# Overall

**Login Euler**

ssh [jfrank@euler.ethz.ch](mailto:jfrank@euler.ethz.ch)

password: normal ETH password

**Run script on euler**

vi run.sh 🡪 copy script 🡪 esc 🡪 :wq 🡪 to save

sbatch run.sh

squeue 🡪 to track queue

sacct --format JobID,State,AllocCPUS,Elapsed,NNodes,NTasks,TotalCPU,REQMEM,MaxRSS,NodeList -j 20203754 🡪 get more info on the job

**Download files of Euler**

scp -r jfrank@euler.ethz.ch:/cluster/scratch/jfrank/rsem C:/Users/julia/Downloads

# Preprocesing

## Trimming

#!/bin/bash

#SBATCH --job-name=trim # Job name

#SBATCH --output=/cluster/scratch/jfrank/scripts/output/trim\_output.log # Output file

#SBATCH --error=/cluster/scratch/jfrank/scripts/output/trim\_error.log # Error log

#SBATCH --time=10:00:00 # Adjust time as needed

#SBATCH --mem-per-cpu=10G # Adjust memory per CPU as needed

#SBATCH --cpus-per-task=1 # Number of CPUs per task

# Initialize Conda for bash shell

source /cluster/home/jfrank/miniconda3/etc/profile.d/conda.sh

conda activate env

# Verify that fastp is available

which fastp || { echo "fastp not found in environment"; exit 1; }

# Navigate to the rawdata directory

cd "/cluster/scratch/jfrank/rawdata" || { echo "Directory not found"; exit 1; }

# Create output directory under processed\_data if it doesn't already exist

output\_dir="/cluster/scratch/jfrank/processed\_data/trimmed\_data"

mkdir -p "$output\_dir"

# Process paired-end files with fastp

for file in \*\_1.fastq.gz; do

# Derive the corresponding R2 file and output file names

sample\_name=$(basename "$file" \_1.fastq.gz)

file\_R1="${sample\_name}\_1.fastq.gz"

file\_R2="${sample\_name}\_2.fastq.gz"

echo "Starting processing $file\_R1 and $file\_R2"

# Set output file paths in trimmed\_data directory within processed\_data

output\_R1="${output\_dir}/${sample\_name}\_trimmed\_1.fastq.gz"

output\_R2="${output\_dir}/${sample\_name}\_trimmed\_2.fastq.gz"

# Run fastp with trimming and quality filtering

fastp -l 30 \

-q 20 \

-i "$file\_R1" -I "$file\_R2" \

-o "$output\_R1" -O "$output\_R2" \

--thread 1

echo "Processed $file\_R1 and $file\_R2"

done

echo "All files processed with trimming!"

## Mapping

**Creating Indexing for the reference genome**

#!/bin/bash

#SBATCH -n 10

#SBATCH --time=10:00:00

#SBATCH --mem-per-cpu=10G

#SBATCH --job-name=star\_genome\_index

#SBATCH --output=star\_genome\_index.out

#SBATCH --error=star\_genome\_index.err

#SBATCH --mail-type=END,FAIL #Email to me when finished

# Change to the appropriate directory

cd /cluster/scratch/jfrank/genome

# Generate genome index

STAR --runThreadN 10 \

--runMode genomeGenerate \

--genomeDir star-index \

--genomeSAindexNbases 12 \

--genomeFastaFiles dm6.fa

**Download anotations**

wget -O genome/dm6.refGene.gtf.gz <https://hgdownload.soe.ucsc.edu/goldenPath/dm6/bigZips/genes/dm6.ncbiRefSeq.gtf.gz>

gzip -d genome/dm6.refGene.gtf.gz

**Creating RSEM Index**

mkdir genome/rsem\_dm6

rsem-prepare-reference --gtf genome/dm6.refGene.gtf \ genome/dm6.fa \ genome/rsem\_dm6/rsem\_dm6

**Actual Mapping**

#!/bin/bash

#SBATCH --job-name=star\_rsem\_pipeline # Job name

#SBATCH --output=/cluster/scratch/jfrank/scripts/output/star\_rsem\_output.log # Output log file

#SBATCH --error=/cluster/scratch/jfrank/scripts/output/star\_rsem\_error.log # Error log file

#SBATCH --time=23:30:00 # Maximum execution time

#SBATCH --mem-per-cpu=8G # Memory per CPU

#SBATCH --cpus-per-task=12 # Number of CPUs to use

# Initialize Conda for bash shell

source /cluster/home/jfrank/miniconda3/etc/profile.d/conda.sh

# Activate the conda environment for STAR and RSEM

conda activate env

# Change directory to the scratch folder

cd /cluster/scratch/jfrank

# Ensure required directories exist

mkdir -p mapping\_transcriptome

mkdir -p rsem

# Loop through each sample ID listed in SRR\_Acc\_List.txt

for id in SRR1197426 SRR1197325 SRR1197425 SRR1197324 SRR1197424 SRR1197326 SRR1197420 SRR1197287 SRR1197419 SRR1197285 SRR1197416 SRR1197286; do

echo "Starting to process sample $id"

# Create a directory for mapping if it doesn't already exist

if [ ! -e mapping\_transcriptome/$id ]; then

echo "Mapping started for $id"

mkdir mapping\_transcriptome/$id

STAR --genomeDir genome/star-index \

--runThreadN 10 \

--readFilesIn processed\_data/trimmed\_data/${id}\_trimmed\_1.fastq.gz processed\_data/trimmed\_data/${id}\_trimmed\_2.fastq.gz \

--readFilesCommand zcat \

--sjdbGTFfile genome/dm6.refGene.gtf \

--quantMode TranscriptomeSAM \

--outSAMtype BAM SortedByCoordinate \

--outFileNamePrefix mapping\_transcriptome/$id/

echo "Mapping done for $id"

fi

# Run RSEM if it hasn't been done for this sample

if [ ! -e rsem/$id ]; then

echo "RSEM started for $id"

mkdir rsem/$id

# Count FASTQ files to check if data is paired-end or single-end

num\_fa=$(ls -1 processed\_data/trimmed\_data/${id}\_trimmed\*.fastq.gz | wc -l)

# Choose RSEM command based on paired-end or single-end data

if [ $num\_fa -eq 1 ]; then

rsem-calculate-expression --alignments \

-p 10 \

mapping\_transcriptome/$id/Aligned.toTranscriptome.out.bam \

genome/rsem\_dm6/rsem\_dm6 \

rsem/$id/$id

else

rsem-calculate-expression --alignments \

--paired-end \

-p 10 \

-q \

mapping\_transcriptome/$id/Aligned.toTranscriptome.out.bam \

genome/rsem\_dm6/rsem\_dm6 \

rsem/$id/$id

fi

echo "RSEM done for $id"

fi

done