RNA-Seq Data Pathway and Gene-set Analysis Workflows

Weijun Luo

September 28, 2013

1 Introduction

In this tutorial, we describe the GAGE (Luo et al., 2009) /Pahview (Luo and Brouwer, 2013) workflows on RNA-Seq data pathway analysis and gene-set analysis. We first cover a full workflow from reads counting, data preprocessing, gene set test, to pathway visualization Section 3. It is called the native workflow, because GAGE/Pahview provides all the functionality for the high level analysis. The same workflow can be used for GO analysis Section 4, and can have different choices of input per gene scores Section 5. GAGE and Pahview are versatile tools. They can take the output from all the major RNA-seq analysis tools and carry on for pathway analysis or visualization. In Section 6, we also describe joint pathway analysis workflows with common RNA-seq analysis tools.

This workflow just covers the most common situations, and never means to be comprehensive. Relevant issues like RNA-seq data quality assessment are not included. Although we focus on RNA-seq data here, but pathway analysis workflow remains similar for microarray, particularly step 3-4 would be the same. Please check *gage* and *pathview* vigenttes for details.

2 Quick start: RNA-seq pathway analysis in about 40 lines

This is concise version, please check the full version below for details.

```
> ##step 0: setup (also need to map the reads outside R)
> source("http://bioconductor.org/biocLite.R")
> biocLite(c("pathview", "gage", "gageData", "Rsamtools",
+              "TxDb.Hsapiens.UCSC.hg19.knownGene"))

> ##step 1: read counts
> library(TxDb.Hsapiens.UCSC.hg19.knownGene)
> exByGn <- exonsBy(TxDb.Hsapiens.UCSC.hg19.knownGene, "gene")
> library(Rsamtools)
> fls <- list.files("tophat_all/", pattern="bam$", full.names =T)
> bamfls <- BamFileList(fls)
> flag <- scanBamFlag(isNotPrimaryRead=FALSE, isProperPair=TRUE)
> param <- ScanBamParam(flag=flag)
> gnCnt <- summarizeOverlaps(exByGn, bamfls, mode="Union",
+              ignore.strand=TRUE, single.end=FALSE, param=param)
> hnrnp.cnts==assay(gnCnt)

> ##step 2: preprocessing
> require(gageData) #demo only
> data(hnrnp.cnts) #demo only
> cnts=hnrnp.cnts
```

```
> sel.rn=rowSums(cnts) != 0
> cnts=cnts[sel.rn,]
> ##joint workflow with DEseq/edgeR/limma/Cufflinks fork here
> libsizes=colSums(cnts)
> size.factor=libsizes/exp(mean(log(libsizes)))
> cnts.norm=t(t(cnts)/size.factor)
> cnts.norm=log2(cnts.norm+8)

> ##step 3: gage
> library(gage)
> ref.idx=5:8
> samp.idx=1:4
> data(kegg.gs)
> cnts.kegg.p <- gage(cnts.norm, gsets = kegg.gs, ref = ref.idx,
+                     samp = samp.idx, compare ="unpaired")

> ##step 4: pathview
> cnts.d= cnts.norm[, samp.idx]-rowMeans(cnts.norm[, ref.idx])
> sel <- cnts.kegg.p$greater[, "q.val"] < 0.1 &
+          !is.na(cnts.kegg.p$greater[,"q.val"])
> path.ids <- rownames(cnts.kegg.p$greater)[sel]
> path.ids2 <- substr(path.ids, 1, 8)
> library(pathview)
> pv.out.list <- sapply(path.ids2, function(pid) pathview(
+                       gene.data = cnts.d, pathway.id = pid,
+                       species = "hsa"))
```

3 The native workflow

This workflow is native to GAGE/Pathview and takes the full advantage of their special design (Luo et al., 2009) (Luo and Brouwer, 2013). The gene expression and pathway level analysis are done using the default setting of the tools, i.e. pair-wise comparison and single sample analysis (Luo et al., 2009). Therefore, this workflow has no limitation on sample size or experimental design. In addition, you can analyze and visualize pathway changes in every single experiment or sample (Figure 2). The test statistics are summarized across samples at pathway level instead of gene level (Luo et al., 2009). The analysis results are more sensitive, informative and consistent than the latter approach.

3.1 Preparation: read mapping and package installation

This preparation step is not part of the RNA-seq data pathway analysis workflow literally. But we have to go through these tasks to prepare for the analysis.

The raw reads data in zipped FASTQ format were downloaded from ArrayExpressBioconductor website. This is the example data used in the at the RNA-seq section of the R/Bioconductor NGS course. They worked with reads mapped to chromosome 14 only, here we work with all reads/genes for a practical pathway analysis.

We use TopHat2 to map the raw reads to the reference human genome (hg19). And then use SAMtools to index the mapped reads. Tophat webpage describes how to install TopHat and SAMtools, prepare the reference genome and use these tools. As an example, below is the code I used to map, index and process the read data for the first sample (ERR127302). You can write a shell script to do so for all samples (ERR127302-9). This step takes a couple of hours on a 8-core processor for each sample. You may run parallel jobs for multiple samples if you have access to HPC.

```
tophat2 -p 8 -o tophat_out_1 ref/hg19 ERR127302_1.fastq.gz ERR127302_2.fastq.gz
cd tophat_out_1
samtools index accepted_hits.bam
mkdir -p tophat_all
ln -s tophat_out_1/accepted_hits.bam tophat_all/ERR127302.bam
ln -s tophat_out_1/accepted_hits.bam.bai tophat_all/ERR127302.bam.bai
```

The pathway analysis workflow is implemented all in R/Bioconductor. You need to install the relevant packages within R if you haven't done so, you will need to work with R 3.0 and Bioconductor 2.13 or newer versions. Note that *gageData* provides the demo RNA-seq data and ready-to-use KEGG and GO gene set data. The installation may take a few minutes. From this point on, we are working fully under R excpet noted otherwise. Of coiurse, you need to start R first.

```
> source("http://bioconductor.org/biocLite.R")
> biocLite(c("pathview", "gage", "gageData", "Rsamtools",
+              "TxDb.Hsapiens.UCSC.hg19.knownGene"))
```

Besides these packages, you will also need to install one of *DESeq*, *DESeq2*, *edgeR*, *limma* and `Cufflinks` (non-Bioconductor) if you want to use the joint workflow described in Section 6.

And we are then ready for the pathway analysis workflow. The workflow has 4 distinct steps, and we describe each of them in details below.

3.2 Step 1: Count the reads mapped to each gene

In this step, we need to extract exon regions by gene (i.e. Annotation of known gene models). It is important to make sure that the gene annotation uses the same version and source of reference genome in the reads mapping step above, in our case, hg19. We then count and summarize the reads mapped to each known gene/exon regions using package *Rsamtools* (and *GenomicRanges*). Here, we used the files `"accepted_hits.bam"` in tophat output directories, which have been renamed after the sample names and collected in to the `"tophat_all"` directory above.

```
> library(TxDb.Hsapiens.UCSC.hg19.knownGene)
> exByGn <- exonsBy(TxDb.Hsapiens.UCSC.hg19.knownGene, "gene")
> library(Rsamtools)
> fls <- list.files("tophat_all/", pattern="bam$", full.names =T)
> bamfls <- BamFileList(fls)
> flag <- scanBamFlag(isNotPrimaryRead=FALSE, isProperPair=TRUE)
> param <- ScanBamParam(flag=flag)
> gnCnt <- summarizeOverlaps(exByGn, bamfls, mode="Union",
+              ignore.strand=TRUE, single.end=FALSE, param=param)
> hnrnp.cnts==assay(gnCnt)
```

3.3 Step 2: Normalize and process read counts

We divide the read counts by the total number of mapped reads for each sample as to normalize over the library size and sequence depth. We don't count for the gene length because it will be cancelled out in the relative expression level for each gene. Genes with 0 counts across all samples are removed (either before or after normalization is the same). We add a small yet appropriate positive count number (+8) to all genes before doing log2 transformation, as to avoid -Inf and stablize the variance at low expression end. The *DEseq* package vignette describes a sophisticated "Variance stabilizing transformation" in detail. We may do MA plot as to check the processed data variances using MA plot (Figure 1). You may further do principle component analysis (PCA) plot to check the overall variances and similarity between samples (not shown).

For this step on, we can actually work on the pre-mapped raw read counts data from steps aboved, i.e. hnrnp.cnts stored in *gageData*.

```
> require(gageData)
> data(hnrnp.cnts)
> cnts=hnrnp.cnts
> dim(cnts)

[1] 22932     8

> sel.rn=rowSums(cnts) != 0
> cnts=cnts[sel.rn,]
> dim(cnts)

[1] 17192     8

> libsizes=colSums(cnts)
> size.factor=libsizes/exp(mean(log(libsizes)))
> cnts.norm=t(t(cnts)/size.factor)
> range(cnts.norm)

[1]      0.0 999179.9

> cnts.norm=log2(cnts.norm+8)
> range(cnts.norm)

[1]  3.0000 19.9304

> #optional MA plot
> pdf("hnrnp.cnts.maplots.pdf", width=8, height=10)
> op=par(lwd=2, cex.axis=1.5, cex.lab=1.5, mfrow=c(2,1))
> plot((cnts.norm[,6]+cnts.norm[,5])/2, (cnts.norm[,6]-cnts.norm[,5]),
+ main="(a) Control vs Control", xlab="mean", ylab="change",
+ ylim=c(-5,5), xlim=c(0,20), lwd=1)
> abline(h=0, lwd=2, col="red", lty="dashed")
> plot((cnts.norm[,1]+cnts.norm[,5])/2, (cnts.norm[,1]-cnts.norm[,5]),
+ main="(b) Knockdown vs Control", xlab="mean", ylab="change",
+ ylim=c(-5,5), xlim=c(0,20), lwd=1)
> abline(h=0, lwd=2, col="red", lty="dashed")
> dev.off()

null device
          1
```

3.4 Step 3: Gene set test with GAGE

Here, we do gene set test to select the signficantly perturbed KEGG pathways using GAGE (Luo et al., 2009). For more information on GAGE analysis please check the main *gage* vignette and the paper (Luo et al., 2009). You may also explore advanced GAGE analysis options and view the results using heatmaps or scatter plot described in the vignette.

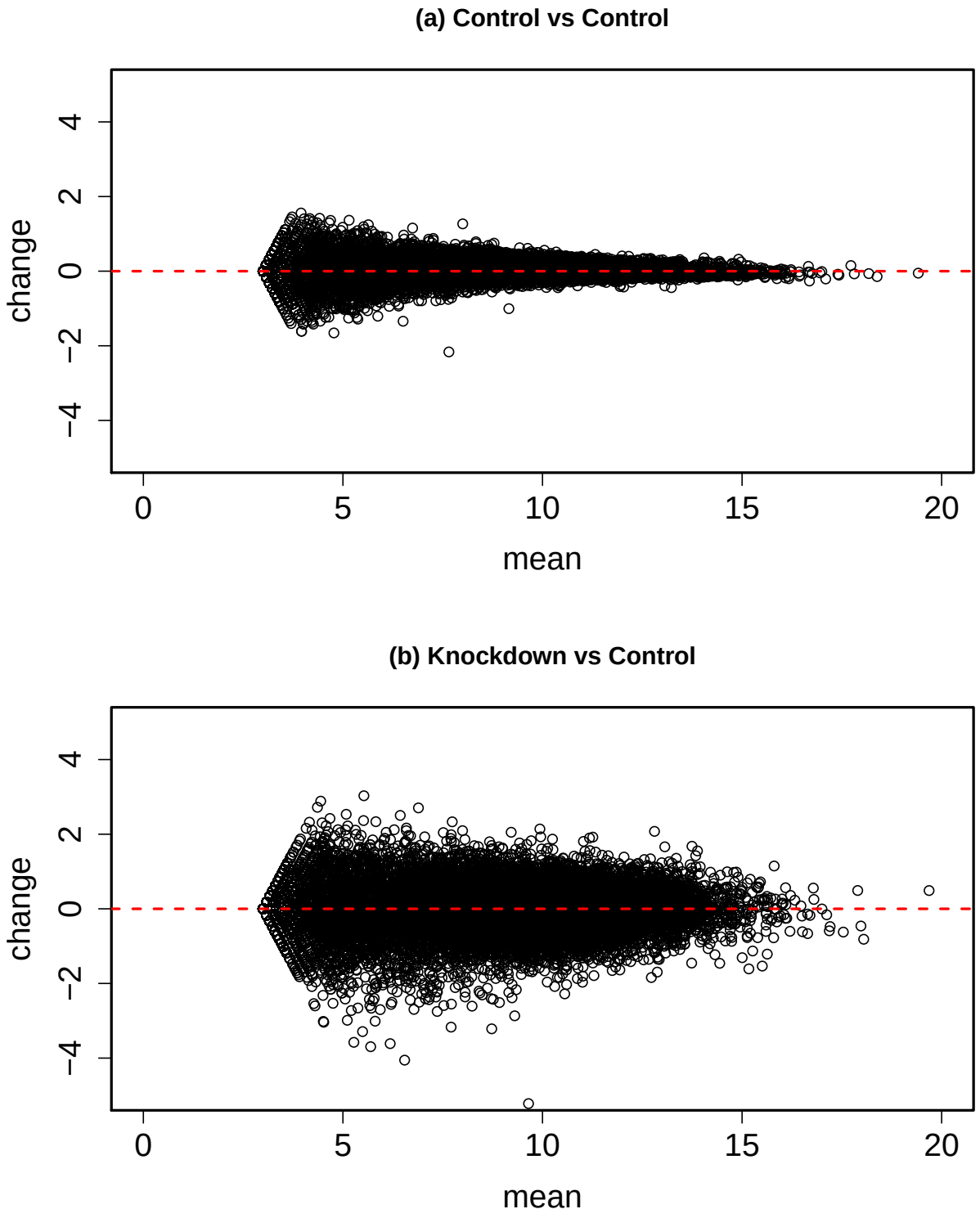


Figure 1: MA plots on processed gene-wise read counts.

```
> library(gage)
> ref.idx=5:8
> samp.idx=1:4
> data(kegg.gs)
> #knockdown and control samples are unpaired
> cnts.kegg.p <- gage(cnts.norm, gsets = kegg.gs, ref = ref.idx,
+                     samp = samp.idx, compare ="unpaired")
```

3.5 Step 4: Pathway visualization with Pathview

We then visualize the gene expression perturbations in significant KEGG pathways using Pathview (Figure 2). For more information on Pathview please check the main *pathview* vignette and the paper (Luo and Brouwer, 2013).

```
> #differential expression: log2 ratio or fold change, uppaired samples
> cnts.d= cnts.norm[, samp.idx]-rowMeans(cnts.norm[, ref.idx])
> #up-regulated pathways (top 3) visualized by pathview
> sel <- cnts.kegg.p$greater[, "q.val"] < 0.1 &
+          !is.na(cnts.kegg.p$greater[,"q.val"])
> path.ids <- rownames(cnts.kegg.p$greater)[sel]
> path.ids2 <- substr(path.ids, 1, 8)
> library(pathview)
> pv.out.list <- sapply(path.ids2[1:3], function(pid) pathview(
+                       gene.data = cnts.d, pathway.id = pid,
+                       species = "hsa"))
```

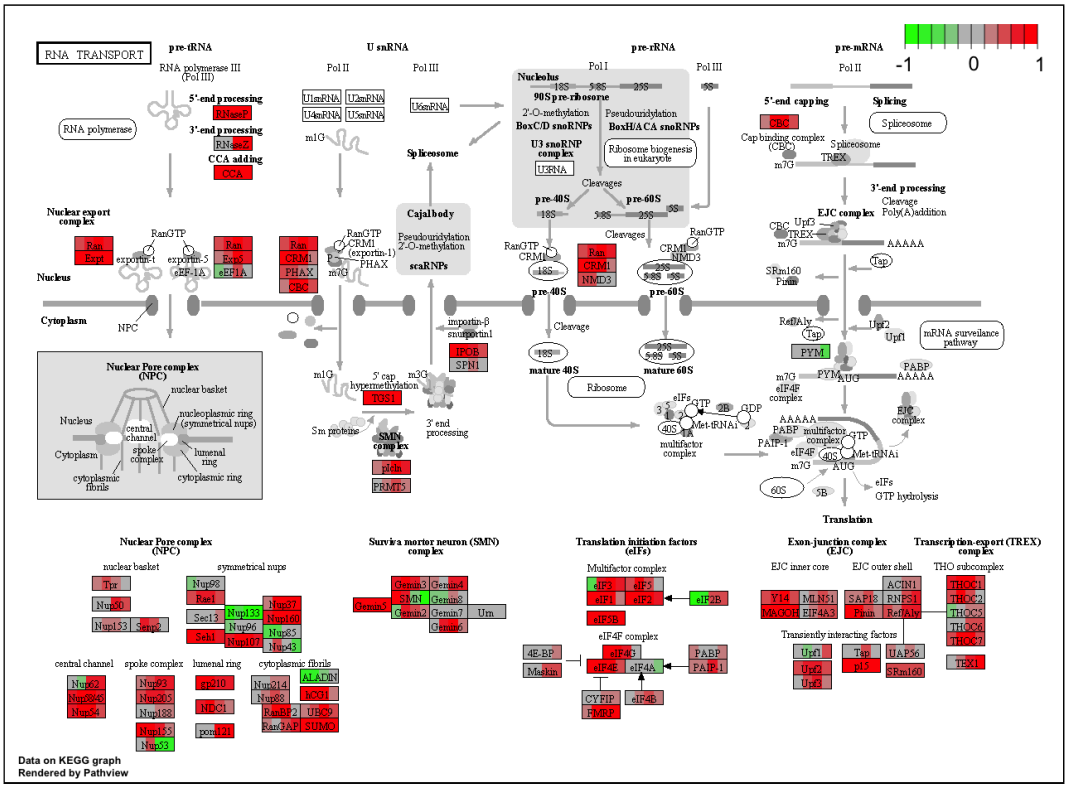
```
[1] "Downloading xml files for hsa03013, 1/1 pathways.."
[1] "Downloading png files for hsa03013, 1/1 pathways.."
[1] "Downloading xml files for hsa03008, 1/1 pathways.."
[1] "Downloading png files for hsa03008, 1/1 pathways.."
```

```
> #down-regulated pathways  (top 3) visualized by pathview
> sel.l <- cnts.kegg.p$less[, "q.val"] < 0.1 &
+            !is.na(cnts.kegg.p$less[,"q.val"])
> path.ids.l <- rownames(cnts.kegg.p$less)[sel.l]
> path.ids.l2 <- substr(path.ids.l, 1, 8)
> pv.out.list.l <- sapply(path.ids.l2[1:3], function(pid) pathview(
+                       gene.data = cnts.d, pathway.id = pid,
+                       species = "hsa"))
```

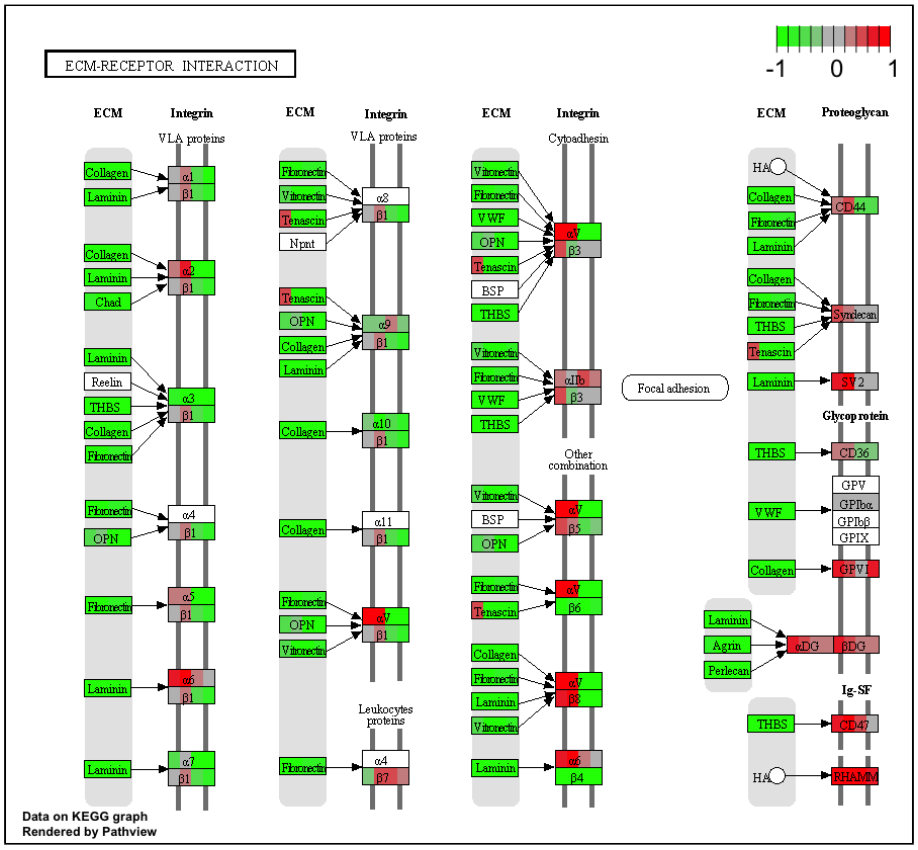
```
[1] "Downloading xml files for hsa04610, 1/1 pathways.."
[1] "Downloading png files for hsa04610, 1/1 pathways.."
[1] "Downloading xml files for hsa04512, 1/1 pathways.."
[1] "Downloading png files for hsa04512, 1/1 pathways.."
[1] "Downloading xml files for hsa04510, 1/1 pathways.."
[1] "Downloading png files for hsa04510, 1/1 pathways.."
```

4 GO analysis

GAGE's capability for GO analysis has long been under-appreciated partially because we always use the term "pathway analysis" (Luo et al., 2009). We have demonstrate the RNA-seq data workflow with KEGG pathway analysis above, as well as in the main *gage* vignette. GAGE is equally well applicable for GO analysis



(a)



(b)

Figure 2: Expression perturbations in signficant pathways visualized by Pathview (a) hsa03013 RNA transport; or (b) hsa04512 ECM-receptor interaction.

(Luo et al., 2009). In fact, the Biological Process terms have similar definitions to KEGG pathways. GAGE analysis on Biological Process could be even more informative as it includes more comprehensive and detailed pathway/process definition. However, GO definition doesn't included molecular interactions hence not able to visulize like KEGG pathway graphs (Figure 2).

```
> library(gageData)
> data(go.sets.hs)
> data(go.subs.hs)
> lapply(go.subs.hs, head)
> #even more informative, take half a minute to run
> cnts.mf.p <- gage(cnts.norm, gsets = go.sets.hs[go.subs.hs$MF],
+     ref = ref.idx, samp = samp.idx, compare ="unpaired")
> cnts.bp.p <- gage(cnts.norm, gsets = go.sets.hs[go.subs.hs$BP],
+     ref = ref.idx, samp = samp.idx, compare ="unpaired")
```

5 Per gene score choices

GAGE does pair-wise comparison between samples by default (Luo et al., 2009), which makes GAGE applicable for arbitrary sample sizes or columan numbers (from 1 to many), and more sensitive than other methods. In other words, GAGE uses sample wise fold change as per gene score/statistics in gene set test. However, per gene scores from group wise comparison including mean fold change, t-stats, f-stats, correlation, etc can all be used with GAGE. We demonstrate GAGE analysis with mean fold change and t-stats as input below. But t-stats are not recommended here because of the small sample size (4). Nontheless, the selected top gene sets or pathways are similar to results from the main workflow above.

```
> cnts.t= apply(cnts.norm, 1, function(x) t.test(x[samp.idx], x[ref.idx],
+               alternative = "two.sided", paired = F)$statistic)
> cnts.meanfc= rowMeans(cnts.norm[, samp.idx]-cnts.norm[, ref.idx])
> range(cnts.t)

[1] -42.14549  59.50588

> range(cnts.meanfc)

[1] -4.818676  2.749263

> cnts.t.kegg.p <- gage(cnts.t, gsets = kegg.gs, ref = NULL, samp = NULL)
> cnts.meanfc.kegg.p <- gage(cnts.meanfc, gsets = kegg.gs, ref = NULL, samp = NULL)
```

6 Joint workflows with common RNA-seq analysis tools

Currently, the most common RNA-seq data analysis tools include *DESeq* (Anders and Huber, 2010) (and *DESeq2*), *edgeR* (Robinson et al., 2010), *Limma* (Smyth, 2004) and `Cufflinks` (Trapnell et al., 2012). Many users are already familiar with them, and countless analyses have been done and published using them. All these tools except `Cufflinks` are implemented in R/Bioconductor. It is very convenient to combine these tools with GAGE/Pahview for joint pathway analysis workflows. In other words, these tools do differential expression analysis first, and GAGE/Pahview then takes their results for pathway or gene set analysis. Compared to the native workflow in Section 3 above, these tools actually take place of step 2 and the internal data preparation of GAGE at step 3. Other steps, i.e. preparation and step 1 and 4 remain the same. Below are example workflows with each one of these analysis tools.

Notice the all of these tools output fold change (log ratio). Some of them, including *Limma* and `Cufflinks`, also output test-statistics. Below we will use fold changes for all workflows. But test-statistics can still be used, particularly for well designed experiments with sufficient sample size. We will show a complete pathway analysis workflow with *DESeq2*. For all other tools, our demos only include the upstream different expression analysis with fold change results, because the downstream GAGE/Pathview steps remain the same.

6.1 DESeq2

DESeq has two versions in Bioconductor, *DEseq* and *DESeq2*. The latter is preferred. It is faster and simpler, more importantly the fold change values are cleaner and more sensible, i.e. no -Inf or Inf values.

First, *DESeq2* for differential expression analysis:

```
> library(DESeq2)
> grp.idx <- rep(c("knockdown", "control"), each=4)
> coldat=DataFrame(grp=factor(grp.idx))
> dds <- DESeqDataSetFromMatrix(cnts, colData=coldat, design = ~ grp)
> dds <- DESeq(dds)
> deseq2.res <- results(dds)
> #direction of fc, depends on levels(coldat$grp), the first level
> #taken as reference (or control) and the second one as experiment.
> deseq2.fc=deseq2.res$log2FoldChange
> names(deseq2.fc)=rownames(deseq2.res)
> exp.fc=deseq2.fc
> out.suffix="deseq2"
```

Next, GAGE for pathway analysis, and Pathview for visualization. We only visualize up-regulated pathways here, down-regulated pathways can be done the same way (see also the above native workflow). Notice that this step (the same code) is identical for *DESeq*, *edgeR*, *Limma* and `Cufflinks` workflows, hence skipped in other workflows below.

```
> require(gage)
> data(kegg.gs)
> fc.kegg.p <- gage(exp.fc, gsets = kegg.gs, ref = NULL, samp = NULL)
> sel <- fc.kegg.p$greater[, "q.val"] < 0.1 &
+         !is.na(fc.kegg.p$greater[, "q.val"])
> path.ids <- rownames(fc.kegg.p$greater)[sel]
> path.ids2 <- substr(path.ids, 1, 8)
> require(pathview)
> pv.out.list <- sapply(path.ids2[1:3], function(pid) pathview(
+                       gene.data =  exp.fc, pathway.id = pid,
+                       species = "hsa", out.suffix=out.suffix))
```

Here we used GAGE to infer the signficant pathways. But we are not limited to these pathways. We can use Pathview to visualize RNA-seq data (exp.fc here) on all interesting pathways directly.

6.2 DESeq

DEseq analysis is bit lengthier and slower, although it is still a most used RNA-seq analysis tool in Bioconductor. Note that we need to remove the -Inf or Inf values in the fold changes.

```
> library(DESeq)
> grp.idx <- rep(c("knockdown", "control"), each=4)
```

```
> cds <- newCountDataSet(cnts, grp.idx)
> cds = estimateSizeFactors(cds)
> cds = estimateDispersions(cds)
> #this line takes several minutes
> system.time(
+ deseq.res <- nbinomTest(cds, "knockdown", "control")
+ )
> deseq.fc=deseq.res$log2FoldChange
> names(deseq.fc)=deseq.res$id
> sum(is.infinite(deseq.fc))
> deseq.fc[deseq.fc>10]=10
> deseq.fc[deseq.fc< -10]=-10
> exp.fc=deseq.fc
> out.suffix="deseq"
```

The folllowing GAGE and Pathview steps remain the same as in Subsection DESeq2.

6.3 edgeR

The *edgeR* package is another most used and the earliest RNA-seq analysis tool in Bioconductor. It is probably the first one using negative binomial distribution to model RNA-seq data (Robinson et al., 2010).

```
> library(edgeR)
> grp.idx <- rep(c("knockdown", "control"), each=4)
> dgel <- DGEList(counts=cnts, group=factor(grp.idx))
> dgel <- calcNormFactors(dgel)
> dgel <- estimateCommonDisp(dgel)
> dgel <- estimateTagwiseDisp(dgel)
> et <- exactTest(dgel)
> edger.fc=et$table$logFC
> names(edger.fc)=rownames(et$table)
> exp.fc=edger.fc
> out.suffix="edger"
```

6.4 Limma

Limma is the most used expression data analysis tool and a most downloaded package in Bioconductor. It is best known for microarray analysis, but RNA-seq data analysis has also been implemented recently. Note that *edgeR* package is needed here too.

```
> library(edgeR)
> grp.idx <- rep(c("knockdown", "control"), each=4)
> dgel2 <- DGEList(counts=cnts, group=factor(grp.idx))
> dgel2 <- calcNormFactors(dgel2)
> library(limma)
> design <- model.matrix(~grp.idx)
> log2.cpm <- voom(dgel2,design)
> fit <- lmFit(log2.cpm,design)
> fit <- eBayes(fit)
> limma.res=topTable(fit,coef=2,n=Inf,sort="p")
> limma.fc=limma.res$logFC
```

```
> names(limma.fc)=limma.res$ID
> exp.fc=limma.fc
> out.suffix="limma"
```

6.5 Cufflinks

`Cufflinks` is a most popular RNA-seq analysis tool. It is developed by the same group as TopHat. It is implemented independent of Bioconductor. However, we can read its gene differential expression analysis results into R easily. The result file is named `gene_exp_diff`, and here is the description is provided on Cufflinks webpage. Notice that the gene sybmols need to converted to Entrez Gene IDs, which are used in KEGG pathways (many research species) and GO gene sets.

```
> cuff.res=read.delim(file="gene_exp.diff", sep="\t")
> cuff.fc=cuff.res$log2.fold_change.
> gnames=cuff.res$gene
> sel=gnames!="-"
> gnames=as.character(gnames[sel])
> cuff.fc=cuff.fc[sel]
> names(cuff.fc)=gnames
> gnames.eg=pathview::id2eg(gnames, category ="symbol")
> sel2=gnames.eg[,2]>""
> cuff.fc=cuff.fc[sel2]
> names(cuff.fc)=gnames.eg[sel2,2]
> range(exp.fc)
> cuff.fc[cuff.fc>10]=10
> cuff.fc[cuff.fc< -10]=-10
> exp.fc=cuff.fc
> out.suffix="cuff"
```

References

Simon Anders and Wolfgang Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11(10):R106, 2010. ISSN 1465-6906. doi: 10.1186/gb-2010-11-10-r106. URL `http://genomebiology.com/2010/11/10/R106`.

Weijun Luo and Cory Brouwer. Pathview: an R/Bioconductor package for pathway-based data integration and visualization. *Bioinformatics*, 29(14):1830–1831, 2013. doi: 10.1093/bioinformatics/btt285. URL `http://bioinformatics.oxfordjournals.org/content/29/14/1830.full`.

Weijun Luo, Michael Friedman, Kerby Shedden, Kurt Hankenson, and Peter Woolf. GAGE: generally applicable gene set enrichment for pathway analysis. *BMC Bioinformatics*, 10(1):161, 2009. URL `http://www.biomedcentral.com/1471-2105/10/161`.

Mark D. Robinson, Davis J. McCarthy, and Gordon K. Smyth. edger: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, 26(1):139–140, 2010. doi: 10.1093/bioinformatics/btp616. URL `http://bioinformatics.oxfordjournals.org/content/26/1/139.abstract`.

Gordon K. Smyth. Linear models and empirical bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, 3(1):Article 3, 2004.

Cole Trapnell, Adam Roberts, Loyal Goff, Geo Pertea, Daehwan Kim, David R. Kelley, Harold Pimentel, Steven L. Salzberg, John L. Rinn, and Lior Pachter. Differential gene and transcript expression analysis of rna-seq experiments with tophat and cufflinks. *Nature Protocols*, 7(3):562–578, 2012.