Exercise 3: Install and run the Emu-based Trana pipeline

(Note that you can view an outline of this document by clicking the icon that looks like a bullet point list, to the far right just above the header above)

In this exercise we will install the Trana pipeline, developed by Frans Wallin, Ryan Kennedy, us at Karolinska and others within Genomic Medicine Sweden, and which includes Emu as a module, together with some quality checking and filtering software, as well as some modules for showing results graphically, such as Krona and the TranaVy reporting tool developed by us, and more.

Since the installation of all these tools can be quite complex, we will be using an Ansible installation script developed by us, which installs all the tools and scripts needed to run the pipeline, with a single command.

The script is optimized for use on GridIon instruments, which Ubuntu Linux, but generally works well on most Ubuntu-like Linuxes, including on GitHub codespaces. Only some adjustments to the available CPUs and memory will be needed below.

The only initial step is installing Ansible, but that is generally very easy.

Installing Trana and TranaVy via Ansible

1. Install Ansible

   Ansible is available via the Ubuntu package repository, so can be installed via apt:

   ```
   sudo apt update
   sudo apt install ansible
   ```

2. Clone and run the install script repo

   The install script repo is supposed to be in a specific directory: `/data/trana/install`, which is under the `/data` folder where all sequencing data is stored on GridIon instruments. This, we start by creating this folder:

   ```
   sudo mkdir -p /data/trana/install
   ```

   And then we change ownership of the whole data folder to our current user, so that we can create files in it without sudo:

   ```
   sudo chown -R codespace:codespace /data
   ```

… and finally we enter it:

```
cd /data/trana/install
```

Here, we clone the repository to the current folder (which is why we end with the `.`, to mark the current folder):

```
git clone https://github.com/genomic-medicine-sweden/trana-setup-
        gridion.git .
```

3. Run the install script

   Now we are ready to run the install script.

   In order to simplify this even more, we have added a Makefile with a rule to run it, so we only need to do:

   ```
   make install
   ```

   This will run the Ansible install script on our local machine.

Run Trana on example data

Now we are ready to try running TRANA on some example data. To start with, we thus need to download some data to run on, and put it in the `/data` folder. Then we can use a script included in the install library, that will detect new data and run Trana on it.

1. Download test data

   Enter the `/data` folder:

   ```
   cd /data
   ```

   Now download the test data tarball:

   ```
   wget https://github.com/kclinmicro/16s-ont-
           course/releases/download/v0.0.1/16s-course-data.tar.gz
   ```

   And untar it:

   ```
   tar -zxvf 16s-course-data.tar.gz
   ```

   Now we copy the `16s_ont_example` folder directly under `/data`, where we are already:

   ```
   cp -r data/16s_ont_example .
   ```

So now, this folder should be in: `/data/16s_ont_example`.

The start script we will now use detects all data folders starting with `16s_ont_*`, so it is important that we keep this part of the name as is.

2. Now another thing we need before staring, is to have a “barcode sheet”, which maps the barcodes used in the sequencing, to sample names. The run script we will use expect such a barcode file in the `/data/trana/barcodesheets` directory, with the same name as the data folder, but ending with `.csv`.

   We have already created a barcodesheet for you, which is available in the data we just downloaded, so we just need to put this file in the correct place:

   So we copy the barcodesheet into place:

   ```
   cp 16s_ont_example/16s_ont_example.csv /data/trana/barcodesheets
   ```

3. Adjust CPU and memory settings for Nextflow

   The last thing we need to do is to adjust the max CPU and memory settings, which are optimized for GridIon in the config file shipped with the test script.

   If we run on GitHub codespaces, or a local laptop, we need to adjust these.

   If you are on GitHub codespaces, open the file `/data/trana/install/gridion.config` and change according to this diff:

   ```
   params {
   -    max_memory = '60.GB'
   -    max_cpus   = 16
   +    max_memory = '6.GB'
   +    max_cpus   = 2
        max_time   = '72.h'
    }

    process {
        resourceLimits = [
   -        cpus: 16,
   -        memory: '60.GB',
   +        cpus: 2,
   +        memory: '6.GB',
            time: 72.h
   ]
   ```

   If you are using a local laptop, you might be able to use a slightly higher number of cores and memory, depending on your laptop’s hardware specs.

4. Adjust downsampling level.

   The scripts right now are a little brittle and requires some downsampling to happen. Since the test datasets are already quite small, we need to change the max_samplesize to something that is surely smaller than the raw datasets.

Open the file `/data/trana/run/detect-data-and-run.sh` and change according to this diff, to set the `max_samplesize` to 500 (reads):

```
 outdir=${tranadir}/output
-max_samplesize=30000
+max_samplesize=500
 sleep_seconds_before_start=1
```

5. Now we are ready to run the pipeline using the start script.

   We first move to the run folder, where the run scripts are and where also all logs will be stored, in a `logs` folder:

   ```
   cd /data/trana/run
   ```

   … and then we run the run script:

   ```
   ./detect-data-and-run.sh
   ```

   Note that this can take possibly several hours, due to very long times for pulling down Singularity images!

   You might want to work on other exercises in parallel.

6. When the above command is complete, we are ready to explore the outputs of the pipeline.

   Go into the output folder:

   ```
   cd /data/trana/output
   ```

   Enter the folder named `16s_ont_example-<date>-<time>`. E.g:

   ```
   cd 16s_ont_example-*
   ```

   Hint: If you for some reason have multiple folder here, e.g. due to interrupted runs or similar, you can check which one of them contains most data with the `du` (disk usage) command:

   ```
   du -sh *
   ```

   Once inside the correct results folder, explore the various output folders.

   We recommend to start with:

   - `reports/`
   - `krona/`

Note that to view HTML files from GitHub codespaces, you might need to download them first (right click in the file viewer to the left pane and click “Download”).

Additionally, if you want to investigate what took the longest time in the pipeline, you might check out `pipeline_info/execution_timeline_*.html`

Hint: Use the TAB key to auto-complete the folder name for you)

7. That’s it - we’re done!