RNA-Seq Data Analysis: From Raw Data Quality Control to Differential Expression Analysis

Weihong Qi, Ralph Schlapbach, and Hubert Rehrauer

Abstract

As a revolutionary technology for life sciences, RNA-seq has many applications and the computation pipeline has also many variations. Here, we describe a protocol to perform RNA-seq data analysis where the aim is to identify differentially expressed genes in comparisons of two conditions. The protocol follows the recently published RNA-seq data analysis best practice and applies quality checkpoints throughout the analysis to ensure reliable data interpretation. It is written to help new RNA-seq users to understand the basic steps necessary to analyze an RNA-seq dataset properly. An extension of the protocol has been implemented as automated workflows in the R package ezRun, available also in the data analysis framework SUSHI, for reliable, repeatable, and easily interpretable analysis results.

Key words RNA-seq, Quality control, Read alignment, Gene expression quantification, Differential gene expression

1 Introduction

RNA-sequencing (RNA-seq) combines simultaneous transcript identification and quantification of a large number of genes in a single assay. It has many different applications in life sciences, ranging from identification of differentially expressed genes and transcripts, analysis of alternative splicing and polyadenylation, to detection of fusion genes and post-transcriptional events [1]. Consequently, the actual RNA-seq data analysis also has many variations, depending on the applications and studied organisms. In this chapter, we describe the data analysis steps in a typical RNA-seq experiment where the aim is to identify differentially expressed genes, including read alignment, quantification of gene expression levels, and differential gene expression analysis. Meanwhile, three quality control (QC) steps, read QC and pre-processing, alignment QC, and count QC, are also described. As pointed out in the recently published RNA-seq data analysis best practice [2], these checkpoints are essential for producing reliable analysis results.

Read QC tools analyze sequence quality and GC content of the sequencing reads, and check the presence of adaptors and other contaminants. Based on the results, reads can be processed accordingly to trim adaptors, remove low quality bases, and discard short and/or low quality reads. Alignment QC monitors key measures of aligned data quality, such as mapping rate, mapping quality, mapped strand, transcript coverage, and so on. Count QC measures the quality of counts by checking count distributions according to GC content, gene length, and/or annotated gene features, as well as by analyzing reproducibility among biological replicates. These checkpoints are essential for researchers to make informed decisions about data interpretation and analysis strategies [3]. For example, quality-controlled reads can improve the alignment results by removing sequencing artifacts and errors that may contribute to incorrect interpretation of data [4]. The mapped strand inferred from alignment QC is an important parameter for accurate counting of the reads [5]. The length and/or GC bias of counts, or batch effects, if detected during the count QC, can be corrected by applying corresponding correcting methods [6]. This protocol is written with the aim to assist new RNA-seq users to understand the basic steps necessary to analyze an RNA-seq dataset properly and can be applied to the analysis of differential gene expression in comparisons of two conditions. An extension of the protocol has been implemented as automated workflows in the R package ezRun (https://github.com/uzh/ezRun), available also through the data analysis framework SUSHI [7], with the aim to provide better workflow documentation, as well as reliable, repeatable and easily interpretable analysis results.

2 Materials

1. RNA-seq Dataset: The *Arabidopsis thaliana* dataset generated from sorted cells within the medial domain of the gynoecium (GSE74458) [8] are used. For simplicity, three biological replicates without technical replicates are selected for each of the two conditions (Table 1).
2. Reference Genome: *Arabidopsis thaliana* genome sequence and updated annotation data (https://www.araport.org/downloads) are used as the reference.
3. Computing Software: Analysis of RNA-seq data consists of multiple steps and is typically done with the help of open source tools (https://omictools.com/rna-seq-category), with each tool performing a single step. The collection of software tools used in this protocol is listed in Table 2.
4. Computing Hardware: Almost all open source software packages for RNA-seq data analysis are developed for Linux

Table 1
RNA-seq read dataset selected from the RNA-seq analysis of *A. thaliana* medial domain of the gynoecium (GSE74458) [8]

Sample name	Run	BioSample	Condition	Biological replicate	Source name	Tissue cell type
GSM1921011	SRR2850572	SAMN04221482	YFP-NEG	BioRep1	Cells from inflorescence_YFP-NEG	Cells from inflorescence
GSM1921013	SRR2850574	SAMN04221484	YFP-NEG	BioRep2	Cells from inflorescence_YFP-NEG	Cells from inflorescence
GSM1921015	SRR2850576	SAMN04221486	YFP-NEG	BioRep3	Cells from inflorescence_YFP-NEG	Cells from inflorescence
GSM1921019	SRR2850580	SAMN04221490	YFP-POS	BioRep1	YFP-POS_Medial domain, Arabidopsis gynoecium	Medial domain, Arabidopsis gynoecium
GSM1921021	SRR2850582	SAMN04221492	YFP-POS	BioRep2	YFP-POS_Medial domain, Arabidopsis gynoecium	Medial domain, Arabidopsis gynoecium
GSM1921023	SRR2850584	SAMN04221494	YFP-POS	BioRep3	YFP-POS_Medial domain, Arabidopsis gynoecium	Medial domain, Arabidopsis gynoecium

Table 2
Open source software packages used in this protocol

Name	Hyperlink to the project home
UCSC utility scripts	http://hgdownload.soe.ucsc.edu/admin/exe/linux.x86_64/
NCBI SRA Toolkit	https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=software
FastQC	http://www.bioinformatics.babraham.ac.uk/projects/fastqc/
Trimmomatic	http://www.usadellab.org/cms/?page=trimmomatic
STAR	https://github.com/alexdobin/STAR
SAMtools	http://samtools.sourceforge.net/
RSeQC	http://rseqc.sourceforge.net/
featureCounts	http://bioinf.wehi.edu.au/featureCounts/
R	https://www.r-project.org/
ezRun	https://github.com/uzh/ezRun
SUSHI	https://github.com/uzh/sushi

Follow the hyperlinks for details about the packages, such as installation and usage

or Unix-based operating systems. Among the multi-step RNA-seq data analysis workflow, the most computing extensive task is alignment of reads, which determine the suitable computing environment. To speed up the analysis, multi-threading is usually supported by these software tools. Consequently, a multi-core Linux/Unix computer server/cluster is needed. Within this protocol, multi-threading is set with 8, the total disk space needed (the raw data, intermediate files, and final results) is about 100 GB. The memory usage varies and its peak is at around 5 GB.

3 Methods

The RNA-seq data analysis protocol described here consists of three major steps: mapping of reads to the reference genome, counting mapped reads for annotated genes, and finding differentially expressed genes. At each step, there is also a quality control (QC) checkpoint: read QC and preprocessing, alignment QC, and count QC, respectively. Before describing each step in details, we start with setting up the computing environment:

3.1 Preparation of the Workspace and Data Files

1. Set up the working space (*see* **Note 1**). Create an empty directory, which will be the working directory. Then create two subdirectories within the working directory, as containers for the reference genome and raw reads, respectively:

```
mkdir tutorial
cd tutorial
mkdir reference
mkdir reads
```

2. Download and unpack the reference genome and annotation data:

```
cd reference
wget https://www.araport.org/downloads/TAIR10_genome_release/
assembly/TAIR10_Chr.all.fasta.gz
wget https://www.araport.org/download_file/Araport11_latest/
annotation/Araport11_GFF3_genes_transposons.201606.gff.gz
wget https://www.araport.org/download_file/Araport11_latest/
annotation/Araport11_GFF3_genes_transposons.201606.gtf.gz
gunzip -d TAIR10_Chr.all.fasta.gz
gunzip -d Araport11_GFF3_genes_transposons.201606.gff.gz
gunzip -d Araport11_GFF3_genes_transposons.201606.gtf.gz
```

3. Transform the annotation data from gff format to bed format, which will be needed during the alignment QC. We use utility scripts from UCSC (http://hgdownload.soe.ucsc.edu/admin/exe/linux.x86_64/) (*see* **Note 2**) for the file format converting:

```
gff3ToGenePred Araport11_GFF3_genes_transposons.201606.gff
Araport11_GFF3_genes_transposons.201606.Gp
genePredToBed Araport11_GFF3_genes_transposons.201606.Gp Ara-
port11_GFF3_genes_transposons.201606.bed
```

4. Download and prepare the raw reads. The read data listed in Table 1 can be downloaded from NCBI Short Read Archive (SRA) and converted to compressed fastq files using the NCBI SRA Toolkit (https://github.com/ncbi/sra-tools):

```
cd ../reads
prefetch -v SRR2850572
fastq-dump -outdir . -gzip /home/[User]/ncbi/public/sra/
SRR2850572.sra
```

5. Here, as well as in many places throughout the protocol, processing of one sample is shown. The same analysis can be applied to other samples by replacing the run IDs (SRR accession numbers listed in Table 1) in the command lines accordingly. One of the advantages of using ezRUN in SUSHI is that the analysis framework manages the meta-information. Once samples are defined in one dataset (*see* **Note 3**), same analysis can be applied to all samples within the dataset, without extra administrative tasks [7].

3.2 Quality Control and Read Alignment

1. For quality control and pre-processing of the reads, we show the usage of FastQC (http://www.bioinformatics.babraham.ac.uk/projects/fastqc/). Because sequencing adaptor

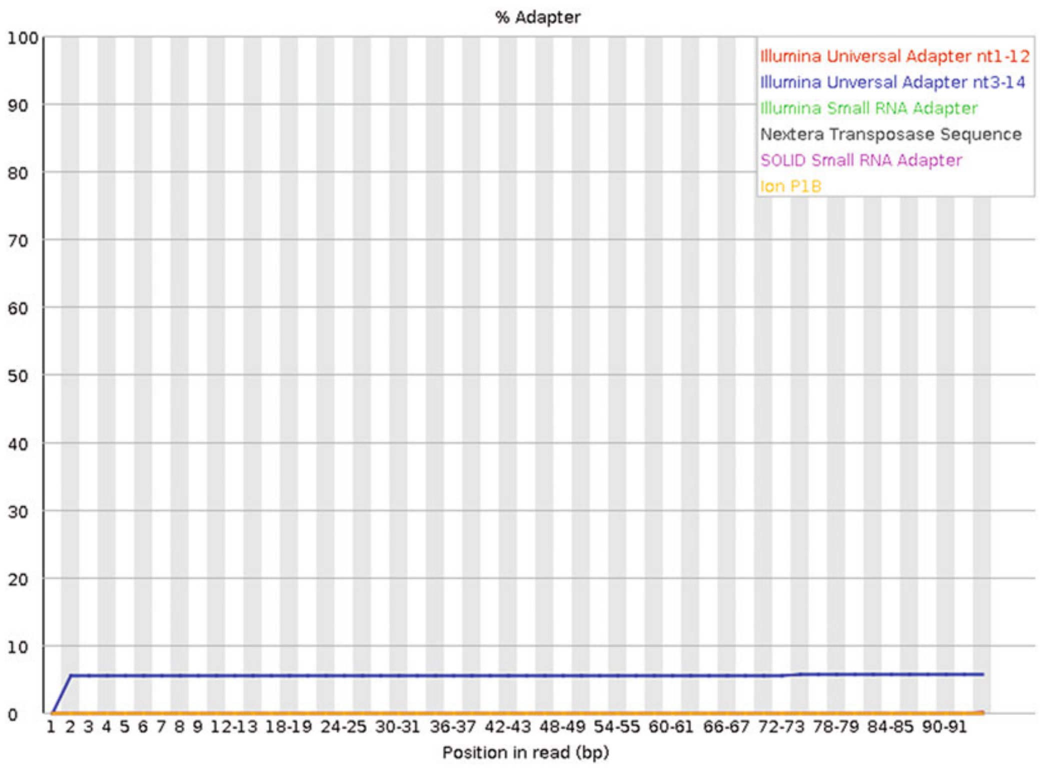

Fig. 1 Adaptor content in sequencing reads

contamination can be found in all six samples (Fig. 1) (*see* **Note 4**), Trimmomatic [9] are applied to generate quality controlled reads (adaptor trimmed, hard trimming of the first 5 bases, minimum length 50, average read quality 20).

```
cd ../
mkdir fastqc
fastqc -t 8 -o fastqc reads/*.fastq.gz
cd reads
trimmomatic-0.36.jar SE -threads 8 -phred33 SRR2850572.fastq.
gz SRR2850572.tr.fastq
ILLUMINACLIP:/usr/local/ngseq/src/Trimmomatic-0.36/adapters/
TruSeq3-SE.fa:2:30:10 HEADCROP:5 MINLEN:50 AVGQUAL:20
```

2. The quality checked reads are aligned to the reference genome using the splice aligner STAR [10]. The alignment result files (bam files) are sorted and indexed using SAMtools [11] for downstream QC and counting analysis.

```
cd ../reference
mkdir STARIndex
STAR –runMode genomeGenerate –genomeDir ./STARIndex –genomeChr-
BinNbits 16 –limitGenomeGenerateRAM 30000000000 –genomeFasta-
Files TAIR10_Chr.all.fasta –sjdbGTFtagExonParentTranscript
```

```
Parent -sjdbGTFfile Araport11_GFF3_genes_transposons.201606.
gff -sjdbOverhang 124 -runThreadN 8
cd ../
mkdir Map_STAR
cd Map_STAR
STAR -genomeDir ../reference/STARIndex/ -sjdbOverhang 124 -read-
FilesIn SRR2850572.tr.fastq -runThreadN 8 -outFilterType BySJout
-outFilterMatchNmin 30 -outFilterMismatchNmax 10 -outFilterMis-
matchNoverLmax 0.05 -alignSJDBoverhangMin 1 -alignSJoverhangMin 8
-alignIntronMax 1000000 -alignMatesGapMax 1000000 -outFilterMul-
timapNmax 50 -chimSegmentMin 15 -chimJunctionOverhangMin 15 -
chimScoreMin 15 -chimScoreSeparation 10 -outSAMstrandField in-
tronMotif -outStd BAM_Unsorted -outSAMtype BAM Unsorted >
SRR2850572.bam
samtools sort -m 3000M -@ 8 SRR2850572.bam > SRR2850572.sorted.bam
samtools index SRR2850572.sorted.bam
rm SRR2850572.bam
```

3.3 Read Alignment Quality Control and Gene Expression Quantification

1. In the following example, we use the mapping QC tool RSeQC [12] to check the mapping rate of the reads, the mapped strand, and the uniformity of read coverage on transcripts (*see* **Note 5**):

```
bam_stat.py -i Map_STAR/SRR2850572.sorted.bam
infer_experiment.py -r reference/Araport11_GFF3_genes_tran-
sposons.201606.bed -i Map_STAR/SRR2850572.sorted.bam
geneBody_coverage.py -r reference/Araport11_GFF3_genes_tran-
sposons.201606.bed -i Map_STAR/ -o BamQC
```

2. The summary statistic showed all reads from SRR2850572 can be mapped, with most being mapped uniquely:

```
Load BAM file... Done
#================================================
#All numbers are READ count
# ===============================================-
=Total records: 13312878
QC failed: 0
Optical/PCR duplicate: 0
Non primary hits 1943411
Unmapped reads: 0
mapq < mapq_cut (non-unique): 1913376

mapq >= mapq_cut (unique): 9456091
Read-1: 0
Read-2: 0
Reads map to '+': 4540183
Reads map to '-': 4915908
Non-splice reads: 6179787
```

```
Splice reads: 3276304
Reads mapped in proper pairs: 0
Proper-paired reads map to different chrom:0
```

3. The mapped strand is "reversely stranded":

```
Reading reference gene model reference/Araport11_GFF3_genes_-
transposons.201606.bed ... Done
Loading SAM/BAM file ... Total 200000 usable reads were sampled
This is SingleEnd Data
Fraction of reads failed to determine: 0.0089
Fraction of reads explained by "++,-": 0.0079
Fraction of reads explained by "+-,-+": 0.9832
```

4. The transcript overage plot (Fig. 2) suggests the transcripts are uniformly covered in all six samples.
5. For gene expression quantification we use featureCounts [13] to count the mapped reads for annotated genes, reversely stranded, allowing both multi-mapping and multi-overlapping reads:

```
cd ../
mkdir Count_featureCounts
featureCounts -T 8 -a reference/Araport11_GFF3_genes_transpo-
sons.201606.gtf -minOverlap 10 -primary -O -M -t exon -g
gene_id -s 2 -o Count_featureCounts/counts.txt Map_STAR/*.bam
```

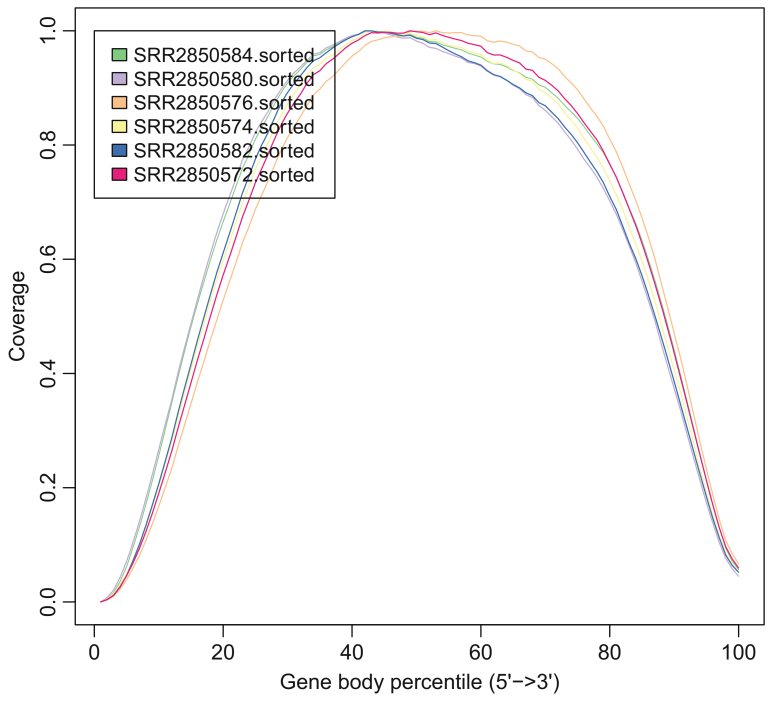

Fig. 2 Gene body coverage in all six samples

3.4 Count QC

We show the example using the R bioconductor package NOISeq [14] to check the length bias and reproducibility among biological replicates (*see* **Note 5**):

1. We first start R in our working directory:

```
R
```

2. After starting R, we install and load NOISeq package:

```
source("https://bioconductor.org/biocLite.R")
biocLite("NOISeq")
library(NOISeq)
```

3. Read in the count table, and convert it to a NOISeq object:

```
mytable<-read.table("Count_featureCounts/counts.txt", skip=1,
header=TRUE)
mycounts<-mytable[, c(7:12)]
row.names(mycounts)<-mytable[, 1]
colnames(mycounts)<-c("YFP-NEG1",  "YFP-NEG2",  "YFP-NEG3",
"YFP-POS1", "YFP-POS2", "YFP-POS3")
myfactors  =  data.frame(condition=c("YFP-NEG",  "YFP-NEG",
"YFP-NEG",  "YFP-POS",  "YFP-POS",  "YFP-POS"),  conditionrun=c
("YFP-NEG-run1",  "YFP-NEG-run1",  "YFP-NEG-run1",  "YFP-POS-
run1",  "YFP-POS-run1",  "YFP-POS-run1"),  run=c("run1",
"run1","run1","run1","run1","run1"))
mydata <- readData(data = mycounts, factors=myfactors, length
= mytable[, c("Geneid", "Length")], chromosome = mytable[, c
("Chr", "Start", "End")])
```

4. Estimate the length bias:

```
mylenbias = dat(mydata, type = "lengthbias")
explo.plot(mylenbias)
```

5. Addressing the reproducibility among biological replicates:

```
mypca=dat(mydata, type="PCA")
explo.plot(mypca)
```

6. As expected, length bias of counts (longer genes have more counts) can be observed in all samples (Fig. 3). Biological replicates of the same condition do cluster together in the Principal Component Analysis (PCA) plot (Fig. 4).

3.5 Differential Expression Analysis

We show the example using the R bioconductor package edgeR [15]:

1. We first start R in our working directory:

```
R
```

2. After starting R, we install and load edgeR package:

```
source("https://bioconductor.org/biocLite.R")
biocLite("edgeR")
library(edgeR)
```

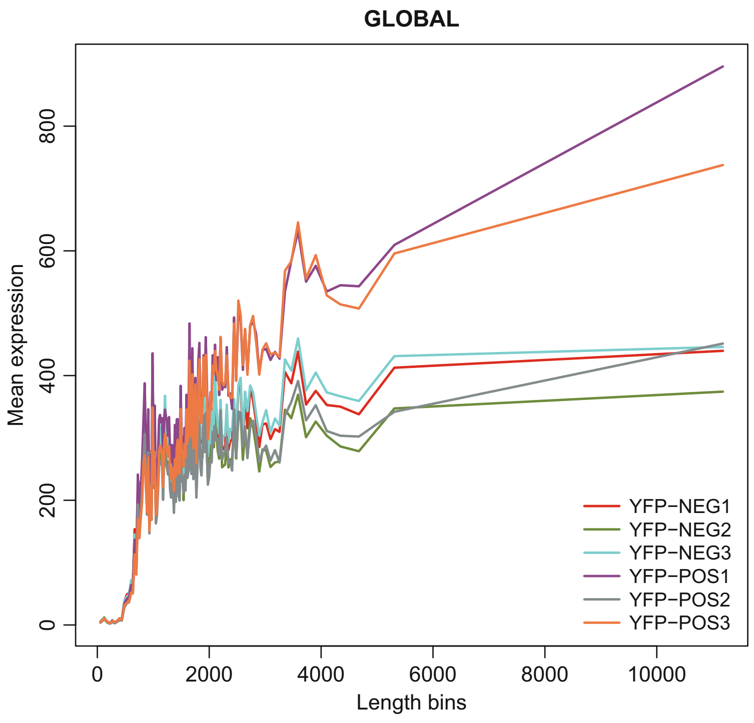

Fig. 3 Gene length versus expression. The genes are divided into bins according to gene length. Each bin contains 200 genes and the middle point of each bin is depicted in *X* axis. For each bin, the 5% trimmed mean of the corresponding expression values is computed and depicted in *Y* axis

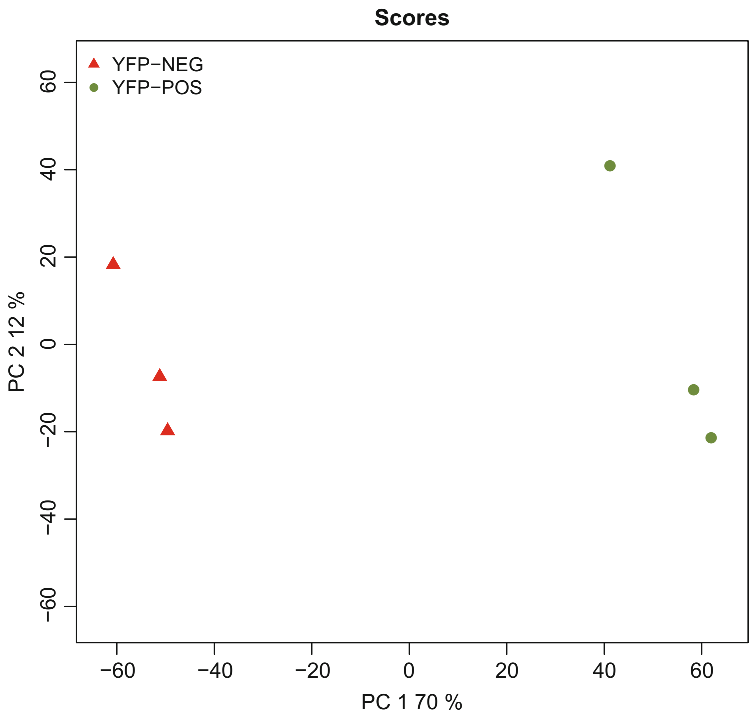

Fig. 4 Principal Component Analysis (PCA) plot colored by conditions

3. Read in the count table, and convert it to an edgeR object:

```
mytable<-read.table("Count_featureCounts/counts.txt", skip=1,
header=TRUE)
mycounts<-mytable[, c(7:12)]
row.names(mycounts)<-mytable[, 1]
colnames(mycounts)<-c("YFP-NEG1", "YFP-NEG2", "YFP-NEG3",
"YFP-POS1", "YFP-POS2", "YFP-POS3")
Condition<-c("YFP-NEG", "YFP-NEG", "YFP-NEG", "YFP-POS",
"YFP-POS", "YFP-POS")
y<-DGEList(counts=mycounts, group=Condition, genes=mytable[,
c("Geneid", "Length")])
```

4. Perform the integrated normalization method trimmed mean of *M* values (TMM) [16]:

```
y<-calcNormFactors(y, method="TMM")
```

5. Estimate common dispersion:

```
y<-estimateCommonDisp(y)
```

6. Estimate tagwise dispersion:

```
y<-estimateTagwiseDisp(y)
```

7. Perform differential expression analysis:

```
et<-exactTest(y, pair=c("YFP-NEG", "YFP-POS"))
res<-as.data.frame(topTags(et, n=37813))
```

8. Generate list of differentially expressed genes:

```
   rese2fold<-res(res$logFC>=1 | res$logFC<=-1, ]
res2foldpadj<-res2fold[res2fold$FDR<=0.01, ]
```

4 Notes

1. This protocol assumes users have basic knowledge about the Unix operation system. There are many books on this topic, as well as free tutorials on the Internet: http://korflab.ucdavis.edu/Unix_and_Perl/
2. The command line examples in this protocol assume the user install the software needed (Table 2) and make all excitable available in his/her PATH environment. Please follow the installation instruction for each software package.
3. A dataset file in SUSHI is a tab-delimited file that records meta-information about the experiment and data [7]. The dataset to initiate the analysis described here using ezRUN in SUSHI can be found here:

   http://fgcz-sushi-demo.uzh.ch/projects/p1000/GSE74458/dataset.tsv

4. Adaptor content analysis is only one of the many analysis modules available in FastQC. Detailed documentation can be found at:

   http://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/

   Here one can access the summary FastQC report on all six samples, generated by ezRUN in SUSHI: http://fgcz-sushi-demo.uzh.ch/projects/p1000/QC_Fastqc_287_2016-10-31–09-29-17/FastQC_Result/00index.html

5. There are many tools available for QC of read alignment results and count data, each with a slightly different set of QC metrics. The feasibility of some QC parameters also depends on the availability of the corresponding annotation data. The ezRun package in SUSHI provides an extensive collection of the QC metrics. For the tutorial dataset, the mapping QC summary report generated by ezRun in SUSHI can be accessed here:

   http://fgcz-sushi-demo.uzh.ch/projects/p1000/QC_RNABamStats_289_2016-10-31–11-41-37/RNA_BAM_Statistics/00index.html

   The link to the count QC summary report generated by ezRun in SUSHI:

   http://fgcz-sushi-demo.uzh.ch/projects/p1000/QC_CountQC_291_2016-10-31–12-21-18/Count_QC/00index.html

References

1. Wang Z, Gerstein M, Snyder M (2009) RNA-Seq: a revolutionary tool for transcriptomics. Nat Rev Genet 10:57–63. doi:10.1038/nrg2484
2. Conesa A, Madrigal P, Tarazona S et al (2016) A survey of best practices for RNA-seq data analysis. Genome Biol 17:13. doi:10.1186/s13059-016-0881-8
3. Rehrauer H, Opitz L, Tan G et al (2013) Blind spots of quantitative RNA-seq: the limits for assessing abundance, differential expression, and isoform switching. BMC Bioinformatics 14:370. doi:10.1186/1471-2105-14-370
4. Li W, Freudenberg J (2014) Mappability and rea d length. Front Genet 5:381. doi:10.3389/fgene.2014.00381
5. Zhao S, Zhang Y, Gordon W et al (2015) Comparison of stranded and non-stranded RNA-seq transcriptome profiling and investigation of gene overlap. BMC Genomics 16 (1):675. doi:10.1186/s12864-015-1876-7
6. Li S, Labaj PP, Zumbo P et al (2014) Detecting and correcting systematic variation in large-scale RNA sequencing data. Nat Biotechnol 32:888–895
7. Hatakeyama M, Opitz L, Russo G et al (2016) SUSHI: an exquisite recipe for fully documented, reproducible and reusable NGS data analysis. BMC Bioinformatics 17:228. doi:10.1186/s12859-016-1104-8
8. Villarino GH, Hu Q, Manrique S et al (2016) Transcriptomic signature of the SHATTERPROOF2 expression domain reveals the meristematic nature of Arabidopsis gynoecial medial domain. Plant Physiol 171:42–61. doi:10.1104/pp.15.01845
9. Bolger AM, Lohse M, Usadel B (2014) Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics 30:2114–2120. doi:10.1093/bioinformatics/btu170
10. Dobin A, Davis CA, Schlesinger F et al (2013) STAR: ultrafast universal RNA-seq aligner.

Bioinformatics 29:15–21. doi:10.1093/bioinformatics/bts635

11. Li H, Handsaker B, Wysoker A et al (2009) The sequence alignment/map format and SAMtools. Bioinformatics 25:2078–2079. doi:10.1093/bioinformatics/btp352
12. Wang L, Wang S, Li W (2012) RSeQC: quality control of RNA-seq experiments. Bioinformatics 28:2184–2185. doi:10.1093/bioinformatics/bts356
13. Liao Y, Smyth GK, Shi W (2014) featureCounts: an efficient general purpose program for assigning sequence reads to genomic features. Bioinformatics 30:923–930. doi:10.1093/bioinformatics/btt656
14. Tarazona S, Furió-Tarí P, Turrà D et al (2015) Data quality aware analysis of differential expression in RNA-seq with NOISeq R/Bioc package. Nucleic Acids Res 43(21):e140. doi:10.1093/nar/gkv711. gkv711
15. Robinson MD, McCarthy DJ, Smyth GK (2010) edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26(1):139–140. doi:10.1093/bioinformatics/btp616
16. Robinson MD, Oshlack A (2010) A scaling normalization method for differential expression analysis of RNA-seq data. Genome Biol 11:R25. doi:10.1186/gb-2010-11-3-r25