SPEAQeasy Differential Expression Analysis

July 10, 2020

Contents

1 Analysis

The following analysis explores a `RangedSummarizedExperiment` object from the SPEAQeasy pipeline. Note that we will use a modified version of the object, which resolved sample identity issues which were present in the raw output from SPEAQeasy. This object also includes phenotype data added after resolving identity issues. Though SPEAQeasy produces objects for several feature types (genes, exons, exon-exon junctions), we will demonstrate an example analysis for just genes. We will perform differential expression across some typical variables of interest (e.g. sex, age, race) and show how to perform principal component analysis (PCA) and visualize findings with plots.

1.1 Load required libraries

```r
library("SummarizedExperiment")
library("recount")
library("edgeR")
library("limma")
library("jaffelab") # GitHub: LieberInstitute/jaffelab
library("RColorBrewer")
library("clusterProfiler")
library("org.Hs.eg.db")
library("pheatmap")
library("here")
```

1.2 Load data and prepare directories to place outputs in

For those who ran SPEAQeasy from the example FASTQ data set, the `RangedSummarizedExperiment` will have a different path, as specified with the `--output` flag.

```r
# Load the RSE gene object
load(here("rse_speaqeasy.RData"), verbose = TRUE)
```

```
## Loading objects:
##   rse_gene
```

```r
# Create directories to organize outputs from this analysis
dir.create(here("DE_analysis", "pdfs"), showWarnings = FALSE)
dir.create(here("DE_analysis", "tables"), showWarnings = FALSE)
dir.create(here("DE_analysis", "rdas"), showWarnings = FALSE)
```

1.3 statistics PCs

1.4 Here we are using principal component analysis to control for the listed variables impact on expression. This will be later added into our linear model

col_names = c('trimmed', 'numReads', 'numMapped', 'numUnmapped', 'overallMapRate', 'concordMapRate', 'totalMapped', 'mitoMapped', 'mitoRate', 'totalAssignedGene') statsPca = prcomp(as.data.frame(colData(rse
rse_gene$PC = statsPca$x[,1] getPcaVars(statsPca)[1] # 87.3

1.5 filter for expressed

rse_gene = rse_gene[rowMeans(getRPKM(rse_gene,"Length")) > 0.2,]

1.6 metrics

1.7 check if certain statistics changed by race or region

pdf(file = here("DE_analysis", "pdfs", "Region_Race_cellcheck.pdf")) boxplot(rse_gener$RNA_rate\ rse_gene$BrainRegion,xlal
boxplot(rse_gene$mitoRate\ rse_gene$BrainRegion,xlab=""") boxplot(rse_gene$gene_Assigned\ rse_gene$BrainRegion,xlab=""")

boxplot(rse_gene$mitoRate\ rse_gene$Race,las=3,xlab=”“) boxplot(rse_gene$gene_Assigned\ rse_gene$Race,las=3,xlab=”") dev.off()

1.7.0.1 explore gene expression geneExprs = log2(getRPKM(rse_gene,“Length”)+1) pca = prcomp(t(geneExprs)) pca_vars = getPcaVars(pca) pca_vars_lab = paste0(“PC”, seq(along=pca_vars), “:”, pca_vars, “% Var Expl”)

pdf(here(“DE_analysis”, “pdfs”, “PCA_plotsExprs.pdf”), w=9) par(mar=c(8,6,2,2),cex.axis=1.8,cex.lab=1.8) palette(brewer.pal(4,“Dark2”))

1.8 pc1 vs pc2

plot(pca$x, pch = 21, bg = factor(rse_gene$PrimaryDx),cex=1.2, xlab = pca_vars_lab[1], ylab = pca_vars_lab[2]) legend(“bottomleft”, levels(rse_gene$PrimaryDx), col=1:2, pch=15,cex=2)

1.9 by line

for(i in 1:10) { boxplot(pca$x[, i]\ rse_gene$Sex, ylab=pca_vars_lab[i], las = 3,xlab=“Sex”,outline=FALSE) points(pca$x[, i]\ jitter(as.numeric(factor(rse_gene$Sex))), pch = 21, bg = rse_gene$PrimaryDx,cex=1.2) }

1.10 by experiment

for(i in 1:10) { boxplot(pca$x[, i]\ rse_gene$Race, ylab=pca_vars_lab[i], las = 3,xlab=“Race”,outline=FALSE) points(pca$x[, i]\ jitter(as.numeric(factor(rse_gene$Race))), pch = 21, bg = rse_gene$PrimaryDx,cex=1.2) } dev.off()

1.11 modeling

dge = DGEList(counts = assays(rse_gene)$counts, genes = rowData(rse_gene)) dge = calcNormFactors(dge)

1.12 mean-variance

mod = model.matrix(~PrimaryDx + PC + BrainRegion, data=colData(rse_gene)) pdf(file = “DE_analysis/pdfs/vGene.pdf” vGene = voom(dge,mod,plot=TRUE) dev.off()

##Get duplicate correlation gene_dupCorr = duplicateCorrelation(vGene$E, mod, block = colData(rse_gene)$SAMPLE_ID) save(gene_dupCorr, file = “DE_analysis/rdas/gene_dupCorr_neurons.rda”)

1.13 fit linear model

fitGeneDupl = lmFit(vGene, correlation=gene_dupCorr$consensus.correlation, block = colData(rse_gene)$SAMPLEID)

1.14 here we perform an empirical bayesian calculation to obtain our significant genes

ebGeneDupl = eBayes(fitGeneDupl) outGeneDupl = topTable(ebGeneDupl,coef=2, p.value = 1,number=nrow(rse_gene),sort="none")

pdf(file = here("DE_analysis", "pdfs", "hist_pval.pdf")) hist(outGeneDupl$P.Value)dev.off()table(outGeneDupl$adj.P.Val < 0.05) table(outGeneDupl$adj.P.Val < 0.1)

sigGeneDupl = topTable(ebGeneDupl,coef=2, p.value = 0.1,number=nrow(rse_gene))

sigGeneDupl[,c("Symbol","logFC", "P.Value","AveExpr")] sigGeneDupl[sigGeneDupl$logFC > 0,c("Symbol","logFC", "P.Value")] sigGeneDupl[sigGeneDupl$logFC < 0,c("Symbol","logFC", "P.Value")]

write.csv(outGeneDupl, file = here("DE_analysis", "tables", "de_stats_allExprs.csv")) write.csv(sigGeneDupl, file = here("DE_analysis", "tables", "de_stats_fdr10_sorted.csv"))

1.15 check plots

exprs = vGene$E[rownames(sigGeneDupl),]

1.15.1 make boxplots

pdf(here("DE_analysis", "pdfs", "DE_boxplots_byDiagnosis.pdf", w=10)) par(mar=c(8,6,4,2),cex.axis=1.8,cex.lab=1.8, cex.main=1.8) palette(brewer.pal(4,"Dark2")) for(i in 1:nrow(sigGeneDupl)) { yy = exprs[i,] boxplot(yy ~ rse_gene$PrimaryDx, outline = FALSE, ylim = range(yy), ylab = "Normalized log2 Exprs", xlab = "", main = paste(sigGeneDupl$Symbol[i], "-", sigGeneDupl$gencodeID[i]))points(yy jitter(as.numeric(rse_gene$PrimaryDx)), pch = 21, bg= rse_gene$PrimaryDx, cex = 1.3)ll = ifelse(sigGeneDupl$logFC[i] > 0, "topleft", "topright") legend(ll, paste0("p=", signif(sigGeneDupl$P.Value[i],3)), cex=1.3) } dev.off()

e = geneExprs[rownames(sigGeneDupl),]

pdf(here("DE_analysis", "pdfs", "DE_boxplots_byGenome_log2RPKM.pdf", w=10)) par(mar=c(8,6,4,2),cex.axis=1.8,cex. cex.main=1.8) palette(brewer.pal(4,"Dark2")) for(i in 1:nrow(sigGeneDupl)) { yy = e[i,] boxplot(yy ~ rse_gene$PrimaryDx, las = 3, outline = FALSE, ylim = range(yy), ylab = "log2(RPKM + 1)", xlab = "", main = paste(sigGeneDupl$Symbol[i], "-", sigGeneDupl$gencodeID[i]))points(yy jitter(as.numeric(rse_gene$PrimaryDx)), pch = 21, bg= rse_gene$PrimaryDx, cex = 1.3)ll = ifelse(sigGeneDupl$logFC[i] > 0, "topleft", "topright") legend(ll, paste0("p=", signif(sigGeneDupl$P.Value[i],3)), cex=1.3) } dev.off()

1.15.2 gene ontology

1.15.3 clusterprofiler is a geneontology package we will use to see if our genes are specifically differentially expressed in certain pathways.

1.16 get significant genes by sign

sigGene = outGeneDupl[outGeneDupl$P.Value < 0.005,] sigGeneList = split(as.character(sigGene$EntrezID), sign(sigGenel sigGeneList = lapply(sigGeneList, function(x) x[!is.na(x)]) geneUniverse = as.character(outGeneDupl$EntrezID) geneUniverse = geneUniverse[!is.na(geneUniverse)]

1.17 do GO and KEGG

goBP_Adj <- compareCluster(sigGeneList, fun = “enrichGO”, universe = geneUniverse, OrgDb = org.Hs.eg.db, ont = “BP”, pAdjustMethod = “BH”, pvalueCutoff = 1, qvalueCutoff = 1, readable= TRUE)

goMF_Adj <- compareCluster(sigGeneList, fun = “enrichGO”, universe = geneUniverse, OrgDb = org.Hs.eg.db, ont = “MF”, pAdjustMethod = “BH”, pvalueCutoff = 1, qvalueCutoff = 1, readable= TRUE)

goCC_Adj <- compareCluster(sigGeneList, fun = “enrichGO”, universe = geneUniverse, OrgDb = org.Hs.eg.db, ont = “CC”, pAdjustMethod = “BH”, pvalueCutoff = 1, qvalueCutoff = 1, readable= TRUE)

kegg_Adj <- compareCluster(sigGeneList, fun = “enrichKEGG”, universe = geneUniverse, pAdjustMethod = “BH”, pvalueCutoff = 1, qvalueCutoff = 1)

save(goBP_Adj, goCC_Adj, goMF_Adj, kegg_Adj, file = here(“DE_analysis”, “rdas”, “gene_set_objects_p005.rda”))

goList = list(BP = goBP_Adj, MF = goMF_Adj, CC = goCC_Adj, KEGG = kegg_Adj) goDf = dplyr::bind_rows(lapply(goList, as.data.frame), .id = “Ontology”) goDf = goDf[order(goDf$pvalue),]

write.csv(goDf, file = here(“DE_analysis”, “tables”, “geneSet_output.csv”), row.names=FALSE)

options(width=130) goDf[goDf$p.adjust < 0.05, c(1:5,7)]

#make heatmap of differentially expressed genes# ##

exprs_heatmap = vGene$E[rownames(sigGene),]

df <- as.data.frame(colData(rse_gene)[,c(“PrimaryDx”)]) rownames(df) <- colnames(exprs_heatmap) colnames(df)<-“diagnosis”

pdf(file=here(“DE_analysis”, “pdfs”, “de_heatmap.pdf”)) pheatmap(exprs_heatmap, cluster_rows=TRUE, show_rownames=FALSE, cluster_cols=TRUE, annotation_col=df) dev.off()