Pipeline of Generating Read Counts Table from NCBI SRA

Gu Mi
Department of Statistics, Oregon State University

March 9, 2014

Contents

1 Introduction **1**
1.1 Software packages required 1
1.2 Hardware parameters 2

2 The Pipeline **2**
2.1 Generate SraRunInfo.csv 2
2.2 Download SRA Files 3
2.3 Convert SRA to FASTQ format 3
2.4 Download reference genome sequence 4
2.5 Download gene model annotations 4
2.6 Build the reference index 5
2.7 (Optional) Assess sequence quality control 5
2.8 Collect meta-data for experimental design 5
2.9 Align reads to the reference genome using Tophat2 6
2.10 Organize, sort and index the BAM files and create SAM files 7
2.11 (Optional) Inspect alignments with IGV 7
2.12 Count reads using HTSeq 8
2.13 Get read counts table using edgeR 8

3 Notes **9**

1 Introduction

In this document, I outline the computational pipeline of generating read counts table from the Short Read Archive (SRA) format (`http://www.ncbi.nlm.nih.gov/sra/`) available at NCBI (`http://www.ncbi.nlm.nih.gov/`). Knowledge of each of the steps in the pipeline helps us understand the kinds of variability that may influence the summarized read counts at the final stage; also, by exploring meta-data that come directly from the sequencing machine, we will get more details on how the experiments are performed. This manual follows the protocol of Anders et al. [2], but is more organized with reproducible codes and computational details.

1.1 Software packages required

The following software/packages/toolkits must be installed locally or available on server. The version numbers I use are in parentheses. Please pay *extreme* attention to the versions of the "Tuxedo Suite" (Bowtie, Tophat and Cufflinks) as non-reproducible results have been reported under different versions (see the supplementary material S.6 in Anders et al. [1]).

1. R (3.0.2): install edgeR for using `readDGE`

2. Python (2.7.3): HTSeq is written in Python; also check out DEXSeq (R) for exon-level inference
3. SRA Toolkit (2.3.4-2): `http://eutils.ncbi.nih.gov/Traces/sra/?view=software`
4. Bowtie2 (2.1.0): `http://bowtie-bio.sourceforge.net/bowtie2/index.shtml`
5. Samtools (0.1.19-44428cd): `http://samtools.sourceforge.net/`
6. Tophat2 (2.0.8b): `http://tophat.cbcb.umd.edu/`
7. HTSeq (0.6.1): `http://www-huber.embl.de/users/anders/HTSeq/doc/overview.html`

Installations of these tools works well in *nix-like environments. Sometimes you may have to add path to the $PATH variable in order to invoke commands globally (otherwise you need to specify the binary file location with absolute path).

1.2 Hardware parameters

All the computations are done using a Macbook Pro with Mac OS X 10.9.2, 2.7 GHz Intel Core i7 processor, 8 GB 1333 MHz DDR3 memory, and 160 GB hard drive available. The number of physical CPU cores is two, so four virtual CPUs are available via multithreading.

2 The Pipeline

Suppose the organism we study is *Mus musculus* (mouse) and the study goal is to compare signals from competent and abnormal human embryos impacted differently on the expression of endometrial receptivity genes in mouse uteri. A comprehensive webpage for this study can be found at `http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE47019`, via the NCBI Gene Expression Omnibus (GEO). Listed below are some key accession numbers:

- the GEO accession number **GSE47019**, which should be available in a journal paper's supplementary material (check PMID), is used to query datasets; we can generate a *gds_result.txt* file: click on "Send to:" (top right corner), select "File", select format "Summary", and click "Create File"
- the SRA number **SRP022850** can be found on the same webpage (under "Relations"); the link directs us to a new page where a *SraRunInfo.csv* file can be generated (see below)

2.1 Generate SraRunInfo.csv

Go to `http://www.ncbi.nlm.nih.gov/sra/?term=SRP022850` (the SRA repository for GSE47019); click on "Send to:" (top right corner), select "File", select format "RunInfo", and click "Create File". The file generated is named "SraRunInfo.csv". Such meta-data from SRA is recommended. Next, we read in this CSV file in R for comprehensive information on the experiments.

```r
## directory set-up:
path = "/Users/mig/OSU/RNASeq/Project_2014/Mus"
path.sra = "/Users/mig/OSU/RNASeq/sratoolkit.2.3.4-2-mac64/bin"
path.d = file.path(path, "Data")
path.o = file.path(path, "Output")
setwd(path)
## read in SraRunInfo.csv and check a subset:
sri = read.csv("./Data/SraRunInfo.csv", stringsAsFactors = FALSE)
sri[1:10, c(1, 2, 12, 15, 18)]
```

```
##          Run         ReleaseDate LibraryStrategy LibraryLayout Platform
## 1  SRR851846 2014-02-11 18:33:08         RNA-Seq        SINGLE ILLUMINA
## 2  SRR851847 2014-02-11 18:33:08         RNA-Seq        SINGLE ILLUMINA
## 3  SRR851848 2014-02-11 18:33:08         RNA-Seq        SINGLE ILLUMINA
## 4  SRR851849 2014-02-11 18:33:08         RNA-Seq        SINGLE ILLUMINA
## 5  SRR851850 2014-02-11 18:33:08         RNA-Seq        SINGLE ILLUMINA
## 6  SRR851851 2014-02-11 18:33:08         RNA-Seq        SINGLE ILLUMINA
## 7  SRR851852 2014-02-11 18:33:08         RNA-Seq        SINGLE ILLUMINA
## 8  SRR863033 2014-01-31 00:00:00         RNA-Seq        SINGLE ILLUMINA
## 9  SRR851853 2014-02-11 18:33:08         RNA-Seq        SINGLE ILLUMINA
## 10 SRR851854 2014-02-11 18:33:08         RNA-Seq        SINGLE ILLUMINA
```

By inspection of the entries and compare with the *gds_result.txt* file, we see Run SRR863033 (the 8th row) is problematic and does not belong to any of the samples. We therefore exclude that row. Many columns are not shown here, but some of them are very useful (e.g. the "LibraryLayout" column indicating single-/paired-end reads).

2.2 Download SRA Files

File format: SRA
File size in total: 8 – 9 GB (1 GB / sample, for nine samples)

The Sequence Read Archive (SRA) stores raw sequencing data from the next generation of sequencing platforms including Roche 454 GS System, Illumina Genome Analyzer, Applied Biosystems SOLiD System, Helicos Heliscope, Complete Genomics, and Pacific Biosciences SMRT. We use R commands to automatically download all SRA files for each of the samples. They are also available at `ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByStudy/sra/SRP/SRP022/SRP022850/`.

```
## download 9 SRA files (excluding the abnormal entry #8)
sri2 = sri[-8, ]
(fs = basename(sri2$download_path))
for (i in 1:nrow(sri2)) {
    download.file(url = sri2$download_path[i], destfile = paste(file.path(path.d,
        "SRA/"), fs[i], sep = ""))
}
```

As flagged by an earlier inspection of the *SraRunInfo.csv* and *gds_result.txt* files, we exclude the 8th row and do not download its SRA file.

2.3 Convert SRA to FASTQ format

File format: SRA to FASTQ
File size in total: 52 GB (4 – 6 GB / sample, for nine samples), the most disk-consuming step
Time for conversion: 4.3 hours

Typically, sequencing data from a sequencing facility will come in (compressed) FASTQ format. The SRA, however, uses its own, compressed, SRA format. In order to convert the example data to FASTQ, use the `fastq-dump` command from the SRA Toolkit on each SRA file. Again, it's convenient to generate the Unix commands using R, and call the `system` function in R to invoke.

```
stopifnot(all(file.exists(file.path(path.d, "SRA/", fs))))  # assure FTP download was successful
setwd(file.path(path.d, "SRA/"))
dir()
```

```
for (f in fs) {
    cmd = paste(file.path(path.sra, "fastq-dump --split-3"), f)
    # use the fastq-dump command from the SRA Toolkit on each SRA file
    cat(cmd, "\n")  # print out current command to track progress
    system(cmd)  # invoke command
}
```

We use the `-split-3` option, which splits mate-pair reads into separate files. After this command, single and paired-end data will produce one and two FASTQ files, respectively. For paired-end data, the file names will be suffixed _1.FASTQ and _2.FASTQ; otherwise, a single file with the extension .FASTQ will be produced. It's common to see experiements having some single-end and some paired-end samples. In this study, we only have single-end reads, and only one .FASTQ file is generated for each sample.

2.4 Download reference genome sequence

File format: (compressed) FASTA
File size: 4.42 GB

Download the reference genome sequence for the organism under study in (compressed) FASTA format. Some useful resources, among others, include: the general Ensembl FTP server (`www.ensembl.org/info/data/ftp/index.html`), the Ensembl plants FTP server (`plants.ensembl.org/info/data/ftp/index.html`), the Ensembl metazoa FTP server (`metazoa.ensembl.org/info/data/ftp/index.html`) and the University of California Santa Cruz (UCSC) current genomes FTP server (`ftp://hgdownload.cse.ucsc.edu/goldenPath/currentGenomes/`).

We use the Ensembl FTP server at `http://uswest.ensembl.org/info/data/ftp/index.html`: choose the "FASTA (DNA)" link instead of "FASTA (cDNA)", since alignments to the genome, not the transcriptome, are desired. On the download page of `ftp://ftp.ensembl.org/pub/release-75/fasta/mus_musculus/dna/Mus_musculus.GRCm38.75.dna.toplevel.fa.gz`, we use "toplevel" as it combines all chromesomes. Do NOT use files with "rm" (repeat-masked) or "sm" in their names, as handling repeat regions is left to alignment algorithm.

To extract the FASTA file, run in terminal:

```
gunzip Mus_musculus.GRCm38.75.dna.toplevel.fa.gz
```

2.5 Download gene model annotations

File format: (compressed) GTF
File size: 456 MB

Download a gene transfer format (GTF) file with gene models for the organism of interest. For species covered by Ensembl, the Ensembl FTP site mentioned above contains links to such files. Run in terminal for the download and uncompression:

```
wget ftp://ftp.ensembl.org/pub/release-75/gtf/mus_musculus/Mus_musculus.GRCm38.75.gtf.gz
gunzip Mus_musculus.GRCm38.75.gtf.gz
```

Make sure that the gene annotation uses the same coordinate system as the reference FASTA file. This can be checked by the file names: the (compressed) FASTA file name is *Mus_musculus.GRCm38.75.dna.toplevel.fa.gz*, and the GTF file name is *Mus_musculus.GRCm38.75.gtf.gz*. They all use *GRCm38.75*, which means the assembly is Genome Reference Consortium Mouse Reference 38, under Ensembl genebuild release 75.

2.6 Build the reference index

File format: BT2 (Bowtie2)
File size: 3.69 GB
Time for building indices: 3 hours

Before reads can be aligned (via Tophat2, see below), the reference FASTA files need to be preprocessed into an index that allows the aligner easy access. To build a bowtie2-specific index from the FASTA file mentioned above, use the command:

```
bowtie2-build -f Mus_musculus.GRCm38.75.dna.toplevel.fa Mmus_GRCm38_75
```

A set of BT2 files will be produced, with names starting with Mmus_GRCm38_75 as specfied above. This procedure needs to be run only once for each reference genome used. Pre-built indices for many commonly used genomes are available from `tophat.cbcb.umd.edu/igenomes.shtml`, but these indices may not built on the latest release.

2.7 (Optional) Assess sequence quality control

We can assess sequence quality control with the R package ShortRead. Recall that we have converted SRA files to FASTQ format files. The FASTQ format extends FASTA with Phred base quality scores. Use a web browser to inspect the generated HTML file (here, stored in the "fastqQAreport" directory) with the quality-assessment report.

```
library("ShortRead")
# quality assessement on short reads
fqQC = qa(dirPath = file.path(path.d, "SRA/"), pattern = ".fastq$", type = "fastq")
report(fqQC, type = "html")
```

2.8 Collect meta-data for experimental design

Refer to *gds_result.txt* for sample information. The SRA files have `NA` for the column `LibraryName`, so we have to refer to the *gds_result.txt* file and specify manually. We use abbreviations for the conditions: DECEM = Developmentally competent embryo conditioned media; AECM = Arrested embryo conditioned media.

```
sri2 = sri[-8, ]
sri2$SampleName

## [1] "GSM1143032" "GSM1143033" "GSM1143034" "GSM1143035" "GSM1143036"
## [6] "GSM1143037" "GSM1143038" "GSM1143039" "GSM1143040"

sri2$LibraryName = c("Control_Salker_A1", "Control_Salker_A2", "Control_Salker_A3",
    "DCECM_Salker_B1", "DCECM_Salker_B2", "DCECM_Salker_B3", "AECM_Salker_C1",
    "AECM_Salker_C2", "AECM_Salker_C3")
samples = unique(sri2[, c("LibraryName", "LibraryLayout")])
for (i in 1:nrow(samples)) {
    rw = (sri2$LibraryName == samples$LibraryName[i])
    if (samples$LibraryLayout[i] == "PAIRED") {
        samples$fastq1[i] = paste0(sri2$Run[rw], "_1.fastq", collapse = ",")
        samples$fastq2[i] = paste0(sri2$Run[rw], "_2.fastq", collapse = ",")
    } else {
        samples$fastq1[i] = paste0(sri2$Run[rw], ".fastq", collapse = ",")
        samples$fastq2[i] = ""
    }
}
samples$condition = "CTL"
samples$condition[grep("DCECM", samples$LibraryName)] = "DCECM"
samples$condition[grep("AECM", samples$LibraryName)] = "AECM"
samples$shortname = paste(substr(samples$condition, 1, 2), substr(samples$LibraryLayout,
    1, 2), 1:nrow(samples), sep = ".")
samples

##           LibraryName LibraryLayout          fastq1 fastq2 condition
## 1  Control_Salker_A1        SINGLE SRR851846.fastq              CTL
## 2  Control_Salker_A2        SINGLE SRR851847.fastq              CTL
## 3  Control_Salker_A3        SINGLE SRR851848.fastq              CTL
## 4    DCECM_Salker_B1        SINGLE SRR851849.fastq            DCECM
## 5    DCECM_Salker_B2        SINGLE SRR851850.fastq            DCECM
## 6    DCECM_Salker_B3        SINGLE SRR851851.fastq            DCECM
## 7     AECM_Salker_C1        SINGLE SRR851852.fastq             AECM
## 9     AECM_Salker_C2        SINGLE SRR851853.fastq             AECM
## 10    AECM_Salker_C3        SINGLE SRR851854.fastq             AECM
##    shortname
## 1    CT.SI.1
## 2    CT.SI.2
## 3    CT.SI.3
## 4    DC.SI.4
## 5    DC.SI.5
## 6    DC.SI.6
## 7    AE.SI.7
## 9    AE.SI.8
## 10   AE.SI.9
```

As the downstream statistical analysis of differential expression relies on this table, carefully inspect (and correct, if necessary) the meta-data table. In particular, verify that there exists one row per sample, that all columns of information are populated and that the file names, labels and experimental conditions are correct.

2.9 Align reads to the reference genome using Tophat2

File format: BAM (compressed, binary version of SAM files)
File size in total: 14.6 GB (1.5 – 2.0 GB / sample, for nine samples)
Time for alignment: 29.3 hours, the most time-consuming step

The meta-data table `samples` we constructed earlier will be used to construct shell commands for calling tophat2, which accepts FASTQ and FASTA sequence files, and the GTF annotation file. Make sure (1) the FASTA (.fa) file and BT2 (.bt2 from Bowtie2) reference index file are in current directory; (2) correctly specify directory for the GTF and FASTQ files.

```
gf = "Mus_musculus.GRCm38.75.gtf"
bowind = "Mmus_GRCm38_75"  # name of the reference index (build in advance)
# set working directory to FASTA containing .fa and .bt2 files!  change
# Mus_musculus.GRCm38.75.dna.toplevel.fa to Mmus_GRCm38_75.fa; o.w. tophat2
# will reconstitute reference FASTA file from Boutie index
setwd(file.path(path.d, "FASTA"))
cmd = with(samples, paste("tophat2 -G", file.path(path.d, "GTF", gf), "-p 5 -o",
    LibraryName, bowind, file.path(path.d, "SRA", fastq1)))
```

```
for (i in 1:length(cmd)) {
    cat(cmd[i], "\n\n")
    system(cmd[i])
}
```

In the call to tophat2, the option `-G` points tophat2 to a GTF file of annotation to facilitate mapping reads across exon-exon junctions (some of which can be found de novo), `-o` specifies the output directory, `-p` specifies the number of threads to use (this may affect run times and can vary depending on the resources available). Other parameters can be specified here, as needed. The first argument, `Mmus_GRCm38_75`, is the name of the index (built in advance), and the second argument is a list of all FASTQ files with reads for the sample. Note that the FASTQ files are concatenated with commas, without spaces. For experiments with paired-end reads, pairs of FASTQ files are given as separate arguments and the order in both arguments must match.

Tophat2 will generate, for each sample, a bunch of files including `accepted_hits.bam`, `deletions.bed`, `insertions.bed`, `junctions.bed`, `prep_reads.info`, `unmapped.bam`, and a log folder. The most important BAM file is the `accepted_hits.bam`, which is a list of read alignments to be used for creating SAM files in the next step. The actual read-counting step by HTSeq is based on SAM files.

2.10 Organize, sort and index the BAM files and create SAM files

File format: SAM
File size in total: 51.4 GB (5 – 7 GB / sample, for nine samples), the most disk-consuming step
Time for creating SAM files: 2.5 hours (15 min / sample)

In this step, we organize, sort and index the BAM files (the `accepted_hits.bam` files generated from Tophat2 in the last step), and create SAM files using samtools. We can sort by name (`-n` option) or by chromosome position (`-p` option). The former is required by using HTSeq to count the reads in the subsequent step; the latter is required to visualize in the Integrative Genomics Viewer (IGV). We only sort by name.

```
for (i in seq_len(nrow(samples))) {
    lib = samples$LibraryName[i]
    ob = file.path(path.d, "FASTA", lib, "accepted_hits.bam")
    # sort by name (_sn), convert to SAM for htseq-count
    cat(paste0("samtools sort -n ", ob, " ", lib, "_sn"), "\n")
    cat(paste0("samtools view -o ", lib, "_sn.sam ", lib, "_sn.bam"), "\n")
    # sort by position (_sp) and index for IGV (optional)
    if (FALSE) {
        cat(paste0("samtools sort ", ob, " ", lib, "_sp", "\n"))
        cat(paste0("samtools index ", lib, "_sp.bam"), "\n\n")
    }
}
```

In the end, we will have many sorted and indexed BAM files (via `samtools sort`) having names as `SampleName_sn.bam`, and SAM files (via `samtools view`) having names as `SampleName_sn.sam`, one for each sample in the same directory.

2.11 (Optional) Inspect alignments with IGV

Once we obtain the BAM files for each sample from the samtools output, we can start IGV, select the correct genome and load the BAM files (file names with "_sp") and the GTF file. Zoom in on an expressed transcript until individual reads are shown and check whether the reads align at and across exon-exon junctions, as expected given the annotation. If any positive and negative controls are known for the system under study (e.g., known differential expression), direct the IGV browser to these regions to confirm that the relative read density is different according to expectation. This step is exploratory and useful for isoform-level inference.

2.12 Count reads using HTSeq

File format: count
File size in total: 7.6 MB
Time for getting counts: 5.8 hours

This is the final step of generating count reads, using the Python package HTSeq and command `htseq-count`. Again, we use R to construct the commands to call htseq-count. The outputs are .count files for each of the nine samples.

```
samples$countf = paste(samples$LibraryName, "count", sep = ".")
gf = file.path(path.d, "GTF", "Mus_musculus.GRCm38.75.gtf")
cmd = paste0("htseq-count -s no -a 10 ", samples$LibraryName, "_sn.sam ", gf,
    " > ", file.path(path.o, samples$countf))
cmd
# [1] 'htseq-count -s no -a 10 Control_Salker_A1_sn.sam
# Mus_musculus.GRCm38.75.gtf > Control_Salker_A1.count'

# ...

# [9] 'htseq-count -s no -a 10 AECM_Salker_C3_sn.sam
# Mus_musculus.GRCm38.75.gtf > AECM_Salker_C3.count'
for (i in length(cmd)) {
    cat(cmd[i], "\n\n")
    system(cmd[i])
}
```

We specify some of the options in the HTSeq Python script: `-s` specifies that the data are not from a stranded protocol (vary by experiment); `-a` specifies a minimum score for the alignment quality. It also prints out progress for every 100,000 GFF lines processed, and prints out progress for every 100,000 SAM alignment records processed.

2.13 Get read counts table using edgeR

The HTSeq outputs nine `.count` files (one for each sample) that can be readily summarized into a big count table using edgeR function `readDGE`.

```
setwd(path.o)
library("edgeR")

## Loading required package:  limma

samples$countf = paste(samples$LibraryName, "count", sep = ".")
counts = readDGE(samples$countf)$counts
colnames(counts) = samples$shortname
head(counts, 3)

##                      CT.SI.1 CT.SI.2 CT.SI.3 DC.SI.4 DC.SI.5 DC.SI.6 AE.SI.7
## ENSMUSG00000000003       0       0       0       0       0       0       0
## ENSMUSG00000000028      89      98      42     175     176     162      93
## ENSMUSG00000000031      65      55     157     129     159     189      75
##                      AE.SI.8 AE.SI.9
## ENSMUSG00000000003       0       0
## ENSMUSG00000000028      99      41
## ENSMUSG00000000031     231      45

tail(counts, 6)

##                          CT.SI.1 CT.SI.2 CT.SI.3 DC.SI.4 DC.SI.5 DC.SI.6
## ENSMUSG00000099334             0       0       0       0       0       0
## __no_feature              931159 1296743 1228456 1485132 1456172 1173435
## __ambiguous               326174  447348  415363  585671  552385  435119
## __too_low_aQual                0       0       0       0       0       0
## __not_aligned                  0       0       0       0       0       0
## __alignment_not_unique 1990750 2774226 2568275 3913371 3405479 3081085
##                          AE.SI.7 AE.SI.8 AE.SI.9
## ENSMUSG00000099334             0       0       0
## __no_feature             1011263 1149689 1426114
## __ambiguous               475543  403870  516604
## __too_low_aQual                0       0       0
## __not_aligned                  0       0       0
## __alignment_not_unique 2905125 2723099 2657654

noint = rownames(counts) %in% c("no_feature", "ambiguous", "too_low_aQual",
    "not_aligned", "alignment_not_unique")
cpms = cpm(counts)  # Returns counts per million (cpm) from a DGEList or matrix object
# remove features without at least 1 read per million in n of the samples,
# where n is the size of the smallest group of replicates
keep = rowSums(cpms > 1) >= 3 & !noint
counts = counts[keep, ]
str(counts)

##  num [1:14142, 1:9] 89 65 59 375 348 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:14142] "ENSMUSG00000000028" "ENSMUSG00000000031" "ENSMUSG00000000037" "ENSMUSG0000000C
##   ..$ : chr [1:9] "CT.SI.1" "CT.SI.2" "CT.SI.3" "DC.SI.4" ...

head(counts, 3)

##                      CT.SI.1 CT.SI.2 CT.SI.3 DC.SI.4 DC.SI.5 DC.SI.6 AE.SI.7
## ENSMUSG00000000028      89      98      42     175     176     162      93
## ENSMUSG00000000031      65      55     157     129     159     189      75
## ENSMUSG00000000037      59      48      26      59      69      72      47
##                      AE.SI.8 AE.SI.9
## ENSMUSG00000000028      99      41
## ENSMUSG00000000031     231      45
## ENSMUSG00000000037      73      24
```

3 Notes

In this pipeline, we obtain “raw” data from SRA, convert to the standard FASTQ format via SRA Toolkit, download reference FASTA file and build the reference index via Bowtie2, align the reads to reference genome via Tophat2, sort/index the alignment BAM files and create SAM files via samtools, count reads via HTSeq, and combine counts via edgeR. There are numerous options in the Tuxedo Suite, samtools and

HTSeq, and the choice largely depends on the scope of research question (gene/exon/isoform) and the question itself. For example, I learned that some options can be specified in Tophat2 which will optimize the task of identifying fusion transcripts, but such options are sub-optimal for studying alternative splicing events. Please refer to relevant online manuals.

In the last step of getting read counts using edgeR, we see extra information about the summarization of read-counting, e.g. the number of ambiguious reads, non-unique alignments, etc. Such information should not be ignored, especially when we are comparing different experiments.

The use of `htseq-count` in this pipeline generates counts at the gene-level (as evidenced in the identifier "ENSMUSG"). For exon-level inference (where we hope to get for each row an exon instead of a gene), the Bioconductor package DEXSeq can be used, where two Python scripts are used for pre-processing the raw data: `dexseq_prepare_annotation.py` and `dexseq_count.py`. See Anders et al. [1].

The computational burden and disk space requirement are relatively high for laptops, but should be moderate on clusters. The most time-consuming step is the alignment via Tophat2 (29.3 hours for nine samples). FASTQ and BAM/SAM files are most disk-consuming (over 100 GB), and with other miscellaneous files, a disk should have a minimum of 150 GB available for this experiment (if we don't delete intermediate files). A CGRB (group) account may be needed in the future to speed up computations for analyses of multiple datasets.

References

[1] Simon Anders, Alejandro Reyes, and Wolfgang Huber. Detecting differential usage of exons from rna-seq data. *Genome Research*, 22(10):2008–2017, 2012.

[2] Simon Anders, Davis J McCarthy, Yunshun Chen, Michal Okoniewski, Gordon K Smyth, Wolfgang Huber, and Mark D Robinson. Count-based differential expression analysis of rna sequencing data using r and bioconductor. *Nature Protocols*, 8(9):1765–1786, 2013.