Sequencing data pre-processing pipeline

Background & Uses

This document details how to take whole genome sequence data and process the reads to ensure they are ready for downstream analysis. You will need access to your reads from the HCGS, which will come in an email with instructions for download, as well as access to UNH premise. Prior to following this pipeline, you should fill out the Brown lab metadata file for this project, as it has naming conventions and read group information that is necessary for data pre-processing. For questions, please contact Alyssa Stasse.

Upload sequences to UNH premise

To begin, reads generated by the HCGS will need to be uploaded to UNH premise. Use your login to access premise and create a directory with your project title using mkdir command. Enter that directory using the cd command. Then, in that directory, create a directory titled Raw_reads. Enter that directory using the cd command. Then, to upload the reads to that file, use the wget command located in the email from HCGS. Below is an example from a previous study:

```
wget -r -np -R "index.html*" --http-user=user --http-password=password
https://cobb.sr.unh.edu/managed/........
```

These files will then be in the Raw_reads directory, but will be in several subdirectories, so you will have to go through and move them with the cp command to the main Raw_reads directory. You will have to manually go through and name each of the old files the new file name generated in the metadata template. This is to ensure that all files will be able to run through the scripts. To rename the files, enter the following code for each file:

```
cp old_file_name.fastq.gz new_file_name.fastq.gz | rm old_file_name.fastq.gz
```

After running this for each file, you should now have all files in the Raw_reads folder following the Brown lab naming convention, allowing them to be used in the scripts later in this document.

Download reference genome from NCBI

In order to analyze population structures, you will need to have a reference genome for the organism of interest. To do this, you can Google search "organism reference genome." This will then take you to the NCBI website for your organism's reference genome. On the right-hand side of the webpage should be a link that says "Download Assembly." Click the link, select "GenBank" from the first drop-down menu, then download to your computer. Once it is downloaded, unzip the file. You will only need the assembled reference genome (.fna file).

Upload reference genome to UNH premise

In order to work with the reference genome, it needs to be uploaded to UNH premise. Move the reference file to a directory in which you know the path. You should create a new directory titled Reference using mkdir and upload the reference genome to that directory. Then, use the following command to upload:

```
scp -r path/on/your/computer/to/reference/reference.fna
login@premise.sr.unh.edu:path/to/directory/Reference/reference.fasta
```

Index the reference genome

Once the reference genome is uploaded to UNH premise, it needs to be indexed in order to work with it. To do this, you will need to copy a slurm script in the Brown lab shared scripts directory to your own directory. Make a new directory in your directory called scripts. Copy the ReferenceIndex.slurm script to your directory using the following command:

```
cp /mnt/home/ecogen/shared/scripts/ReferenceIndex.slurm
/path/to/your/directory/scripts/ReferenceIndex.slurm
```

Open ReferenceIndex.slurm using vim ReferenceIndex.slurm. To edit in vim, hit the i key on your keyboard. To exit editing mode, hit the escape key. To close vim without saving, type :q and hit enter. To save and quit, type :wq and hit enter. In ReferenceIndex.slurm, the script should look like the following:

```
## Load the appropriate modules first.  Linuxbrew/colsa contains most
## programs, though some are contained within the anaconda/colsa
## module.  Refer to http://premise.sr.unh.edu for more info.
module purge
module load linuxbrew/colsa
bwa -h
samtools -h
picard -h

## Instruct your program to make use of the number of desired threads.
## As your job will be allocated an entire node, this should normally
## be 24.
cd /path/to/Reference

bwa index reference.fasta

samtools faidx reference.fasta

picard CreateSequenceDictionary \
R=reference.fasta
```

In the script, add your file path and the name to your reference genome. To run the script, use the following command:

```
sbatch --exclude=node\[117-118\] ReferenceIndex.slurm
```

When this is complete, you should have a slew of different reference files in your reference directory. Do not move these files.

Data pre-processing: mapping reads to reference and sorting

Now that the reference genome is uploaded into your premise directory and all files needed have been created, you can start pre-processing your data. This will include mapping the reads to the reference and sorting those reads in the correct .bam file. Ensure that your files are named in the Brown lab file naming convention, or else this script will not work. Write out the following command to get the first pre-processing script in

your script directory:

```
cp /mnt/home/ecogen/shared/scripts/ReadsPreProcess_1.slurm
/path/to/your/script/directory/ReadsPreProcess_1.slurm
```

Navigate into your scripts directory and open the script using vim ReadsPreProcess_1.slurm. The slurm script will look like the following:

```
## Normal Slurm options
## SBATCH -p shared
#SBATCH --job-name="RPP1"
#SBATCH --output="RPP1.output"

## Load the appropriate modules first.  Linuxbrew/colsa contains most
## programs, though some are contained within the anaconda/colsa
## module.  Refer to http://premise.sr.unh.edu for more info.
module purge
module load linuxbrew/colsa
bwa
samtools

## Instruct your program to make use of the number of desired threads.
## As your job will be allocated an entire node, this should normally
## be 24.

for fname in /path/to/your/directory/Raw_reads/*_R1_#.fastq.gz
do
        base=${fname%_R1_#.fastq.gz}
        bwa mem \
        -M \
        -t 24 \
        -v 3 \
        /path/to/your/reference/reference.fasta \
        "${base}_R1_#.fastq.gz" "${base}_R2_#.fastq.gz" -t 24 > "${base}_mem.bam"
done

wait

for fname in /path/to/your/directory/Raw_reads/*_mem.bam
do
        base=${fname%_mem.bam}
        samtools sort "${base}_mem.bam" > "${base}_mem2.bam"
done
```

You will have to fill in the path to your directory in the places indicated in the script, and where there is a # in the for loop, put the last number in your new file name. Make the necessary edits to your script using the i key to get into edit mode, then the escape key to exit edit mode. Once edits are made, save and exit using :wq and hitting enter. Now, the code should be ready to run.
To run the code, type the following command:

```
sbatch --exclude=node\[117-118\] ReadsPreProcess_1.slurm
```

The script will then start running. This could take a while depending on sample size. To check the status

of the run and how long it has been running, use the command squeue ReadsPreProcess_1.slurm. When the command is done running, you should have 2 new files for every set of reads in your Raw_reads directory. Move all of the new files (_mem.bam and _mem2.bam) to a new directory in this project titled Aligned_reads using the following command:

```
cp /path/to/your/Raw_reads/*_mem.bam /path/to/your/Aligned_reads/*_mem.bam
```

When all files have copied to the new directory, ensure that they are in the Aligned_reads directory. Once you know they are all there, use the following command to remove them from the Raw_reads directory:

```
rm /path/to/your/Raw_reads/*_mem.bam
```

Now, you should have sorted and aligned reads for each sample in the Aligned_reads directory. If there are any issues at this point, address them before moving on.

Data pre-processing: removing and adding new header groups and indexing files

In sequencing data, you need read groups to process the data. However, we need to reformat the read groups slightly in order for downstream analyses to work. To do this, we will be using the script below and the metadata sheet. If you open one of your raw data files, you will see a string of letters and numbers (referenced in the metadata file). You should have already used this to fill out the metadata and should know where to find each section. In the below script, you should fill in the path (as always), as well as the sections were it says sec1, sec2, sec3, and sec4. This corresponds to the information in your metadata sheet. Also in your metadata sheet is each of the RG components, so you can just copy and paste.

Copy the template script from the shared script folder using the following command:

```
cp /mnt/home/ecogen/shared/scripts/ReadsPreProcess_2.slurm
/path/to/your/directory/scripts/ReadsPreProcess_2.slurm
```

Once this is copied, go into the script using the vim command and edit the directory paths as well as the RGID and parts of the RGPU. Make sure that no quotations, brackets, or dollar signs get removed or moved around. The template script should look like this:

```
## Normal Slurm options
## SBATCH -p shared
#SBATCH --job-name="RPP2"
#SBATCH --output="RPP2.output"

## Load the appropriate modules first.  Linuxbrew/colsa contains most
## programs, though some are contained within the anaconda/colsa
## module.  Refer to http://premise.sr.unh.edu for more info.
module purge
module load linuxbrew/colsa
bwa
samtools
picard

## Instruct your program to make use of the number of desired threads.
## As your job will be allocated an entire node, this should normally
## be 24.

for fname in /path/to/your/Aligned_reads/*_mem2.bam
```

```
do
        base=${fname%_mem2.bam}
        samtools view -H "${base}_mem2.bam" | grep -v "^@RG" |
        samtools reheader - "${base}_mem2.bam" > "${base}_mem3.bam"
done

wait

for fname in /path/to/your/Aligned_reads/*_mem3.bam
do
        base=${fname%_mem3.bam}
        ID="${fname:2:4}"
        picard AddOrReplaceReadGroups \
        I="${base}_mem3.bam" \
        O="${base}_align.bam" \
        RGID=sec1.sec2.sec3 \
        RGLB="${ID}.${ID}" \
        RGPL=illumina \
        RGPU="sec1.sec2.sec3.sec4.${ID}.${ID}" \
        RGSM="${ID}"
done

wait

for fname in /path/to/your/Aligned_reads/*_align.bam
do
        base=${fname%_align.bam}
        samtools index "${base}_align.bam"
done
```

Once you are done editing, save your script and run it with the following command:

```
sbatch --exclude=node\[117-118\] ReadsPreProcess_2.slurm
```

The script will then start running. Once complete, you should have 2 new files for each sample, _mem3.bam and _align.bam. The latter is what you will be using for validation, and if it passes validation, for downstream analyses.

Validating pre-processed files

Luckily, there is a tool built into picard that validates .sam or .bam files for use in downstream analyses. To do this, you will need to copy the validate script to your scripts directory using the following command:

```
cp /mnt/home/ecogen/shared/scripts/ValidateFiles.slurm
/path/to/your/script/directory/ValidateFiles.slurm
```

When you open the script with vim, it should look like the following:

```
## Normal Slurm options
## SBATCH -p shared
#SBATCH --job-name="validate"
#SBATCH --output=validate.output
```

```
## Load the appropriate modules first.  Linuxbrew/colsa contains most
## programs, though some are contained within the anaconda/colsa
## module.  Refer to http://premise.sr.unh.edu for more info.
module purge
module load linuxbrew/colsa
picard -h

## Instruct your program to make use of the number of desired threads.
## As your job will be allocated an entire node, this should normally
## be 24.

for file in /path/to/your/Aligned_reads/*_align.bam
        do
        picard ValidateSamFile \
        I=$file \
        IGNORE_WARNINGS=TRUE \
        MODE=VERBOSE
done
```

Make sure to insert your path information into the script. To run the script, type the following command:

```
sbatch --exclude=node\[117-118\] ValidateFiles.slurm
```

The output, which will be in the scripts directory as validate.output, will show any errors once the script is done running. Open the output file using the vim command. Read through the output file. If there are no errors, there will be several lines saying "No errors found." However, if there are errors, though, it will pop up with ERROR and list out the error. Make sure to address these errors before any downstream analyses, or else programs will not work.

Now you have pre-processed sequencing data. Once you have validated with no errors, you are more than welcome to delete all intermediate files and just keep the _align.bam and the raw reads files. If not, I recommend moving all intermediate files into a different directory. If there are any issues, make sure to not move on before the next step and reach out to Alyssa Stasse if you need help. Most errors are due to a typo, and most error messages can be fixed with a simple Google. Best of luck!