Huws lab Metagenomic Protocol v1.0.0

Thursday, 26 September 2024

This document will include filenames of scripts that can be taken, modified and then submitted using the Kelvin HPC. They should work with most modern HPC clusters using the SLURM submission system.

Below I will provide the general commands used to run each tool as if on a single sample. However, as each study has many samples, we want to run each tool in a *for loop* that loops through all samples in a study and runs the same command for each one.

**Some generally useful links:**

* **Kelvin Resources and descriptions:** <https://ni-hpc.github.io/nihpc-documentation/Modules%20%26%20Jobscripts/#job-scheduler-instructions>
* **Illumina Info page on adapters:** <https://knowledge.illumina.com/software/general/software-general-reference_material-list/000002905>
* **Overviews of different Metagenome Assembly tools:** <https://doi.org/10.1371/journal.pone.0169662> | <https://doi.org/10.1186%2Fs12864-017-3918-9>

**GitHub links for the tools we will be using:** Github is a database of

* **FastQC:** <https://github.com/s-andrews/FastQC>
* **MultiQC:** https://github.com/ewels/MultiQC
* **Trimmomatic**: <https://github.com/usadellab/Trimmomatic>
* **Bowtie2:**  <https://github.com/BenLangmead/bowtie2>
* **Meta/Spades:** <https://github.com/ablab/spades>
* **Kraken2:** <https://github.com/DerrickWood/kraken2>
* **MetaPhlan:** https://github.com/biobakery/MetaPhlAn
* **Prodigal/Pyrodigal:** https://github.com/althonos/pyrodigal <https://github.com/hyattpd/Prodigal>
* **Eggnog-mapper:** <https://github.com/eggnogdb/eggnog-mapper>
* **Python3:** https://www.python.org/

**Anaconda environments available on Kelvin:** Several anaconda (conda) environments have been preinstalled on Kelvin and are available for everyone to use.

* Please request access to /mnt/scratch2/igfs-anaconda via the Kelvin support ticket system (<https://www.qub.ac.uk/directorates/InformationServices/Services/ITServiceDesk/>)
* There are different anaconda versions available on Kelvin and I am currently testing which is the best version and so far it seems to be ‘2022.10’ – To activate this module use ‘**module load apps/anaconda3/2022.10/bin’**.

**Kelvin Job Submission:** Kelvin uses the ‘SLURM’ job submission system (<https://slurm.schedmd.com/sbatch.html>) to handle users and their jobs.

* We submit ‘Jobs’ to Kelvin using the SLURM system with the ‘sbatch’ command.
  + For example, **sbatch trimmomatic\_script.slurm**

General Notes:

* Where I have placed ‘…/’ (3 dots and a forward slash) in the code, I am referring to wherever YOUR data is. You must change this according to where your data is as copying this directly will not work.
* Where you see ‘$$’ (Double dollar signs), I am showing you where you need to add your own parameters. For example, ‘#SBATCH --mail-user=**$$**@qub.ac.uk’ is asking you to replace ‘$$’ with your QUB email address.

Databases used:

* Kraken2 Database:
  + June 2023 – PlusPFP - https://genome-idx.s3.amazonaws.com/kraken/k2\_pluspfp\_20230605.tar.gz

Conda Environments:

* We have a number of preinstall anaconda environments in the igfs-anaconda directory with all the tools needed for this pipeline:
  + To view them you can use ‘conda env list’
* A screen shot of a computer

  Description automatically generated

**Metagenomic Workflow:**

**Kelvin SLURM batch file handling:**

* The Kelvin HPC uses the SLURM job scheduling system to submit jobs into a queue that then run when ‘space’ becomes available.
* These few lines at the top of each SLURM submission script configures a job submission to a SLURM-managed compute cluster with specific resource requests, job parameters, partition preferences, and email notifications for job status updates. The actual computational tasks to be performed by the job would follow these SLURM directives in the script.**A screen shot of a computer program

  Description automatically generated**
* The SLURM directives are as follows:
  + #SBATCH --cpus-per-task=8: Requests 8 CPU cores for the task, specifying the number of cores allocated per task.
  + #SBATCH --mem=20G: Requests 20 GB of memory for the job, setting the maximum amount of memory the job can use.
  + #SBATCH --job-name=trimm: Sets the name of the job to "trimm", which is useful for identifying the job in the job queue and logs.
  + #SBATCH --error=Trim\_$$.error: Directs the standard error output (errors) to a file named Trim\_$$.error, where $$ is replaced with the job ID, ensuring a unique error file for each job instance.
  + #SBATCH --output=Trim\_$$.txt: Directs the standard output (regular output) to a file named Trim\_$$.txt. The $$ is replaced with the job ID, ensuring a unique output file.
  + #SBATCH --partition=bio-compute,k2-medpri: Specifies that the job should be run on either the "bio-compute" or "k2-medpri" partition, which are groups of compute resources in the cluster. The job scheduler will place the job on any available node in these partitions.
  + #SBATCH --nodes=1: Requests one compute node for the job, where a node is a single machine in the cluster.
  + #SBATCH --time=24:00:00: Sets a time limit of 24 hours for the job, ensuring the job will be terminated if it exceeds this time.
  + #SBATCH --mail-user=$$@qub.ac.uk: Specifies the email address to send notifications to. The $$ will be replaced with the job ID. This placeholder should be replaced with an actual email address before running the script.
  + #SBATCH --mail-type=BEGIN,END,FAIL: Configures SLURM to send email notifications at the beginning, end, and failure of the job, which is useful for monitoring job status without needing to manually check the job queue.

1. **(*Optional*) – Read quality checking with FastQC:**
   1. **FastQC** is a tool that reads the quality data provided by the Illumina sequencing machine (embedded in the FastQ files) and provides a visual overview of the overall ‘predicted’ quality of the sample (The sequencing machine is not 100% certain of its ‘own’ accuracy).
   2. To run **FastQC**:A screen shot of a computer code

      Description automatically generated
      1. Activate the multiqc environment (FastQC and MultiQC are installed together in a single environment ‘multiqc’).
      2. Set working directory and output directory.
      3. Find all ‘.fastq’ files and run the fastqc command on them individually.
   3. See Submission file **fastqc\_v1.slurm** to run **FastQC** on a collection of samples.
2. **(*Optional*) – Overview FastQC quality check with MultiQC:**
   1. **MultiQC** takes the individual output files created by **FastQC** and collates them into a single output that allows for easier comparison across a study.
   2. To run **MultiQC:**A screen shot of a computer code

      Description automatically generated
      1. Activate the multiqc environment.
      2. Set the output directory.
      3. multiqc --outdir /path/to/output\_directory.
3. **Trimming and pairing reads with Trimmomatic:**
   1. We are using the Trimmomatic provided adapter sequences (TruSeq3-PE-2) available at: <https://github.com/timflutre/trimmomatic/blob/master/adapters/TruSeq3-PE-2.fa>
   2. To run Trimmomatic: **A computer screen shot of a code

      Description automatically generated**
      1. **trimmomatic PE -phred33 -threads 8 sample\_name\_R1.fastq sample\_name\_R2.fastq sample\_name\_R1\_trimmed.fastq.gz sample\_name\_R1\_unpaired.fastq.gz sample\_name\_R2\_trimmed.fastq.gz sample\_name\_R2\_unpaired.fastq.gz ILLUMINACLIP: TruSeq3-PE-2:2:30:10 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:20 MINLEN:50**
   3. The Trimmomatic command is described below:
      1. **trimmomatic**: This is the command to run Trimmomatic.
      2. **PE**: Indicates that the input data is in paired-end format.
      3. **-phred33**: Specifies that the quality scores are in Phred+33 format. This is a common quality score encoding for Illumina sequencing data.
      4. **-threads 8**: Specifies that the analysis should use 8 threads or CPU cores for parallel processing.
      5. **sample\_name\_R1.fastq** and **sample\_name\_R2.fastq**: Input paired-end FASTQ files containing the reads from the two ends of the sequencing fragments.
      6. **sample\_name\_R1\_trimmed.fastq.gz** and **sample\_name\_R2\_trimmed.fastq.gz**: Output paired-end FASTQ files containing the trimmed reads after processing.
      7. **sample\_name\_R1\_unpaired.fastq.gz** and **sample\_name\_R2\_unpaired.fastq.gz**: Output FASTQ files containing unpaired reads that were not part of a valid paired-end fragment after processing.
      8. **ILLUMINACLIP:TruSeq3-PE-2:2:30:10**: Adapter clipping options. This specifies the adapter sequences to be removed. **TruSeq3-PE-2** is the adapter file, and **2:30:10** specifies the seed mismatches, palindrome clip threshold, and simple clip threshold.
      9. **LEADING:3**: Removes low-quality bases at the beginning of the reads if the quality drops below a threshold of 3.
      10. **TRAILING:3**: Removes low-quality bases at the end of the reads if the quality drops below a threshold of 3.
      11. **SLIDINGWINDOW:4:20**: Performs a sliding window trimming approach, where a 4-base window scans through the read and trims when the average quality within the window drops below 20.
      12. **MINLEN:50**: Discards reads that become shorter than 50 bases after trimming.
      13. This command aims to improve the quality of the reads, remove adapter sequences, and ensure that the reads are of a minimum length and have a mate. It's commonly used as a preprocessing step before downstream analysis of sequencing data.
   4. See Submission file **trimmomatic\_script.slurm** to run **Trimmomatic** on a collection of samples.
4. **Removing contamination and preparing reads for assembly:**
   1. First, we need to identify the host and environmental DNA we might expect to be contaminating our samples. Most of our animal studies will likely be with cows or sheep along with human operators. Therefore, the standard dataset we will use to remove contamination is a combination of the bovine, ovine and human genomes.
   2. ***\*\*This has already been done\*\**:** To prepare these genomes for contamination removal we must first download them from NCBI (See links below) and next create a bowtie2 database from them.
      1. These genomes and the bowtie2 database are already available in **‘/mnt/scratch2/igfs-databases/reference\_genomes/combined\_cow\_sheep\_human**’ and the bowtie2 database has already been created using the following command: bowtie2-build Human\_Sheep\_Cow\_Combined.fna Human\_Sheep\_Cow\_Combined\_Bowtie2
         1. Bovine: https://www.ncbi.nlm.nih.gov/datasets/taxonomy/9913/
         2. Ovine: https://www.ncbi.nlm.nih.gov/datasets/taxonomy/9940/
         3. Human: <https://www.ncbi.nlm.nih.gov/datasets/taxonomy/9606/>
   3. A screen shot of a computer code

      Description automatically generated
   4. This script is described below:
      1. **bowtie2 --threads 20 -x …/Human\_Sheep\_Cow\_Combined -1 sample\_name\_1 -2 sample\_name\_2 | samtools view -Sbh - > mapped\_bam;**
         1. **bowtie2 --threads 20 -x /Human\_Sheep\_Cow\_Combined -1 sample\_name\_1 -2 sample\_name\_2**: This command uses Bowtie2 to align paired-end reads from **sample\_name\_1** and **sample\_name\_2** to the reference genome indexed by **/Human\_Sheep\_Cow\_Combined**, using 20 threads for parallel processing.
         2. **|**: This is a pipe symbol, which takes the output of the previous command and uses it as input for the next command.
         3. **samtools view -Sbh - > mapped\_bam**: This command uses **samtools view** to convert the Bowtie2 alignment output (in SAM format) from the previous command into a BAM file (**mapped\_bam**) in binary format (**-b**). The **-h** option retains the header, and **-S** indicates input in SAM format. The **-** indicates that the input comes from the standard input (i.e., the output of the previous command).
      2. **samtools view -b -f 12 mapped\_bam > unmapped\_bam ;**
         1. **samtools view -b -f 12 mapped\_bam > unmapped\_bam**: This command uses **samtools view** to filter the BAM file **mapped\_bam**. The **-f 12** flag filters for reads that are not mapped (**0x4** flag) and have both mate pairs unmapped (**0x8** flag). The resulting filtered reads are saved in a new BAM file named **unmapped\_bam**.
      3. **samtools sort -n unmapped\_bam -o unmapped\_sorted\_bam ;**
         1. **samtools sort -n unmapped\_bam -o unmapped\_sorted\_bam**: This command sorts the **unmapped\_bam** file by read name (**-n** option) and saves the sorted output in a new BAM file named **unmapped\_sorted\_bam**.
      4. **bedtools bamtofastq -i unmapped\_sorted\_bam -fq unmapped\_R1 -fq2 unmapped\_R2**
         1. **bedtools bamtofastq -i unmapped\_sorted\_bam -fq unmapped\_R1 -fq2 unmapped\_R2**: This command uses **bedtools** to convert the sorted BAM file **unmapped\_sorted\_bam** into two FASTQ files, **unmapped\_R1** and **unmapped\_R2**, containing the unmapped reads' paired-end data.
      5. In summary, this code aligns paired-end reads using Bowtie2, processes the alignment results to extract unmapped reads, sorts the unmapped reads by name, and then converts the sorted unmapped reads into separate FASTQ files for both read pairs using **samtools** and **bedtools**.
5. Top of Form
6. **Assemble reads into Metagenomic contigs: metaspades\_v1.slurm**
   1. A computer code on a dark background

      Description automatically generated
   2. The command is described below:
      1. **metaspades.py -1 sample\_name\_1 -2 sample\_name\_2 -o outdir -t 24**
      2. **metaspades.py**: This is the command to run the MetaSPAdes assembler.
      3. **-1 sample\_name\_1 -2 sample\_name\_2**: These options specify the input paired-end FASTQ files for the assembly. **sample\_name\_1** and **sample\_name\_2** are the filenames of the first and second reads, respectively.
      4. **-o outdir**: This option specifies the output directory where the assembly results will be saved. **outdir** is the name of the directory where the output will be stored.
      5. **-t 24**: This option sets the number of threads (CPU cores) to use for parallel processing. In this case, it's set to **24**.
      6. In summary, this command runs the MetaSPAdes metagenomic assembler on paired-end sequencing data provided in **sample\_name\_1** and **sample\_name\_2** files. The assembled contigs and other output files will be saved in the specified output directory **outdir**, and the assembly process will use 24 threads for parallel processing.
7. **Taxonomically annotate contigs with Kraken2:**
   1. A computer screen shot of a code

      Description automatically generated