Analyzing Illumina RNA-seq Data Using The CRI HPC

Wen-Ching Chan (http://cri.uchicago.edu/people/#chan)

Nov 2018

CRI RNAseq Pipelines

- CRI-RNAseq-2016 (https://github.com/riyuebao/CRI-Workshop-Nov2016-RNAseq/blob/master/Run_RNAseq.tutorial.ipynb) by Riyue Bao (http://cri.uchicago.edu/people/#bao). Last modified on **November 12, 2016**.
- CRI-RNAseq-2014 (https://wiki.uchicago.edu/pages/viewpage.action?pageId=95855827) by Lei Huang (http://cri.uchicago.edu/people/#huang). Last modified on **Apr 29, 2014**.

RUN AS PRACTICE

> **Pipeline package is available on GitHub (https://github.com/wenching/cri_rnaseq_2018/archive/cri_rnaseq_2018.tar.gz)**

The Center for Research Informatics (http://cri.uchicago.edu/) (CRI) provides computational resources and expertise in biomedical informatics for researchers in the Biological Sciences Division (BSD) of the University of Chicago.

As a bioinformatics core (http://cri.uchicago.edu/bioinformatics/), we are actively improving our pipelines and expanding pipeline functions. The tutorials will be updated in a timely manner but may not reflect the newest updates of the pipelines. Stay tuned with us for the latest pipeline release.

If you have any questions, comments, or suggestions, feel free to contact our core at bioinformatics@bsd.uchicago.edu (mailto:bioinformatics@bsd.uchicago.edu) or one of our bioinformaticians.

Quick Start | Top

1. modify the generator script **Build_RNAseq.DLBC.sh** accordingly
    1. project="PROJECT_AS_PREFIX" (e.g., **DLBC** which is used as a prefix of metadata file **DLBC**.metadata.txt and configuration file **DLBC**.pipeline.yaml)
    2. padding="DIRECTORY_NAME_CONTAINING_PROJECT_DATA" (e.g., **example** which is the folder name to accommodate metadata file, configuration file, sequencing data folder, and references folder)
2. prepare metadata file as the example **DLBC.metadata.txt (https://github.com/wenching/cri_rnaseq_2018/blob/master/example/DLBC.metadata.txt)**
    1. **Single End (SE)** Library
        1. Set Flavor column as 1xReadLength (e.g., 1x50)
        2. Set Seqfile1 column as the file name of the repective sequencing file
    2. **Paired End (PE)** Library
        1. Set Flavor column as 2xReadLength (e.g., 2x50)
        2. Set Seqfile1 column as the file name of the repective read 1 (R1) sequencing file
        3. Set an additional column named '**Seqfile2**' as the file name of the repective read 2 (R2) sequencing file
    3. **Non strand-specific** Library
        1. Set LibType column to NS
    4. **Strand-specific** Library
        1. Inquire the library type from your seuqencing center and set LibType column to FR (the left-most end of the fragment (in transcript coordinates, or the first-strand synthesis) is the first sequenced) or RF (the right-most end of the fragment (in transcript coordinates) is the first sequenced, or the second-strand synthesis). You can read this blog (http://onetipperday.sterding.com/2012/07/how-to-tell-which-library-type-to-use.html) for more details of strand-specific RNA-seq.
3. prepare reference files as the example reference hg38 under **[/CRI/HPC/cri_rnaseq_2018_ex/example/references/v28_92_GRCh38.p12)**
4. prepare pre-built STAR index files as the example reference hg38 under **[/CRI/HPC/cri_rnaseq_2018_ex/example/references/v28_92_GRCh38.p12/STAR)**

Introduction | Top

RNA sequencing (RNA-seq) is a revolutionary approach that uses the next-generation sequencing technologies to detect and quantify expressed transcripts in biological samples. Compared to other methods such as microarrays, RNA-seq provides a more unbiased assessment of the full range of transcripts and their isoforms with a greater dynamic range in expression quantification.

In this tutorial, you will learn how to use the CRI's RNA-seq pipeline (available on both CRI HPC cluster (http://cri.uchicago.edu/hpc/) and GitHub (https://github.com/wenching/cri_rnaseq_2018.git))) to analyze Illumina RNA sequencing data. The tutorial comprises the following Steps:

- Assess the sequencing data quality using FastQC (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/)
- Align the short reads to the target genome (e.g., Human) using STAR (https://github.com/alexdobin/STAR)
- Measure alignment result using Picard (https://broadinstitute.github.io/picard/) and RSeQC (http://rseqc.sourceforge.net/)
- Quantify expression using Subread (http://subread.sourceforge.net/)::featureCounts (http://bioinf.wehi.edu.au/featureCounts/)
- Identify differentially expressed genes using edgeR (https://bioconductor.org/packages/release/bioc/html/edgeR.html), DESeq2 (https://bioconductor.org/packages/release/bioc/html/DESeq2.html), limma (https://bioconductor.org/help/search/index.html?q=limma/)
- Heat Map using pheatmap (https://cran.r-project.org/web/packages/pheatmap/index.html)
- Conduct functional enrichment analysis using clusterProfiler (https://bioconductor.org/packages/release/bioc/html/clusterProfiler.html)

By the end of this tutorial, you will:

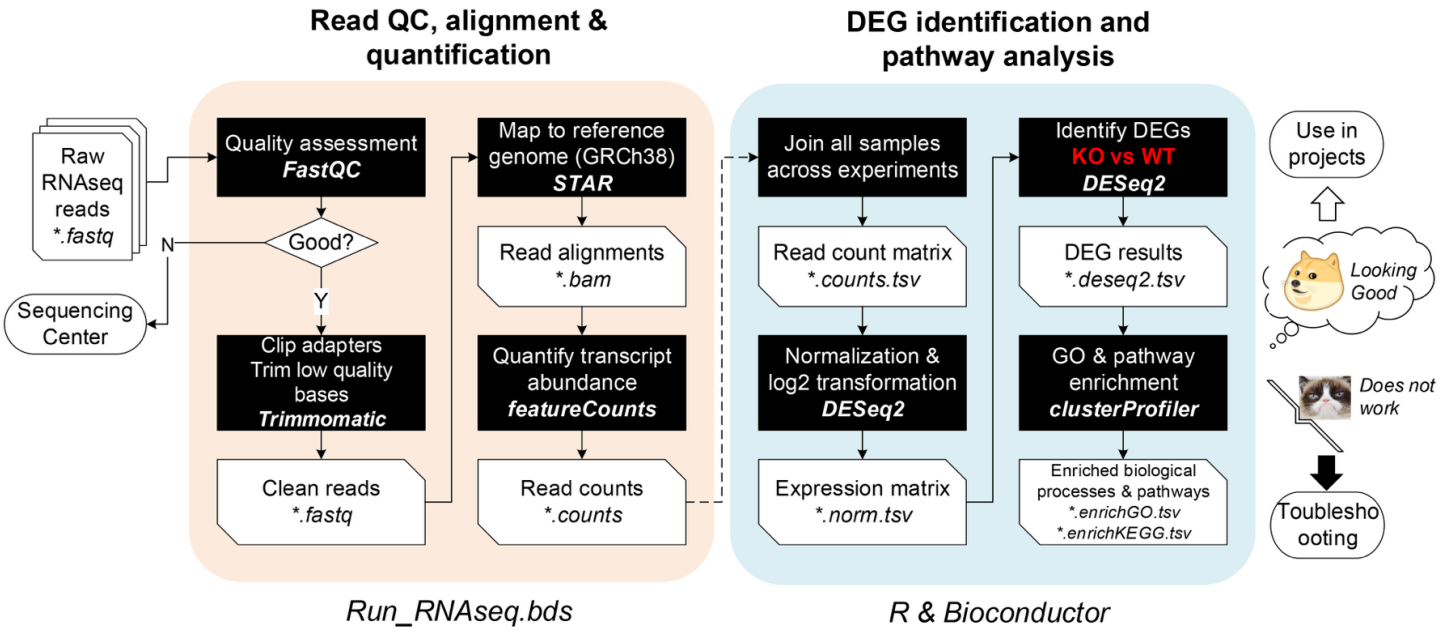
- Understand the basics of RNA-seq experimental design
- Be familiar with the common data formats and standards in RNA-seq analysis
- Know computational tools for RNA-seq data processing
- Be able to align short sequence reads on a reference genome
- Be able to perform the basic analysis such as gene quantification and differential expression analysis

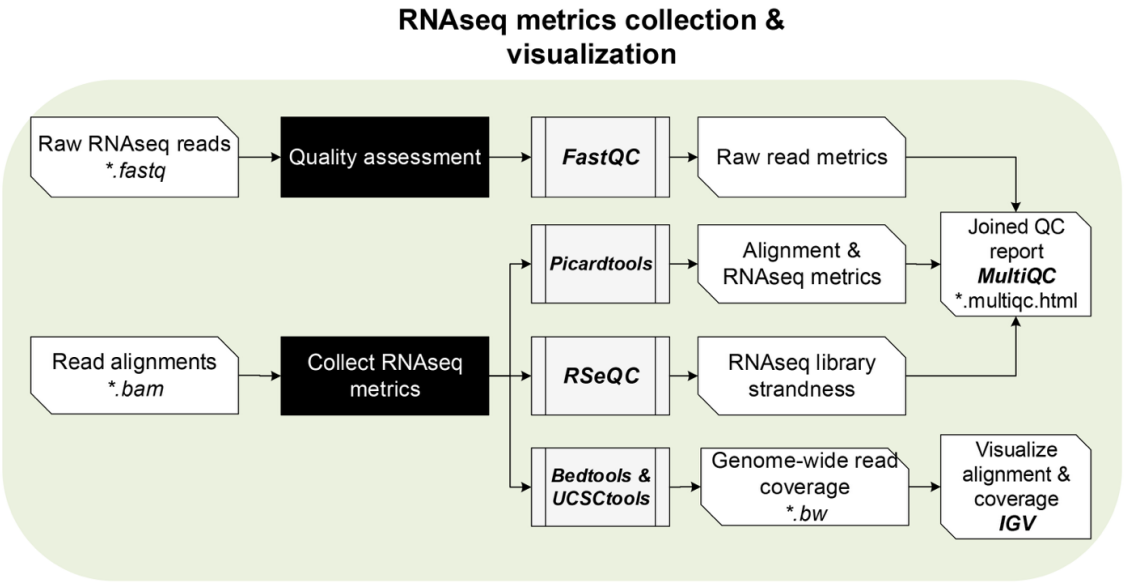
This tutorial is based on CRI's high-performance computing (HPC) cluster. If you are not familiar with this newly assembled cluster, a concise user's guide can be found here (http://cri.uchicago.edu/wp-content/uploads/2017/04/Gardner-Part-1.pdf).

Work Flow | Top

The RNA-seq data used in this tutorial are from **DLBC**.

In this tutorial, we use the sequencing reads in the project DLBC in mouse as example. The sample information are saved in the file **`DLBC.metadata.txt`** (see below).





Work Flow

Data Description | Top

There are six (partial) single-end RNA-seq sequencing libraries will be used as the example dataset In this tutorial. The respective sample information is described in the metadata table example/ **`DLBC.metadata.txt`** .

Sample Description

Sample	Library	ReadGroup	LibType	Platform	SequencingCenter	Date	Lane	Unit
KO01	KO01	SRR1205282	NS	Illumina	SRA	2015-07-22	7	FCC2B5CACXX
KO02	KO02	SRR1205283	NS	Illumina	SRA	2015-07-22	7	FCC2B5CACXX
KO03	KO03	SRR1205284	NS	Illumina	SRA	2015-07-22	7	FCC2B5CACXX
WT01	WT01	SRR1205285	NS	Illumina	SRA	2015-07-22	4	FCC2C3CACXX

Prerequisites | Top

We will use SSH (Secure Shell) to connect to CRI's HPC. SSH now is included or can be installed in all standard operating systems (Windows, Linux, and OS X).

Login and Setup Tutorial Working Directory | Top

The login procedure varies slightly depending on whether you use a Mac/Unix/Linux computer or a Windows computer.

- Log into one of entry nodes in CRI HPC
    1. Open a terminal session.
    2. Connect to the login node of the CRI HPC cluster:

    ```
    $ ssh -l username@hpc.cri.uchicago.edu
    ```

    3. If it's your first time to log in, you will be asked to accept the ssh key. Type "**yes**"
    4. Type in the password when prompted

    > Make sure that you replace ***username*** with your login name.

- **CAUTION**
  - THIS PACKAGE IS LARGE, PLEASE DO NOT DOWNLOAD IT TO YOUR HOME DIRECTORY
  - USE OTHER LOCATION LIKE /CRI/HPC/username

- Set up a tutorial directory
  1. You should be in your home directory after logging in

     ```
     $ pwd
     /home/username
     ```

  2. Instead of downloading the pipeline package to your local home directory, use other location like /CRI/HPC/username

     ```
     $ cd /CRI/HPC/username; pwd
     /CRI/HPC/username
     ```

- Download the pipeline package
  1. One way to download the pipeline package via `git clone`

     ```
     $ git clone git@github.com:wenching/cri_rnaseq_2018.git
     ```

  2. Or, copy the latest package via 'wget'

     ```
     $ wget https://github.com/wenching/cri_rnaseq_2018/archive/cri_rnaseq_2018.
     tar.gz .
     ```

  3. Uncompress the tarball file

     ```
     $ tar -zxvf cri_rnaseq_2018.tgz
     ```

  4. Change working directory to pipeline dirctory

     ```
     $ cd cri_rnaseq_2018
     $ tree -d -L 4
     ```

```
## ../
## ├── SRC
## |   ├── Python
## |   |   ├── lib
## |   |   ├── module
## |   |   └── util
## |   └── R
## |       ├── module
## |       └── util
## ├── docs
## |   ├── IMG
## |   └── result
## └── example
##     ├── data
##     └── references
##         └── v28_92_GRCh38.p12
##             └── STAR
##
## 16 directories
```

- File structure
  - Raw sequencing data files (*.fastq.gz) are located at **`example/data/`**

```
$ tree example/data/
|-- SRR1205282.fastq.gz
|-- SRR1205283.fastq.gz
|-- SRR1205284.fastq.gz
|-- SRR1205285.fastq.gz
|-- SRR1205286.fastq.gz
`-- SRR1205287.fastq.gz
```

  - Genome data are located at
    `/CRI/HPC/ReferenceData/cri_rnaseq_2018/vM18_93_GRCm38.p6`

```
$ tree example/references/v28_92_GRCh38.p12
|-- GRCh38_rRNA.bed
|-- GRCh38_rRNA.bed.interval_list
|-- STAR
|   |-- Genome
|   |-- Log.out
|   |-- SA
|   |-- SAindex
|   |-- chrLength.txt
|   |-- chrName.txt
|   |-- chrNameLength.txt
|   |-- chrStart.txt
|   |-- exonGeTrInfo.tab
|   |-- exonInfo.tab
|   |-- geneInfo.tab
|   |-- genomeParameters.txt
|   |-- run_genome_generate.logs
|   |-- sjdbInfo.txt
|   |-- sjdbList.fromGTF.out.tab
|   |-- sjdbList.out.tab
|   `-- transcriptInfo.tab
|-- genes.gtf
|-- genes.gtf.bed12
|-- genes.refFlat.txt
|-- genome.chrom.sizes
|-- genome.dict
`-- genome.fa
```

- Pipeline/project related files
  - project related files (i.e., metadata & configuration file) as used in this tutorial are located under **`example/`**

```
$ ls -l example/DLBC.*
```

```
## example/DLBC.metadata.txt
## example/DLBC.pipeline.yaml
```

    - Here are the first few lines in the configuration example file example/ **`DLBC.pipeline.yaml`**

```
---
pipeline:
  flags:
    aligners:
      run_star: True
    quantifiers:
      run_featurecounts: True
      run_rsem: False
      run_kallisto: False
    callers:
      run_edger: True
      run_deseq2: True
      run_limma: True
  software:
    main:
      use_module: 0
      adapter_pe: AGATCGGAAGAGCGGTTCAG,AGATCGGAAGAGCGTCGTGT
      adapter_se: AGATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTA
      fastq_format: 33
      genome_assembly: hg38
```

When running on another dataset, you will need to modify these `two files` and the master pipeline script (i.e., `Build_RNAseq.DLBC.sh`) (as described below) accordingly.

For instance, if you would like to turn off the DE analysis tool limma, you can set the respecitve paramter to ‘False’ in configuration file - run_limma: False

For metadata file, you might pay attendtion on the following settings

1. **Single End (SE)** Library
    1. Set Flavor column as 1xReadLength (e.g., 1x50)
    2. Set Seqfile1 column as the file name of the repective sequencing file
2. **Paired End (PE)** Library
    1. Set Flavor column as 2xReadLength (e.g., 2x50)
    2. Set Seqfile1 column as the file name of the repective read 1 (R1) sequencing file
    3. Set an additional column named '**Seqfile2**' as the file name of the repective read 2 (R2) sequencing file
3. **Non strand-specific** Library
    1. Set LibType column to NS
4. **Strand-specific** Library
    1. Inquire the library type from your seuqencing center and set LibType column to FR (the left-most end of the fragment (in transcript coordinates, or the first-strand synthesis) is the first sequenced) or RF (the right-most end of the fragment (in transcript coordinates) is the first sequenced, or the second-strand synthesis). You can read this blog (http://onetipperday.sterding.com/2012/07/how-to-tell-which-library-type-to-use.html) for more details of strand-specific RNA-seq.

- Master pipeline script

```
$ cat Build_RNAseq.DLBC.sh
```

```
##
##
## ## build pipeline scripts
##
## now=$(date +"%m-%d-%Y_%H:%M:%S")
##
## ## project info
## project="DLBC"
## SubmitRNAseqExe="Submit_${PWD##*/}.sh"
## padding="example/"
##
## ## command
## echo "START" `date` " Running build_rnaseq.py"
## python3 SRC/Python/build_rnaseq.py \
##  --projdir $PWD \
##  --metadata $PWD/${padding}$project.metadata.txt \
##  --config $PWD/${padding}$project.pipeline.yaml \
##  --systype cluster \
##  --threads 8 \
##  --log_file $PWD/Build_RNAseq.$project.$now.log
##
## ## submit pipeline master script
## echo "START" `date` " Running $SubmitRNAseqExe"
## echo "bash $SubmitRNAseqExe"
##
## echo "END" `date`
```

Basically, when running on your own dataset, you will need to modify this master pipeline script (i.e., `Build_RNAseq.DLBC.sh`) accordingly.

For instance, you can change respective parameters as follows. - project="PROJECT_AS_PREFIX" (e.g., **DLBC** which is used as a prefix of metadata file **DLBC**.metadata.txt and configuration file **DLBC**.pipeline.yaml) - padding="DIRECTORY_NAME_CONTAINING_PROJECT_DATA" (e.g., **example** which is the folder name to accommodate metadata file, configuration file, sequencing data folder, and references folder)

Pipeline Steps | Top

- Before running, you need to know

> The master BigDataScript script can be `ONLY run on a head/entry node` other than any other compuatation node. But, you can step by step run individual sub-task bash scripts in any computation node interactively.

- Generate sub-task scirpts of the pipepline

```
# load modules
$ module purge;module load gcc udunits R/3.5.0 python/3.6.0; module update

# This step is optional but it will install all necessary R packages ahead.
# In case the pipeline was terminated due to the failure of R package installati
on later when running the pipeline.
# A successful execution result will show the package versions of all necessary
packages from devtools to pheatmap
$ Rscript --vanilla SRC/R/util/prerequisite.packages.R

# create directories and generate all necessary scripts
$ bash Build_RNAseq.DLBC.sh
```

  - this step will execute **`SRC/Python/build_rnaseq.py`** using python3 to generate all sub-task bash scripts and directories according to the provided metadata and configuration files (i.e., DLBC/ **`DLBC.metadata.txt`** and DLBC/ **`DLBC.pipeline.yaml`**)

```
$ tree -d RNAseq
RNAseq
|-- Aln
|   `-- star
|       |-- KO01
|       |   `-- SRR1205282
|       |-- KO02
|       |   `-- SRR1205283
|       |-- KO03
|       |   `-- SRR1205284
|       |-- WT01
|       |   `-- SRR1205285
|       |-- WT02
|       |   `-- SRR1205286
|       `-- WT03
|           `-- SRR1205287
|-- AlnQC
|   |-- picard
|   |   `-- star
```

```
|   |       |-- KO01
|   |       |   `-- tmp
|   |       |-- KO02
|   |       |   `-- tmp
|   |       |-- KO03
|   |       |   `-- tmp
|   |       |-- WT01
|   |       |   `-- tmp
|   |       |-- WT02
|   |       |   `-- tmp
|   |       `-- WT03
|   |           `-- tmp
|   `-- rseqc
|       `-- star
|           |-- KO01
|           |   `-- tmp
|           |-- KO02
|           |   `-- tmp
|           |-- KO03
|           |   `-- tmp
|           |-- WT01
|           |   `-- tmp
|           |-- WT02
|           |   `-- tmp
|           `-- WT03
|               `-- tmp
|-- DEG
|   |-- deseq2
|   |   `-- featurecounts
|   |       `-- star
|   |-- edger
|   |   `-- featurecounts
|   |       `-- star
|   `-- limma
|       `-- featurecounts
|           `-- star
|-- LociStat
|   `-- featurecounts
|       `-- star
|           `-- cri_rnaseq_2018
|-- PostAna
|   |-- clusterprofiler
|   |   `-- featurecounts
|   |       `-- star
|   |           `-- cri_rnaseq_2018
|   `-- pheatmap
|       `-- featurecounts
```

```
|           `-- star
|               `-- cri_rnaseq_2018
|-- QuantQC
|   `-- featurecounts
|       `-- star
|-- Quantification
|   `-- featurecounts
|       `-- star
|           |-- KO01
|           |-- KO02
|           |-- KO03
|           |-- WT01
|           |-- WT02
|           `-- WT03
|-- RawReadQC
|   |-- KO01
|   |   `-- SRR1205282
|   |       `-- SRR1205282_fastqc
|   |           |-- Icons
|   |           `-- Images
|   |-- KO02
|   |   `-- SRR1205283
|   |       `-- SRR1205283_fastqc
|   |           |-- Icons
|   |           `-- Images
|   |-- KO03
|   |   `-- SRR1205284
|   |       `-- SRR1205284_fastqc
|   |           |-- Icons
|   |           `-- Images
|   |-- WT01
|   |   `-- SRR1205285
|   |       `-- SRR1205285_fastqc
|   |           |-- Icons
|   |           `-- Images
|   |-- WT02
|   |   `-- SRR1205286
|   |       `-- SRR1205286_fastqc
|   |           |-- Icons
|   |           `-- Images
|   `-- WT03
|       `-- SRR1205287
|           `-- SRR1205287_fastqc
|               |-- Icons
|               `-- Images
|
`-- shell_scripts
```

- Execute the entire analysis with just one command

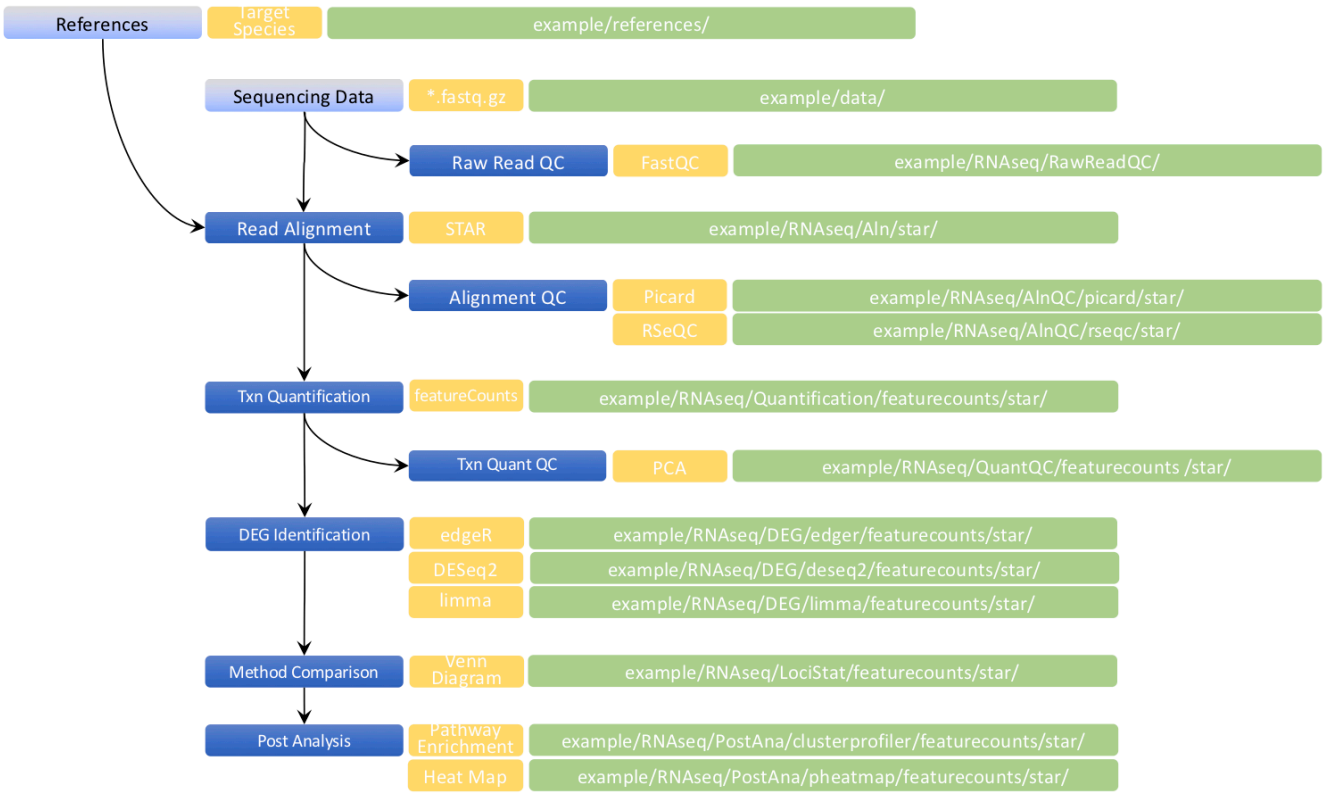
```
# This will start to run the entire pipeline.
# You can chekc teh BDS report to know the running status.
$ bash Submit_cri_rnaseq_2018.sh
```

  - Again, this step will execute the master BigDataScript script **`Submit_cri_rnaseq_2018.bds`** , so you will need to run this command **on a head/entry node**.

Check running status

1. $ qstat -a
2. tail *.bds.log

Navigation Map | Top



Navigation Map

Step 1: Quality Control | Top

For the first step, the pipeline will perform quality assessment on the raw fastq files.

The BDS code snippet for the sample KO01 will look like:

```
$ grep -A1 run.RawReadQC.FastQC.SRR1205282.sh Submit_*.bds
```

```
## dep( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/RawReadQC/KO01/SRR1205282/SRR1205282_fastq
c.zip' ] <- [ '/CRI/HPC/cri_rnaseq_2018/example/data/SRR1205282.fastq.gz' ], cpus :=
1, mem := 16*G, timeout := 72*hour, taskName := "FastQC.SRR1205282") sys bash /CRI/HP
C/cri_rnaseq_2018/RNAseq/shell_scripts/run.RawReadQC.FastQC.SRR1205282.sh; sleep 2
## goal( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/RawReadQC/KO01/SRR1205282/SRR1205282_fast
qc.zip' ] )
```

This code chunk will invoke the bash script RNAseq/shell_scripts/ **`run.RawReadQC.FastQC.SRR1205282.sh`** to execute FastQC on the KO01(BK-AA-1_S11_L007_R355) sequencing library.

After the completion of the entire pipeline, you can check FastQC report per individual libraries; for instance, the partial report of KO01 will be as follows or a full report (result/SRR1205282_fastqc.html).

FastQC Report

Wed 14 Nov 2018
SRR1205282.fastq.gz

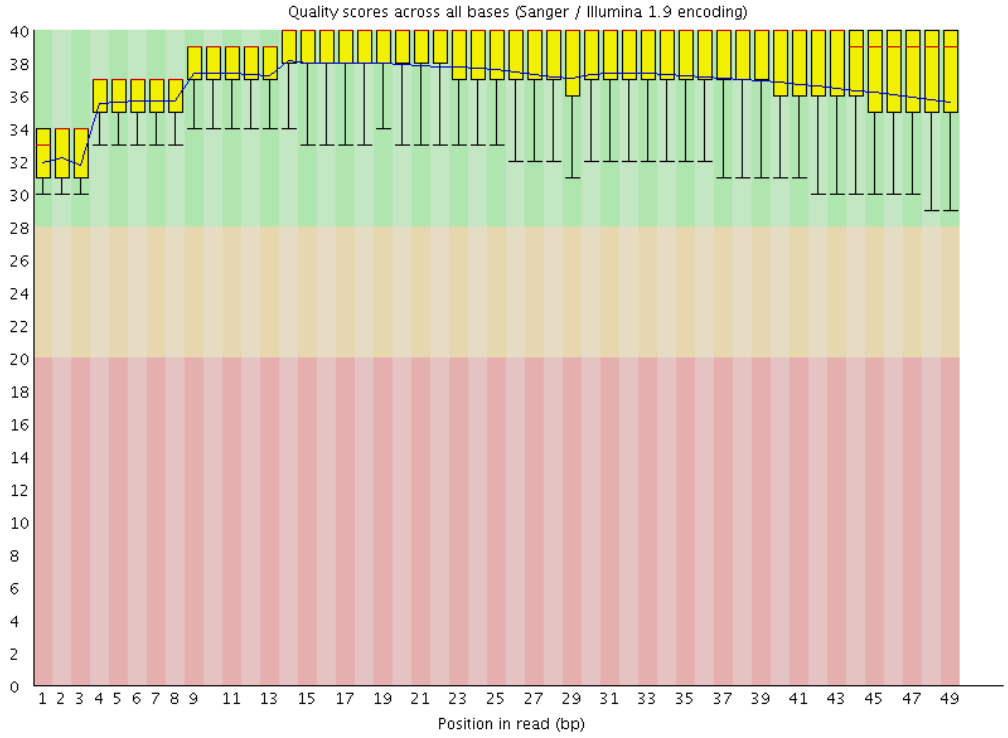
Summary

- Basic Statistics
- Per base sequence quality
- Per tile sequence quality
- Per sequence quality scores
- Per base sequence content
- Per sequence GC content
- Per base N content
- Sequence Length Distribution
- Sequence Duplication Levels
- Overrepresented sequences
- Adapter Content
- Kmer Content

Basic Statistics

Measure	Value
Filename	SRR1205282.fastq.gz
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	74126025
Sequences flagged as poor quality	0
Sequence length	49
%GC	49

Per base sequence quality



Produced by FastQC (version 0.11.5)

You can check FastQC Help (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/) for more details about how to interpret a FastQC report.

Or, compare your reports to the example reports provided by FastQC for a Good Illumina Data (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/good_sequence_short_fastqc.html) or Bad Illumina Data (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/bad_sequence_fastqc.html).

Step 2.1: Read Alignment | Top

In this step, the pipeline will conduct read alignment on the raw fastq files.

The BDS code snippet for the sample KO01 will look like:

```
$ grep -A1 run.alignRead.star.SRR1205282.sh Submit_*.bds
```

```
## dep( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/Aln/star/KO01/SRR1205282/SRR1205282.star.b
am' ] <- [ '/CRI/HPC/cri_rnaseq_2018/example/data/SRR1205282.fastq.gz' ], cpus := 4,
mem := 64*G, timeout := 72*hour, taskName := "star.SRR1205282") sys bash /CRI/HPC/cri
_rnaseq_2018/RNAseq/shell_scripts/run.alignRead.star.SRR1205282.sh; sleep 2
## goal( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/Aln/star/KO01/SRR1205282/SRR1205282.star.
bam' ] )
```

This code chunk will invoke the bash script RNAseq/shell_scripts/ **`run.alignRead.star.SRR1205282.sh`** to execute STAR on the KO01(SRR1205282) sequencing library.

After the completion of the entire pipeline, you can check the alignment result of each individual libraries; for instance, the result of KO01(SRR1205282) will be as follows.

```
$ tree RNAseq/Aln/star/KO01/SRR1205282
RNAseq/Aln/star/KO01/SRR1205282
|-- SRR1205282.star.Aligned.sortedByCoord.out.bam
|-- SRR1205282.star.Log.final.out
|-- SRR1205282.star.Log.out
|-- SRR1205282.star.Log.progress.out
|-- SRR1205282.star.SJ.out.tab
|-- SRR1205282.star.Unmapped.out.mate1
|-- SRR1205282.star.bai
|-- SRR1205282.star.bam -> SRR1205282.star.Aligned.sortedByCoord.out.bam
`-- run.alignRead.star.SRR1205282.log
```

You can check a log file (e.g., **RNAseq/Aln/star/KO01/SRR1205282/ `SRR1205282.star.Log.final.out`**) for more alignment information provided by STAR.

```
$ cat RNAseq/Aln/star/KO01/SRR1205282/SRR1205282.star.Log.final.out
```

```
##                                 Started job on |    Nov 14 10:39:46
##                              Started mapping on |    Nov 14 10:40:01
##                                     Finished on |    Nov 14 10:50:31
##        Mapping speed, Million of reads per hour |    423.58
##
##                            Number of input reads |    74126025
##                        Average input read length |    49
##                                     UNIQUE READS:
##                    Uniquely mapped reads number |    63912102
##                         Uniquely mapped reads % |    86.22%
##                           Average mapped length |    48.82
##                        Number of splices: Total |    10010945
##             Number of splices: Annotated (sjdb) |    9913245
##                        Number of splices: GT/AG |    9915467
##                        Number of splices: GC/AG |    77874
##                        Number of splices: AT/AC |    11550
##                Number of splices: Non-canonical |    6054
##                       Mismatch rate per base, % |    0.25%
##                          Deletion rate per base |    0.01%
##                         Deletion average length |    1.43
##                         Insertion rate per base |    0.00%
##                        Insertion average length |    1.22
##                              MULTI-MAPPING READS:
##         Number of reads mapped to multiple loci |    0
##              % of reads mapped to multiple loci |    0.00%
##         Number of reads mapped to too many loci |    9697272
##              % of reads mapped to too many loci |    13.08%
##                                   UNMAPPED READS:
##        % of reads unmapped: too many mismatches |    0.00%
##                  % of reads unmapped: too short |    0.44%
##                      % of reads unmapped: other |    0.26%
##                                   CHIMERIC READS:
##                        Number of chimeric reads |    0
##                             % of chimeric reads |    0.00%
```

Step 2.2: Alignment QC | Top

In this step, the pipeline will conduct a QC on alignment result.

The BDS code snippets for the sample KO01 will look like:

```
$ grep -A1 run.alnQC.*.star.KO01.*.sh Submit_*.bds
```

```
## dep( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/AlnQC/picard/star/KO01/KO01.star.picard.RN
A_Metrics', '/CRI/HPC/cri_rnaseq_2018/RNAseq/AlnQC/picard/star/KO01/KO01.star.picard.
RNA_Metrics.pdf' ] <- [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/Aln/star/KO01/KO01.star.bai'
], cpus := 4, mem := 32*G, timeout := 72*hour, taskName := "picard.star.KO01") sys ba
sh /CRI/HPC/cri_rnaseq_2018/RNAseq/shell_scripts/run.alnQC.picard.star.KO01.CollectRn
aSeqMetrics.sh; sleep 2
## goal( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/AlnQC/picard/star/KO01/KO01.star.picard.R
NA_Metrics', '/CRI/HPC/cri_rnaseq_2018/RNAseq/AlnQC/picard/star/KO01/KO01.star.picard
.RNA_Metrics.pdf' ] )
## --
## dep( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/AlnQC/rseqc/star/KO01/KO01.star.rseqc.clip
ping_profile.xls', '/CRI/HPC/cri_rnaseq_2018/RNAseq/AlnQC/rseqc/star/KO01/KO01.star.r
seqc.clipping_profile.r', '/CRI/HPC/cri_rnaseq_2018/RNAseq/AlnQC/rseqc/star/KO01/KO01
.star.rseqc.clipping_profile.pdf' ] <- [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/Aln/star/KO
01/KO01.star.bai' ], cpus := 8, mem := 64*G, timeout := 72*hour, taskName := "rseqc.s
tar.KO01") sys bash /CRI/HPC/cri_rnaseq_2018/RNAseq/shell_scripts/run.alnQC.rseqc.sta
r.KO01.clipping_profile.py.sh; sleep 2
## goal( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/AlnQC/rseqc/star/KO01/KO01.star.rseqc.cli
pping_profile.xls', '/CRI/HPC/cri_rnaseq_2018/RNAseq/AlnQC/rseqc/star/KO01/KO01.star.
rseqc.clipping_profile.r', '/CRI/HPC/cri_rnaseq_2018/RNAseq/AlnQC/rseqc/star/KO01/KO0
1.star.rseqc.clipping_profile.pdf' ] )
## --
## dep( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/AlnQC/rseqc/star/KO01/KO01.star.rseqc.infe
r_experiment.txt' ] <- [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/Aln/star/KO01/KO01.star.bai
' ], cpus := 8, mem := 64*G, timeout := 72*hour, taskName := "rseqc.star.KO01") sys b
ash /CRI/HPC/cri_rnaseq_2018/RNAseq/shell_scripts/run.alnQC.rseqc.star.KO01.infer_exp
eriment.py.sh; sleep 2
## goal( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/AlnQC/rseqc/star/KO01/KO01.star.rseqc.inf
er_experiment.txt' ] )
```

This code chunk will invoke few bash scripts (e.g., RNAseq/shell_scripts/ **`run.alnQC.picard.star.KO01.CollectRnaSeqMetrics.sh`** , **`run.alnQC.rseqc.star.KO01.clipping_profile.py.sh`** , and **`run.alnQC.rseqc.star.KO01.infer_experiment.py.sh`**) to execute alignment QC tools (i.e., Picard (https://broadinstitute.github.io/picard/) and RSeQC (http://rseqc.sourceforge.net/)) on the sample KO01.

After the completion of the entire pipeline, you can check the alignment QC results of each individual samples; for instance, the results of KO01 will be as follows.

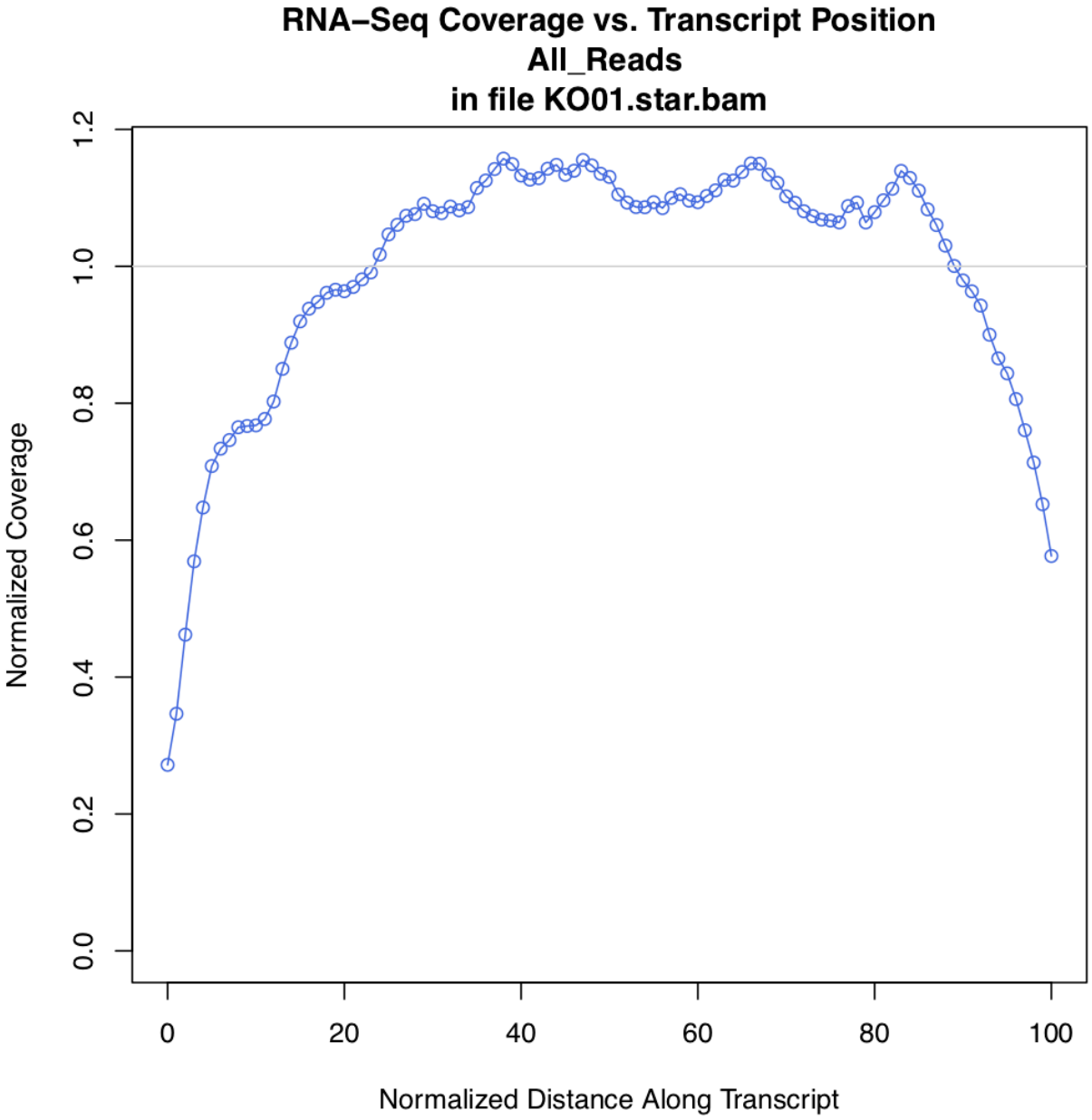
```
$ tree RNAseq/AlnQC/*/star/KO01
RNAseq/AlnQC/picard/star/KO01
|-- KO01.star.picard.RNA_Metrics
|-- KO01.star.picard.RNA_Metrics.pdf
|-- run.alnQC.picard.star.KO01.CollectRnaSeqMetrics.log
`-- tmp
RNAseq/AlnQC/rseqc/star/KO01
|-- KO01.star.rseqc.clipping_profile.pdf
|-- KO01.star.rseqc.clipping_profile.r
|-- KO01.star.rseqc.clipping_profile.xls
|-- KO01.star.rseqc.infer_experiment.txt
|-- run.alnQC.rseqc.star.KO01.clipping_profile.py.log
`-- tmp
```

You can check alignment statistics (e.g., **RNAseq/AlnQC/picard/star/KO01/ `KO01.star.picard.RNA_Metrics`**) for more information provided by Picard (https://broadinstitute.github.io/picard/).

```
$ head RNAseq/AlnQC/picard/star/KO01/KO01.star.picard.RNA_Metrics
```

```
## ## htsjdk.samtools.metrics.StringHeader
## # picard.analysis.CollectRnaSeqMetrics REF_FLAT=/CRI/HPC/cri_rnaseq_2018/example/r
eferences/v28_92_GRCh38.p12/genes.refFlat.txt RIBOSOMAL_INTERVALS=/CRI/HPC/cri_rnaseq
_2018/example/references/v28_92_GRCh38.p12/GRCh38_rRNA.bed.interval_list STRAND_SPECI
FICITY=NONE CHART_OUTPUT=/CRI/HPC/cri_rnaseq_2018/RNAseq/AlnQC/picard/star/KO01/KO01.
star.picard.RNA_Metrics.pdf METRIC_ACCUMULATION_LEVEL=[SAMPLE, ALL_READS] INPUT=/CRI/
HPC/cri_rnaseq_2018/RNAseq/Aln/star/KO01/KO01.star.bam OUTPUT=/CRI/HPC/cri_rnaseq_201
8/RNAseq/AlnQC/picard/star/KO01/KO01.star.picard.RNA_Metrics TMP_DIR=[/CRI/HPC/cri_rn
aseq_2018/RNAseq/AlnQC/picard/star/KO01/tmp]    MINIMUM_LENGTH=500 RRNA_FRAGMENT_PERC
ENTAGE=0.8 ASSUME_SORTED=true STOP_AFTER=0 VERBOSITY=INFO QUIET=false VALIDATION_STRI
NGENCY=STRICT COMPRESSION_LEVEL=5 MAX_RECORDS_IN_RAM=500000 CREATE_INDEX=false CREATE
_MD5_FILE=false GA4GH_CLIENT_SECRETS=client_secrets.json
## ## htsjdk.samtools.metrics.StringHeader
## # Started on: Wed Nov 14 10:53:18 CST 2018
##
## ## METRICS CLASS picard.analysis.RnaSeqMetrics
## PF_BASES PF_ALIGNED_BASES    RIBOSOMAL_BASES CODING_BASES    UTR_BASES   INTRONIC_
BASES  INTERGENIC_BASES    IGNORED_READS   CORRECT_STRAND_READS    INCORRECT_STRAND_R
EADS  PCT_RIBOSOMAL_BASES PCT_CODING_BASES    PCT_UTR_BASES   PCT_INTRONIC_BASES  PCT
_INTERGENIC_BASES    PCT_MRNA_BASES  PCT_USABLE_BASES    PCT_CORRECT_STRAND_READS
MEDIAN_CV_COVERAGE  MEDIAN_5PRIME_BIAS  MEDIAN_3PRIME_BIAS  MEDIAN_5PRIME_TO_3PRIME_B
IAS    SAMPLE  LIBRARY READ_GROUP
## 3131692998   3120045478  1072267 1902187708  1009131413  164870020   42790598    0
0   0   0.000344    0.609667    0.323435    0.052842    0.013715    0.933102    0.929
631    0   0.783501    0.448148    0.690783    0.59721
## 3131692998   3120045478  1072267 1902187708  1009131413  164870020   42790598    0
0   0   0.000344    0.609667    0.323435    0.052842    0.013715    0.933102    0.929
631    0   0.783501    0.448148    0.690783    0.59721 unknown
```

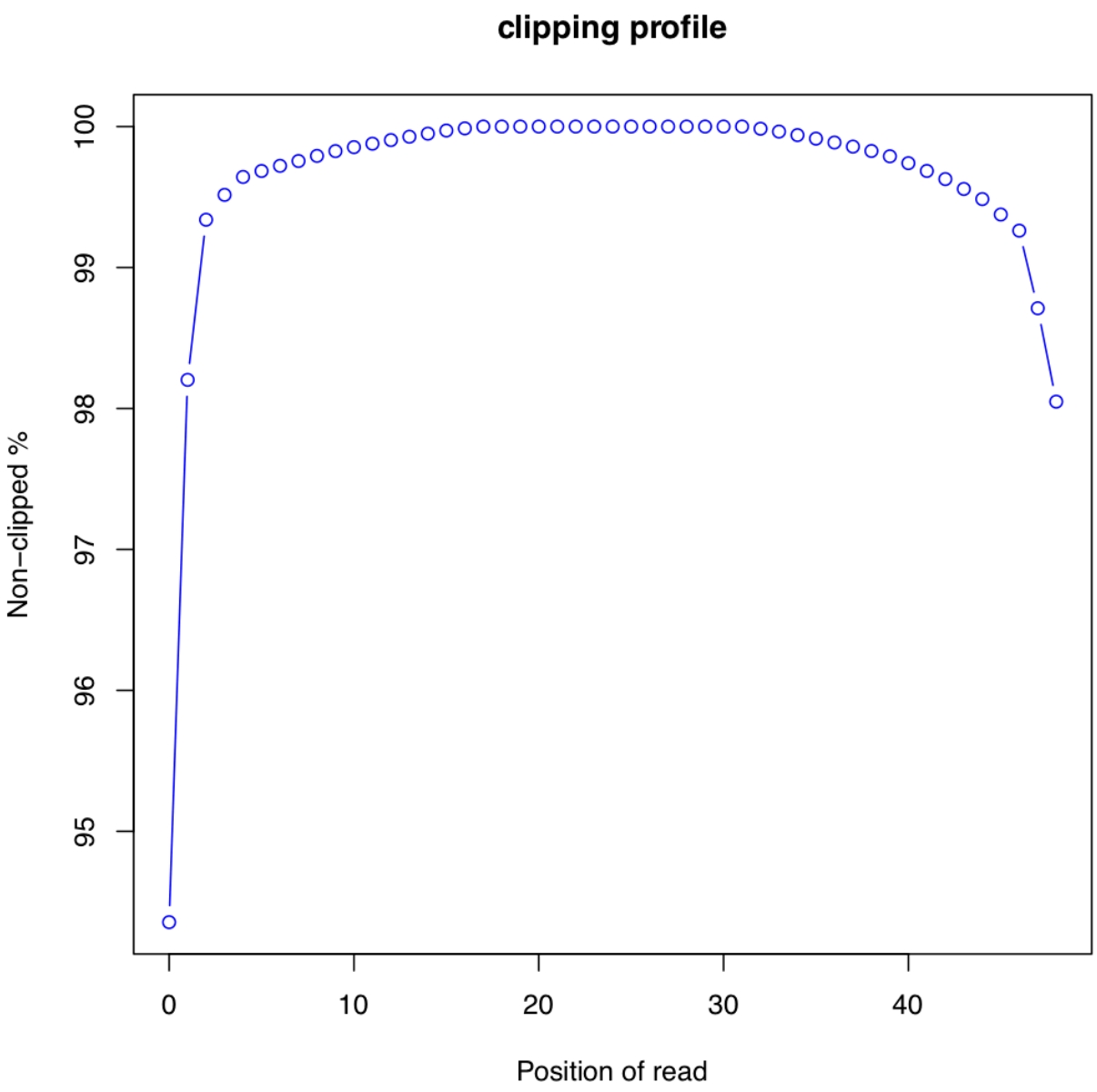
Or, the respective coverage plot of the sample KO01 produced by Picard (https://broadinstitute.github.io/picard/) will be as follows.



There are two alignment measurements performed using RSeQC (http://rseqc.sourceforge.net).

1. clipping_profile.py (http://rseqc.sourceforge.net/#clipping-profile-py)
    - Calculate the distributions of clipped nucleotides across reads
2. infer_experiment.py (http://rseqc.sourceforge.net/#infer-experiment-py)
    - Use to “guess” how RNA-seq sequencing was configured, particularly how reads were stranded for strand-specific RNA-seq data, through comparing the “strandedness of reads” with the “strandedness of transcripts”.

The results will be as follows. Please check the RSeQC (http://rseqc.sourceforge.net) website for more measurements and details.




```
$cat RNAseq/AlnQC/rseqc/star/KO01/KO01.star.rseqc.infer_experiment.txt
```

```
##
##
## This is SingleEnd Data
## Fraction of reads failed to determine: 0.2143
## Fraction of reads explained by "++,--": 0.4025
## Fraction of reads explained by "+-,-+": 0.3832
```

Here, you can confirm again the sample KO01 is a single-end, strand-specific library using the second-strand synthesis.

Step 3: Expression Quantification | Top

In this step, the pipeline will conduct expression quantification over alignments.

The BDS code snippet for the sample KO01 will look like:

```
$ grep -A1 run.quant.featurecounts.star.KO01.sh Submit_*.bds
```

```
## dep( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/Quantification/featurecounts/star/KO01/KO0
1.star.featurecounts.count' ] <- [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/Aln/star/KO01/KO0
1.star.bai' ], cpus := 4, mem := 32*G, timeout := 72*hour, taskName := "featurecounts
.star.KO01") sys bash /CRI/HPC/cri_rnaseq_2018/RNAseq/shell_scripts/run.quant.feature
counts.star.KO01.sh; sleep 2
## goal( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/Quantification/featurecounts/star/KO01/KO
01.star.featurecounts.count' ] )
```

This code chunk will invoke the bash script (e.g., RNAseq/shell_scripts/ **`run.quant.featurecounts.star.KO01.sh`**) to execute expression quantification tool (i.e., Subread (http://subread.sourceforge.net/)::featureCounts (http://bioinf.wehi.edu.au/featureCounts/) on the sample KO01.

After the completion of the entire pipeline, you can check the quantification results of each individual samples; for instance, the results of KO01 will be as follows.

```
$ tree RNAseq/Quantification/featurecounts/star/KO01
RNAseq/Quantification/featurecounts/star/KO01
|-- KO01.star.featurecounts.count
|-- KO01.star.featurecounts.count.jcounts
|-- KO01.star.featurecounts.count.summary
`-- run.quant.featurecounts.star.KO01.log
```

You can check quantification statistics (e.g., **RNAseq/Quantification/featurecounts/star/KO01** **`KO01.star.featurecounts.count.summary`**) for more information provided by Subread (http://subread.sourceforge.net/)::featureCounts (http://bioinf.wehi.edu.au/featureCounts/)

```
$ cat RNAseq/Quantification/featurecounts/star/KO01/KO01.star.featurecounts.count.sum
mary
```

```
## Status   /CRI/HPC/cri_rnaseq_2018/RNAseq/Aln/star/KO01/KO01.star.bam
## Assigned 55814495
## Unassigned_Unmapped  0
## Unassigned_MappingQuality    0
## Unassigned_Chimera   0
## Unassigned_FragmentLength    0
## Unassigned_Duplicate 0
## Unassigned_MultiMapping  0
## Unassigned_Secondary 0
## Unassigned_Nonjunction   0
## Unassigned_NoFeatures    4118361
## Unassigned_Overlapping_Length    0
## Unassigned_Ambiguity 3979246
```

Or, the top 10 most abundant genes in the sample KO01 will be as follows.

```
$ cat <(head -n2 RNAseq/Quantification/featurecounts/star/KO01/KO01.star.featurecount
s.count | tail -n+2 | cut -f1,7) <(cut -f1,7 RNAseq/Quantification/featurecounts/star
/KO01/KO01.star.featurecounts.count | sort -k2,2nr | head)
```

Top 10 most abundant genes in KO01

Geneid	Chr	Start	End	Strand	Length	KO01
ENSG00000198886.2	chrM	10760	12137	•	1378	655558
ENSG00000211899.9	chr14	105851708	105856218	•	1868	523725
ENSG00000210082.2	chrM	1671	3229	•	1559	428124
ENSG00000198804.2	chrM	5904	7445	•	1542	420494
ENSG00000167658.15	chr19	3976056	3985469	•	4027	339682

Step 4-1: Identify Differentially Expressed Genes (DEGs) | Top

In this step, the pipeline will identify differentially expressed genes (DEG) according to the alignment result files (i.e., BAM files) after the alignment step.

The BDS code snippets for the example dataset will look like:

```
$ grep -A1 run.call.*.featurecounts.star.*.sh Submit_*.bds
```

```
## dep( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/DEG/edger/featurecounts/star/cri_rnaseq_20
18.star.featurecounts.edger.count.txt', '/CRI/HPC/cri_rnaseq_2018/RNAseq/DEG/edger/fe
aturecounts/star/cri_rnaseq_2018.star.featurecounts.edger.test.DEG.txt' ] <- [ '/CRI/
HPC/cri_rnaseq_2018/RNAseq/Quantification/featurecounts/star/KO01/KO01.star.featureco
unts.count', '/CRI/HPC/cri_rnaseq_2018/RNAseq/Quantification/featurecounts/star/KO02/
KO02.star.featurecounts.count', '/CRI/HPC/cri_rnaseq_2018/RNAseq/Quantification/featu
recounts/star/KO03/KO03.star.featurecounts.count', '/CRI/HPC/cri_rnaseq_2018/RNAseq/Q
uantification/featurecounts/star/WT01/WT01.star.featurecounts.count', '/CRI/HPC/cri_r
naseq_2018/RNAseq/Quantification/featurecounts/star/WT02/WT02.star.featurecounts.coun
t', '/CRI/HPC/cri_rnaseq_2018/RNAseq/Quantification/featurecounts/star/WT03/WT03.star
.featurecounts.count' ], cpus := 4, mem := 32*G, timeout := 72*hour, taskName := "edg
er.featurecounts.star.cri_rnaseq_2018") sys bash /CRI/HPC/cri_rnaseq_2018/RNAseq/shel
l_scripts/run.call.edger.featurecounts.star.cri_rnaseq_2018.sh; sleep 2
## goal( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/DEG/edger/featurecounts/star/cri_rnaseq_2
018.star.featurecounts.edger.count.txt' ] )
## --
## dep( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/DEG/deseq2/featurecounts/star/cri_rnaseq_2
018.star.featurecounts.deseq2.count.txt', '/CRI/HPC/cri_rnaseq_2018/RNAseq/DEG/deseq2
/featurecounts/star/cri_rnaseq_2018.star.featurecounts.deseq2.test.DEG.txt' ] <- [ '/
CRI/HPC/cri_rnaseq_2018/RNAseq/Quantification/featurecounts/star/KO01/KO01.star.featu
recounts.count', '/CRI/HPC/cri_rnaseq_2018/RNAseq/Quantification/featurecounts/star/K
O02/KO02.star.featurecounts.count', '/CRI/HPC/cri_rnaseq_2018/RNAseq/Quantification/f
eaturecounts/star/KO03/KO03.star.featurecounts.count', '/CRI/HPC/cri_rnaseq_2018/RNAs
eq/Quantification/featurecounts/star/WT01/WT01.star.featurecounts.count', '/CRI/HPC/c
ri_rnaseq_2018/RNAseq/Quantification/featurecounts/star/WT02/WT02.star.featurecounts.
count', '/CRI/HPC/cri_rnaseq_2018/RNAseq/Quantification/featurecounts/star/WT03/WT03.
star.featurecounts.count' ], cpus := 4, mem := 32*G, timeout := 72*hour, taskName :=
"deseq2.featurecounts.star.cri_rnaseq_2018") sys bash /CRI/HPC/cri_rnaseq_2018/RNAseq
/shell_scripts/run.call.deseq2.featurecounts.star.cri_rnaseq_2018.sh; sleep 2
## goal( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/DEG/deseq2/featurecounts/star/cri_rnaseq_
2018.star.featurecounts.deseq2.count.txt' ] )
## --
## dep( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/DEG/limma/featurecounts/star/cri_rnaseq_20
18.star.featurecounts.limma.count.txt', '/CRI/HPC/cri_rnaseq_2018/RNAseq/DEG/limma/fe
aturecounts/star/cri_rnaseq_2018.star.featurecounts.limma.test.DEG.txt' ] <- [ '/CRI/
HPC/cri_rnaseq_2018/RNAseq/Quantification/featurecounts/star/KO01/KO01.star.featureco
unts.count', '/CRI/HPC/cri_rnaseq_2018/RNAseq/Quantification/featurecounts/star/KO02/
KO02.star.featurecounts.count', '/CRI/HPC/cri_rnaseq_2018/RNAseq/Quantification/featu
recounts/star/KO03/KO03.star.featurecounts.count', '/CRI/HPC/cri_rnaseq_2018/RNAseq/Q
uantification/featurecounts/star/WT01/WT01.star.featurecounts.count', '/CRI/HPC/cri_r
naseq_2018/RNAseq/Quantification/featurecounts/star/WT02/WT02.star.featurecounts.coun
t', '/CRI/HPC/cri_rnaseq_2018/RNAseq/Quantification/featurecounts/star/WT03/WT03.star
.featurecounts.count' ], cpus := 4, mem := 32*G, timeout := 72*hour, taskName := "lim
ma.featurecounts.star.cri_rnaseq_2018") sys bash /CRI/HPC/cri_rnaseq_2018/RNAseq/shel
l_scripts/run.call.limma.featurecounts.star.cri_rnaseq_2018.sh; sleep 2
## goal( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/DEG/limma/featurecounts/star/cri_rnaseq_2
018.star.featurecounts.limma.count.txt' ] )
```

This code chunk will invoke few bash scripts (e.g., RNAseq/shell_scripts/ **`run.call.edger.featurecounts.star.cri_rnaseq_2018.sh`** , **`run.call.deseq2.featurecounts.star.cri_rnaseq_2018.sh`** , and **`run.call.limma.featurecounts.star.cri_rnaseq_2018.sh`**) to execute differential expression (DE) analysis using three the state-of-the-art tools (i.e., edgeR (https://bioconductor.org/packages/release/bioc/html/edgeR.html), DESeq2 (https://bioconductor.org/packages/release/bioc/html/DESeq2.html), and limma (https://bioconductor.org/packages/release/bioc/html/limma.html)) on the example dataset of six samples from KO01 to WT03.

There are three DE analysis tools used in the current pipeline, including

1. edgeR (https://bioconductor.org/packages/release/bioc/html/edgeR.html): Empirical Analysis of Digital Gene Expression Data in R
2. DESeq2 (https://bioconductor.org/packages/release/bioc/html/DESeq2.html): Differential gene expression analysis based on the negative binomial distribution
3. limma (https://bioconductor.org/packages/release/bioc/html/limma.html): Linear Models for Microarray Data

After the completion of the entire pipeline, you can check the calling results of each individual methods; for instance, the analysis results of the example dataset will be as follows.

```
$ tree RNAseq/DEG/*/featurecounts/star/
RNAseq/DEG/deseq2/featurecounts/star/
|-- cri_rnaseq_2018.star.featurecounts.deseq2.RData
|-- cri_rnaseq_2018.star.featurecounts.deseq2.count.ntd.meanSdPlot.pdf
|-- cri_rnaseq_2018.star.featurecounts.deseq2.count.ntd.txt
|-- cri_rnaseq_2018.star.featurecounts.deseq2.count.rld.meanSdPlot.pdf
|-- cri_rnaseq_2018.star.featurecounts.deseq2.count.rld.txt
|-- cri_rnaseq_2018.star.featurecounts.deseq2.count.txt
|-- cri_rnaseq_2018.star.featurecounts.deseq2.count.vst.meanSdPlot.pdf
|-- cri_rnaseq_2018.star.featurecounts.deseq2.count.vst.txt
|-- cri_rnaseq_2018.star.featurecounts.deseq2.plotDispEsts.pdf
|-- cri_rnaseq_2018.star.featurecounts.deseq2.plotMA.pdf
|-- cri_rnaseq_2018.star.featurecounts.deseq2.test.DEG.txt
|-- cri_rnaseq_2018.star.featurecounts.deseq2.test.txt
`-- run.call.deseq2.featurecounts.star.cri_rnaseq_2018.log
RNAseq/DEG/edger/featurecounts/star/
|-- cri_rnaseq_2018.star.featurecounts.edger.RData
|-- cri_rnaseq_2018.star.featurecounts.edger.count.txt
|-- cri_rnaseq_2018.star.featurecounts.edger.plotBCV.pdf
|-- cri_rnaseq_2018.star.featurecounts.edger.plotMA.pdf
|-- cri_rnaseq_2018.star.featurecounts.edger.plotSmear.pdf
|-- cri_rnaseq_2018.star.featurecounts.edger.test.DEG.txt
|-- cri_rnaseq_2018.star.featurecounts.edger.test.txt
`-- run.call.edger.featurecounts.star.cri_rnaseq_2018.log
RNAseq/DEG/limma/featurecounts/star/
|-- cri_rnaseq_2018.star.featurecounts.limma.RData
|-- cri_rnaseq_2018.star.featurecounts.limma.count.txt
|-- cri_rnaseq_2018.star.featurecounts.limma.count.voom.meanSdPlot.pdf
|-- cri_rnaseq_2018.star.featurecounts.limma.plotMA.pdf
|-- cri_rnaseq_2018.star.featurecounts.limma.test.DEG.txt
|-- cri_rnaseq_2018.star.featurecounts.limma.test.txt
|-- cri_rnaseq_2018.star.featurecounts.limma.voom.mean-variance.pdf
`-- run.call.limma.featurecounts.star.cri_rnaseq_2018.log
```

You can check statistical test results per gene (e.g., **RNAseq/DEG/deseq2/featurecounts/star/ `cri_rnaseq_2018.star.featurecounts.deseq2.test.txt`**) for more information generated by each method.

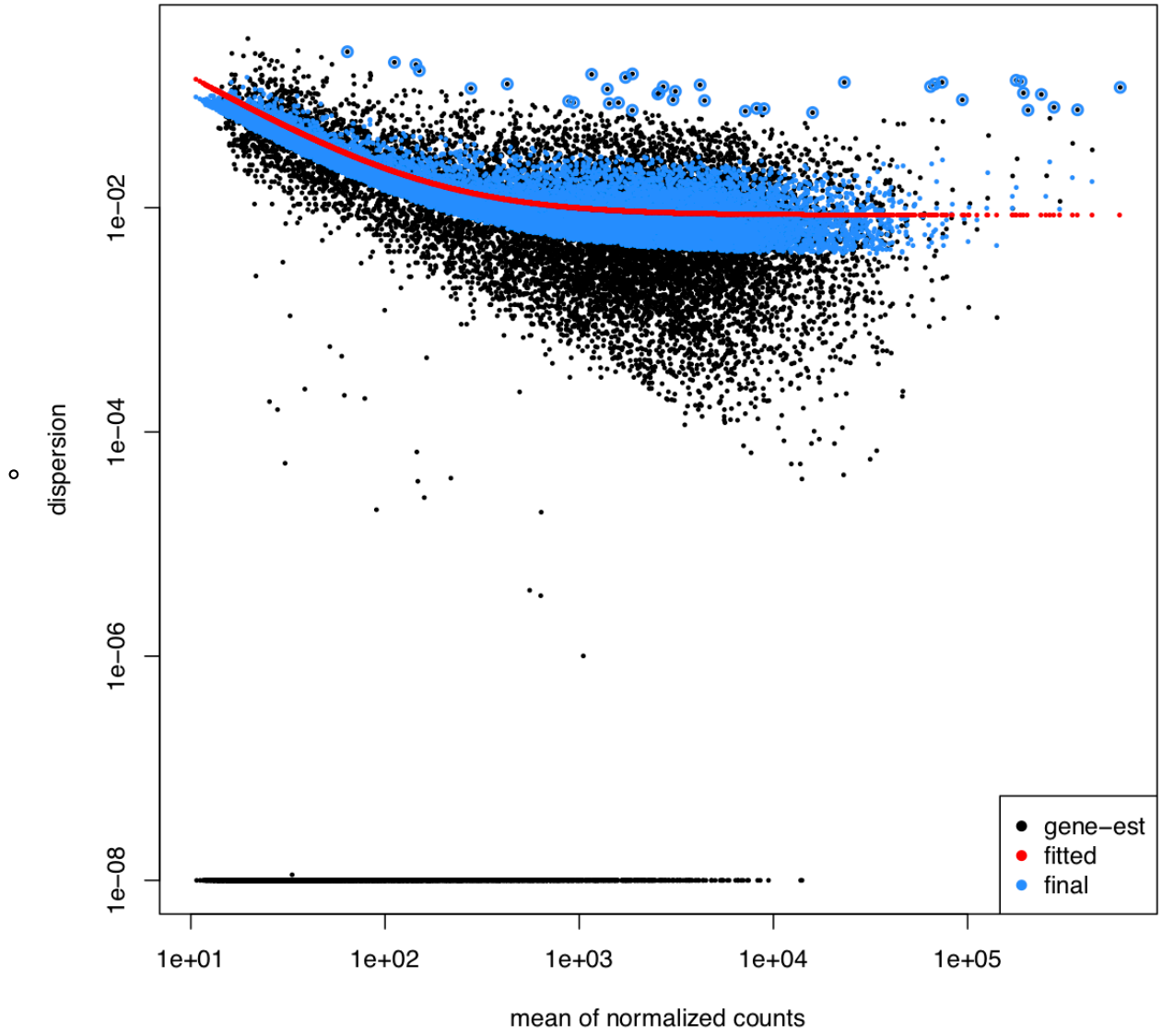
```
$ head RNAseq/DEG/deseq2/featurecounts/star/cri_rnaseq_2018.star.featurecounts.deseq2
.test.txt
```

Statistical test result per gene using DESeq2

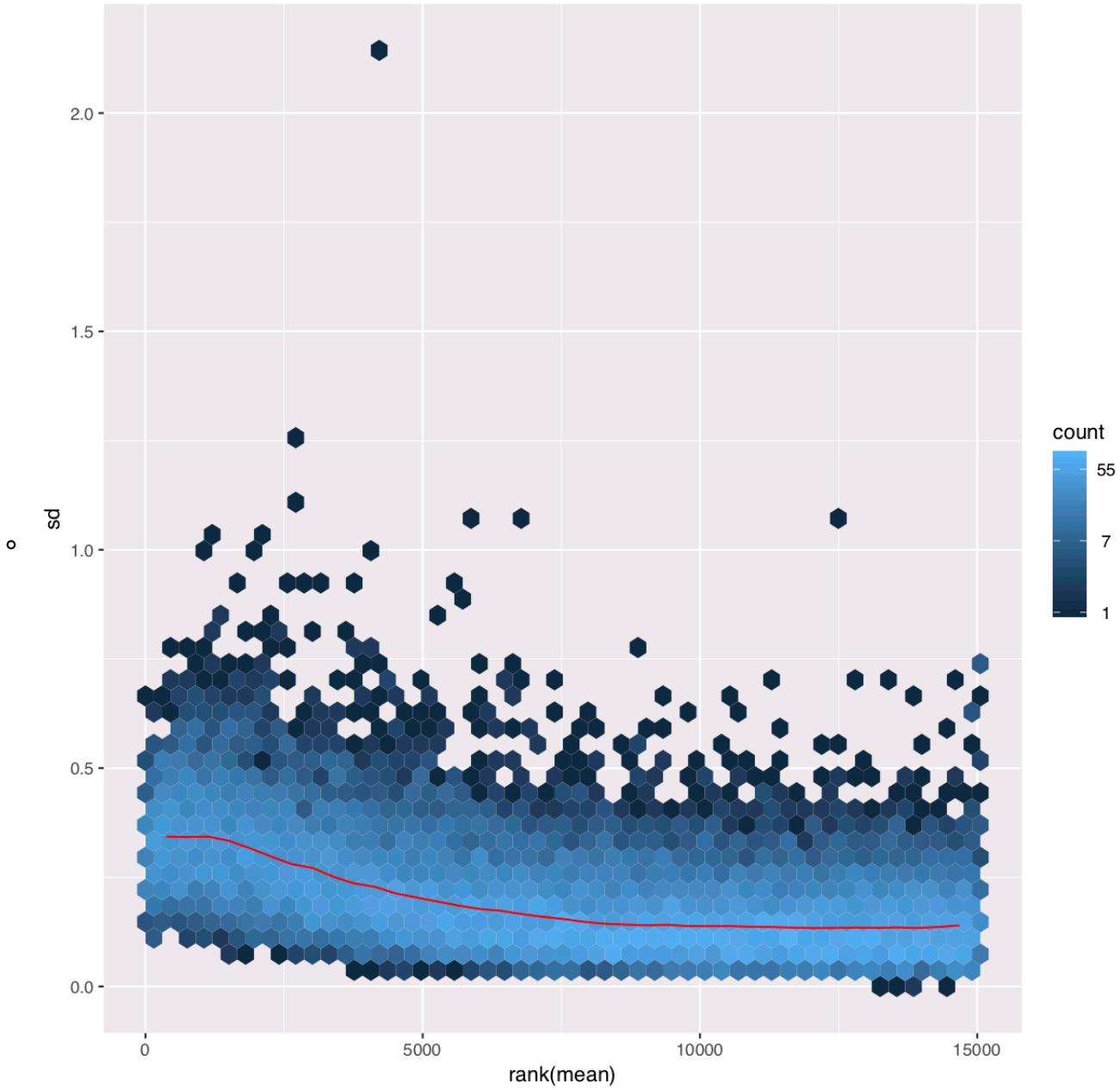
	Geneid	baseMean	log2FoldChange	lfcSE	stat	pvalue	FDR	DE(
4249	ENSG00000120738.7	5193.2844	1.960273	0.1147987	17.07574	0	0	
9872	ENSG00000170345.9	422.7627	1.960006	0.1246756	15.72085	0	0	
4273	ENSG00000113070.7	1441.0826	1.386086	0.1041939	13.30296	0	0	
9601	ENSG00000100867.14	356.3080	1.707939	0.1339178	12.75364	0	0	
8830	ENSG00000229117.8	7121.3363	-1.270634	0.1061629	-11.96872	0	0	-
7342	ENSG00000035403.17	855.0647	-1.158327	0.1024377	-11.30763	0	0	-
587	ENSG00000177606.6	556.6779	1.267243	0.1122316	11.29132	0	0	

Or, the exploratory plots of the example dataset produced by DESeq2 (https://bioconductor.org/packages/release/bioc/html/DESeq2.html) will be as follows.

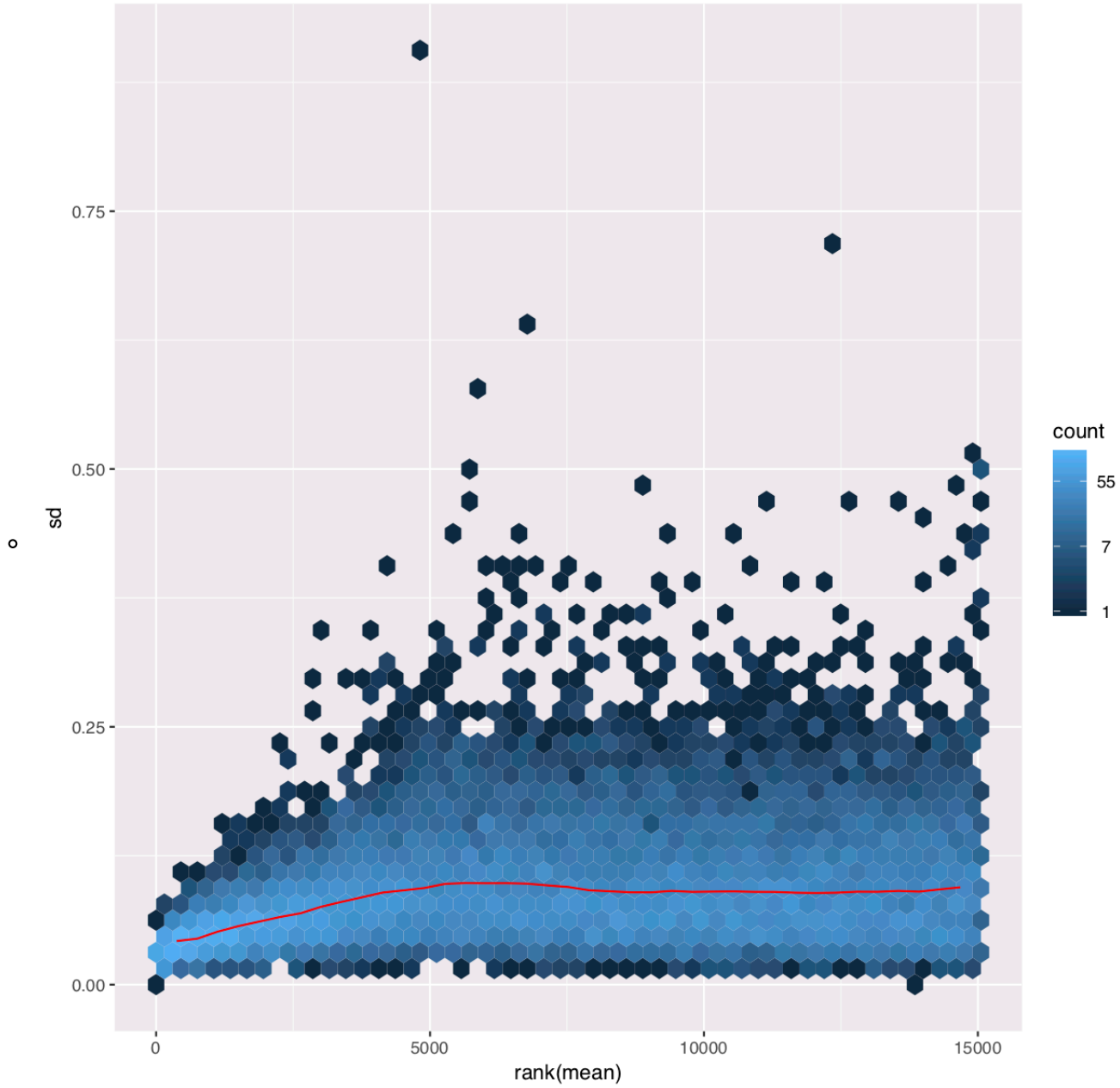
1. An exploratory plot of the per-gene dispersion estimates together with the fitted mean-dispersion relationship



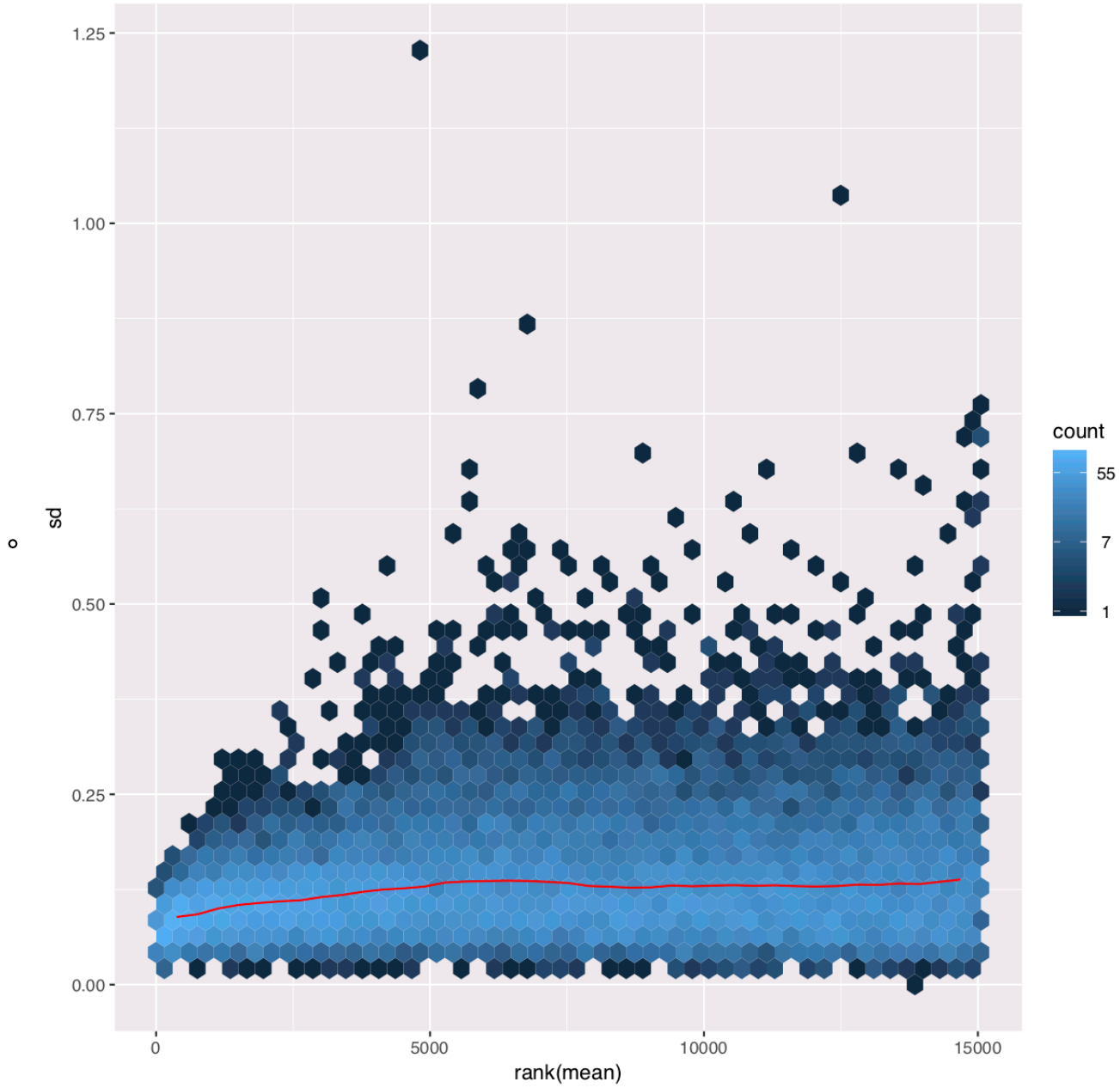
2. An exploratory plot of row standard deviations versus row means using the normalized counts transformation (f(count + pc))



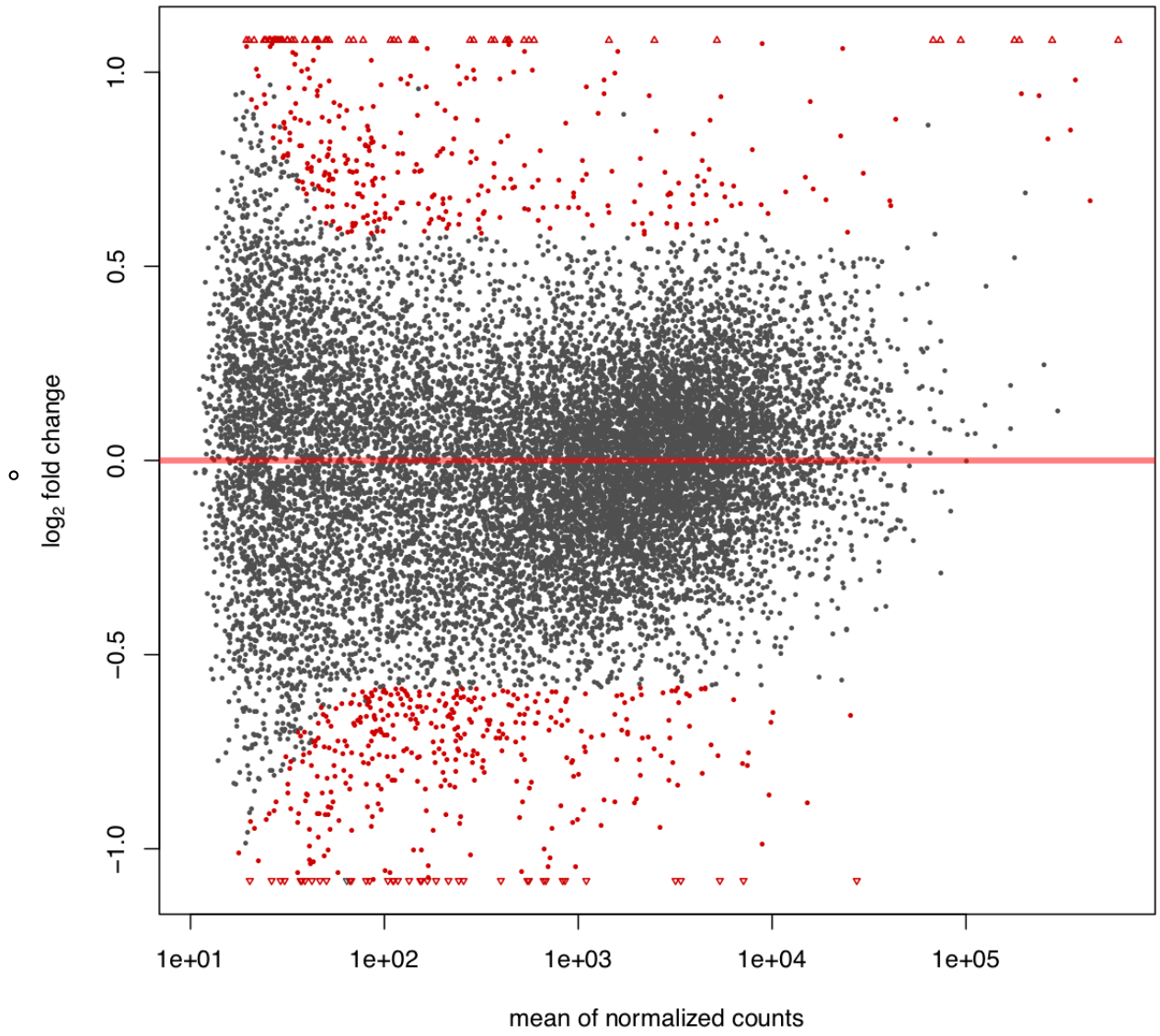
3. An exploratory plot of row standard deviations versus row means using the variance stabilizing transformation



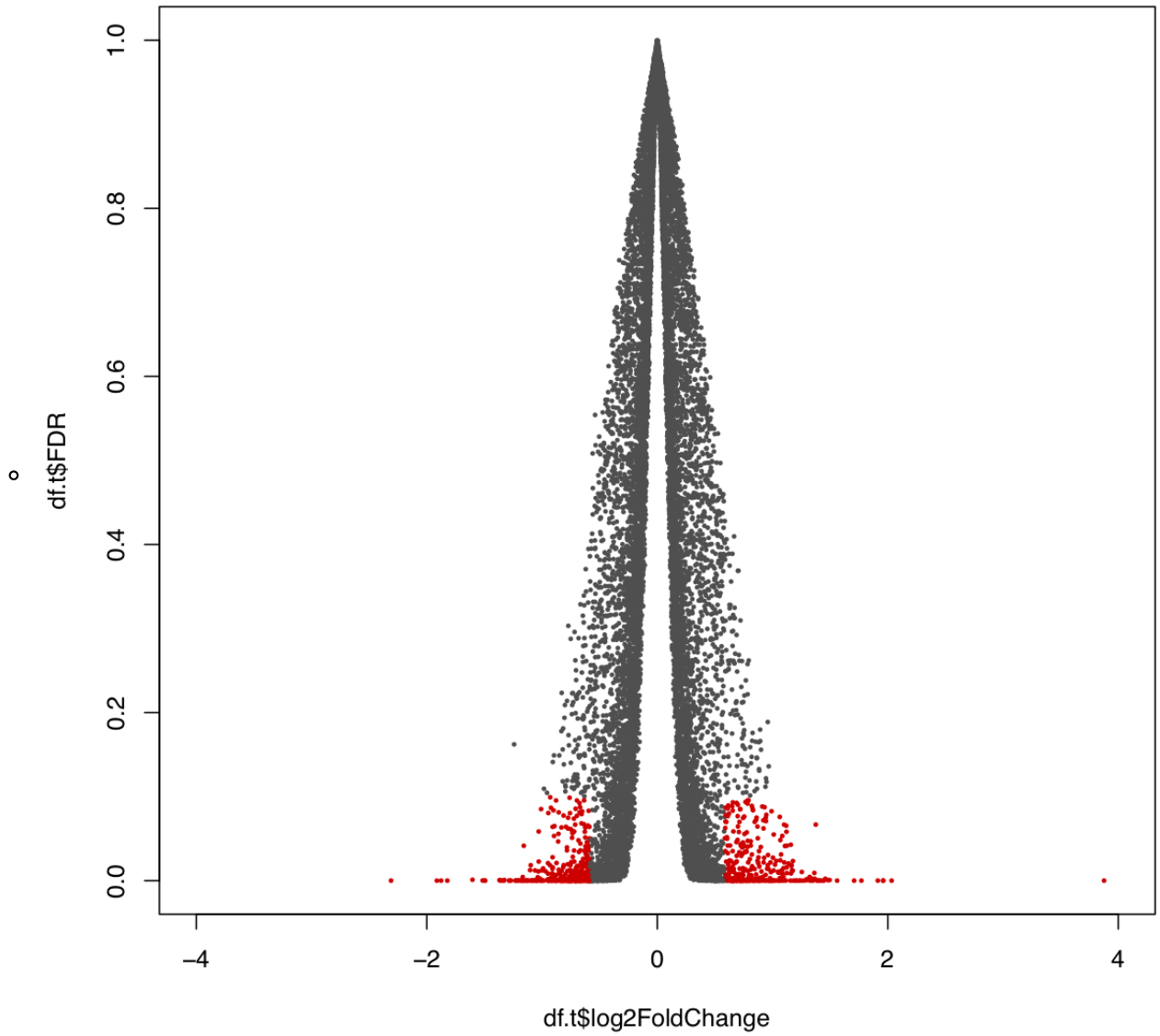
4. An exploratory plot of row standard deviations versus row means using the 'regularized log' transformation



5. A MA plot of log2 fold changes (on the y-axis) versus the mean of normalized counts (on the x-axis)



6. A scatter plot of log2 fold changes (on the y-axis) versus the FDR (on the x-axis)



Step 4-2: DEG Statistics | Top

In this step, the pipeline will collect DEG statistics and identify the overlapping set of identified DEGs from the previous methods.

The BDS code snippet for the sample KO01 will look like:

```
$ grep -A1 run.lociStat.featurecounts.star.*.sh Submit_*.bds
```

```
## dep( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/LociStat/featurecounts/star/cri_rnaseq_201
8/cri_rnaseq_2018.star.featurecounts.overlap.txt' ] <- [ '/CRI/HPC/cri_rnaseq_2018/RN
Aseq/DEG/edger/featurecounts/star/cri_rnaseq_2018.star.featurecounts.edger.test.DEG.t
xt', '/CRI/HPC/cri_rnaseq_2018/RNAseq/DEG/deseq2/featurecounts/star/cri_rnaseq_2018.s
tar.featurecounts.deseq2.test.DEG.txt', '/CRI/HPC/cri_rnaseq_2018/RNAseq/DEG/limma/fe
aturecounts/star/cri_rnaseq_2018.star.featurecounts.limma.test.DEG.txt' ], cpus := 4,
mem := 32*G, timeout := 72*hour, taskName := "lociStat.featurecounts.star.cri_rnaseq_
2018") sys bash /CRI/HPC/cri_rnaseq_2018/RNAseq/shell_scripts/run.lociStat.featurecou
nts.star.cri_rnaseq_2018.sh; sleep 2
## goal( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/LociStat/featurecounts/star/cri_rnaseq_20
18/cri_rnaseq_2018.star.featurecounts.overlap.txt' ] )
```

This code chunk will invoke the bash script (e.g., RNAseq/shell_scripts/ **`run.lociStat.featurecounts.star.cri_rnaseq_2018.sh`**) to collect DEG statistics and to make a Venn diagram plot.

After the completion of the entire pipeline, you can check the statistics result of DEGs per method; for instance, the example dataset DLBC will be as follows.

```
$ grep -A5 'Up/Down regulated DEGs per methods'
RNAseq/LociStat/featurecounts/star/*/run.lociStat.featurecounts.star.*.log | tail -n+
2
```

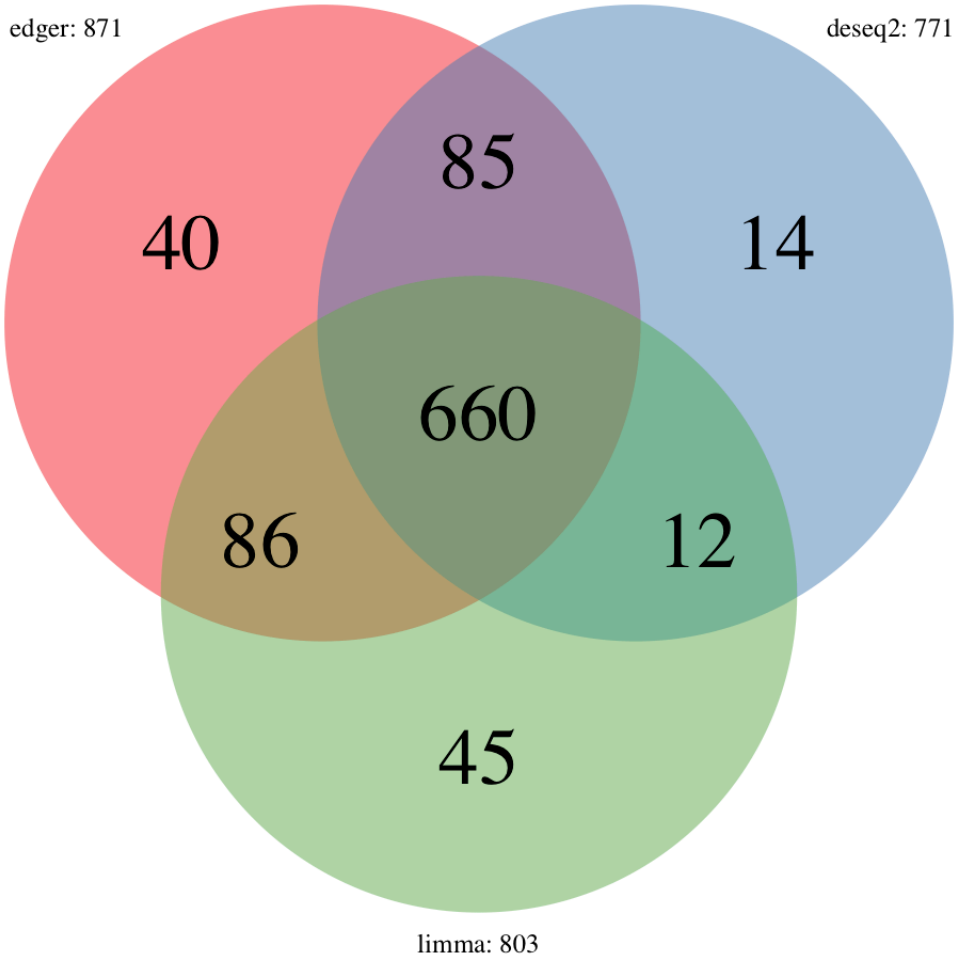
```
## grep: result/run.lociStat.featurecounts.star.*.log: No such file or directory
```

```
$ cut -f1,2,4 RNAseq/LociStat/featurecounts/star/*/*.star.featurecounts.VennList.txt
```

DEG Statistics

Methods	Method.Num	ID.Num
edger	1	40
deseq2	1	14
limma	1	45
edger&deseq2	2	85
edger&limma	2	86
deseq2&limma	2	12
edger&deseq2&limma	3	660

There is a Venn diagram plot will be generated after this step.



Step 5: Sample Correlation | Top

In this step, the pipeline will make a PCA plot based on the transcriptional profiling of all samples.

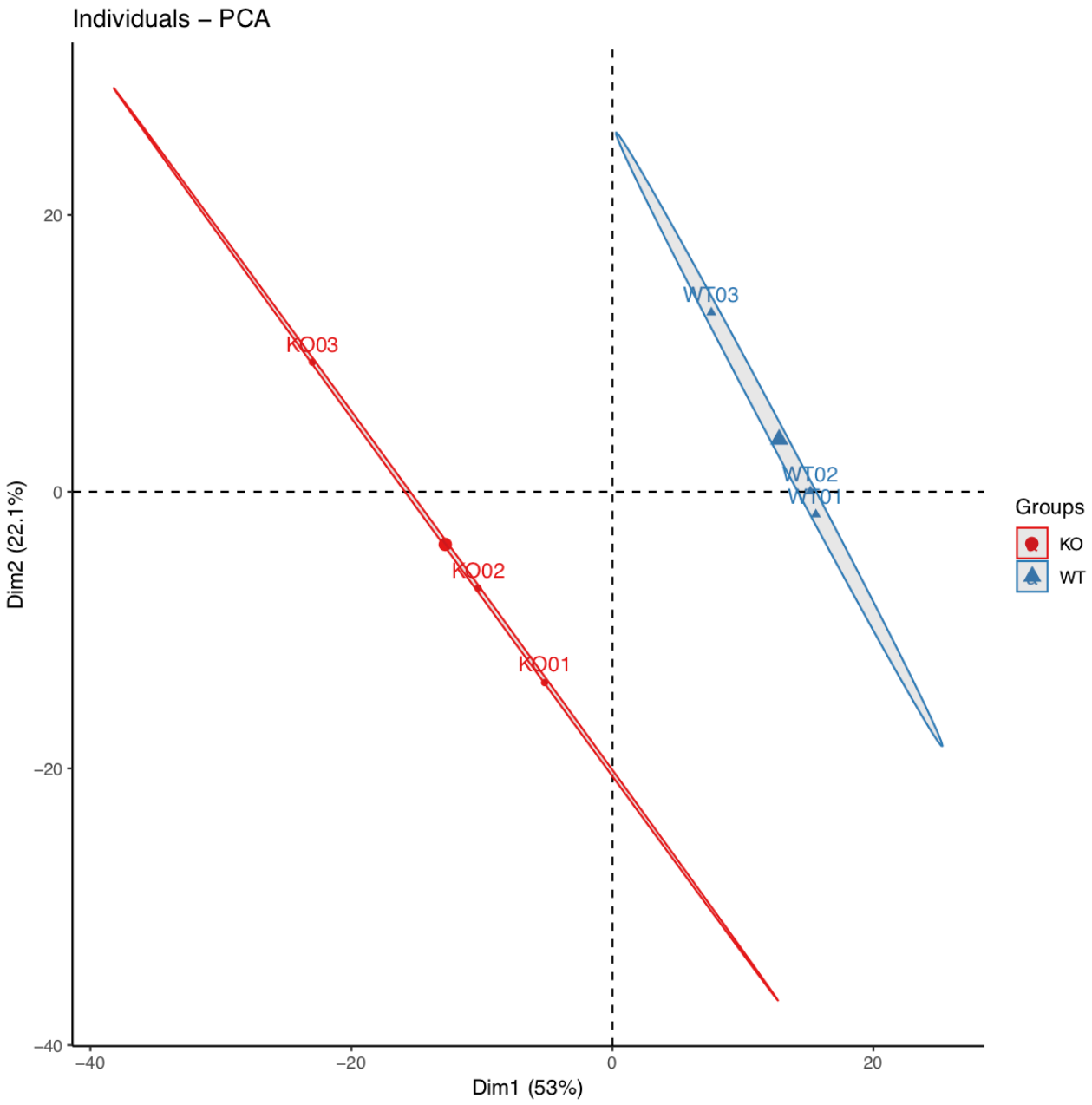
The BDS code snippet for the sample KO01 will look like:

```
$ grep -A1 run.quantQC.pca.featurecounts.star.*.sh Submit_*.bds
```

```
## dep( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/QuantQC/featurecounts/star/cri_rnaseq_2018
.star.featurecounts.pca.pdf' ] <- [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/DEG/deseq2/featu
recounts/star/cri_rnaseq_2018.star.featurecounts.deseq2.count.txt' ], cpus := 8, mem
:= 64*G, timeout := 72*hour, taskName := "pca.featurecounts.star.cri_rnaseq_2018") sy
s bash /CRI/HPC/cri_rnaseq_2018/RNAseq/shell_scripts/run.quantQC.pca.featurecounts.st
ar.cri_rnaseq_2018.sh; sleep 2
## goal( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/QuantQC/featurecounts/star/cri_rnaseq_201
8.star.featurecounts.pca.pdf' ] )
```

This code chunk will invoke the bash script (e.g., RNAseq/shell_scripts/ **`run.quantQC.pca.featurecounts.star.cri_rnaseq_2018.sh`**) to make a PCA plot based on the alignment quantification result generated by DESeq2 (https://bioconductor.org/packages/release/bioc/html/DESeq2.html) or one of DE analysis tools.

After the completion of the entire pipeline, you can check the PCA plot under the folder of `QuantQC/` .



Step 6: Heat Map | Top

In this step, the pipeline will make a heat map based on the overlapping set of DEGs identified across different DE analysis tools.

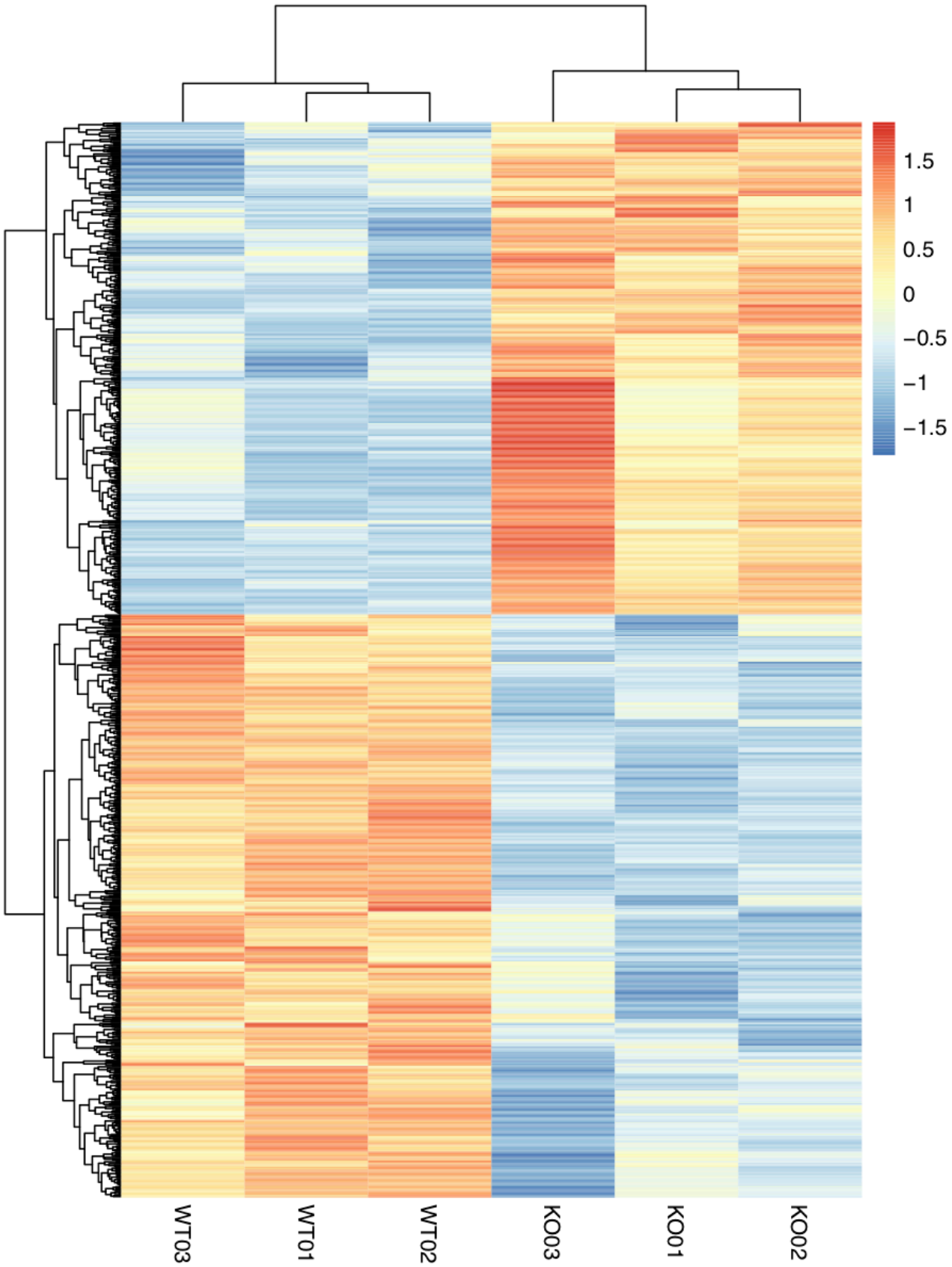
The BDS code snippet for the sample KO01 will look like:

```
$ grep -A1 run.postAna.pheatmap.featurecounts.star.*.sh Submit_*.bds
```

```
## dep( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/PostAna/pheatmap/featurecounts/star/cri_rn
aseq_2018/cri_rnaseq_2018.star.featurecounts.heatmap.pdf' ] <- [ '/CRI/HPC/cri_rnaseq
_2018/RNAseq/LociStat/featurecounts/star/cri_rnaseq_2018/cri_rnaseq_2018.star.feature
counts.overlap.txt' ], cpus := 4, mem := 32*G, timeout := 72*hour, taskName := "postA
na.pheatmap.featurecounts.star.cri_rnaseq_2018") sys bash /CRI/HPC/cri_rnaseq_2018/RN
Aseq/shell_scripts/run.postAna.pheatmap.featurecounts.star.cri_rnaseq_2018.sh; sleep
2
## goal( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/PostAna/pheatmap/featurecounts/star/cri_r
naseq_2018/cri_rnaseq_2018.star.featurecounts.heatmap.pdf' ] )
```

This code chunk will invoke the bash script (e.g., RNAseq/shell_scripts/ **`run.postAna.pheatmap.featurecounts.star.cri_rnaseq_2018.sh`**) to make a heat map plot based on the overlapping set of DEGs identified across different DE analysis tools (i.e., edgeR (https://bioconductor.org/packages/release/bioc/html/edgeR.html), DESeq2 (https://bioconductor.org/packages/release/bioc/html/DESeq2.html), and limma (https://bioconductor.org/packages/release/bioc/html/limma.html)).

After the completion of the entire pipeline, you can check the heat map under the folder of `PostAna/pheatmap` .



Step 7: Functional Enrichment Analysis | Top

In this step, the pipeline will conduct enrichment analysis and make several exploratory plots based on the overlapping set of DEGs identified across different DE analysis tools.

The BDS code snippet for the projet DLBC will look like:

```
$ grep -A1 run.postAna.clusterprofiler.featurecounts.star.*.sh Submit_*.bds
```

```
## dep( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/PostAna/clusterprofiler/featurecounts/star
/cri_rnaseq_2018/cri_rnaseq_2018.star.featurecounts.enrichGO.ALL.txt' ] <- [ '/CRI/HP
C/cri_rnaseq_2018/RNAseq/LociStat/featurecounts/star/cri_rnaseq_2018/cri_rnaseq_2018.
star.featurecounts.overlap.txt' ], cpus := 4, mem := 32*G, timeout := 72*hour, taskNa
me := "postAna.clusterprofiler.featurecounts.star.cri_rnaseq_2018") sys bash /CRI/HPC
/cri_rnaseq_2018/RNAseq/shell_scripts/run.postAna.clusterprofiler.featurecounts.star.
cri_rnaseq_2018.sh; sleep 2
## goal( [ '/CRI/HPC/cri_rnaseq_2018/RNAseq/PostAna/clusterprofiler/featurecounts/sta
r/cri_rnaseq_2018/cri_rnaseq_2018.star.featurecounts.enrichGO.ALL.txt' ] )
```

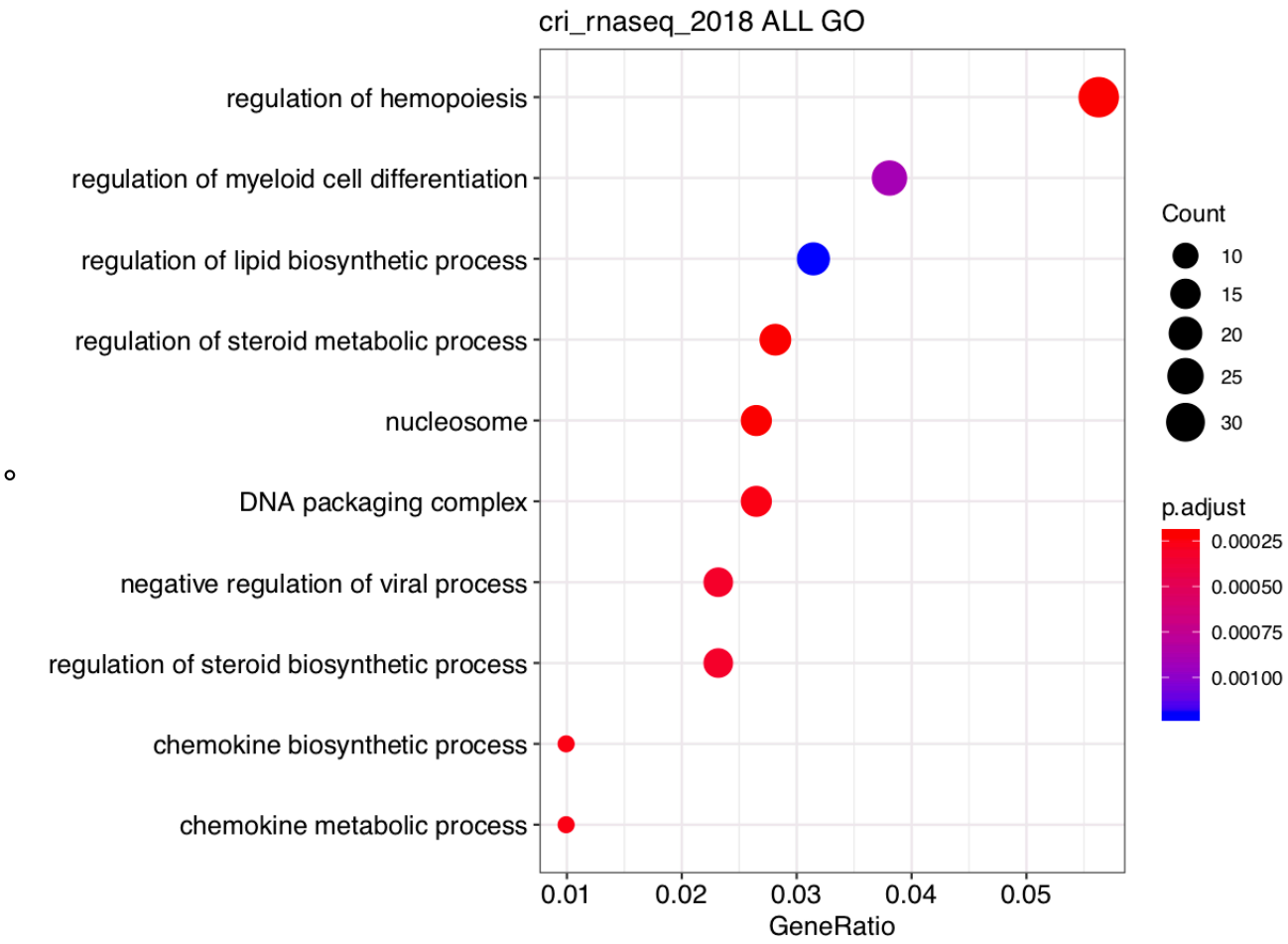
This code chunk will invoke the bash script (e.g., RNAseq/shell_scripts/ **`run.postAna.clusterprofiler.featurecounts.star.cri_rnaseq_2018.sh`**) to conduct enrichment analyses including GO (http://www.geneontology.org/) and KEGG (https://www.genome.jp/kegg/) pathway erichment analyses as well as gene set enrichment analysis (GSEA).

After the completion of the entire pipeline, you can check the heat map under the folder of `PostAna/pheatmap` .

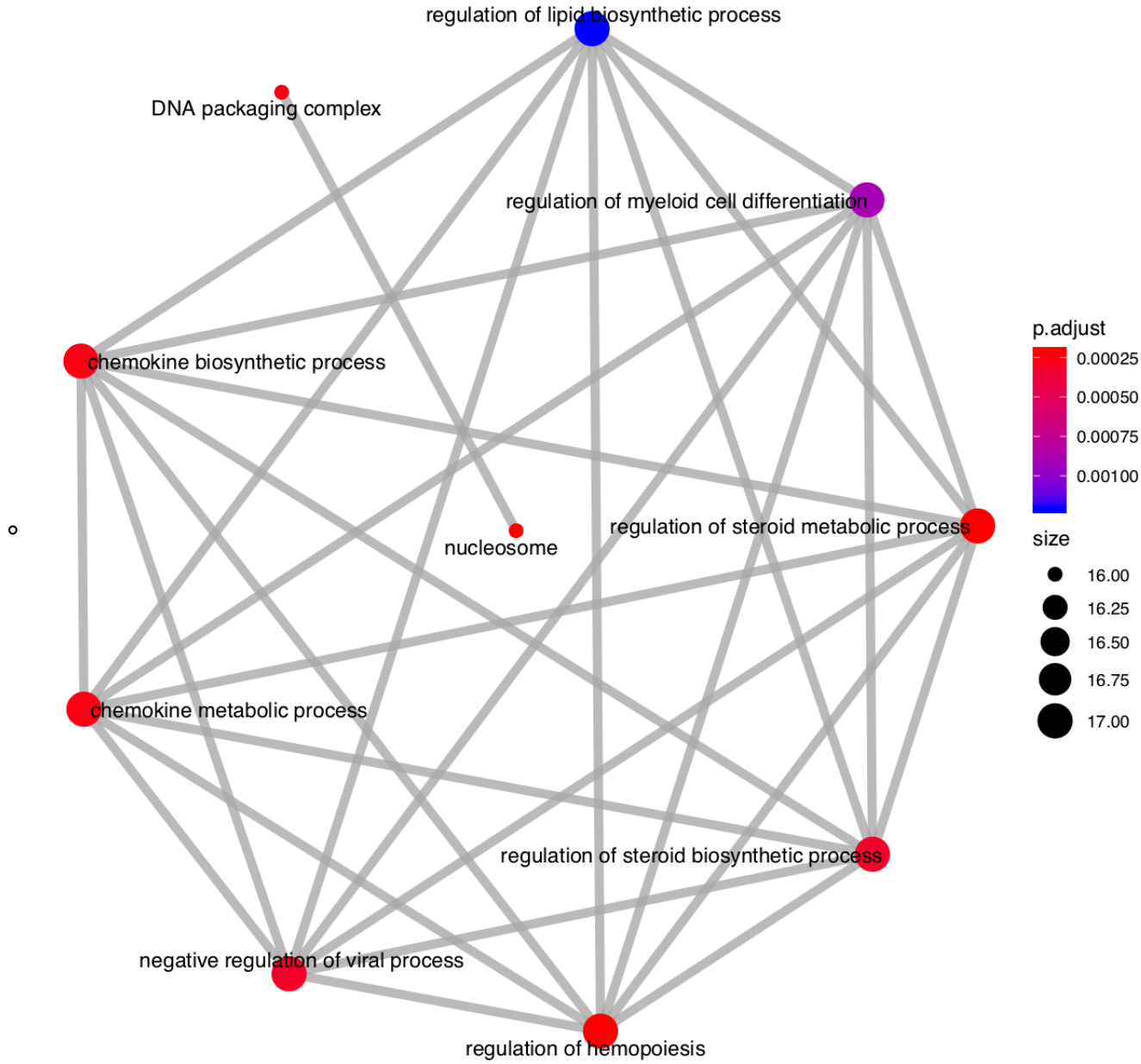
```
$ tree RNAseq/PostAna/clusterprofiler/featurecounts/star/*/
RNAseq/PostAna/clusterprofiler/featurecounts/star/cri_rnaseq_2018/
|-- cri_rnaseq_2018.star.featurecounts.enrichGO.ALL.cnetplot.pdf
|-- cri_rnaseq_2018.star.featurecounts.enrichGO.ALL.dotplot.pdf
|-- cri_rnaseq_2018.star.featurecounts.enrichGO.ALL.emapplot.pdf
|-- cri_rnaseq_2018.star.featurecounts.enrichGO.ALL.txt
|-- cri_rnaseq_2018.star.featurecounts.enrichGSEAGO.ALL.neg001.pdf
|-- cri_rnaseq_2018.star.featurecounts.enrichGSEAGO.ALL.pos001.pdf
|-- cri_rnaseq_2018.star.featurecounts.enrichGSEAGO.ALL.txt
|-- cri_rnaseq_2018.star.featurecounts.enrichGSEAKEGG.pos001.pdf
|-- cri_rnaseq_2018.star.featurecounts.enrichGSEAKEGG.txt
|-- cri_rnaseq_2018.star.featurecounts.enrichKEGG.cnetplot.pdf
|-- cri_rnaseq_2018.star.featurecounts.enrichKEGG.dotplot.pdf
|-- cri_rnaseq_2018.star.featurecounts.enrichKEGG.emapplot.pdf
|-- cri_rnaseq_2018.star.featurecounts.enrichKEGG.txt
`-- run.postAna.clusterprofiler.featurecounts.star.cri_rnaseq_2018.log
```

Below, several plots will be generated based on the overlapping set of DEGs using GO (http://www.geneontology.org/) database as example.

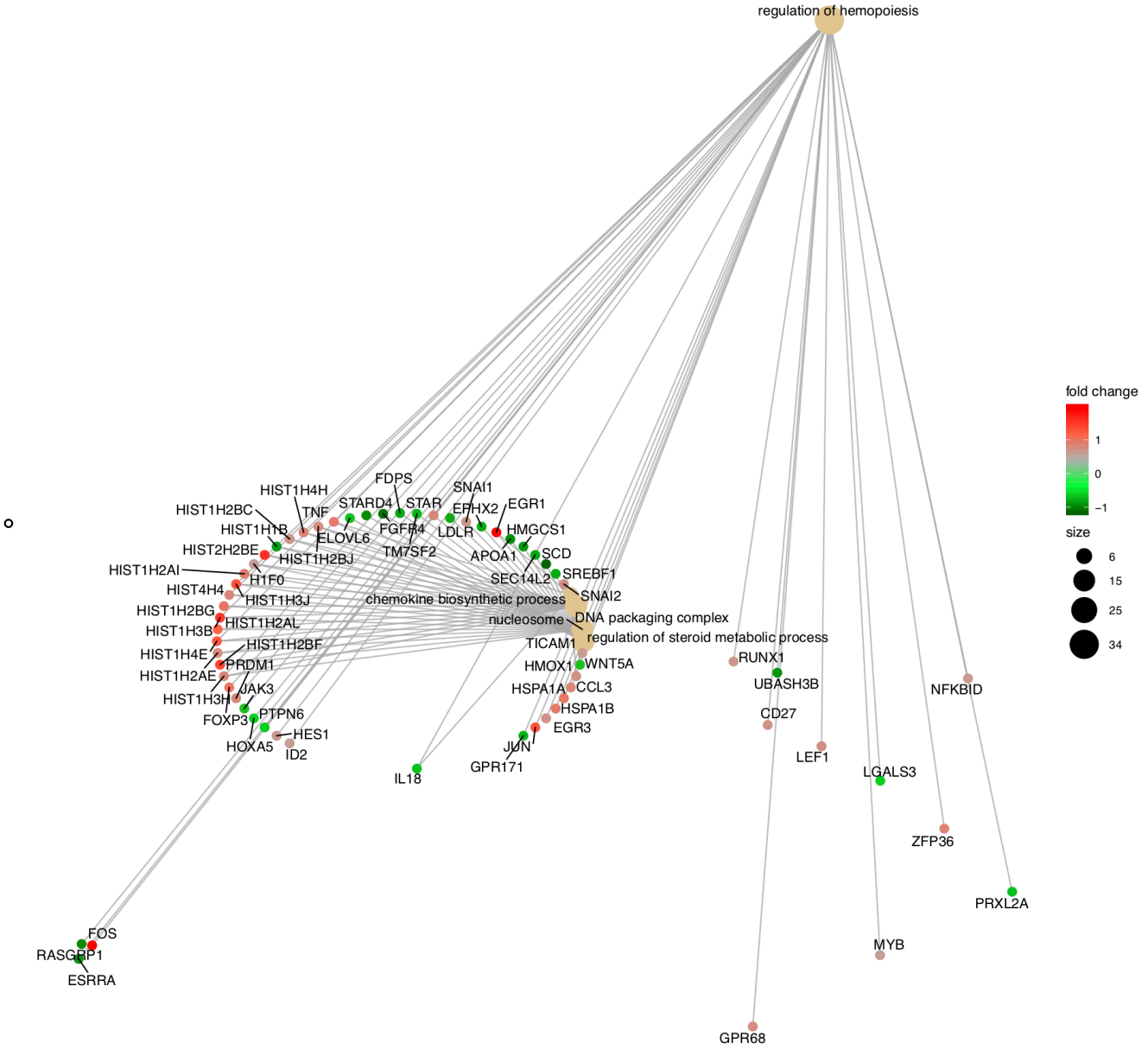
1. An exploratory dot plot for enrichment result



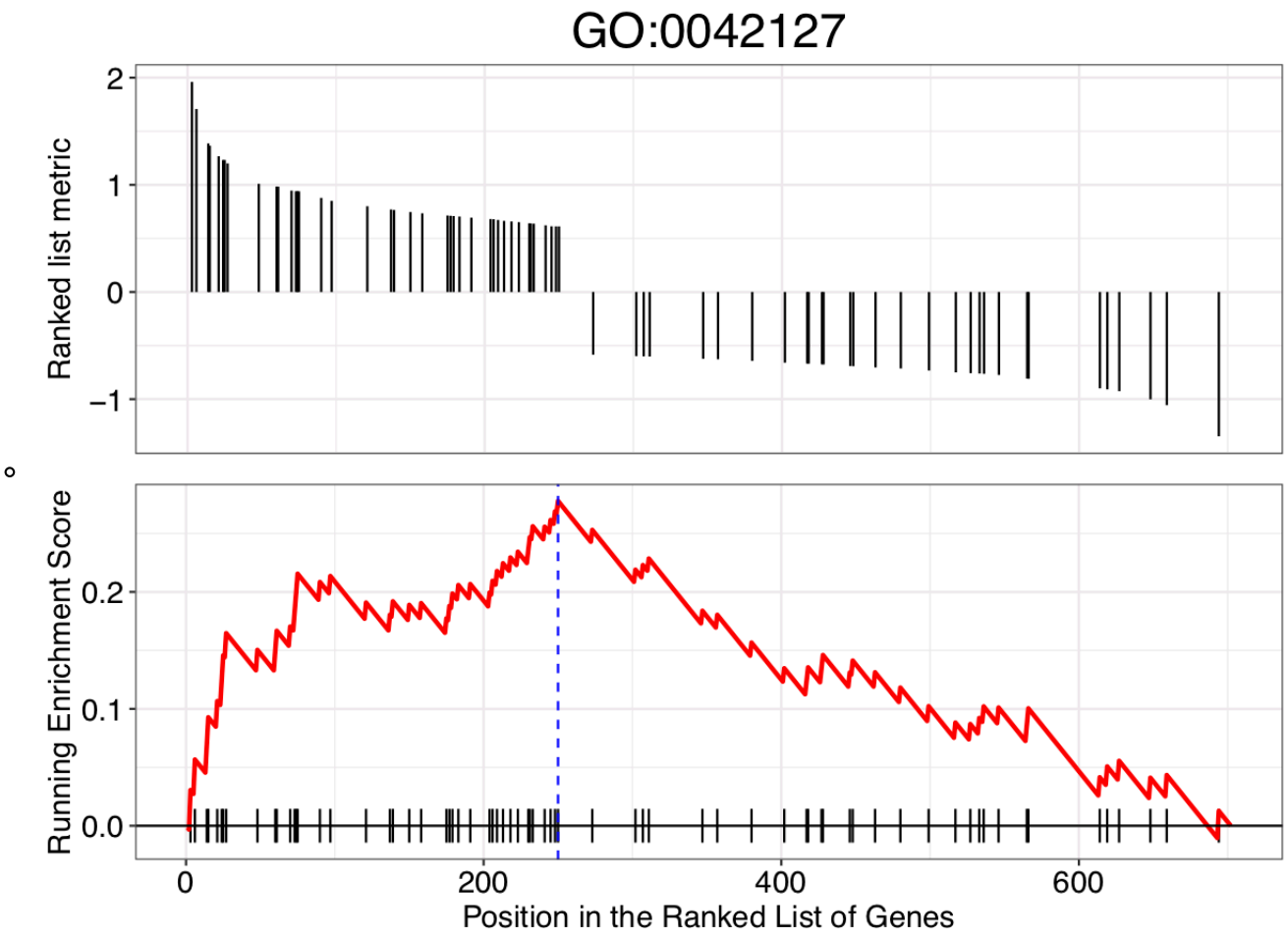
2. An exploratory enrichment map for enrichment result of over-representation test



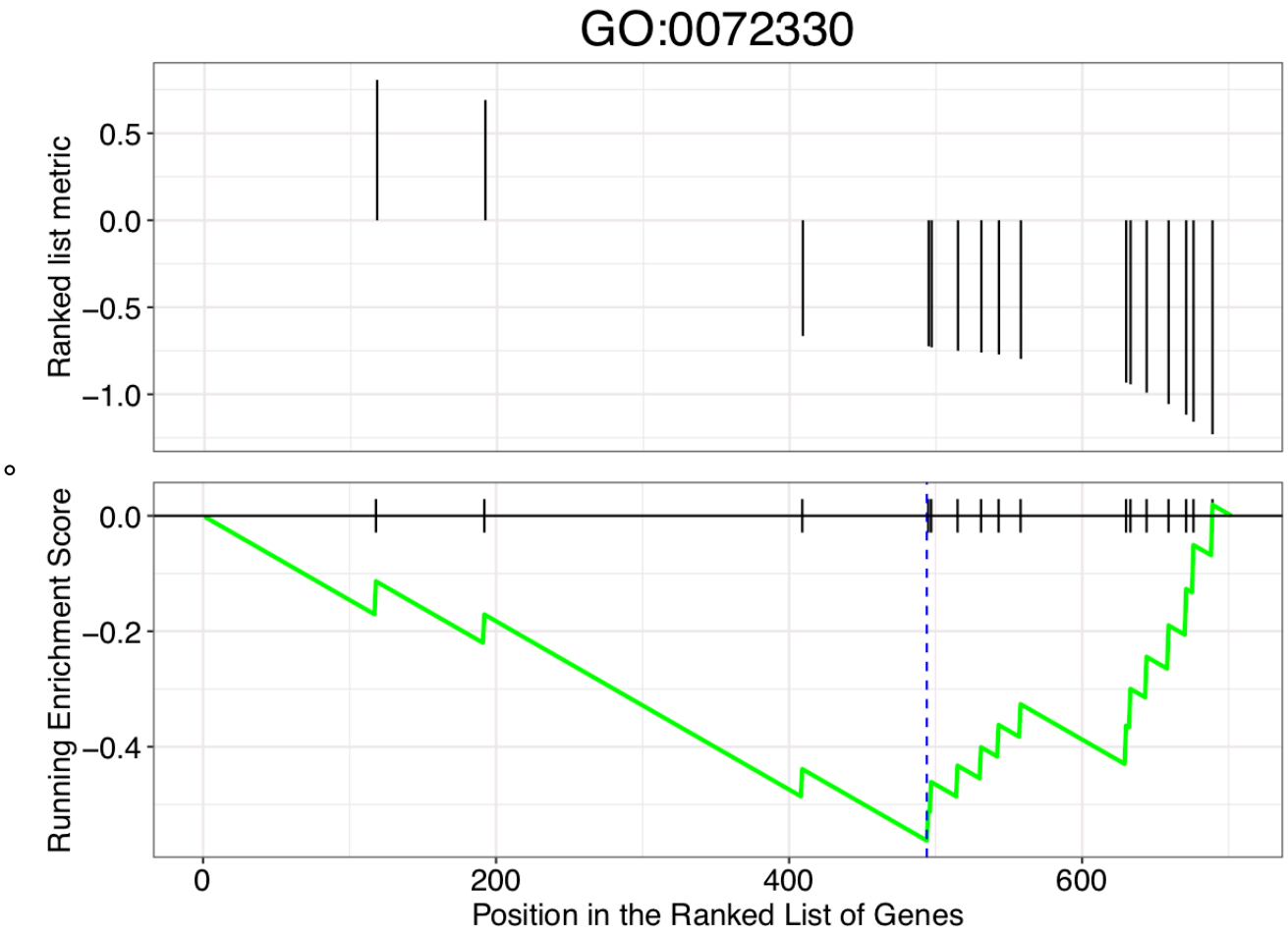
3. An exploratory Gene-Concept Network plot of over-representation test



4. An exploratory plot of GSEA result with NES great than zero



5. An exploratory plot of GSEA result with NES less than zero



BigDataScript Report | Top

Considering the environment setting in the CRI HPC system, BigDataScript (https://pcingola.github.io/BigDataScript/) was used as a job management system in the current development to achieve an automatic pipeline. It can handle the execution dependency of all sub-task bash scripts and resume from a failed point, if any.

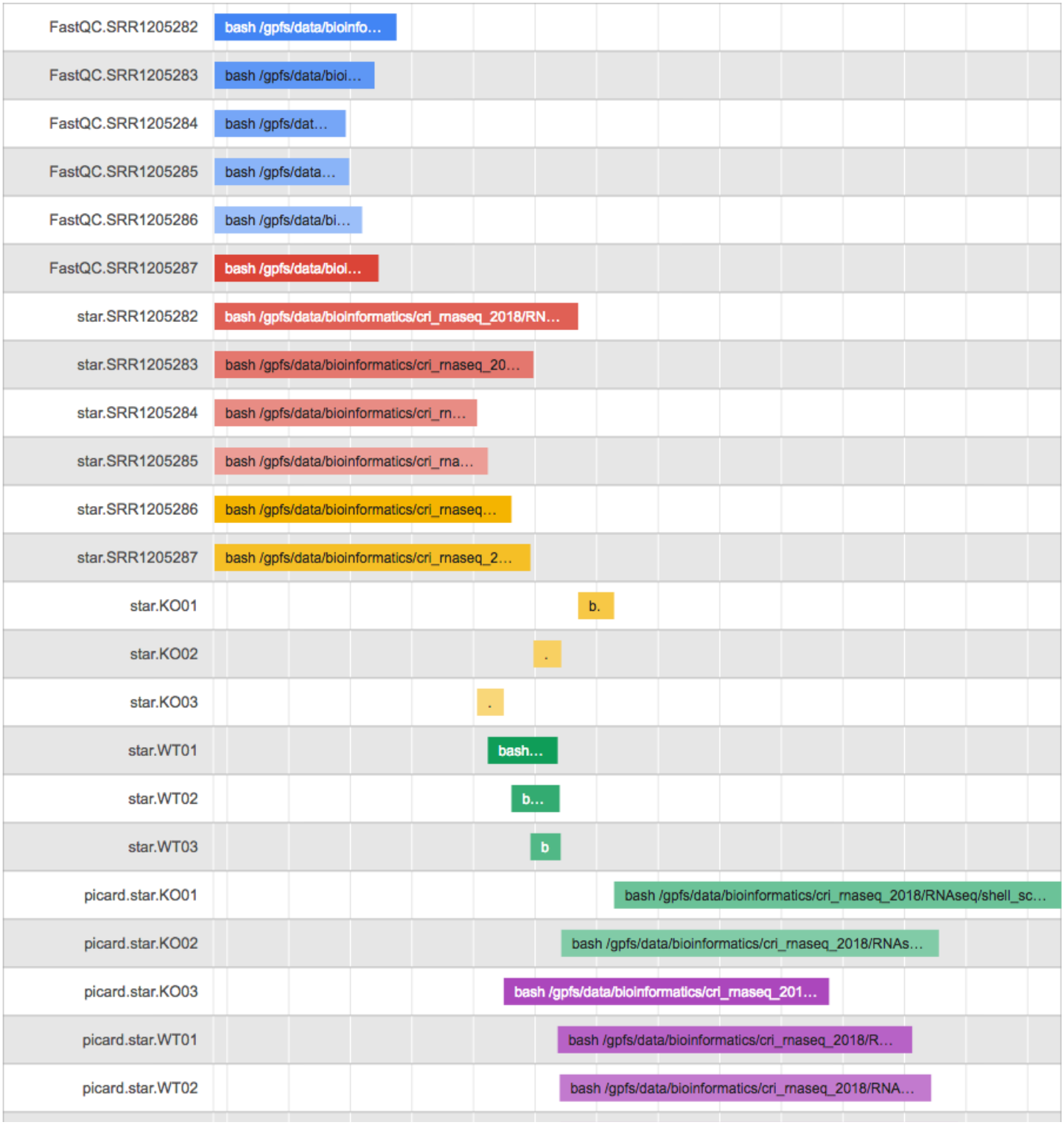
After the completion of the entire pipeline, you will see a BigDataScript report in HTML under the pipeline folder. For instance, this is the report from one test run. The graphic timeline will tell you the execution time per sub-task script.

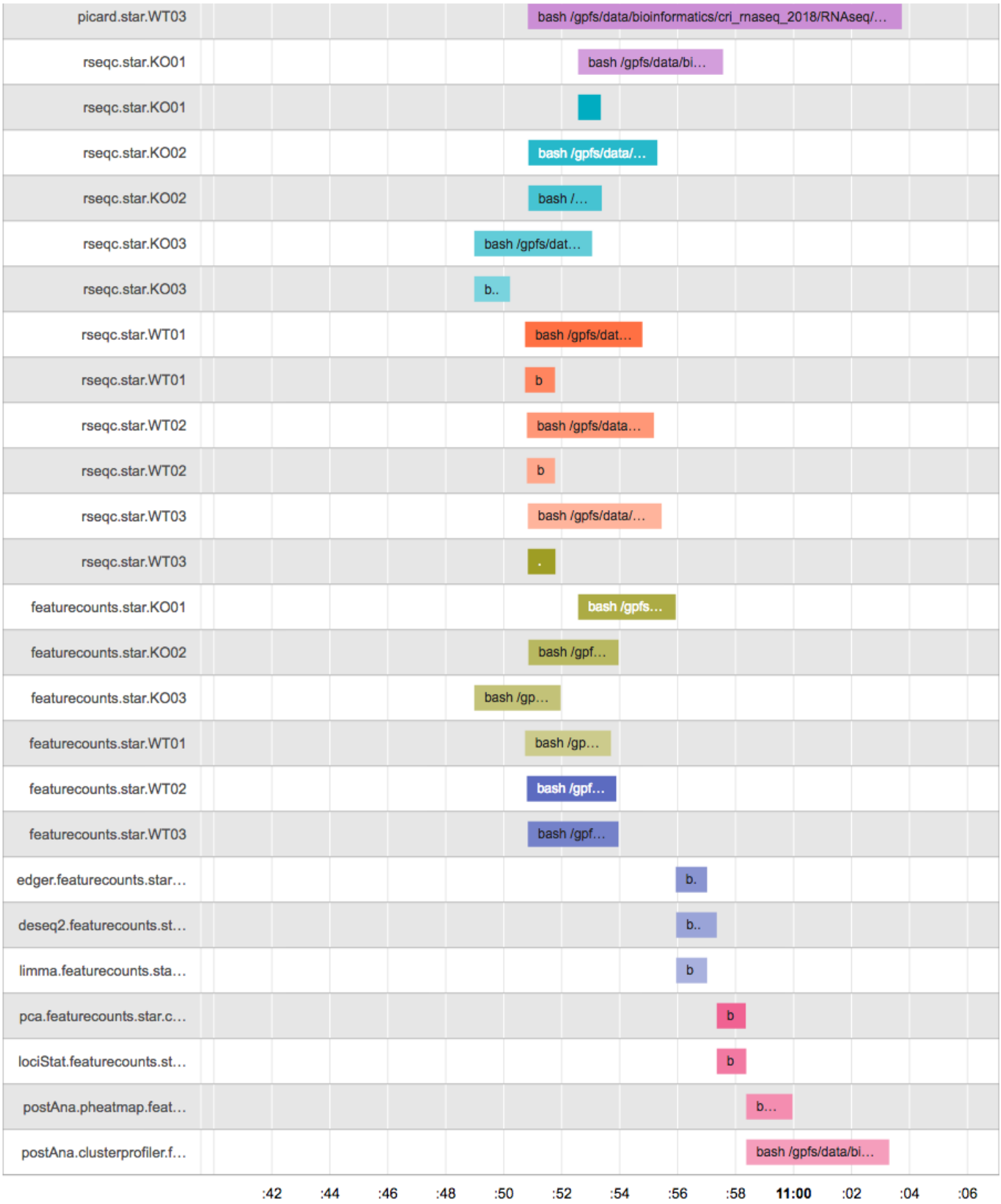
BigDataScript report: Submit_cri_rnaseq_2018.bds

Script file	/CRI/HPC/cri_rnaseq_2018/Submit_cri_rnaseq_2018.bds
Program ID	Submit_cri_rnaseq_2018.bds.20181114_103932_986
Start time	2018-11-14 10:39:32
Run time	00:27:33.131
Tasks executed	49
Tasks failed	0
Tasks failed names	

Arguments* []
System* cluster
Cpus* 1
Exit value 0
* Values in global scope when program finished execution.

Timeline





[Last Updated on 2018/11/14] | Top