Huws lab Metagenomic Protocol v1.0.0

Friday, 25 July 2025

This document will include filenames of scripts that can be modified and then submitted using the Kelvin2 HPC. They `should’ work with most modern HPC clusters using the SLURM submission system.

Below are the general commands used to run each tool or step, through a set of samples that are housed separately in individual directories.

**Some useful links:**

* **Kelvin Resources and descriptions:** [https://ni-hpc.github.io/nihpc-documentation/Modules%20%26%20Jobscripts/#job-scheduler-instructions](https://ni-hpc.github.io/nihpc-documentation/Modules%20%26%20Jobscripts/" \l "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> - Being replaced by FastP
* **FastP**:
* **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/>
* **MetaPont:** https://github.com/TheHuwsLab/MetaPont

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

* Users no longer need request access to /mnt/scratch2/igfs-anaconda and should have read-only access. If this is not the case, request access via the Kelvin support ticket system (<https://www.qub.ac.uk/directorates/InformationServices/Services/ITServiceDesk/>)
* There are different anaconda versions available on Kelvin2 and I the current workflow seems to run fine ‘2024.06’ – To ensure that this is the version of anaconda you are using, please enure this line is in all submission scripts ‘**module load apps/anaconda3/2024.06/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 fastqc\_script.slurm**
* Submission scripts are written in Bash which is the same language you use to navigate the terminal so commands such as `cd’ and `mkdir’ work the same in a submission file and on the terminal.

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:

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

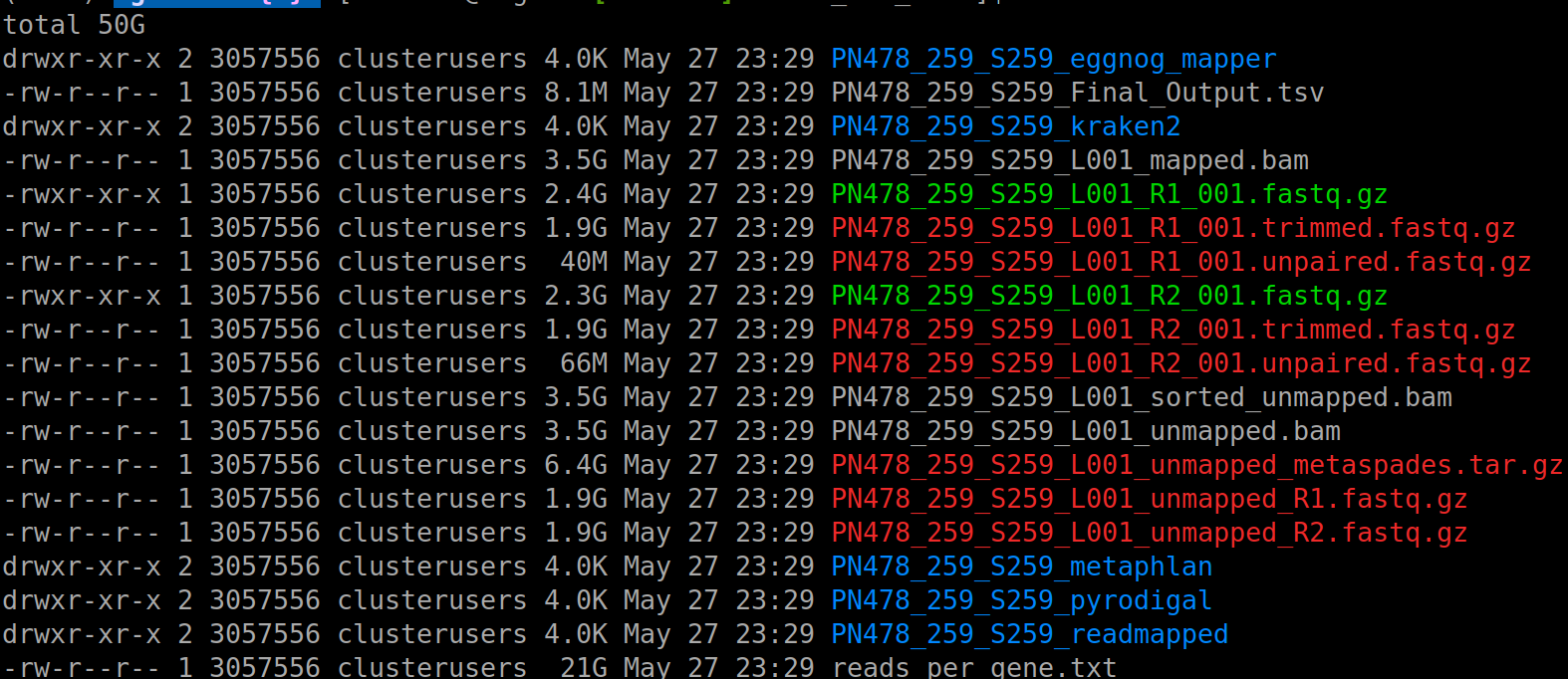
Conda Environments:

* 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:**

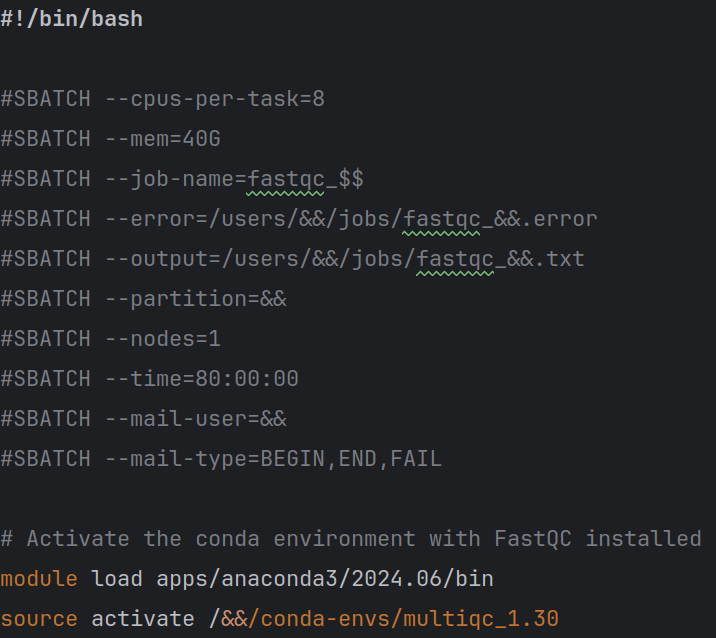
The workflow has been written to conform to a very specific format where each sample is separated and processed in its own directory. Outputs such as assemblies and annotations are given their own directory within the sample directory. The submission scripts rely on this directory structure and as such should not be changed. An example can be seen below:

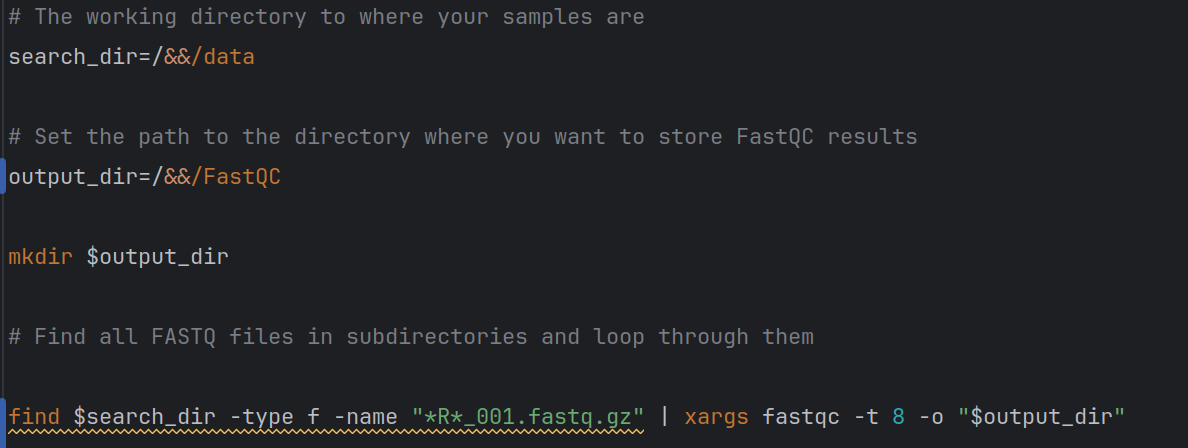


**Kelvin SLURM batch file handling:**

* 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

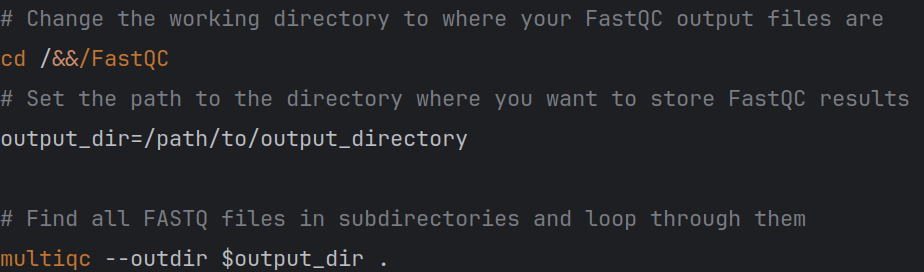
  Description automatically generated
* The SLURM directives are as follows:
  + #SBATCH --cpus-per-task=8: Requests 8 CPU cores for the task, specifying the number of cores allocated per task.
  + #SBATCH --mem=20G: Requests 20 GB of memory for the job, setting the maximum amount of memory the job can 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. **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**:
      1. Activate the multiqc environment (FastQC and MultiQC are installed together in a single environment ‘multiqc’ on the Kelvin cluster).
      2. Set working directory and output directory and find all ‘.fastq’ files and run the fastqc command on them individually.



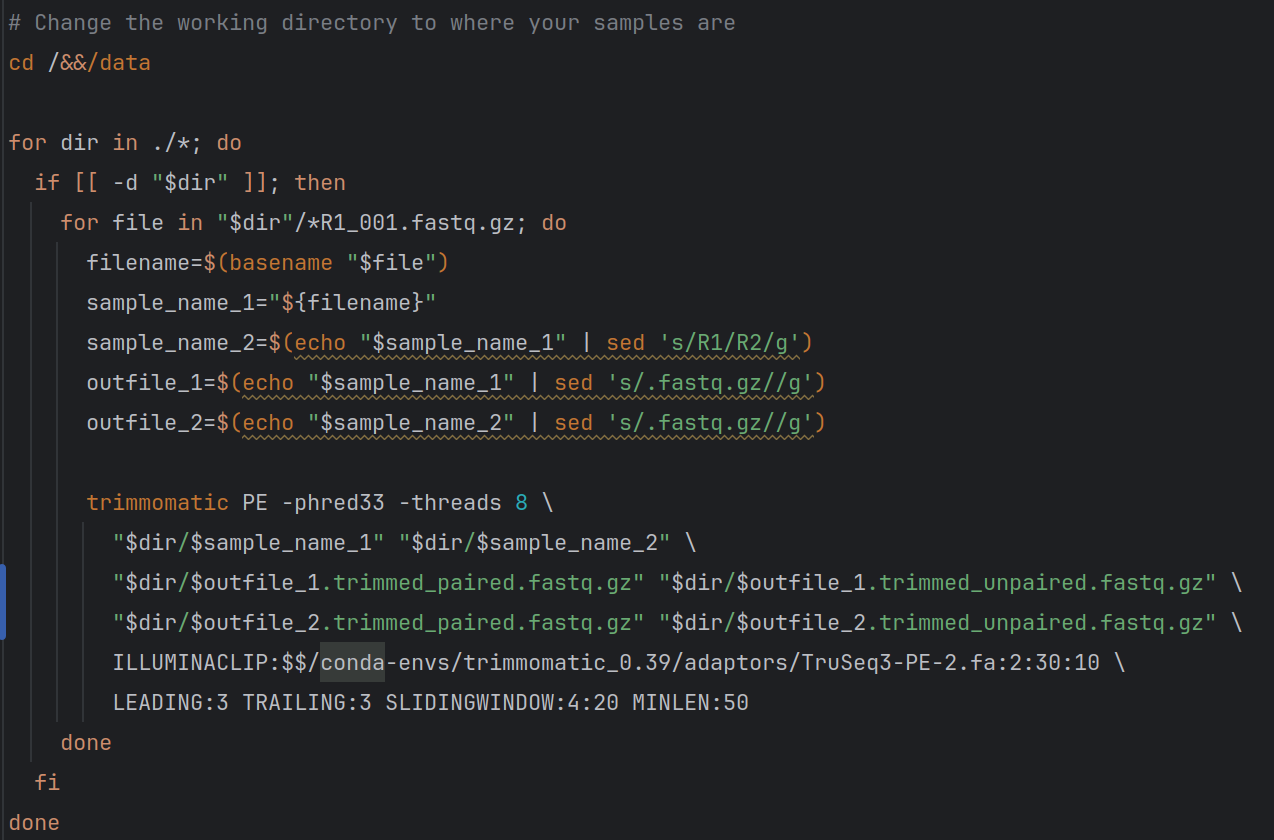
* 1. See Submission file **fastqc\_v1.slurm** to run **FastQC** on a collection of samples.

1. **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:**

****

* + 1. Activate the multiqc environment.
    2. Set the output directory.
    3. multiqc --outdir /path/to/output\_directory.

1. **(To be replaced with fastp) 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:

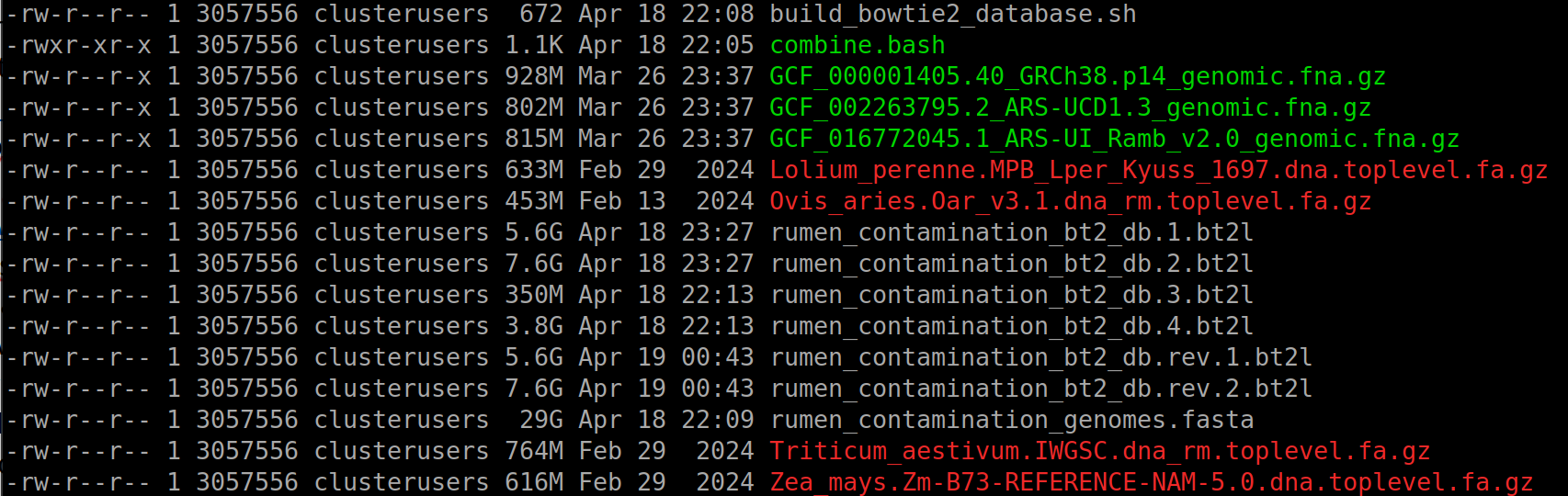
****

* + 1. 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 – Check the manual for further details.
  1. See Submission file **trimmomatic\_v1.slurm** to run **Trimmomatic** on a collection of samples.

1. **(Replacing Trimmomatic) Trimming and pairing reads with Fastp:**
   1. As a replacement for trimmomatic, we can use Fastp to trim and pair our reads.
      1. As we may be unsure of which adapters were used, we use the –detect\_adapter\_for\_pe option. Users can provide their own adapters in fasta format.
   2. See Submission file **fastp\_v1.slurm.**



1. **Removing contamination and preparing reads for assembly:**
   1. 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. We have also included some typical agricultural plant genomes. The full list can be found in the ‘rumen\_genomes.txt’ file inside the Contamination\_Control dir.
   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/$$/reference\_genomes/rumen\_contamination\_genomes/rumen\_contamination\_bt2\_db** and the bowtie2 database has already been created. See build\_bowtie2\_database.slurm in the Contamination\_Control dir for more details.
         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/
         4. [Lolium perenne: https://www.ncbi.nlm.nih.gov/datasets/taxonomy/4522/](https://www.ncbi.nlm.nih.gov/datasets/genome/GCF_019359855.1/)
         5. Triticum aestivum: https://www.ncbi.nlm.nih.gov/datasets/taxonomy/4565/
         6. Zea mays: https://www.ncbi.nlm.nih.gov/datasets/taxonomy/4577/



* 1. Due to the time each sample will take to process this script is written differently where each sample has its own SLURM submission. This allows concurrent processing of each sample to significantly speed up processing of very large numbers of samples.
  2. The script is described below:
     1. Its main purpose is to automatically create and submit separate jobs to check for contamination in sequencing data.
     2. Below are details for the script *contamination\_removal\_array\_v1.slurm*:
        1. WORK\_DIR: where the raw sequencing data is stored.
        2. JOB\_DIR: where the new job scripts and their logs will be saved.
        3. Makes sure the job directory exists (mkdir -p).
        4. Moves into the working directory to look for data.
        5. Looks for folders starting with “PN” (e.g., PN12345) – Change accordingly.
        6. Inside each folder, looks for files ending in R1\_001.trimmed\_paired.fastq.gz – Change accordingly.
        7. For each such file:
           1. Figures out the name of the matching “R2” file.
           2. Uses the folder name as the sample name (PNxxx).
        8. For each sample, creates placeholder variable names for:
           1. A sorted BAM file of reads that did not match contaminants.
           2. Two new FASTQ files for those unmapped paired reads.
        9. For each sample, it writes a separate SLURM submission script that:
           1. Requests computing resources (30 CPUs, 120 GB RAM, 3-hour runtime).
           2. Loads the Anaconda3 module and activates a Conda environment with Bowtie2 installed.
           3. Runs Bowtie2 to align reads against known contamination genomes.
           4. Keeps only the read pairs where both reads didn’t match (using samtools view -f 12).
           5. Sorts the resulting BAM file by read name (with samtools sort -n).
           6. Converts this sorted BAM file back to paired FASTQ files (with bedtools bamtofastq).

Uses pigz to compress the two fastq files to .gz files.

* + - 1. After creating each sample’s SLURM script, the main script uses sbatch to submit it to the cluster.
      2. Prints info about each job so the user knows what’s happening.
      3. At the end, you have:
         1. A cleaned BAM file of reads that didn’t match known contaminants.
         2. New paired FASTQ files of these clean reads.
         3. Log files recording what happened during the job.
         4. The SLURM job script itself (for reproducibility).
    1. **In short:**
       1. This script automates the process of removing known contaminant sequences from many sequencing samples by:
          1. Creating one job script per sample.
          2. Submitting each to the cluster.
          3. Keeping only clean, unmapped reads for further analysis.

See submission file **contamination\_removal\_array\_v1.slurm** scriptin the Contamination\_Control directory for more details.

1. **Assemble reads into Metagenomic contigs: metaspades\_array\_v1.slurm**
   1. Here is where we will assemble our cleaned reads into metagenomic contigs. As with the decontamination step, this SLURM script will automatically create individual submission scripts for each sample.
   2. The script is described below:
      1. Its main purpose is to automatically create and submit separate jobs to assemble each samples set of cleaned reads.
      2. Below are details for the script metaspades\_array\_v1*.slurm*:
         1. WORK\_DIR: where the raw sequencing data is stored.
         2. JOB\_DIR: where the new job scripts and their logs will be saved.
         3. Makes sure the job directory exists (mkdir -p).
         4. Moves into the working directory to look for data.
         5. Looks for folders starting with “PN” (e.g., PN12345) – Change accordingly.
         6. Inside each folder, looks for files ending in \_unmapped\_R1.fastq.gz (the first read in paired-end sequencing) – Change accordingly (This is the output format provided in the decontamination script).
         7. For each such file:
            1. Figures out the name of the matching “R2” file.
            2. Uses the folder name as the sample name (PNxxx).
         8. For each sample, creates placeholder variable name for:
            1. Output directory name for assembly results (replacing \_unmapped\_R1.fastq.gz with \_metaspades)
         9. Checks if contigs.fasta already exists in the output folder:
            1. **If it does →** skip this sample (don’t submit a new job).
            2. **If it doesn’t →** prepare a new SLURM job script.
         10. For each sample, it writes a separate SLURM submission script that:
             1. Requests computing resources (30 CPUs, 350 GB RAM, 24-hour runtime).
             2. Loads the Anaconda3 module and activates a Conda environment with spades\_4.2.0 installed.
             3. Runs metaspades to metagenomically assemble the cleaned and decontamination reads.
             4. Keeps only the read pairs where both reads didn’t match (using samtools view -f 12).
             5. Sorts the resulting BAM file by read name (with samtools sort -n).
             6. Converts this sorted BAM file back to paired FASTQ files (with bedtools bamtofastq).
         11. After creating each sample’s SLURM script, the main script uses sbatch to submit it to the cluster.
         12. Prints info about each job so the user knows what’s happening.
         13. At the end, you have:
             1. A cleaned BAM file of reads that didn’t match known contaminants.
             2. New paired FASTQ files of these clean reads.
             3. Log files recording what happened during the job.
             4. The SLURM job script itself (for reproducibility).
      3. **In short:**
         1. This script automates the process of removing known contaminant sequences from many sequencing samples by:
            1. Creating one job script per sample.
            2. Submitting each to the cluster.
            3. Keeping only clean, unmapped reads for further analysis.
   3. 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.
2. **Taxonomically annotate contigs with Kraken2:**
   1. A computer screen shot of a code

      Description automatically generated