`DEG.comparison`: A comparison of methods for DEG analysis of RNA-seq data

Project ID: RNAseq1
Author of Report:
Daniela Cassol (danicassol@gmail.com)
Lichao Li (lli024@ucr.edu)

June 10, 2015

Contents

1 Introduction

This report describes a comparison of 7 methods to detect differentially expressed genes (DEG) in RNA-seq data from cell-type specific RNAs analysis study in *Arabidopsis thaliana* engaged by Jiao Y et. al (Jiao and Meyerowitz, 2010). The first part of this report will be described the workflow systemPipeR package (Girke, 2014). systemPipeR was used to obtain the table of the counts reads and then the workflow created in the package DEG.comparison.

2 systemPipeR

2.1 Environment settings and input data

Typically, the user wants to record here the sources and versions of the reference genome sequence along with the corresponding annotations. In the provided sample data set all data inputs are stored in a data subdirectory and all results will be written to a separate results directory, while the systemPipeRNAseq.Rnw script and the targets file are expected to be located in the parent directory. The R session is expected to run from this parent directory.

The chosen data set SRP003234 contains 16 singel-end (SE) read sets from *Arabidposis thaliana* (Jiao and Meyerowitz, 2010).

2.2 Required packages and resources

The *DEG.comparison* package needs to be loaded to perform the analysis steps shown in this report (Cassol and Li, 2015).

```
> library(DEG.comparison)
```

2.3 Experiment definition provided by targets file

The targets file defines all FASTQ files and sample comparisons of the analysis workflow.

```
> library(systemPipeR)
> targetspath <- system.file("extdata", "targets.txt", package="DEG.comparison")
> targets <- read.delim(targetspath, comment.char = "#")
> targets
                 FileName SampleName Factor     SampleLong Experiment          Date
1  ./data/SRR064149.fastq     AP1.4A  AP1.4  AP1.stage4.R1          1 7-April-2015
2  ./data/SRR064150.fastq     AP1.4B  AP1.4  AP1.stage4.R2          1 7-April-2015
3  ./data/SRR064151.fastq    AP1.67A AP1.67 AP1.stage67.R1          1 7-April-2015
4  ./data/SRR064153.fastq    AP1.67B AP1.67 AP1.stage67.R2          1 7-April-2015
5  ./data/SRR064154.fastq     AP3.4A  AP3.4  AP3.stage4.R1          1 7-April-2015
6  ./data/SRR064155.fastq     AP3.4B  AP3.4  AP3.stage4.R2          1 7-April-2015
7  ./data/SRR064158.fastq    AP3.67A AP3.67 AP3.stage67.R1          1 7-April-2015
8  ./data/SRR064159.fastq    AP3.67B AP3.67 AP3.stage67.R2          1 7-April-2015
9  ./data/SRR064160.fastq     AG.67A  AG.67  AG.stage67.R1          1 7-April-2015
10 ./data/SRR064161.fastq     AG.67B  AG.67  AG.stage67.R2          1 7-April-2015
11 ./data/SRR064162.fastq      AG.4A   AG.4   AG.stage4.R1          1 7-April-2015
```

```
12 ./data/SRR064163.fastq      AG.4B   AG.4   AG.stage4.R2          1 7-April-2015
13 ./data/SRR064164.fastq     FTC.4A  FTC.4 F.TC.stage4.R1          1 7-April-2015
14 ./data/SRR064165.fastq     FTC.4B  FTC.4 F.TC.stage4.R2          1 7-April-2015
15 ./data/SRR064166.fastq     FTL.4A  FTL.4 F.TL.stage4.R1          1 7-April-2015
16 ./data/SRR064167.fastq     FTL.4B  FTL.4 F.TL.stage4.R2          1 7-April-2015
```

```
> cmp <- readComp(file=targetspath, format="matrix", delim="-")
```

2.4 Structure of `param` file and `SYSargs` container

The `param` file defines the parameters of the command-line software. The following shows the format of a sample `param` file provided by this package.

```
> parampath <- system.file("extdata", "tophat.param", package="DEG.comparison")
> read.delim(parampath, comment.char = "#")
```

```
     PairSet         Name                                 Value
1    modules         <NA>                         bowtie2/2.1.0
2    modules         <NA>                         tophat/2.0.8b
3   software         <NA>                                tophat
4      cores           -p                                     4
5      other         <NA> -g 1 --segment-length 25 -i 30 -I 3000
6   outfile1           -o                           <FileName1>
7   outfile1         path                             ./results/
8   outfile1       remove                                  <NA>
9   outfile1       append                               .tophat
10  outfile1 outextension              .tophat/accepted_hits.bam
11 reference         <NA>              ./data/TAIR10_chr_all.fas
12   infile1         <NA>                           <FileName1>
13   infile1         path                                  <NA>
14   infile2         <NA>                           <FileName2>
15   infile2         path                                  <NA>
```

The `systemArgs` function imports the definitions of both the `param` file and the `targets` file, and stores all relevant information as `SYSargs` object. To run the pipeline without command-line software, one can assign `NULL` to `sysma` instead of a `param` file. In addition, one can start the *systemPipeR* workflow with pregenerated BAM files by providing a targets file where the `FileName` column gives the paths to the BAM files and `sysma` is assigned `NULL`.

```
> args <- systemArgs(sysma=parampath, mytargets=targetspath)
> args
```

```
An instance of 'SYSargs' for running 'tophat' on 16 samples
```

Several accessor functions are available that are named after the slot names of the `SYSargs` object class.

```
> names(args)
```

```
 [1] "targetsin"     "targetsout"    "targetsheader" "modules"       "software"
 [6] "cores"         "other"         "reference"     "results"       "infile1"
[11] "infile2"       "outfile1"      "sysargs"       "outpaths"
```

```
> modules(args)
```

```
[1] "bowtie2/2.1.0" "tophat/2.0.8b"
```

```
> cores(args)
```

```
[1] 4
```

```
> outpaths(args)[1]
```

```
                                                                                       AP:
"/tmp/Rtmp3rgMnA/Rbuild6e3d58091fda/DEG.comparison/vignettes/results/SRR064149.fastq.tophat/accepted_hits.
> sysargs(args)[1]

"tophat -p 4 -g 1 --segment-length 25 -i 30 -I 3000 -o /tmp/Rtmp3rgMnA/Rbuild6e3d58091fda/DEG.comparison/v:
```

The content of the param file can be returned as JSON object as follows (requires *rjson* package).

```
> systemArgs(sysma=parampath, mytargets=targetspath, type="json")
[1] "{\"modules\":{\"n1\":\"\",\"v2\":\"bowtie2/2.1.0\",\"n1\":\"\",\"v2\":\"tophat/2.0.8b\"},\"software\"
```

2.5 Read preprocessing

2.5.1 Construct SYSargs object from param and targets files.

```
> trim.param <- system.file("extdata", "trim.param", package="DEG.comparison")
> trim.param <- read.delim(trim.param, comment.char = "#")
> args <- systemArgs(sysma="trim.param", mytargets=targetspath)
```

2.5.2 preprocessReads

The function preprocessReads allows to apply predefined or custom read preprocessing functions to all FASTQ files referenced in a SYSargs container, such as quality filtering or adaptor trimming routines. The paths to the resulting output FASTQ files are stored in the outpaths slot of the SYSargs object. Internally, preprocessReads uses the FastqStreamer function from the *ShortRead* package to stream through large FASTQ files in a memory-efficient manner. The following example performs adaptor trimming with the trimLRPatterns function from the *Biostrings* package. After the trimming step a new targets file is generated (here targets_trim.txt) containing the paths to the trimmed FASTQ files. The new targets file can be used for the next workflow step with an updated SYSargs instance, *e.g.* running the NGS alignments using the trimmed FASTQ files.

```
> preprocessReads(args=args, Fct="trimLRPatterns(Rpattern='GCCCGGGTAA', subject=fq)",
+                  batchsize=100000, overwrite=TRUE, compress=TRUE)
> writeTargetsout(x=args, file="targets_trim.txt")
```

The following example shows how one can design a custom read preprocessing function using utilities provided by the *ShortRead* package, and then run it in batch mode with the preprocessReads function.

```
> filterFct <- function(fq) {
+   filter1 <- nFilter(threshold=1) # Keeps only reads without Ns
+         filter2 <- polynFilter(threshold=20, nuc=c("A","T","G","C")) # Removes low complexity reads
+         filter <- compose(filter1, filter2)
+         fq[filter(fq)]
+ }
> preprocessReads(args=args, Fct="filterFct(fq)", batchsize=100000)
```

2.5.3 FASTQ quality report

The following seeFastq and seeFastqPlot functions generate and plot a series of useful quality statistics for a set of FASTQ files including per cycle quality box plots, base proportions, base-level quality trends, relative k-mer diversity, length and occurrence distribution of reads, number of reads above quality cutoffs and mean quality distribution.

```
> fqlist <- seeFastq(fastq=infile1(args), batchsize=10000, klength=8)
> pdf("./results/fastqReport.pdf", height=18, width=4*length(fqlist))
> seeFastqPlot(fqlist)
> dev.off()
```

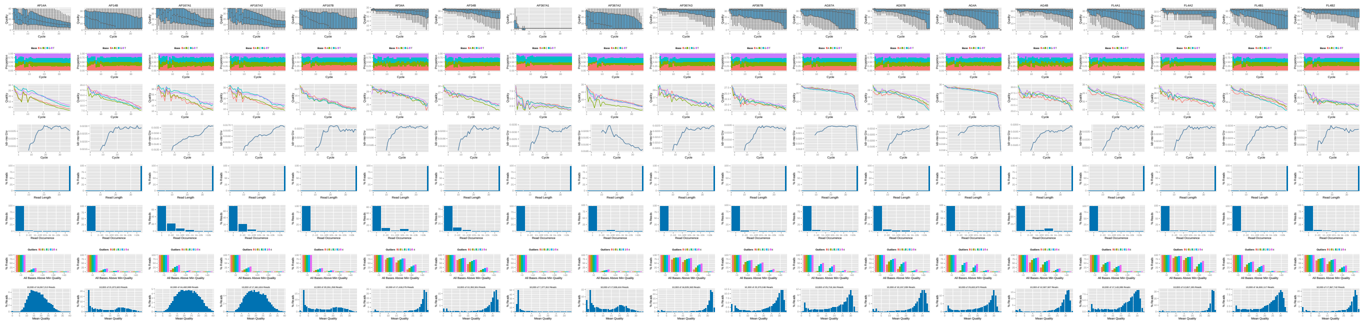

Figure 1: QC report for 19 FASTQ files.

2.6 Alignments

2.6.1 Read mapping with `Bowtie2/Tophat2`

The NGS reads of this project will be aligned against the reference genome sequence using `Bowtie2/TopHat2` (Kim et al., 2013; Langmead and Salzberg, 2012). The parameter settings of the aligner are defined in the `tophat.param` file.

```
> args <- systemArgs(sysma="tophat.param", mytargets="targets.txt")
> sysargs(args)[1] # Command-line parameters for first FASTQ file
```

Submission of alignment jobs to compute cluster, here using 72 CPU cores (19 qsub processes each with 4 CPU cores).

```
> moduleload(modules(args))
> system("bowtie2-build ./data/TAIR10_chr_all.fas ./data/TAIR10_chr_all.fas")
> bampaths <- runCommandline(args=args)
> file.copy(paste0("./.BatchJobs.R"), ".")
> file.copy(paste0("./torque.tmpl"), ".")
> resources <- list(walltime="20:00:00", nodes=paste0("1:ppn=", cores(args)), memory="16gb")
> reg <- clusterRun(args, conffile=".BatchJobs.R", template="torque.tmpl", Njobs=18, runid="01",
+                   resourceList=resources)
> showStatus(reg)
```

Check whether all BAM files have been created

```
> file.exists(outpaths(args))
> sapply(1:length(args), function(x) loadResult(reg, x)) # Works after job completion
```

2.7 Read and alignment stats

The following provides an overview of the number of reads in each sample and how many of them aligned to the reference.

```
> read_statsDF <- alignStats(args=args)
> write.table(read_statsDF, "results/alignStats.xls", row.names=FALSE, quote=FALSE, sep="\t")
```

```
> read.table(system.file("extdata", "alignStats.xls", package="DEG.comparison"), header=TRUE)[1:4,]
```

```
  FileName   Nreads   Nalign Perc_Aligned Nalign_Primary Perc_Aligned_Primary
1    AP14A 16557213  7623900     46.04579        7623900             46.04579
2    AP14B 21872633  3382762     15.46573        3382762             15.46573
3  AP167A1 16482999 10764418     65.30619       10764418             65.30619
4  AP167A2 17461634 10855588     62.16823       10855588             62.16823
```

2.8 Create symbolic links for viewing BAM files in IGV

The symLink2bam function creates symbolic links to view the BAM alignment files in a genome browser such as IGV. The corresponding URLs are written to a file with a path specified under urlfile, here IGVurl.txt.

```
> symLink2bam(sysargs=args, htmldir=c("~/.html/", "cassol_GEN242/"),
+             urlbase="http://biocluster.ucr.edu/~dcassol/", urlfile="IGVurl.txt")
>
```

2.9 Read quantification per annotation range

2.9.1 Read counting with summarizeOverlaps in parallel mode using multiple cores

Reads overlapping with annotation ranges of interest are counted for each sample using the summarizeOverlaps function (Lawrence et al., 2013). The read counting is preformed for exonic gene regions in a non-strand-specific manner while ignoring overlaps among different genes. Subsequently, the expression count values are normalized by *reads per kp per million mapped reads* (RPKM). The raw read count table (countDFeByg.xls) and the correspoding RPKM table (rpkmDFeByg.xls) are written to separate files in the results directory of this project. Parallelization is achieved with the *BiocParallel* package, here using 8 CPU cores.

```
> library("GenomicFeatures"); library(BiocParallel)
> txdb <- makeTranscriptDbFromGFF(file="data/TAIR10_GFF3_genes.gff", format="gff3", dataSource="TAIR", spec
> saveDb(txdb, file="./data/tair10.sqlite")
> txdb <- loadDb("./data/tair10.sqlite")
> eByg <- exonsBy(txdb, by=c("gene"))
> bfl <- BamFileList(outpaths(args), yieldSize=50000, index=character())
> multicoreParam <- MulticoreParam(workers=4); register(multicoreParam); registered()
> # Note: for strand-specific RNA-Seq set 'ignore.strand=FALSE' and for PE data set 'singleEnd=FALSE'
> counteByg <- bplapply(bfl, function(x) summarizeOverlaps(eByg, x, mode="Union",
+                                                          ignore.strand=TRUE,
+                                                          inter.feature=TRUE,
+                                                          singleEnd=TRUE))
> countDFeByg <- sapply(seq(along=counteByg), function(x) assays(counteByg[[x]])$counts)
> rownames(countDFeByg) <- names(rowData(counteByg[[1]])); colnames(countDFeByg) <- names(bfl)
> countDFeByg[1:4,1:12]
> write.table(countDFeByg, "results/countDFeByg.xls", col.names=NA, quote=FALSE, sep="\t")
> rpkmDFeByg <- apply(countDFeByg, 2, function(x) returnRPKM(counts=x, ranges=eByg))
> write.table(rpkmDFeByg, "results/rpkmDFeByg.xls", col.names=NA, quote=FALSE, sep="\t")
> rpkmDFeByg[1:4,1:7]
```

2.10 Sample-wise correlation analysis

The following computes the sample-wise Spearman correlation coefficients from the RPKM normalized expression values. After transformation to a distance matrix, hierarchical clustering is performed with the hclust function and the result is plotted as a dendrogram (sample_tree.pdf).

```
> library(ape)
> rpkmDFeBygpath <- system.file("extdata", "rpkmDFeByg.xls", package="DEG.comparison")
> rpkmDFeByg <- read.delim(rpkmDFeBygpath, row.names=1)
> rpkmDFeByg <- rpkmDFeByg[rowMeans(rpkmDFeByg) > 50,]
> d <- cor(rpkmDFeByg, method="spearman")
> hc <- hclust(as.dist(1-d))
> pdf("results/sample_tree.pdf")
> plot.phylo(as.phylo(hc), type="p", edge.col="blue", edge.width=2, show.node.label=TRUE, no.margin=TRUE)
> dev.off()
```

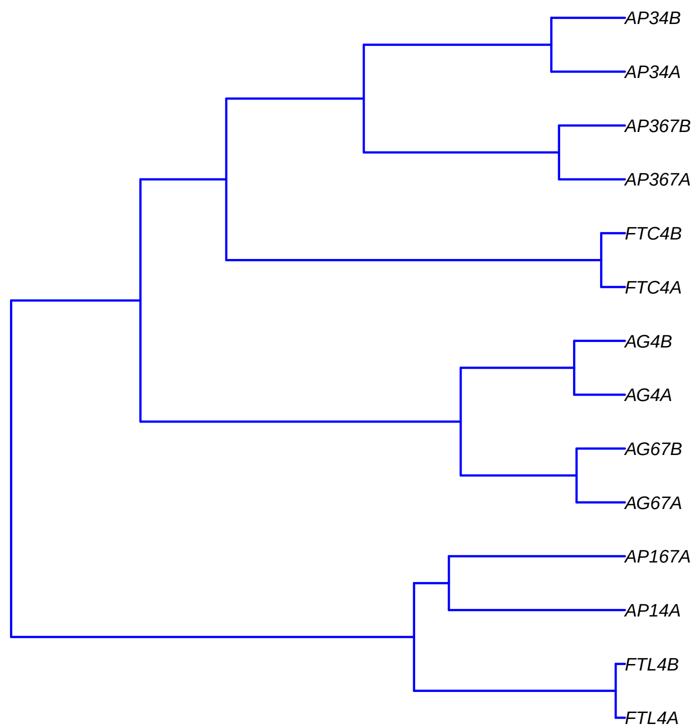

Figure 2: Correlation dendrogram of samples.

3 DEG.comparison

3.1 Analysis of differentially expressed genes

3.1.1 Structure of data and comparisons

The data to input to analysis of differentially expressed genes come from *systemePipeR* results and is the first step in the package `DEG.comparison`.

```
> ##Data input
> countDFeBygpath <- system.file("extdata", "countDFeByg.xls", package="DEG.comparison")
> countDFeByg <- read.delim(countDFeBygpath, row.names=1)
> rpkmDFeBygpath <- system.file("extdata", "rpkmDFeByg.xls", package="DEG.comparison")
> rpkmDFeByg <- read.delim(rpkmDFeBygpath, row.names=1)
```

For all the DEGs methods except RPKM, we construct a list contains all the paired comparison of samples. These paired comparison is based on samples from different stage of same organ or different organ in the same stage in this case. Therefore for the package we need the count table (`countDFeByg`) and list comparisons (`Comp3`).

```
> ##Comparisons
> Comp3 <- list(AP1.4_AP1.67=c("AP1.4A","AP1.4B", "AP1.67A", "AP1.67B"),
+               AP3.4_AP3.67=c("AP3.4A","AP3.4B", "AP3.67A" ,"AP3.67B"),
+               AG.4_AG.67=c("AG.4A", "AG.4B", "AG.67A","AG.67B"),
+               AP1.4_AP3.4=c("AP1.4A","AP1.4B", "AP3.4A","AP3.4B"),
+               AP1.4_AG.4=c("AP1.4A", "AP1.4B", "AG.4A", "AG.4B"),
+               AP3.4_AG.4=c("AP3.4A", "AP3.4B", "AG.4A", "AG.4B"),
+               AP1.67_AP3.67=c("AP1.67A","AP1.67B", "AP3.67A", "AP3.67B"),
+               AP1.67_AG.67=c("AP1.67A","AP1.67B", "AG.67A", "AG.67B"),
+               AP3.67_AG.67=c("AP3.67A", "AP3.67B", "AG.67A", "AG.67B"))
```

3.1.2 DEG1: Simple Fold Change Method - RPKM

RPKM (Reads Per Kilobase per Million mapped reads) is a method of quantifying gene expression by simply normalizing for total read length and the number of sequencing reads (Mortazavi et al., 2008). For running DEGs directly from RPKM counts reads, we need to set two list.

```
> ##Settings
> Comp1 <- list(Factor=(Reduce(union, targets$Factor)), Sample=c(colnames(rpkmDFeByg)),
+               group=c(1,1,2,2,3,3,4,4,5,5,6,6,7,7,8,8))
> Comp2 <- list(AP1.4_AP1.67=c("AP1.4", "AP1.67"), AP3.4_AP3.67=c("AP3.4", "AP3.67"),
+               AG.4_AG.67=c("AG.4", "AG.67"), AP1.4_AP3.4=c("AP1.4", "AP3.4"),
+               AP1.4_AG.4=c("AP1.4", "AG.4"), AP3.4_AG.4=c("AP3.4", "AG.4"),
+               AP1.67_AP3.67=c("AP1.67", "AP3.67"), AP1.67_AG.67=c("AP1.67", "AG.67"),
+               AP3.67_AG.67=c("AP3.67", "AG.67"))
```

In our package, we create `run_RPKM` to compute mean values for replicates samples and log2 values of fold change between paired comparisons, then collect significant genes using `filterDEG_logFC`.

```
> RPKM_FC <- run_RPKM (rpkmDFeByg, Comp1, Comp2)
> write.table(RPKM_FC, "./results/RPKM_FC.xls", quote=FALSE, sep="\t", col.names = NA)
> pdf("./results/DEG_list_RPKM.pdf")
> DEG_list_RPKM <- filterDEG_logFC(degDF=RPKM_FC, filter=c(Fold=2), method="RPKM")
> dev.off()
> DEG_list_RPKM$Summary[1:4,]
```

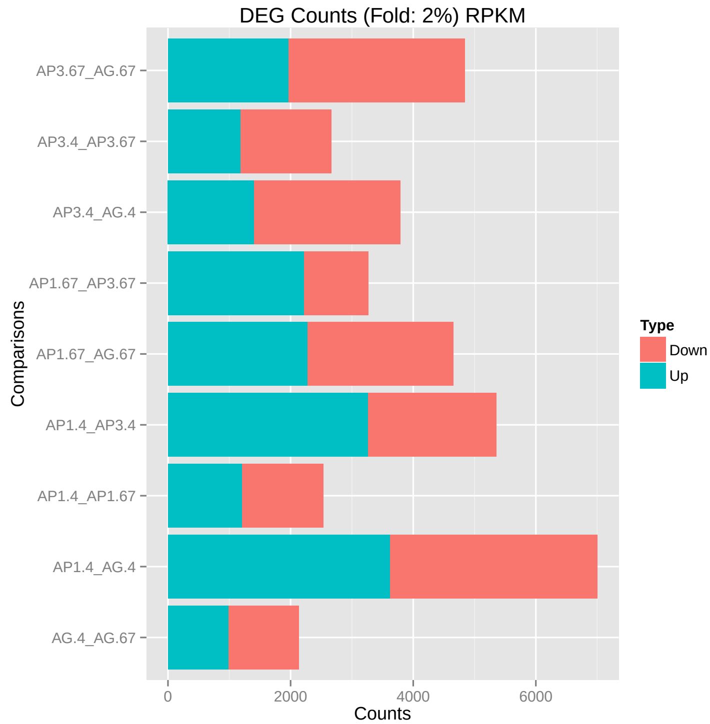

Figure 3: Up and down regulated DEGs.

3.1.3 DEG2: edgeR

edgeR is a DEG methods that implement a range of statistical methodology based on the negative binomial distributions, including empirical Bayes estimation, exact tests, generalized linear models and quasi-likelihood tests (McCarthy et al., 2012). Here we use generalized linear models (glms) method from the *edgeR* package (Robinson et al., 2010), which is suitable for multifactor experiments of any complexity. The function `run_edgeR` is defined in *systemPipeR* package (Girke, 2014). The sample comparisons used by this analysis are defined in the header lines of the `targets` file starting with <CMP>.

```
> edgeDF <- run_edgeR(countDF=countDFeByg, targets=targets, cmp=cmp[[1]], independent=FALSE, mdsplot="")
> write.table(edgeDF, "results/edgeDF.xls", col.names=NA, quote=FALSE, sep="\t")
> pdf("./results/DEG_list_edgeR.pdf")
> DEG_list_edgeR <- filterDEGnew(degDF=edgeDF, filter=c(Fold=2, FDR=1), method="edgeR")
> dev.off()
> DEG_list_edgeR$Summary[1:4,]
```

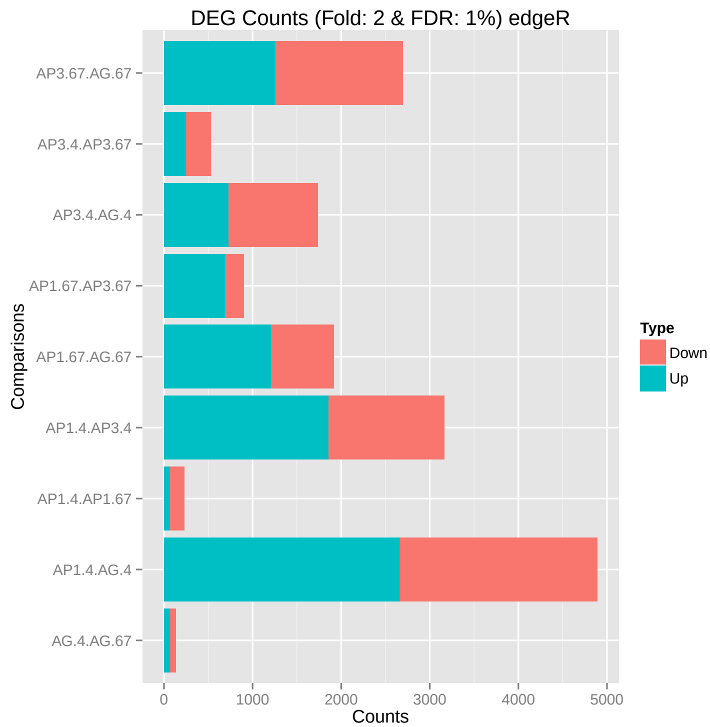

Figure 4: Up and down regulated DEGs.

3.1.4 DEG3: DESeq2

DESeq2 estimate variance-mean dependence in count data and test for differential expression based on a model using the negative binomial distribution (Love et al., 2014). Similar to edgeR, he function `run_DESeq2` is defined in *systemPipeR* package (Girke, 2014).

```
> deseq2DF <- run_DESeq2(countDF=countDFeByg, targets=targets, cmp=cmp[[1]], independent=FALSE)
> write.table(deseq2DF, "results/deseq2DF.xls", col.names=NA, quote=FALSE, sep="\t")
> pdf("./results/DEG_list_DESeq2.pdf")
> DEG_list_DESeq2 <- filterDEGnew(degDF=deseq2DF, filter=c(Fold=2, FDR=1), method="DESeq2")
> dev.off()
> DEG_list_DESeq2$Summary[1:4,]
```

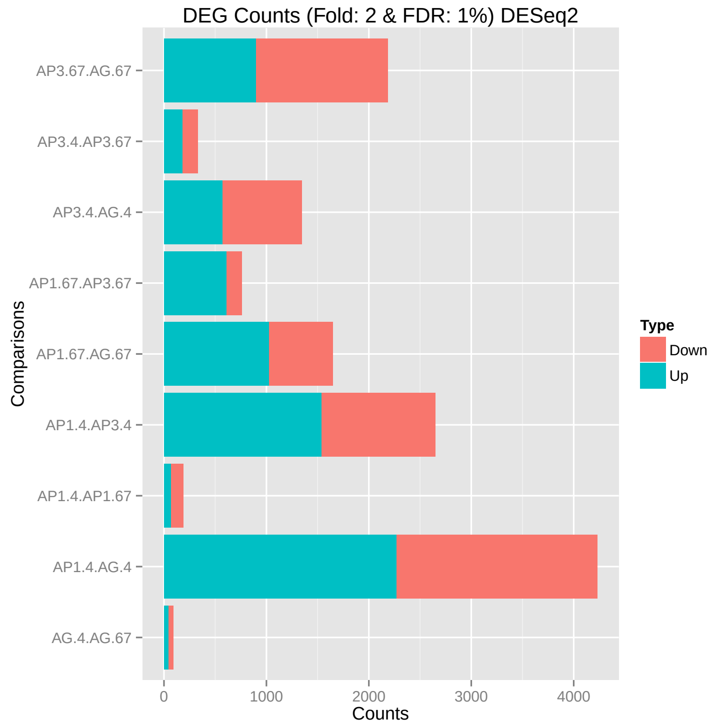

Figure 5: Up and down regulated DEGs.

3.1.5 DEG4: baySeq

baySeq calculating estimated posterior likelihoods of differential expression (or more complex hypotheses) via empirical Bayesian methods (Hardcastle and Kelly, 2010). This approach begins by considering a distribution for the row defined by a set of underlying parameters for which some prior distribution exists. By estimating this prior distribution from the data, we are able to assess the posterior likelihood of the model.

```
> dim(countDFeByg)
> bayseqDF <- run_BaySeq(countDFeByg, Comp3, number=27416)
> write.table(bayseqDF, "results/bayseqDF.xls", col.names=NA, quote=FALSE, sep="\t")
> pdf("./results/DEG_list_bayseqDF.pdf")
> DEG_list_bayseqDF <- filterDEG_FDR(degDF=bayseqDF, filter=c(FDR=1), method="BaySeq")
> dev.off()
> DEG_list_bayseqDF$Summary[1:4,]
```

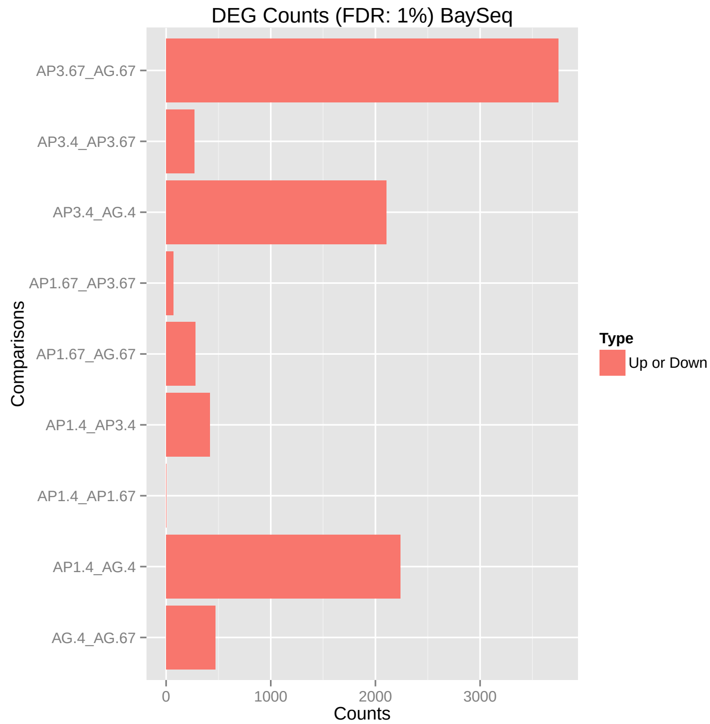

Figure 6: Up and down regulated DEGs.

3.1.6 DEG5: NBPSeq

NBPSeq is a negative binomial (NB) models for two-group comparisons and regression inferences from RNA-Seq data (Di et al., 2013). There are several NBPseq test methods in the package. Here we try classic and generalized linear model corrected NBPseq methods. For these two methods, we create `run_NBPSeq_glm` and `run_NBPSeq_nbp` respectively.

NBPSeq.glm For each row of the input data matrix, nb.glm.test fits an NB log-linear regression model and performs large-sample tests for a one-dimensional regression coefficient.

```
> NBPSeq.glmDF <- run_NBPSeq_glm (countDFeByg, Comp3)
> write.table(NBPSeq.glmDF, "results/NBPSeq_glmDF.xls", col.names=NA, quote=FALSE, sep="\t")
> pdf("./results/DEG_list_NBPSeq_glmDF.pdf")
> DEG_list_NBPSeq.glmDF <- filterDEGnew(degDF=NBPSeq.glmDF, filter=c(Fold=2, FDR=1), method="NBPSeq.glm")
> dev.off()
> DEG_list_NBPSeq.glmDF$Summary[1:4,]
```

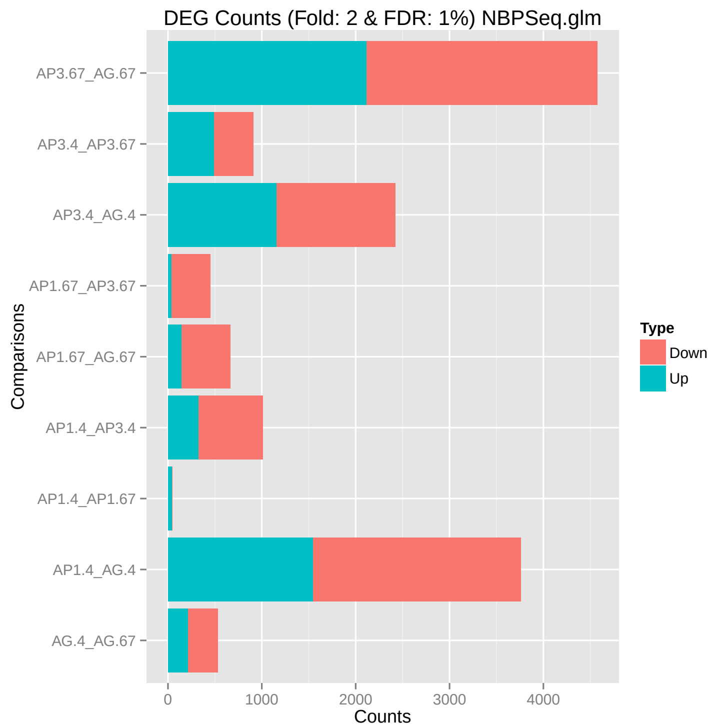

Figure 7: Up and down regulated DEGs.

NBPSeq.nbp.test nbp.test fits an NBP model to the RNA-Seq counts and performs Robinson and Smyth's exact NB test on each gene to assess differential gene expression between two groups (Robinson and Smyth, 2008).

```
> NBPSeq.nbpDF <- run_NBPSeq_nbp (countDFeByg, Comp3)
> write.table(NBPSeq.nbpDF, "results/NBPSeq.nbpDF.xls", col.names=NA, quote=FALSE, sep="\t")
> pdf("./results/DEG_list_NBPSeq.nbpDF.pdf")
> DEG_list_NBPSeq.nbpDF <- filterDEGnew(degDF=NBPSeq.nbpDF, filter=c(Fold=2, FDR=1), method="NBPSeq.nbp")
> dev.off()
> DEG_list_NBPSeq.nbpDF$Summary[1:4,]
```

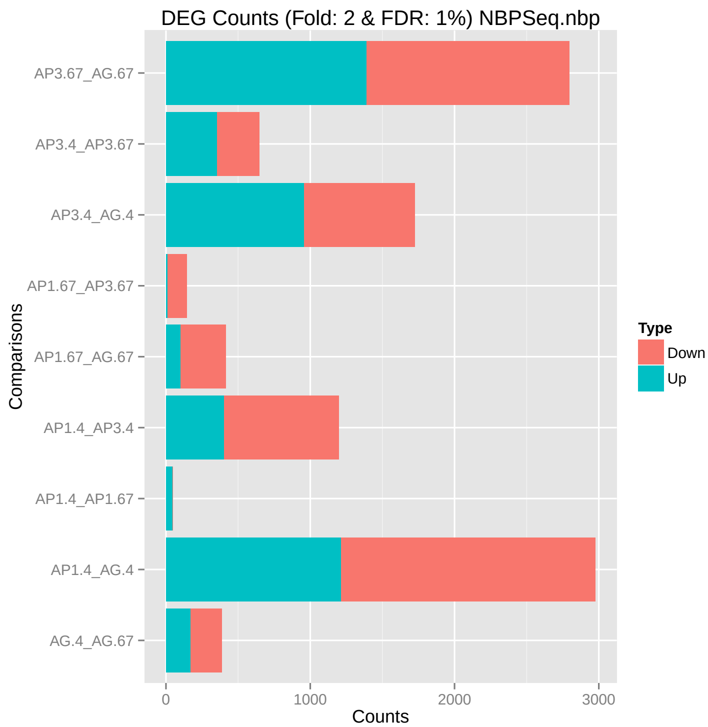

Figure 8: Up and down regulated DEGs.

3.1.7 DEG6: TSPM

TSPM is a simple and powerful statistical approach, based on a two-stage Poisson model that extends the Poisson GLM and provides a powerful and flexible alternative for analyzing RNA-Seq data of moderately small sample sizes Auer and Doerge (2011). We create Rfunctionrun_TSPM for compute all comparison.

```
> TSPMDF <- run_TSPM(countDFeByg, Comp3)
> write.table(TSPMDF, "results/TSPMDF.xls", col.names=NA, quote=FALSE, sep="\t")
> pdf("./results/DEG_list_TSPM.pdf")
> DEG_list_TSPM <- filterDEGnew(degDF=TSPMDF, filter=c(Fold=2, FDR=1), method="TSPM")
> dev.off()
> DEG_list_TSPM$Summary[1:4,]
```

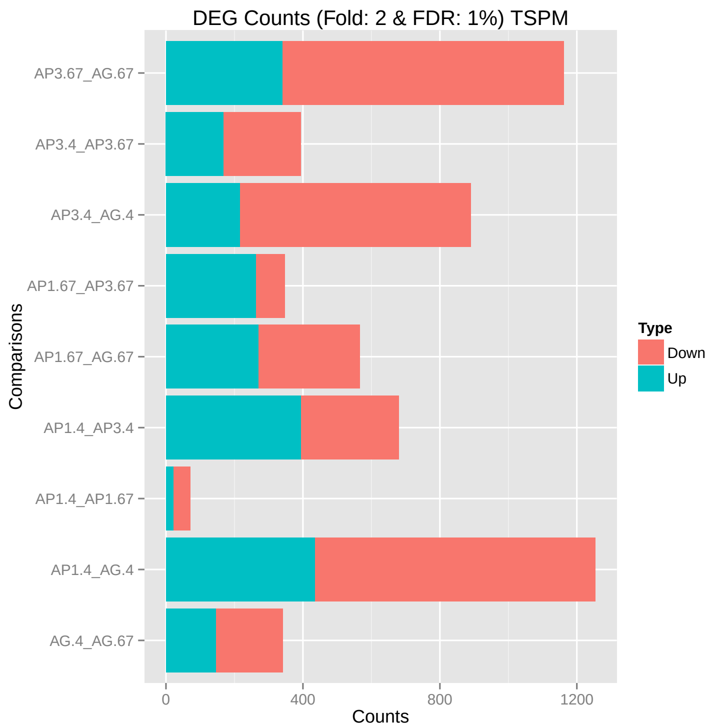

Figure 9: Up and down regulated DEGs.

3.2 Comparisons

In this section, we will introduce four strategies to compare results from various DEGs methods: Venn diagram, pairwise Spearman's correlation coefficient, Area under ROC curve and correlation dendrogram. For Venn diagram, we use predicted up and down regulated DEGs in all paired-comparison samples. Readers can create Venn figure for specific paired-comparison by changing the setlist for plot. For correlation coefficient scatterplot, we compare results between methods for a specific paired-comparison based on logFC and FDR respectively. For ROC curve, we use logFC data in one paired-comparison.

3.2.1 Total number of significantly differentially expressed

```
> totalgenes <- system.file("extdata", "totalgenes.csv", package="DEG.comparison")
> totalgenes <- read.delim (totalgenes, sep=",")
> pdf("./results/TotalGenes.pdf")
> plot <- ggplot(totalgenes, aes(Package, Number.of.significantly.differentially.expressed)) + geom_bar(aes
> print(plot)
> dev.off()
```

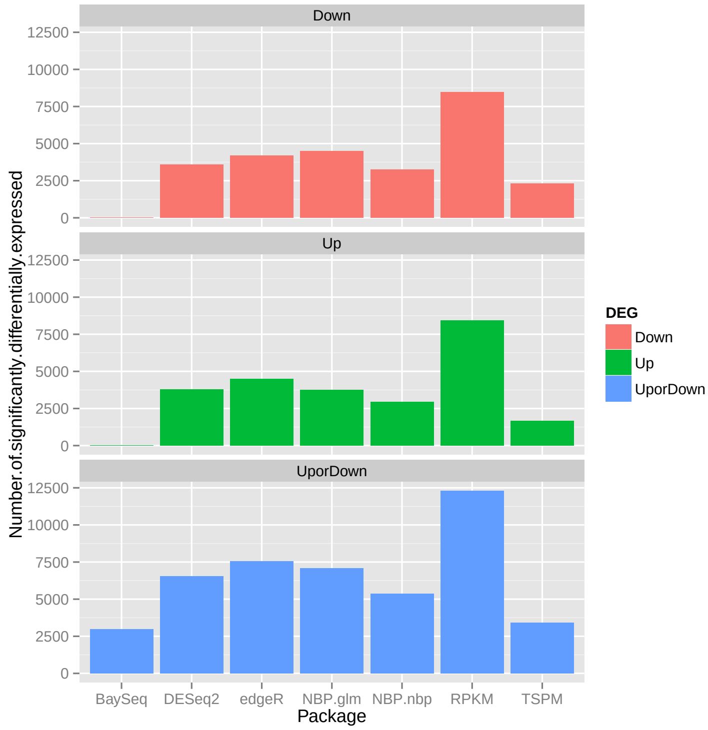

Figure 10: Total number of significantly differentially expressed

3.2.2 List and data.frame with all DEG in all comparisons

Before we construct Venn diagram for methods, we need to merge all up-regulated, down-regulated or total genes founded by seven methods into data frames, respectively.

```
> ###UporDown
> List_RPKM_UporDown <- Reduce(union, (DEG_list_RPKM$UporDown))
> RPKM_UporDown <- data.frame(List_RPKM_UporDown); rownames(RPKM_UporDown) <- RPKM_UporDown[[1]]
> List_edgeR_UporDown <- Reduce(union, (DEG_list_edgeR$UporDown))
> edgeR_UporDown <- data.frame(List_edgeR_UporDown); rownames(edgeR_UporDown) <- edgeR_UporDown[[1]]
> List_DESeq2_UporDown <- Reduce(union, (DEG_list_DESeq2$UporDown))
> DESeq2_UporDown <- data.frame(List_DESeq2_UporDown); rownames(DESeq2_UporDown) <- DESeq2_UporDown[[1]]
> List_bayseqDF_UporDown <- Reduce(union, (DEG_list_bayseqDF$UporDown))
> bayseqDF_UporDown <- data.frame(List_bayseqDF_UporDown); rownames(bayseqDF_UporDown) <- bayseqDF_UporDow
> List_NBPSeq.glmDF_UporDown <- Reduce(union, (DEG_list_NBPSeq.glmDF$UporDown))
> NBPSeq.glmDF_UporDown <- data.frame(List_NBPSeq.glmDF_UporDown); rownames(NBPSeq.glmDF_UporDown) <- NBPSe
> List_NBPSeq.nbpDF_UporDown <- Reduce(union, (DEG_list_NBPSeq.nbpDF$UporDown))
> NBPSeq.nbpDF_UporDown <- data.frame(List_NBPSeq.nbpDF_UporDown); rownames(NBPSeq.nbpDF_UporDown) <- NBPSe
> List_TSPM_UporDown <- Reduce(union, (DEG_list_TSPM$UporDown))
> TSPM_UporDown <- data.frame(List_TSPM_UporDown); rownames(TSPM_UporDown) <- TSPM_UporDown[[1]]
> ###Up
> List_RPKM_Up <- Reduce(union, (DEG_list_RPKM$Up))
> RPKM_Up <- data.frame(List_RPKM_Up); rownames(RPKM_Up) <- RPKM_Up[[1]]
> List_edgeR_Up <- Reduce(union, (DEG_list_edgeR$Up))
> edgeR_Up <- data.frame(List_edgeR_Up); rownames(edgeR_Up) <- edgeR_Up[[1]]
> List_DESeq2_Up <- Reduce(union, (DEG_list_DESeq2$Up))
> DESeq2_Up <- data.frame(List_DESeq2_Up); rownames(DESeq2_Up) <- DESeq2_Up[[1]]
```

```
> List_NBPSeq.glmDF_Up <- Reduce(union, (DEG_list_NBPSeq.glmDF$Up))
> NBPSeq.glmDF_Up <- data.frame(List_NBPSeq.glmDF_Up); rownames(NBPSeq.glmDF_Up) <- NBPSeq.glmDF_Up[[1]]
> List_NBPSeq.nbpDF_Up <- Reduce(union, (DEG_list_NBPSeq.nbpDF$Up))
> NBPSeq.nbpDF_Up <- data.frame(List_NBPSeq.nbpDF_Up); rownames(NBPSeq.nbpDF_Up) <- NBPSeq.nbpDF_Up[[1]]
> List_TSPM_Up <- Reduce(union, (DEG_list_TSPM$Up))
> TSPM_Up <- data.frame(List_TSPM_Up); rownames(TSPM_Up) <- TSPM_Up[[1]]
> ###Down
> List_RPKM_Down <- Reduce(union, (DEG_list_RPKM$Down))
> RPKM_Down <- data.frame(List_RPKM_Down); rownames(RPKM_Down) <- RPKM_Down[[1]]
> List_edgeR_Down <- Reduce(union, (DEG_list_edgeR$Down))
> edgeR_Down <- data.frame(List_edgeR_Down); rownames(edgeR_Down) <- edgeR_Down[[1]]
> List_DESeq2_Down <- Reduce(union, (DEG_list_DESeq2$Down))
> DESeq2_Down <- data.frame(List_DESeq2_Down); rownames(DESeq2_Down) <- DESeq2_Down[[1]]
> List_NBPSeq.glmDF_Down <- Reduce(union, (DEG_list_NBPSeq.glmDF$Down))
> NBPSeq.glmDF_Down <- data.frame(List_NBPSeq.glmDF_Down); rownames(NBPSeq.glmDF_Down) <- NBPSeq.glmDF_Dow
> List_NBPSeq.nbpDF_Down <- Reduce(union, (DEG_list_NBPSeq.nbpDF$Down))
> NBPSeq.nbpDF_Down <- data.frame(List_NBPSeq.nbpDF_Down); rownames(NBPSeq.nbpDF_Down) <- NBPSeq.nbpDF_Dow
> List_TSPM_Down <- Reduce(union, (DEG_list_TSPM$Down))
> TSPM_Down <- data.frame(List_TSPM_Down); rownames(TSPM_Down) <- TSPM_Down[[1]]
>
```

3.2.3 Venn diagram

We union up-regulated or down-regulated data from different methods into setlists,respectively. Then perform Venn diagram for 5 methods without baySeq and RPKM. We exclude baySeq because the methods only show differentially expressed genes without up or down regulated label, and exclude RPKM because of over-sensitivity compare to other methods.

```
> ##Up
> setlist2 <- list(edgeR=rownames(edgeR_Up), DESeq2=rownames(DESeq2_Up),
+                  TSPM=rownames(TSPM_Up), NBPSeq.glm=rownames(NBPSeq.glmDF_Up),
+                  NBPSeq.nbp=rownames(NBPSeq.nbpDF_Up))
> OLlist2 <- overLapper(setlist=setlist2, sep="_", type="vennsets")
> counts2 <- sapply(OLlist2$Venn_List, length)
> pdf("./results/Venn_diagram2.pdf")
> vennPlot(counts=counts2, mymain="DEG Comparison Up")
> dev.off()
```

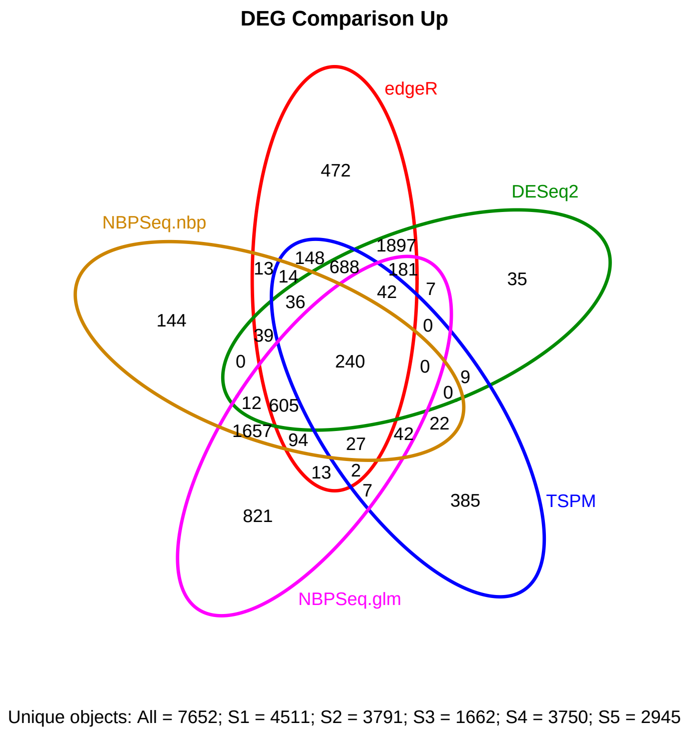

Figure 11: Venn diagram of the overlap in differentially up-regulated genes among five methods (excluding baySeq).

```
> ##Down
> setlist4 <- list(edgeR=rownames(edgeR_Down), DESeq2=rownames(DESeq2_Down),
+                  TSPM=rownames(TSPM_Down), NBPSeq.glm=rownames(NBPSeq.glmDF_Down),
+                  NBPSeq.nbp=rownames(NBPSeq.nbpDF_Down))
> OLlist4 <- overLapper(setlist=setlist4, sep="_", type="vennsets")
> counts4 <- sapply(OLlist4$Venn_List, length)
> pdf("./results/Venn_diagram4.pdf")
> vennPlot(counts=counts4, mymain="DEG Comparison Down")
> dev.off()
```

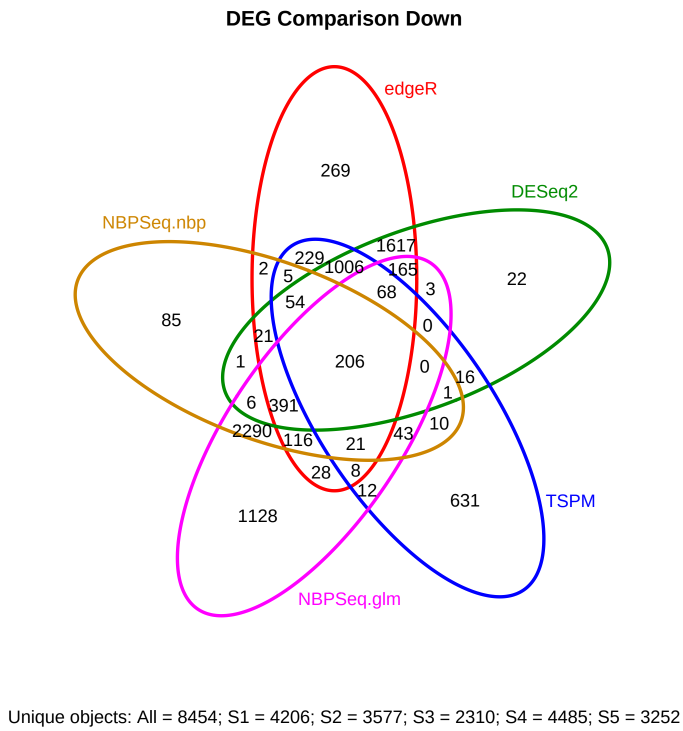

Figure 12: Venn diagram of the overlap in differentially down-regulated genes among five methods (excluding baySeq).

We also construct a Venn plot for DEGs in paired-comparison between organs. The result shows that DEGs are various between organs paired-comparisons even in the same stage.

```
> ###Specific comparison
> hh4<- (DEG_list_RPKM$Up$"AP1.4_AP3.4"); hh5<- (DEG_list_RPKM$Up$"AP1.4_AG.4"); hh6<- (DEG_list_RPKM$Up$"A
> RPKMhh4 <- data.frame(hh4); rownames(RPKMhh4) <- RPKMhh4[[1]]
> RPKMhh5 <- data.frame(hh5); rownames(RPKMhh5) <- RPKMhh5[[1]]
> RPKMhh6 <- data.frame(hh6); rownames(RPKMhh6) <- RPKMhh6[[1]]
> setlist6 <- list("AP1.4-AP3.4"=rownames(RPKMhh4), "AP1.4-AG.4"=rownames(RPKMhh5), "AP3.4-AG.4"=rownames(R
> OLlist6 <- overLapper(setlist=setlist6, sep="_", type="vennsets")
> counts6 <- sapply(OLlist6$Venn_List, length)
> pdf("./results/Venn_diagram_RPKM.pdf")
> vennPlot(counts=counts6, mymain="DEG Comparison RPKM")
> dev.off()
```

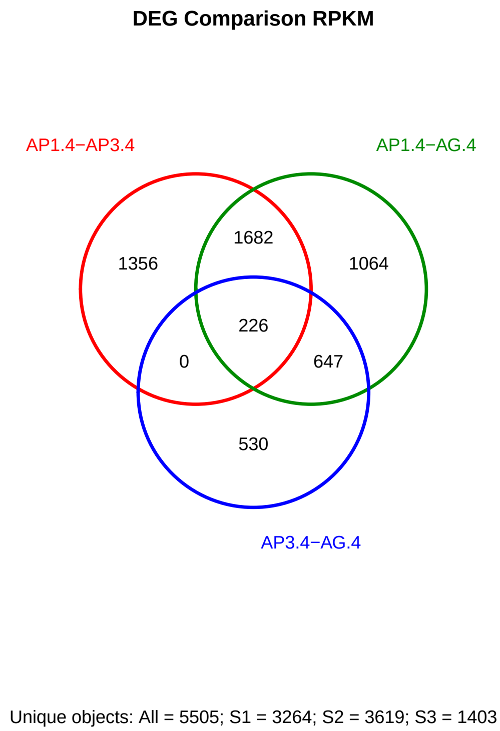

Figure 13: Venn diagram of the overlap in differentially genes among different comparisons in RPKM.

3.2.4 Scatterplot

Spearman's correlation coefficient is a nonparametric measure of statistical dependence between two variables. It assesses how well the relationship between two variables can be described using a monotonic function. Here we build a paired-wise scatterplot based on fold change and FDR of genes in AG-67 compare to AP3-67 that show correlation between methods.

```
> ###Data: scatterDEG
> RPKM.S <- data.frame(RPKM_FC[unlist(List_RPKM_UporDown),])
> bayseq.S <- data.frame(bayseqDF [unlist(List_bayseqDF_UporDown),])
> edgeR.S <- data.frame(edgeDF[unlist(List_edgeR_UporDown),])
> NBPSeq.glm.S <- data.frame(NBPSeq.glmDF[unlist(List_NBPSeq.glmDF_UporDown),])
> deseq2.S <- data.frame(deseq2DF[unlist(List_DESeq2_UporDown),])
> TSPM.S <- data.frame(TSPMDF [unlist(List_TSPM_UporDown),])
> NBPSeq.nbp.S <- data.frame(NBPSeq.nbpDF[unlist(List_NBPSeq.nbpDF_UporDown),])
```

```
> ##logFC
> RPKM.logFC <- data.frame(rownames(RPKM.S), RPKM.S$"AP3.67_AG.67_logFC", row.names=1)
> edgeR.logFC <- data.frame(rownames(edgeR.S), edgeR.S$"AP3.67.AG.67_logFC", row.names=1)
> deseq2.logFC <- data.frame(rownames(deseq2.S), deseq2.S$"AP3.67.AG.67_logFC" , row.names=1)
> NBPSeq.glm.logFC <- data.frame(rownames(NBPSeq.glm.S), NBPSeq.glm.S$"AP3.67_AG.67_logFC" , row.names=1)
> NBPSeq.nbp.logFC <- data.frame(rownames(NBPSeq.nbp.S), NBPSeq.nbp.S$"AP3.67_AG.67_logFC" , row.names=1)
> TSPM.logFC <- data.frame(rownames(TSPM.S), TSPM.S$"AP3.67_AG.67_logFC", row.names=1)
> scatterDEG1 <- merge(edgeR.logFC, deseq2.logFC, by='row.names', all=TRUE); rownames(scatterDEG1) <- scat
> scatterDEG2 <- merge(scatterDEG1, RPKM.logFC , by='row.names', all=TRUE); rownames(scatterDEG2) <- scatte
> scatterDEG3 <- merge(scatterDEG2, NBPSeq.glm.logFC, by='row.names', all=TRUE); rownames(scatterDEG3) <- s
> scatterDEG4 <- merge(scatterDEG3, NBPSeq.nbp.logFC, by='row.names', all=TRUE); rownames(scatterDEG4) <- s
> scatterDEG5 <- merge(scatterDEG4, TSPM.logFC, by='row.names', all=TRUE); rownames(scatterDEG5) <- scatter
```

```
> scatterDEG_logFC <- scatterDEG5
> colnames(scatterDEG_logFC) <- c("edgeR", "DESeq2", "RPKM", "NBPSeq.glm", "NBPSeq.nbp", "TSPM")
> scatterDEG_logFC[is.na(scatterDEG_logFC)] <- 0
> ###Scatterplot
> scatterDEG_logFC[1:4,]
> pdf("./results/Scatterplot_AP367_AG67_logFC.pdf")
> pairs(scatterDEG_logFC, lower.panel=panel.smooth, upper.panel=panel.cor, pch=20, main="Scatterplot Matri
> dev.off()
```

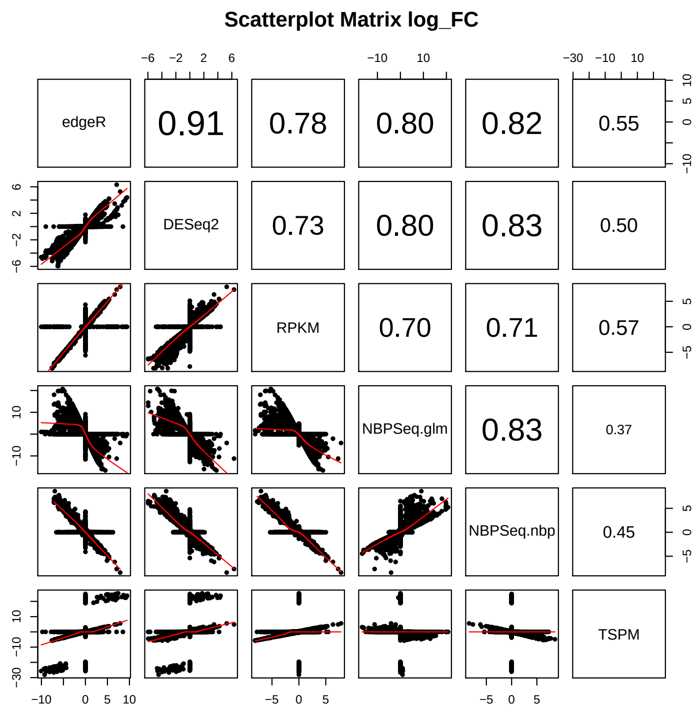

Figure 14: Pair-wise Spearman's correlation coefficients of fold change computed among six methods.

```
> ##FDR
> bayseqFDR <- data.frame(rownames(bayseq.S), bayseq.S$"AP3.67_AG.67_FDR", row.names=1)
> edgeRFDR <- data.frame(rownames(edgeR.S), edgeR.S$"AP3.67.AG.67_FDR", row.names=1)
> deseq2FDR <- data.frame(rownames(deseq2.S), deseq2.S$"AP3.67.AG.67_FDR" , row.names=1)
> NBPSeq.glmFDR <- data.frame(rownames(NBPSeq.glm.S), NBPSeq.glm.S$"AP3.67_AG.67_FDR" , row.names=1)
> NBPSeq.nbpFDR <- data.frame(rownames(NBPSeq.nbp.S), NBPSeq.nbp.S$"AP3.67_AG.67_FDR" , row.names=1)
> TSPMFDR <- data.frame(rownames(TSPM.S), TSPM.S$"AP3.67_AG.67_FDR", row.names=1)
> scatterDEG1 <- merge(edgeRFDR, deseq2FDR, by='row.names', all=TRUE); rownames(scatterDEG1) <- scatterDEG
> scatterDEG2 <- merge(scatterDEG1, bayseqFDR, by='row.names', all=TRUE); rownames(scatterDEG2) <- scatter
> scatterDEG3 <- merge(scatterDEG2, NBPSeq.glmFDR, by='row.names', all=TRUE); rownames(scatterDEG3) <- sca
> scatterDEG4 <- merge(scatterDEG3, NBPSeq.nbpFDR, by='row.names', all=TRUE); rownames(scatterDEG4) <- sca
> scatterDEG5 <- merge(scatterDEG4, TSPMFDR, by='row.names', all=TRUE); rownames(scatterDEG5) <- scatterDE
> scatterDEG_FDR <- scatterDEG5
> colnames(scatterDEG_FDR) <- c("edgeR", "DESeq2", "baySeq", "NBPSeq.glm", "NBPSeq.nbp", "TSPM")
> scatterDEG_FDR[is.na(scatterDEG_FDR)] <- 1
> ###Scatterplot
> scatterDEG_FDR[1:4,]
> pdf("./results/Scatterplot_AP367_AG67_FDR.pdf")
> pairs(scatterDEG_FDR, lower.panel=panel.smooth, upper.panel=panel.cor, pch=20, main="Scatterplot Matrix
```

```
> dev.off()
```

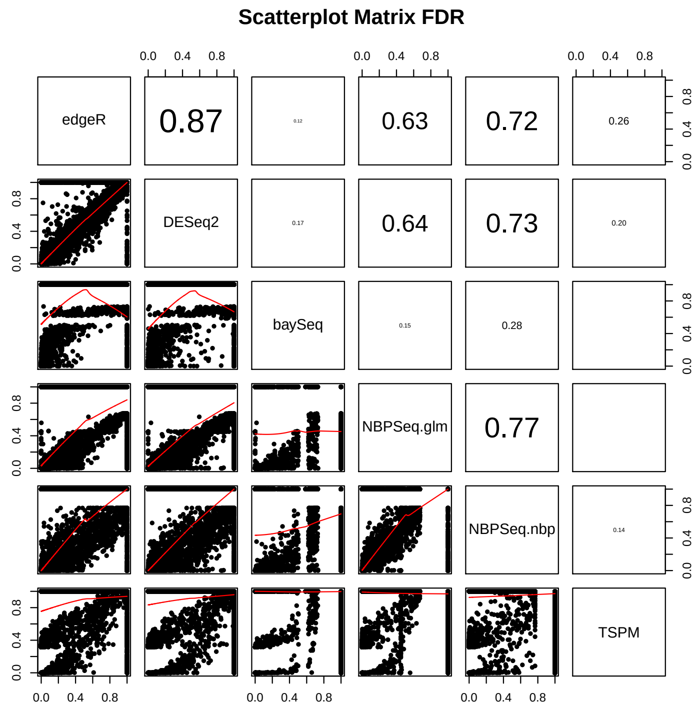

Figure 15: Pair-wise Spearman's correlation coefficients of FRD computed among six methods.

From logFC scatterplot figure, we can say edgeR is highly correlated with DESeq2, a possible reason could be the same negative binominal distribution model that they based on. NBPSeq.glm and classic NBPSeq have similar DEGs results because they both derive from NBPSeq method.

3.2.5 ROC Curve

Receiver operating characteristic (ROC), or ROC curve, is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate against the false positive rate at various threshold settings. For TPR and FPR calculation, we need to give true results compare to predicted results from methods. So we set results from edgeR as the true results.

Firstly, we need to construct a data frame containing fold change data and another data frame containing binary data that assigning significant genes as 1 and non-significant genes as 0.

```
> ##Data
> data.ROC <- scatterDEG5
> scatterDEG1[is.na(scatterDEG1)] <- 1;
> data.ROC <- merge(data.ROC, tmp[1], by='row.names', all=TRUE ); rownames(data.ROC) <- data.ROC[[1]] ; da
> colnames(data.ROC) <- c("edgeR", "DESeq2", "Bayseq", "NBPSeq.glm", "NBPSeq.nbp", "TSPM", "Common")
> write.table(data.ROC, "results/data_ROC.xls", col.names=NA, quote=FALSE, sep="\t")
> data.class <- data.ROC
> data.class[!is.na(data.class)] <- 0
> data.class[is.na(data.class)] <- 1
> write.table(data.class, "results/data.class.xls", col.names=NA, quote=FALSE, sep="\t")
```

```
> data.ROC1 <- data.ROC
> data.ROC1[,1:7][is.na(data.ROC1[,1:7])] <- 1
```

Here we use *ROCR* to draw ROC curves of methods in one figure.

```
> ### ROCR
> pdf("./results/ROC.pdf")
> pred1 <- prediction(data.ROC1$Common, data.class$edgeR)
> perf1 <- performance(pred1, "tpr", "fpr")
> plot(perf1, avg= "threshold", col="black", lty=4,lwd= 2, main= "ROC curves compare with edgeR")
> par(new = TRUE)
> pred2 <- prediction(data.ROC1$Common, data.class$Bayseq)
> perf2 <- performance(pred2, "tpr", "fpr")
> plot(perf2, avg= "threshold", lty=4, lwd= 2, col="dodgerblue4")
> par(new = TRUE)
> pred3 <- prediction(data.ROC1$Common, data.class$NBPSeq.glm)
> perf3 <- performance(pred3, "tpr", "fpr")
> plot(perf3, avg= "threshold",  lty=4,lwd= 2, col="darkgreen")
> par(new = TRUE)
> pred4 <- prediction(data.ROC1$Common, data.class$NBPSeq.nbp)
> perf4 <- performance(pred4, "tpr", "fpr")
> plot(perf4, avg= "threshold",  lty=4,lwd= 2, col="darkviolet")
> par(new = TRUE)
> pred5 <- prediction(data.ROC1$Common, data.class$TSPM)
> perf5 <- performance(pred5, "tpr", "fpr")
> plot(perf5, avg= "threshold", lty=4, lwd= 2, col="firebrick")
> par(new = TRUE)
> pred6 <- prediction(data.ROC1$Common, data.class$DESeq2)
> perf6 <- performance(pred6, "tpr", "fpr")
> plot(perf6, avg= "threshold",  lty=4,lwd= 2, col="deeppink1")
> legend("bottomright", legend=c("edgeR / 1", "BaySeq / 0.749", "NBPSeq.glm / 0.533", "NBPSeq.nbp / 0.872
+        col=c("black", "dodgerblue4", "darkgreen", "darkviolet", "firebrick", "deeppink1"), lty=4, lwd=3,
> dev.off()
```

The area under the ROC curve(AUC-ROC) is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one. Basically, AUC-ROC correspond to the efficiency of a statistical separation method. Larger AUC value means higher accuracy of the method.

```
> ##calculate AUC
> auc.tmp1 <- performance(pred1,"auc"); auc1 <- as.numeric(auc.tmp1@y.values)
> auc.tmp2 <- performance(pred2,"auc"); auc2 <- as.numeric(auc.tmp2@y.values)
> auc.tmp3 <- performance(pred3,"auc"); auc3 <- as.numeric(auc.tmp3@y.values)
> auc.tmp4 <- performance(pred4,"auc"); auc4 <- as.numeric(auc.tmp4@y.values)
> auc.tmp5 <- performance(pred5,"auc"); auc5 <- as.numeric(auc.tmp5@y.values)
> auc.tmp6 <- performance(pred6,"auc"); auc6 <- as.numeric(auc.tmp6@y.values)
>
```

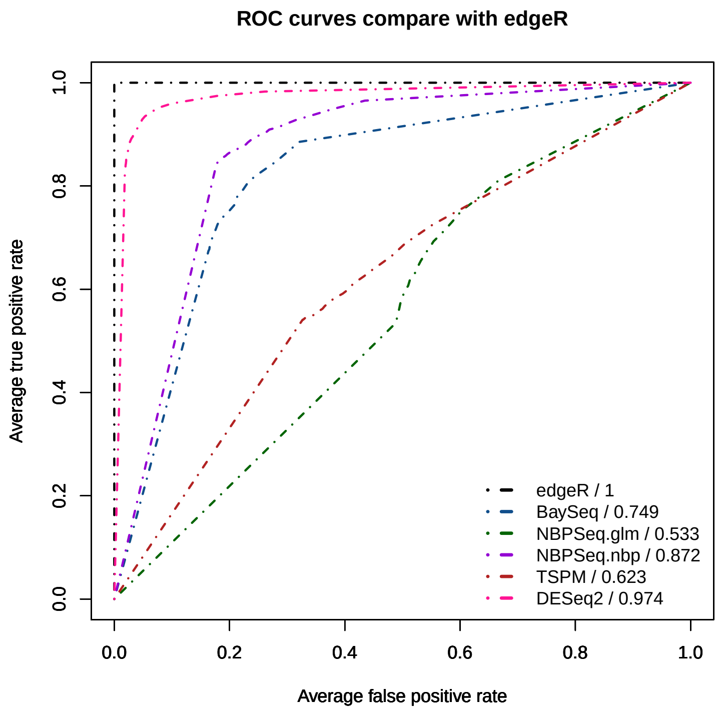

Figure 16: ROC curve

3.2.6 Correlation dendrogram of methods

```
> dFDR <- cor(data.ROC1, method="spearman")
> hc1 <- hclust(as.dist(1-dFDR))
> pdf("./results/methodsCorrelation_FDR.pdf")
> plot.phylo(as.phylo(hc1), type="p", edge.col="blue", edge.width=2, show.node.label=TRUE,  main="Correlati
> dev.off()
```

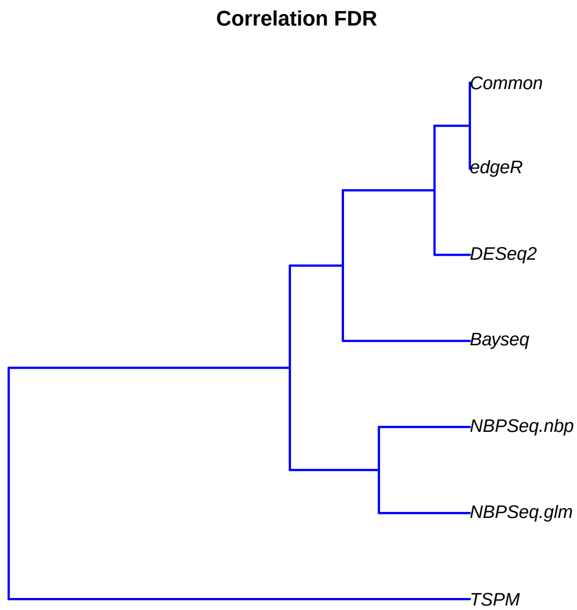

Figure 17: Overall similarity between the methods based on Spearman correlation of gene ranks.

```
> dlogFC <- cor(scatterDEG_logFC, method="spearman")
> hc2 <- hclust(as.dist(1-dlogFC))
> pdf("./results/methodsCorrelation_logFC.pdf")
> plot.phylo(as.phylo(hc2), type="p", edge.col="blue", edge.width=2, show.node.label=TRUE, main="Correlati
> dev.off()
```

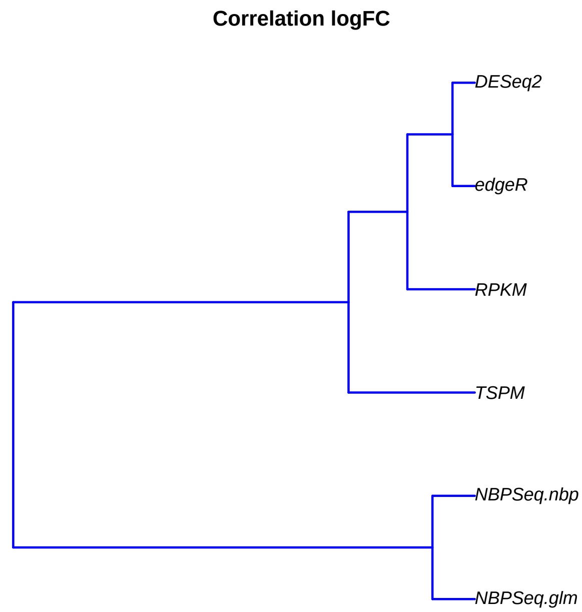

Figure 18: Overall similarity between the methods based on Spearman correlation of gene ranks.

4 Version Information

```
> toLatex(sessionInfo())
```

- R version 3.1.2 (2014-10-31), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.28.2, Biobase 2.26.0, BiocGenerics 0.12.1, BiocParallel 1.0.3, Biostrings 2.34.1, DBI 0.3.1, DEG.comparison 1.0, GenomeInfoDb 1.2.5, GenomicAlignments 1.2.2, GenomicRanges 1.18.4, IRanges 2.0.1, NBPSeq 0.3.0, ROCR 1.0-5, RSQLite 1.0.0, Rsamtools 1.18.3, S4Vectors 0.4.0, ShortRead 1.24.0, XVector 0.6.0, abind 1.4-3, ape 3.2, baySeq 2.0.50, ggplot2 1.0.1, gplots 2.17.0, systemPipeR 1.0.12
- Loaded via a namespace (and not attached): AnnotationForge 1.8.2, BBmisc 1.9, BatchJobs 1.6, BiocStyle 1.4.1, Category 2.32.0, GO.db 3.0.0, GOstats 2.32.0, GSEABase 1.28.0, KernSmooth 2.23-14, MASS 7.3-40, Matrix 1.2-0, RBGL 1.42.0, RColorBrewer 1.1-2, Rcpp 0.11.6, XML 3.98-1.1, annotate 1.44.0, base64enc 0.1-2, bitops 1.0-6, brew 1.0-6, caTools 1.17.1, checkmate 1.5.3, codetools 0.2-11, colorspace 1.2-6, digest 0.6.8, edgeR 3.8.6, fail 1.2, foreach 1.4.2, gdata 2.16.1, genefilter 1.48.1, graph 1.44.1, grid 3.1.2, gtable 0.1.2, gtools 3.4.2, hwriter 1.3.2, iterators 1.0.7, lattice 0.20-31, latticeExtra 0.6-26, limma 3.22.7, magrittr 1.5, munsell 0.4.2, nlme 3.1-120, pheatmap 1.0.2, plyr 1.8.2, proto 0.3-10, qvalue 1.43.0, reshape2 1.4.1, rjson 0.2.15, scales 0.2.4, sendmailR 1.2-1, splines 3.1.2, stringi 0.4-1, stringr 1.0.0, survival 2.38-1, tools 3.1.2, xtable 1.7-4, zlibbioc 1.12.0

5 References

Paul L Auer and Rebecca W Doerge. A Two-Stage poisson model for testing RNA-Seq data. *Stat. Appl. Genet. Mol. Biol.*, 10(1):1–26, 2011.

Daniela Cassol and Lichao Li. DEG.comparison: A comparison of methods for deg analysis of rna-seq data, 10 June 2015. URL https://github.com/dcassol/DEG.comparison.

Yanming Di, Sarah C. Emerson, Daniel W. Schafer, Jeffrey a. Kimbrel, and Jeff H. Chang. Higher order asymptotics for negative binomial regression inferences from RNA-sequencing data. *Statistical Applications in Genetics and Molecular Biology*, 12(1):49–70, 2013. ISSN 15446115. doi: 10.1515/sagmb-2012-0071.

Thomas Girke. systemPipeR: NGS workflow and report generation environment, 28 June 2014. URL https://github.com/tgirke/systemPipeR.

Thomas J Hardcastle and Krystyna a Kelly. baySeq: empirical Bayesian methods for identifying differential expression in sequence count data. *BMC bioinformatics*, 11:422, 2010. ISSN 1471-2105. doi: 10.1186/1471-2105-11-422.

Yuling Jiao and Elliot M Meyerowitz. Cell-type specific analysis of translating RNAs in developing flowers reveals new levels of control. *Molecular systems biology*, 6(419):419, 2010. ISSN 1744-4292. doi: 10.1038/msb.2010.76. URL http://dx.doi.org/10.1038/msb.2010.76.

Daehwan Kim, Geo Pertea, Cole Trapnell, Harold Pimentel, Ryan Kelley, and Steven L Salzberg. TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biol.*, 14(4):R36, 25 April 2013. ISSN 1465-6906. doi: 10.1186/gb-2013-14-4-r36. URL http://dx.doi.org/10.1186/gb-2013-14-4-r36.

Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nat. Methods*, 9(4):357–359, April 2012. ISSN 1548-7091. doi: 10.1038/nmeth.1923. URL http://dx.doi.org/10.1038/nmeth.1923.

Michael Lawrence, Wolfgang Huber, Hervé Pagès, Patrick Aboyoun, Marc Carlson, Robert Gentleman, Martin T Morgan, and Vincent J Carey. Software for computing and annotating genomic ranges. *PLoS Comput. Biol.*, 9(8):e1003118, 8 August 2013. ISSN 1553-734X. doi: 10.1371/journal.pcbi.1003118. URL http://dx.doi.org/10.1371/journal.pcbi.1003118.

Michael Love, Wolfgang Huber, and Simon Anders. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biol.*, 15(12):550, 2014. ISSN 1465-6906. doi: 10.1186/s13059-014-0550-8. URL http://genomebiology.com/2014/15/12/550.

Davis J McCarthy, Yunshun Chen, and Gordon K Smyth. Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation. *Nucleic Acids Res.*, 40(10):4288–4297, May 2012.

Ali Mortazavi, Brian a Williams, Kenneth McCue, Lorian Schaeffer, and Barbara Wold. Mapping and quantifying mammalian transcriptomes by RNA-Seq. *Nature methods*, 5(7):621–628, 2008. ISSN 1548-7091. doi: 10.1038/nmeth.1226.

M D Robinson, D J McCarthy, and G K Smyth. edger: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140, January 2010. ISSN 1367-4803. doi: 10.1093/bioinformatics/btp616. URL http://dx.doi.org/10.1093/bioinformatics/btp616.

Mark D Robinson and Gordon K Smyth. Small-sample estimation of negative binomial dispersion, with applications to SAGE data. *Biostatistics*, 9(2):321–332, April 2008.