Short-reads data analysis

Tomasz Mądry

2025-07-11

Overview

- Abstract
- Data Description
- Important directories
- Indexing genome
- Quality check
  - FastQC
  - MultiQC
- Mapping reads
- Read genome coverage
- Salmon
  - Overview
  - Indexing transcriptome
  - Quantification

Abstract

The RNA-seq data used for the analysis comes from samples of a human lung epithelial adenocarcinoma cell line (A549). Samples were treated with 100 nM of dexamethasone, a synthetic glucocorticoid.

Polyadenylated mRNA was collected from treated samples at two different time intervals following treatment initiation, and from control samples. For each time point there are 4 biological replicates. Hence, in total we have 12 samples, 4 samples for 0 hours (control), 4 samples for 2 hours after treatment, and 4 samples for 4 hours after treatment.

Paired-end RNA sequencing was conducted to examine the effect of dexamethasone on human cells.

Data Description

- This is a subset of a larger dataset available from the ENCODE project (https://www.encodeproject.org/treatment-time-series/ENCSR897XFT/)
- Human lung epithelial adenocarcinoma cell line A549
- RNA-seq experiments after treatment with 100 nM dexamethasone
- Includes control samples (0 exposure to dex) and samples treated for 2 and 4 hours
- 4 biological replicates per condition
- Polyadenylated mRNA
- Strand-specific (forward)
- Paired-end reads

Important directories

Raw data directory:

```
~/pl0534-01/project_data/data/short-reads/raw_data/
```

Required software:

```
~/pl0534-01/project_data/soft/
```

Required scripts:

```
~/pl0534-01/project_data/scripts/
```

Add software to PATH - SIMPLIFIED

Just paste this in your terminal:

```
# Add to PATH
cat $HOME/pl0534-01/project_data/soft/paths.txt >> $HOME/.bashrc

# Reload .bashrc
source $HOME/.bashrc
```

Add software to PATH

Read paths from file:

```
cat ~/pl0534-01/project_data/soft/paths.txt
```

Select them and copy using `ctrl` + `shift` + `c`

Open `~/.bashrc` file using nano text editor:

```
nano ~/.bashrc
```

Navigate to the bottom of the file using DOWN arrow key. Then paste necessary paths pressing `ctrl` + `shift` + `v`.

Save changes by pressing `ctrl` + `x`, then `y` and confirm with `enter`

Reload `~/.bashrc` file:

```
source ~/.bashrc
```

From now on, you can run software from different directories.

Create working directory

Navigate to `scratch` dir:

```
cd ~/pl0534-01/scratch/
```

Create your working directory and navigate there:

```
mkdir $USER
cd $USER
```

Here you will store all your results.

Copy necessary scripts

Copy `scripts` directory into your `$HOME/pl0534-01/scratch/$USER/` dir:

```
cp -r $HOME/pl0534-01/project_data/scripts/ $HOME/pl0534-01/scratch/$USER/
```

Indexing genome - directories

Create gencode directory to store annotations, genome etc.

```
cd ~/pl0534-01/scratch/
mkdir gencode && cd gencode
```

Now, we need to download the latest annotation and genome files from GENCODE.

Indexing genome - download files

Go to GENCODE website: https://www.gencodegenes.org/

Download annotation in GTF format:

Download human genome from the same website:

Check the files:

```
ls -l
```

```
GRCh38.p14.genome.fa.gz      gencode.v48.annotation.gtf.gz
```

Indexing genome - run interactive job

Don't run compute jobs on log-in nodes because:

- They're shared by all users;
- They're not managed by Slurm's scheduler;
- Heavy usage can harm performance cluster-wide;
- It may violate your cluster's usage policies;

Use `sbatch`, `salloc`, or `srun` on compute nodes instead.

How to start an interactive job:

```
tmux new-session -s star                              # start new session called s
srun --pty --cpus-per-task=33 --mem=64G /bin/bash     # allocate resources
```

You should see the following:

```
klug3@eagle:~$ srun --pty --cpus-per-task=33 --mem=64G /bin/bash
srun: job 35144745 queued and waiting for resources
srun: job 35144745 has been allocated resources
```

Indexing genome - decompress input files

Navigate to /gencode directory and create `index_genome` directory

```
cd ~/pl0534-01/scratch/$USER/gencode/
mkdir genome_index
```

Decompress genome file:

```
gzip -d GRCh38.p14.genome.fa.gz
```

Decompress annotation file:

```
gzip -dk gencode.v48.annotation.gtf.gz # but keep original one as well
```

Indexing genome - run STAR

Now, run STAR with parameters:

```
STAR --runThreadN 32 \
--runMode genomeGenerate \
--genomeDir genome_index \ # dir to store output
--genomeFastaFiles GRCh38.p14.genome.fa \
--sjdbGTFfile gencode.v48.annotation.gtf \
```

Same, but easy to copy:

```
STAR --runThreadN 32 --runMode genomeGenerate --genomeDir genome_index
--genomeFastaFiles GRCh38.p14.genome.fa --sjdbGTFfile
gencode.v48.annotation.gtf
```

After completion (~30 mins) `genom_index` directory should contain 16 files:

```
klug3@eagle:~$ ls ~/pl0534-01/scratch/$USER/gencode/genome_index/ | wc -l
16
```

We can free allocated resources:

```
exit # quit interactive slurm job
```

Quality check - FastQC

A quality control tool for high throughput sequence data.

Navigate to `$HOME/pl0534-01/scratch/$USER/scripts/` directory:

```
cd $HOME/pl0534-01/scratch/$USER/scripts/
```

Since we have 3 samples with 4 replicates each, and sequencing was performed using a paired-end reads protocol, there are a total of **24** input files. To speed up the data processing step, we will schedule 24 parallel jobs using `sbatch` Slurm command.

Run `FastQC` in parallel using the Slurm `sbatch` job scheduler:

```
for file in
$HOME/pl0534-01/project_data/data/short-reads/raw_data/*fastq.gz; do
sbatch
--output="$HOME/pl0534-01/scratch/$USER/scripts/logs/slurm-%j.out"
--error="$HOME/pl0534-01/scratch/$USER/scripts/logs/slurm-%j.err"
fastqc.sh $file; done
```

FastQC - monitor progress

To monitor progress, use the following command:

```
squeue -u $USER
```

Or, we can use `check_jobs.sh` shell script to refresh the list every 3 seconds:

```
./check_jobs.sh fastqc
```

After a couple of minutes (~4 mins per job) output should be available in `$HOME/pl0534-01/scrach/$USER/fastqc/` directory.

FastQC - download summaries

To preview the summaries, they need to be downloaded first. Open another terminal window on your local machine, navigate to the `~/Desktop/MITGEST_training/` directory, and create a `fastqc_output/` folder:

```
mkdir ~/Desktop/MITGEST_training/fastqc_output/
cd ~/Desktop/MITGEST_training/fastqc_output/
```

Next, use the SCP protocol to download the generated plots:

```
scp <your_name>@eagle.man.poznan.pl:~/pl0534-01/scratch/<your_name>/fastqc/*.ht
```

Use your system's default browser (e.g., Chrome) to preview the results.

MutliQC - Python 3 venv - SIMPLIFIED

Aggregate bioinformatics results across many samples into a single report

MultiQC is a Python 3 package and can be installed using the pip package manager. However, installing Python packages globally is generally considered bad practice, as it can lead to conflicts between dependencies. It's recommended to use a virtual environment or a tool like conda to manage project-specific installations.

Activate Python 3 virtual environment:

```
# Allocate resources for interactive job:
srun --pty --cpus-per-task=2 --mem=8G /bin/bash
# Activate Python 3 venv:
source $HOME/pl0534-01/project_data/multiqc_venv/bin/activate
```

You should see the following:

```
(multiqc_venv)
klug3@e1958:/mnt/storage_3/home/klug3/pl0534-01/scratch/klug3/scripts$
```

MutliQC - Python 3 venv

Aggregate bioinformatics results across many samples into a single report

MultiQC is a Python 3 package and can be installed using the pip package manager. However, installing Python packages globally is generally considered bad practice, as it can lead to conflicts between dependencies. It's recommended to use a virtual environment or a tool like conda to manage project-specific installations.

Creating Python 3 virtual environment (venv):

```
# Allocate resources for interactive job:
srun --pty --cpus-per-task=2 --mem=8G /bin/bash

# Navigate to your scripts/ directory
cd $HOME/pl0534-01/scratch/$USER/scripts/

# Create new Python 3 virtual environment:
python3 -m venv multiqc_venv

# Activate created venv
source multiqc_venv/bin/activate
```

You should see the following:

```
(multiqc_venv)
klug3@e1958:/mnt/storage_3/home/klug3/pl0534-01/scratch/klug3/scripts$
```

MultiQC - installation in multiqc_venv

Make sure, that your virtual environment is active by:

```
which python3
```

You should see the path to your multiqc_venv:

```
mnt/storage_5/scratch/pl0534-01/klug3/scripts/multiqc_venv/bin/python3
```

MultiQC installation:

```
# Upgrade pip manager
python3 -m pip install --upgrade pip

# Install MultiQC
python3 -m pip install multiqc
```

Run MultiQC with --version parameter to verify the installation:

```
multiqc --version
```

MultiQC - create interactive report

Run MultiQC on the previously generated FastQC output files:

```
multiqc $HOME/pl0534-01/scratch/$USER/fastqc/ --outdir $HOME/pl0534-01/scratch/
```

To preview the report, it needs to be downloaded first. Open another terminal window on your local machine, create ~/Desktop/MITGEST_training/multiqc_output directory, and navigate there:

```
cd ~/Desktop/MITGEST_training && mkdir multiqc_output/ && cd multiqc_output/
```

Next, use the SCP protocol to download the generated plots:

```
scp -r <your_name>@eagle.man.poznan.pl:~/pl0534-01/scratch/<your_name>/multiqc/
```

Use your system's default browser (e.g., Chrome) to preview the results.

We can now deactivate the multiqc_venv virtual environment and release the allocated resources:

```
deactivate # deactivate virtual environment
exit       # quit interactive job session
```

Mapping reads - STAR parameters

Navigate back to the ~/pl0534-01/scratch/$USER/ directory

Copy ~/pl0534-01/project_data/scripts/ directory and go there:

```
cp -r ~/pl0534-01/project_data/scripts/ .
cd scripts/
```

STAR mapping parameters:

```
STAR --runThreadN 16 \
--genomeDir $GENOMEDIR \
--readFilesIn ${SAMPLE}_r1.fastq.gz ${SAMPLE}_r2.fastq.gz \
--outFileNamePrefix $SAMPLE \
--readFilesCommand zcat \
--outReadsUnmapped Fastx \
--outFilterMultimapNmax 10 \
--outFilterMultimapScoreRange 0 \
--outSAMunmapped None \
--outSAMtype BAM Unsorted \
--outFilterScoreMinOverLread 0 \
--outFilterMatchNminOverLread 0 \
--outFilterMatchNmin 16 \
--outFilterMismatchNmax 1 \
--alignSJDBoverhangMin 1000 \
--alignIntronMax 1
```

Mapping reads - scheduling jobs

Run star_mapping script for each sample:

```
for file in $(ls
$HOME/pl0534-01/project_data/data/short-reads/raw_data/*_r1.fastq.gz |
awk -F'.' '{print$1}' | sed 's/_r1//g'); do sbatch
--output="$HOME/pl0534-01/scratch/$USER/scripts/logs/slurm-%j.out"
--error="$HOME/pl0534-01/scratch/$USER/scripts/logs/slurm-%j.err"
star_mapping.sh $file; done
```

To monitor progress, use the following command:

```
./check_jobs.sh star_mapping
```

Results will appear in `$HOME/pl0534-01/scratch/$USER/mapped/` directory.

Read genome coverage - Sorting bam files by coordinates

Used samtools sort parameters:

```
samtools sort -@8 \
${IN_FILE_NAME} \
-o ${OUT_FILE_NAME}
```

To run `Python2 read_genome_coverage` script, we need to sort our `bam` files:

```
for file in $HOME/pl0534-01/scratch/$USER/mapped/*Aligned.out.bam; do sbatch --
```

To monitor progress, use the following command:

```
./check_jobs.sh sort_bams
```

Output will be in `$HOME/pl0534-01/scratch/$USER/mapped_sorted/` directory.

Sorting should take about 10 minutes.

Read genome coverage - index genome

Besides sorted BAM files, we also need the reference genome index file (.fa.fai)

To do so we will use `samtools faidx` software.

Navigate to `$HOME/pl0534-01/scratch/$USER/gencode/` directory and allocate resources.

```
cd $HOME/pl0534-01/scratch/$USER/gencode/
srun --pty --cpus-per-task=16 --mem=24G /bin/bash
```

Run indexing on `GRCh38.p14.genome.fa` genome file.

```
samtools faidx -@16 GRCh38.p14.genome.fa -o GRCh38.p14.genome.fa.fai
```

Free allocated resources:

```
exit
```

Read genome coverage - Python 2

With all the necessary files in place, we can finally run `read_genome_coverage.sh` in parallel:

Navigate to `$HOME/pl0534-01/scratch/$USER/scripts/` directory:

```
cd $HOME/pl0534-01/scratch/$USER/scripts/
```

And run `read_genome_coverage.sh` (~15 mins):

```
for file in $HOME/pl0534-01/scratch/$USER/mapped_sorted/*sorted.bam; do
sbatch
--output="$HOME/pl0534-01/scratch/$USER/scripts/logs/slurm-%j.out"
--error="$HOME/pl0534-01/scratch/$USER/scripts/logs/slurm-%j.err"
read_genome_coverage.sh $file; done
```

To monitor progress, use the following command:

```
./check_jobs.sh read_genome_coverage
```

Output is located in `$HOME/pl0534-01/scratch/$USER/genome_coverage/` directory.

Read genome coverage - prepare R

First, we need to install the necessary R packages. To do this, we need to create a .Renviron file with the following content:

```
echo "R_LIBS_USER=~/pl0534-01/scratch/$USER/scripts/Rpackages" > $HOME/.Renviro
```

Then, we need to allocate the necessary resources, load the R module and start R:

```
srun --pty --cpus-per-task=2 --mem=6G /bin/bash
module avail # check available modules
module load r/4.4.1-gcc-14.2.0 # load R module
R # start R
```

Finally, we can start installing packages:

```
install.packages(c("optparse", "reshape2", "ggplot2", "plyr"))
```

To begin the installation, answer **"Yes"** to the first two prompts.

Then, select a CRAN mirror—option **"57"** for Poland.

The installation should take about 5 minutes. To exit R without saving your workspace, use the `quit(save = 'no')` command.

Read genome coverage - plot using R

The final step is to plot the data. To do this, we need to navigate to the `$HOME/pl0534-01/scratch/$USER/scripts/` directory, and run the R `read.genomic.coverage.R` script:

```
cd $HOME/pl0534-01/scratch/$USER/scripts/
for file in $HOME/pl0534-01/scratch/$USER/genome_coverage/*.txt; do name=`echo
```

Plots in `pdf` format are saved to `$HOME/pl0534-01/scratch/$USER/genome_coverage/` directory.

To preview the plots, they need to be downloaded first. Open another terminal window on your local machine, navigate to the `Desktop/` directory, and create a `genome_coverage_plots/` folder:

```
mkdir ~/Desktop/MITGEST_training/genome_coverage_plots/ && cd ~/Desktop/MITGEST
```

Next, use the SCP protocol to download the generated plots:

```
scp <your_name>@eagle.man.poznan.pl:~/pl0534-01/scratch/<your_name>/genome_cove
```

Use your system's default PDF viewer to preview the plots.

Salmon

Overview

Salmon is a fast and accurate bioinformatics tool for quantifying transcript-level expression from RNA-seq data. It uses lightweight quasi-mapping or pre-aligned reads to estimate transcript abundances while correcting for sequencing biases such as GC content and positional effects. Salmon outputs TPM, raw counts, and effective counts, making it compatible with downstream tools like `DESeq2`. Its speed, scalability, and bias-aware algorithms make it a popular choice for large-scale transcriptomic studies.

The typical Salmon workflow includes:

- Indexing a reference transcriptome;
- Quantifying expression using RNA-seq reads;
- Exporting results for downstream analysis (e.g., differential expression with DESeq2 or edgeR).

Salmon: indexing transcriptome

Allocate resources and navigate to gencode/ directory:

```
srun --pty --cpus-per-task=17 --mem=32G /bin/bash
cd $HOME/pl0534-01/scratch/$USER/gencode/
```

Download transcriptome from GENCODE:

```
wget
https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_48/gencode.v4
```

And decompress it:

```
gzip -d gencode.v48.transcripts.fa.gz
```

Run indexing using salmon

```
salmon index -p 16 -t gencode.v48.transcripts.fa -i salmon_index --gencode
```

After completion, you can free allocated resources by `exit` command.

Output will be in `salmon_index` directory.

Salmon: quantification

Salmon quant parameters:

```
salmon quant -p 16 \
-i ${SALMON_INDEX} \
-l A \
-1 ${SAMPLE}_r1.fastq.gz \
-2 ${SAMPLE}_r2.fastq.gz \
--geneMap $ANNOT \
-o $SAMPLE
```

Running `salmon_quant.sh` script for each sample:

```
for file in $(ls
$HOME/pl0534-01/project_data/data/short-reads/raw_data/*_r1.fastq.gz |
awk -F'.' '{print$1}' | sed 's/_r1//g'); do sbatch
--output="$HOME/pl0534-01/scratch/$USER/scripts/logs/slurm-%j.out"
--error="$HOME/pl0534-01/scratch/$USER/scripts/logs/slurm-%j.err"
salmon_quant.sh $file; done
```

To monitor progress, use the following command (~11 mins per job):

```
./check_jobs.sh salmon_quant
```

Analyse data using DESeq2

To analyze the output from quantification using Salmon, we need to download the data. The desired `quant.sf` files for each sample are located in the `$HOME/pl0534-01/scratch/$USER/salmon_quant/` directory. To download them, we will use the SCP (Secure Copy Protocol).

First, open a new terminal window on your local machine. Navigate to the `~/Desktop` directory and create a new directory to store the Salmon output:

```
mkdir ~/Desktop/MITGEST_training/salmon_output/ && cd ~/Desktop/MITGEST_trainin
```

Then, download files:

```
scp <your_name>@eagle.man.poznan.pl:~/pl0534-01/scratch/<your_name>/salmon_quan
```

scp klug3@eagle.man.poznan.pl:~/pl0534-01/scratch/klug3/salmon_quant/*_quant.sf .

install.packages("renv")

renv::init()

Quality check - FastQC

Navigate to `$HOME/pl0534-01/scratch/$USER/scripts/` directory:

```
cd $HOME/pl0534-01/scratch/$USER/scripts/
```

Run FastQC in parallel using the Slurm `sbatch` job scheduler:

```
for file in
$HOME/pl0534-01/project_data/data/short-reads/raw_data/*fastq.gz; do
sbatch
--output="$HOME/pl0534-01/scratch/$USER/scripts/logs/slurm-%j.out"
--error="$HOME/pl0534-01/scratch/$USER/scripts/logs/slurm-%j.err"
fastqc.sh $file; done
```

To monitor progress, use the following command:

```
./check_jobs.sh fastqc
```

After a couple of minutes (~4 mins per job) output should be available in `$HOME/pl0534-01/scrach/$USER/fastqc/` directory.

FastQC - download summaries

To preview the summaries, they need to be downloaded first. Open another terminal window on your local machine, navigate to the `Desktop/` directory, and create a `fastqc_output/` folder:

```
mkdir ~/Desktop/MITGEST_training/fastqc_output/
cd ~/Desktop/MITGEST_training/fastqc_output/
```

Next, use the SCP protocol to download the generated plots:

```
scp <your_name>@eagle.man.poznan.pl:~/pl0534-01/scratch/<your_name>/fastqc/*.ht
```

Use your system's default browser (e.g.: Chrome) to preview the results.