniasm/miniasm –f /your/path/barcode1.fastq barcode1.paf > barcode1.gfa
```

To get the reads from .gfa format to .fasta (also called .fa), you'll need to use this awk command (be careful to get it exactly right)

In [ ]:
```
awk '$1=="S" {print ">"$2"\n"$3}' barcode1.gfa > barcode1.fa
```

You can now check out your new assembly file. Try some of the following:

In [ ]:
```
wc
head
ls –l
tail
```

You can also use scripts to assess your assembly. Here, we have already installed a script called assembly-stats

In [ ]:
```
assembly-stats barcode1.fa
```

You can find this at https://github.com/sanger-pathogens/assembly-stats if you want to install it on your own machines. It’s slightly more complicated to install than what we have done so far, but it’s similar, and there are instructions on the website. Just make sure to start with git clone . To use assembly stats, type in the name of your fasta or fastq file (instead of reads.fa above)

Next steps

There are two other sample fastqs in the ONT_data folder. In order to assemble these in a short time frame, and without using too much of your computer’s memory, you will need to down-sample the data. You can choose which sample to work on, or try more than one, or try a couple of different depths of coverage if you have time.

Use the tool chopper, which is already installed for you. You can filter your data on quality score (using -q) or length (using -l) - you might like to filter to remove all reads shorter than 500bp, or with quality score higher than ten. Use chopper -h to read the help file.

In [ ]:
```
cat barcode2.fastq | chopper -q 10 -l 500 > barcode2_filtered.fastq
```

Things to consider when filtering or downsampling data for assembly:

You can check how much data you have using the size of your assembly from the first part of the experiment, and running assembly-stats on your filtered fastq file.

e.g. assembly-stats your_assembly_sample_1.fa n=4000000

Your assembly is 4Mb

e.g. assembly-stats your_filtered.fastq n=100000000

Your filtered fastq is 100Mb. This gives you a "theoretical depth of coverage" (TDOC) of 25x, i.e. your dataset is 25x your genome size.

You should typically provide an absolute minimum of 20x TDOC when running an assembly. For many applications, it’s not wise to use lower than 30x. There’s no upper limit on the maximum depth of coverage, but you are likely to find more than 100x won’t finish within this session (or length of this course! Or within the capability of your laptop!)

Once you have down-sampled your data, go through the steps in the first task to create an assembly and assess its size.

Acknowledgements

With thanks to the following people who were involved in generating the data used today:

Trevor Lawley & team, Yan Shao, Sanger Long Read sequencing team

This is unpublished data and should not be used for anything outside this session, and practicing the skills you have learnt on this course