SPIDR Pipeline V1

Darvesh Gorhe

June 27th, 2024

Contents

1 Notes

- `UNI` is always a placeholder for your actual UNI, please replace it when running commands
- The `test` directory is a placeholder as well, but feel free to use it as is
- It's recommended that you type out the commands (labeled with cmd). Sometimes the underscores and spaces don't get copied correctly.
- If you run into issues with conda environments not setting up properly, you may need to set your channel priority to strict. You can do this with the following command:
  - cmd: `conda config --set channel_priority strict`
  - Alternatively you can create a `.condarc` file where you installed miniforge. Put the following into the `.condarc` file:

```
channels:
    - conda-forge
    - bioconda

channel_priority: strict
```

- Assuming you set up your conda environment correctly, this file should be located in `/burg/mjlab/projects/cli-tools/UNI/miniforge3/.condarc`

2 Pre-requisites

1. Ensure that you have `mamba` installed. You can check this by running:
   - cmd: `mamba --version`
   - If you get a version number, then `mamba` is installed
   - If `mamba` is not installed go to the Installing `mamba` subsection for instructions on how to install `mamba`
2. Ensure that you have `snakemake` installed in your base environment. You can check this by running:

- cmd: `snakemake --version`
- If you get a version number, then `snakemake` is installed
- If `snakemake` is not installed go to the Installing `snakemake` subsection for instructions on how to install `snakemake`

2.1 Installing `mamba`

1. Log in to Ginsburg
2. Create a directory for your command-line tools if it doesn't already exist
    - cmd: `mkdir -p /burg/mjlab/projects/cli-tools/UNI`
3. Navigate to the directory you created
    - cmd: `cd /burg/mjlab/projects/cli-tools/UNI`
4. Download the installation script
    - cmd: `curl -L -O "https://github.com/conda-forge/miniforge/releases/latest/download/Miniforge3-\$(uname)-\$(uname-m).sh"`
5. Run the installation script
    - cmd: `bash Miniforge3-$(uname)-$(uname -m).sh`
    - Type 'yes' and hit enter to agree to the terms of service
    - When asked where to install `miniforge3`, type `/burg/mjlab/projects/cli-tools/\UNI` instead of the default path (usually `/burg/\UNI/miniforge3`)
    - Type 'y' or 'yes' for any of the following prompts to agree to them
6. Set `$CONDA_ROOT` in your `/.bashrc` file to the directory where miniforge3 is installed
    - cmd: `echo export "CONDA_ROOT=/burg/mjlab/projects/cli-tools/UNI/miniforge3" >> ~/.bashrc`
    - This is important because there is a specific precedence for `.condarc` file which can change which version of packages are installed
    - `.condarc` files tell `conda` and `mamba` which channels to look at, in what order to look at channels, and how strictly to look at channels
    - You can learn more about the `.condarc` specifics here.
    - You can learn more about channels here.
7. Re-start your shell to make sure the changes take effect
    - cmd: `source ~/.bashrc`
8. Ensure your `.condarc` file has only

2.2 Installing `snakemake`

1. Log in to Ginsburg
2. Make sure you have `mamba` installed and running, you can check this with the following command
    - cmd: `mamba --version`
    - If you get a version number, then `mamba` is installed
    - If `mamba` is not installed go to the Installing `mamba` subsection for instructions on how to install `mamba`
3. Install `snakemake` from the `bioconda` channel
    - cmd: `mamba install -c bioconda snakemake`
    - This will list all the dependencies that `mamba` will download and install, type 'Y' and hit enter to confirm
4. Confirm that `snakemake` is installed properly
    - cmd: `snakemake --version`
    - If you get a version number (probably 7.32.4 or higher), then `snakemake` is installed correctly

3 SPIDR Run with `HEK_MTOR_QC` Data - Ginsburg HPC

1. Create a directory `/burg/mjlab/projects/spidr-runs/\UNI/\DIRNAME` if it doesn't exist.
   - cmd: `mkdir -p /burg/mjlab/projects/spidr-runs/UNI/test`
   - For the remainder of the document we'll call this directory `\DIRNAME`
2. Navigate to that directory (test in our case)
   - cmd: `/burg/mjlab/projects/spidr-runs/\UNI/\DIRNAME`
3. Clone the SPIDR repository with all the code into this folder.
   - cmd: `git clone https://github.com/mjlab-Columbia/spidr.git`
   - This will create a directory called `spidr`
4. Navigate to the new `spidr` directory
   - cmd: `cd spidr`
5. Create the `experiments.json` file to tell the pipeline where your read files are.
   - cmd: `python scripts/python/fastq2json.py --fastq_dir /burg/mjlab/projects/HEK_MTOR_QC`
   - This will be the directory containing the paired-end files from your sequencing run
   - Avoid copying the files from where they are since they're often large
6. Copy the example barcode configuration file `config_6_rounds_mTOR.txt` file located at `/burg/mjlab/projects/spidr-barcoding-files/config_6_rounds_mTOR.txt`
   - cmd: `cp/burg/mjlab/projects/spidr-barcoding-files/config_6_rounds_mTOR.txt./config_6_rounds_mTOR.txt`
   - This file contains 4 columns separate by tabs: barcode type, barcode name, barcode sequence, and barcode mismatch tolerance
   - You can learn more about the barcode structure on the wiki.
7. Copy the barcode format file `format_6_rounds_mTOR.txt` file located at `/burg/mjlab/projects/spidr-barcoding-files/format_6_rounds_mTOR.txt`
   - cmd: `cp /burg/mjlab/projects/spidr-barcoding-files/format_6_rounds_mTOR.txt ./format_6_rounds_mTOR.txt`
   - This file should contain 3 columns separated by tabs: the expected position of the combinatorial barcode (with 1 being the first barcode), the barcode name, and the barcode sequence
   - This file is used to remove complete barcodes with barcodes in the incorrect order.
   - There may be a 4th column in the example file, but you can ignore that column.
8. Copy the example `config.yaml` file located at `/burg/mjlab/projects/spidr-barcoding-files/config.yaml`
   - cmd: `cp /burg/mjlab/projects/spidr-barcoding-files/config.yaml ./config.yaml`
   - Editing the file is covered in the next step
9. Edit the `config.yaml` file to ensure all the settings are appropriate for your experiment.
   - In this case since we're using example files, you only need to change the email section (i.e. change `email: "dsg2157@columbia.edu"`)
   - Please visit the wiki for more information on all the parameters.
   - Alignment indices are located at `/burg/mjlab/projects/spidr-alignment-indices`. For the STAR index, point it to the directory containing the index files and for bowtie2, make sure you include the file name prefix in addition to the full path. (Note: add more clarification here later)
   - For help navigating and editing files on Ginsburg, see the `ginsburg` repository on GitHub.
10. Request more resources within an interactive job
    - cmd: `srun --pty -t 0-04:00 -A mjlab --mem=32G -N 1 -c 4 /bin/bash`

- By default, you're put into a login node when logging into Ginsburg. The login node has limited resources and if you use too many resources, Ginsburg may automatically terminate your session (i.e. they kick you out). We need to get off the login node to avoid this issue.
- This command requests a single node with 4 CPU cores and 32 Gb of RAM for 4 hours
- More details about the `srun` command can be found here.
- Ginsburg specific limitations/info can be found here.

11. Perform a "dry" run of the pipeline
    - cmd: `bash run.sh --dry_run`
    - This command will show what steps will be run and the input/output file names for each step without actually running the steps
    - This will help catch basic errors like missing files, misnamed files, etc.
12. Run the actual pipeline
    (a) (Foreground) cmd: `bash run.sh`
    (b) (Background) cmd: `sbatch run.sh`
    - When running the pipeline in the foreground you will need to keep your terminal open. If you close your terminal, the pipeline will stop (but it will finish whatever steps it already started).
    - When running the pipeline in the background a file called `spidr.log` will be created to capture outputs and errors from the run. It captures the equivalent of what you would see on your screen when executing in the foreground.

---

4 SPIDR Run with Non-test Data - Ginsburg HPC

1. Create a directory, ideally within `/burg/mjlab/projects/spidr-runs` under a folder with UNI.
   - cmd: `mkdir /burg/mjlab/projects/spidr-runs/UNI/test`
   - For the remainder of the document we'll call this directory `\DIRNAME`
2. Navigate to that directory (test in our case)
   - cmd: `/burg/mjlab/projects/spidr-runs/\UNI/\DIRNAME`
3. Clone the SPIDR repository with all the code into this folder.
   - cmd: `git clone https://github.com/mjlab-Columbia/spidr.git`
   - This will create a directory called `spidr`
4. Navigate to the new `spidr` directory
   - cmd: `cd spidr`
5. Create the `experiments.json` file to tell the pipeline where your read files are.
   - cmd: `python scripts/python/fastq2json.py --fastq_dir <Path to directory containing read 1 and read 2>`
   - This will be the directory containing the paired-end files from your sequencing run
   - Avoid copying the files from where they are since they're often large
6. Create a barcode config file
   - Other than the first 3 rows, this file should contain 4 columns separated by tabs in the following order: barcode category, barcode name, barcode sequence, and mismatch tolerance
   - The first row has the read 1 format (usually just "READ1=DPM")
   - The second row has the read 2 format (something like "READ2=Y|SPACER|ODD|SPACER|EVEN")
   - The third row is empty
   - Please see the wiki for more information.

- You can see an example of barcode config file at `/burg/mjlab/projects/spidr-barcoding-files/config_6_rounds_mTOR.txt`

7. Create a barcode format file
   - This file should contain 3 columns separated by tabs in the following order: barcode positon (first barcode is position 1), barcode name, barcode sequence
   - Please see the wiki for more information.
   - You can see an example of barcode format file at `/burg/mjlab/projects/spidr-barcoding-files/format_6_rounds_mTOR.txt`
8. Create a `config.yaml` file and ensure all the settings are correct.
   - There is an example `config.yaml` located here `/burg/mjlab/projects/spidr-barcoding-files/config.yaml`
   - Please visit the wiki for more information on all the parameters.
   - Alignment indices are located at `/burg/mjlab/projects/spidr-alignment-indices`. For the STAR index, point it to the directory containing the index files and for bowtie2, make sure you include the file name prefix in addition to the full path. (Note: add more clarification here later)
   - For help navigating and editing files on Ginsburg, see the `ginsburg` repository on GitHub.
9. Request more resources within an interactive job
   - cmd: `srun --pty -t 0-04:00 -A mjlab --mem=32G -N 1 -c 4 /bin/bash`
   - By default, you're put into a login node when logging into Ginsburg. The login node has limited resources and if you use too many resources, Ginsburg may automatically terminate your session (i.e. they kick you out). We need to get off the login node to avoid this issue.
   - This command requests a single node with 4 CPU cores and 32 Gb of RAM for 4 hours
   - More details about the `srun` command can be found here.
   - Ginsburg specific limitations/info can be found here.
10. Perform a "dry" run of the pipeline
    - cmd: `bash run.sh --dry_run`
    - This command will show what steps will be run and the input/output file names for each step without actually running the steps
    - This will help catch basic errors like missing files, misnamed files, etc.
11. Run the pipeline
    (a) (Foreground) cmd: `bash run.sh`
    (b) (Background) cmd: `sbatch run.sh`
    - When running the pipeline in the foreground you will need to keep your terminal open. If you close your terminal, the pipeline will stop (but it will finish whatever steps it already started).
    - When running the pipeline in the background a file called `spidr.log` will be created to capture outputs and errors from the run. It captures the equivalent of what you would see on your screen when executing in the foreground.

---

5 SPIDR Run with `HEK_MTOR_QC Data` - MacOS (Experimental and Incomplete)

There is an experimental script called `local.sh` which is an attempt to run the SPIDR pipeline on a Mac. This Mac should ideally have an M-series chip and 32Gb+ of RAM and 1Tb+ of SSD space. This method requires a lot of manual work, so it's not recommended. It was originally conceived as an exercise in hubris, but it might actually be useful.

1. Install the following packages via Homebrew
   - cmd: `brew install parallel bowtie2 samtools gcc@11 libomp`

- If you don't have Homebrew installed on your Mac, visit https://brew.sh/ to learn how to install it.
- `gcc` version 11 is the latest stable GCC compiler that compiles `STAR` on MacOS, so the `@11` is important.

2. Create a conda environment
   - cmd: `mamba create -n spidr_local -c bioconda -c conda-forge cutadapt fastqsplitter python`
3. Activate the conda environment
   - cmd: `mamba activate spidr_local`
4. Install `trim_galore`
   - cmd: `cd $HOME`
   - cmd: `curl -fsSL https://github.com/FelixKrueger/TrimGalore/archive/0.6.10.tar.gz -o trim_galore.tar.gz`
   - cmd: `tar xvzf trim_galore.tar.gz`
   - cmd: `echo "export PATH=$PATH:$HOME/TrimGalore-0.6.10" >> ~/.zshrc`
   - Note: If you're using the `bash` shell then `$\sim$/.zshrc` should be replaced with `$\sim$/.bashrc` (`zsh` is the default shell which is why it's in the instructions).
5. Download the source code for `STAR` and compile it
   - cmd: `cd $HOME`
   - cmd: `git clone https://github.com/alexdobin/STAR.git`
   - cmd: `cd STAR/source`
   - cmd: `make STARforMacStatic CXX=/opt/homebrew/Cellar/gcc@11/11.4.0/bin/g++-11`
   - cmd: `echo "export PATH=$PATH:$HOME/STAR/bin/MacOSX_x86_64/STAR" >> ~/.zshrc`
   - Note: This compilation from scratch is necessary because there are no distributions for `STAR` on conda for MacOS. That is to say, things like `conda install STAR` or `mamba install STAR` won't work on MacOS.
   - Note: Double-check the exact path to the `gcc` version installed by Homebrew (i.e. it may have a slightly different number than 11.4.0)
6. Download the `HEK_MTOR_QC` data to your Mac.
   - cmd: `wget "S3 URL HERE"`
7. Un-archive the folder
   - cmd: `tar -xvf "FILE FROM S3 HERE"`
8. Run `local.sh`
   - cmd: `./local.sh HEK_MTOR_QC/HEK_MTOR_QC_R1.fastq.gz HEK_MTOR_QC/HEK_MTOR_QC_R2.fastq.gz HEK_MTOR_QC 10`
   - The arguments are as follows
     1 Path to gzipped read 1 fastq file
     2 Path to gzipped read 2 fastq file
     3 Prefix for filenames to be created
     4 Number of equal chunks to split the reads into
   - This executes the same steps as the snakemake based pipeline

---

6 Debugging

- You can get a visual representation of the pipeline by creating a rulegraph. You can run the following:
  - cmd: `snakemake --rulegraph | dot -Tpdf > rulegraph.pdf`
  - You can run this on Ginsburg, and then copy it on to your local computer via 'scp' or using the Globus portal