Using ReportingTools in an Analysis of RNA-seq Data

Jessica L. Larson and Christina Chaivorapol

April 4, 2013

Contents

1 Introduction

The ReportingTools package can be used with differential gene expression results from RNA-sequencing analysis. In this vignette we show how to publish output from an edgeR, Gene Ontology (GO) and/or Protein family (PFAM) analysis. In the final section we publish all our pages onto one, creating a comprehensive output page.

2 Differential expression analysis with edgeR

In this section we demonstrate how to use the ReportingTools package to generate a table of differentially expressed genes as determined by the edgeR software. We begin by loading our library and data set. The mockRnaSeqData contains random RNA-seq output for random mouse genes.

```
> library(ReportingTools)
> data(mockRnaSeqData)
```

Next, we run edgeR to find differentially expressed genes.

```
> library(edgeR)
> conditions <- c(rep("case",3), rep("control", 3))
> d <- DGEList(counts = mockRnaSeqData, group = conditions)
> d <- calcNormFactors(d)
> d <- estimateCommonDisp(d)
> ## Get an edgeR object
> edgeR.de <- exactTest(d)
```

Now the results can be written to a report using the DGEExact object.

```
> library(lattice)
> rep.theme <- reporting.theme()
> ## Change symbol colors in plots
> rep.theme$superpose.symbol$col <- c("blue", "red")
> rep.theme$superpose.symbol$fill <- c("blue", "red")
> lattice.options(default.theme = rep.theme)
> ## Publish a report of the top 10 genes with p-values < 0.05 and log-fold change > 2
> ## In this case, the plots contain the counts from mockRnaSeqData, which are not normalized.
> ## The publish function does not normalize counts for the countTable argument to allow for
> ## flexibility in plotting various units (e.g. RPKM instead of counts).
>
> deReport <- HTMLReport(shortName = 'RNAseq_analysis_with_edgeR',
+     title = 'RNA-seq analysis of differential expression using edgeR',
+     reportDirectory = "./reports")
> publish(edgeR.de, deReport, countTable=mockRnaSeqData,
+         conditions=conditions, annotation.db = 'org.Mm.eg',
+         pvalueCutoff = .05, lfc = 2, n = 10, name="edgeR")
> finish(deReport)
>
> ## If you would like to plot normalized counts, run the following commands instead:
> ## mockRnaSeqData.norm <- d$pseudo.counts
> ## publish(edgeR.de, deReport, mockRnaSeqData.norm,
> ##        conditions, annotation.db = 'org.Mm.eg',
> ##           pvalueCutoff = .05, lfc = 2, n = 10)
> ## finish(deReport)
```

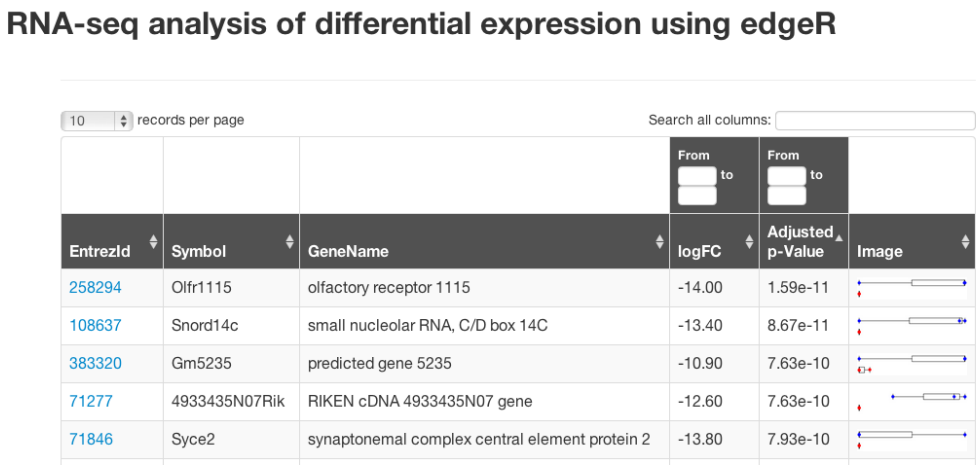

Figure 1: Resulting page created by `publish` for `edgeR.de`.

We can also ouput results of the LRT test from edgeR.

```
> d <- DGEList(counts = mockRnaSeqData, group = conditions)
> d <- calcNormFactors(d)
> design <- model.matrix(~conditions)
> d <- estimateGLMCommonDisp(d, design)
> d <- estimateGLMTrendedDisp(d, design)
> d <- estimateGLMTagwiseDisp(d, design)
> fit <- glmFit(d,design)
> edgeR.lrt <- glmLRT(fit, coef=2)
> deReport2 <- HTMLReport(shortName = 'RNAseq_analysis_with_edgeR_2',
+     title = 'RNA-seq analysis of differential expression using edgeR (LRT)',
+     reportDirectory = "./reports")
> publish(edgeR.lrt, deReport2, countTable=mockRnaSeqData,
+         conditions=conditions, annotation.db = 'org.Mm.eg',
+         pvalueCutoff = .05, lfc = 2, n = 10, name="edgeRlrt")
> finish(deReport2)
```

3 Differential expression analysis with DESeq

In this section we demonstrate how to use the `ReportingTools` package to generate a table of differentially expressed genes as determined by the `DESeq` package.

First, we run `DESeq` to find differentially expressed genes.

```
> library(DESeq)
> cds<-newCountDataSet(mockRnaSeqData, conditions)
> cds<-estimateSizeFactors(cds)
> cds<-estimateDispersions(cds)
> res<-nbinomTest(cds,"control", "case" )
```

Now the results can be written to a report after converting the `DESeq` output to a data frame. This is done using the `makeDESeqDF` command, which is a built-in function to convert DESeq differential expression output to a more meaningful data frame with plots, details about the genes, etc. With `ReportingTools` ,

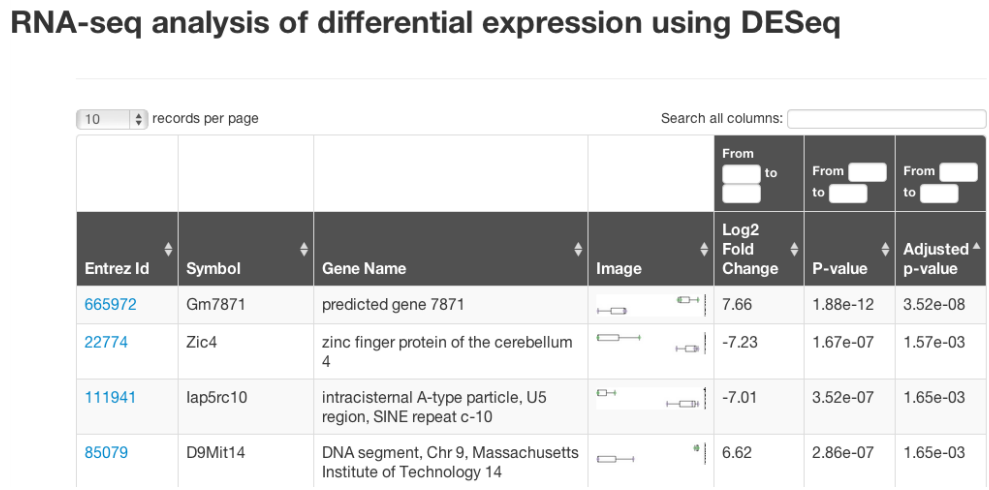

Figure 2: Resulting page created by `makeDESeqDF`

you can replace the `makeDESeqDf` with any function you like for more flexibility (see the basic vignette for more details and examples).

```
> desReport <- HTMLReport(shortName = 'RNAseq_analysis_with_DESeq',
+     title = 'RNA-seq analysis of differential expression using DESeq',
+     reportDirectory = "./reports")
> publish(res,desReport,name="df",countTable=mockRnaSeqData, pvalueCutoff=0.05, conditions=conditions,ann
> finish(desReport)
```

4 GO analysis using GOstats

This section will demonstrate how to use `ReportingTools` to write a table of GO analysis results to an html file. First we select our genes of interest, and then run the `hyperGTest`.

```
> library(GOstats)
> library(org.Mm.eg.db)
> tt <- topTags(edgeR.de, n = 1000, adjust.method = 'BH', sort.by = 'p.value')
> selectedIDs <- rownames(tt$table)
> universeIDs <- rownames(mockRnaSeqData)
> goParams <- new("GOHyperGParams",
+     geneIds = selectedIDs,
+     universeGeneIds = universeIDs,
+     annotation ="org.Mm.eg" ,
+     ontology = "MF",
+     pvalueCutoff = 0.01,
+     conditional = TRUE,
+     testDirection = "over")
> goResults <- hyperGTest(goParams)
```

With these results, we can then make the GO report.

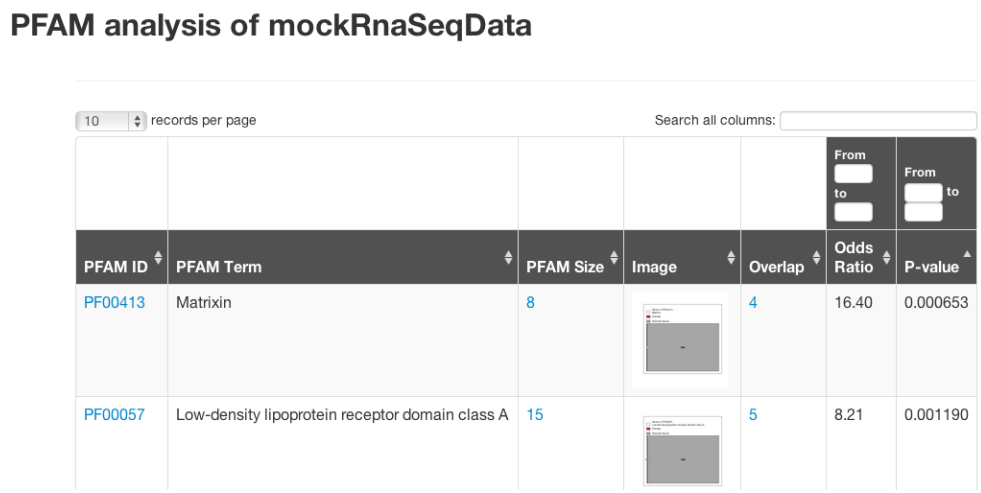

Figure 3: Resulting page created by `publish` for `PFAMResults`

```
> goReport <- HTMLReport(shortName = 'go_analysis_rnaseq',
+         title = "GO analysis of mockRnaSeqData",
+         reportDirectory = "./reports")
> publish(goResults, goReport, selectedIDs=selectedIDs, annotation.db="org.Mm.eg",
+         pvalueCutoff= 0.05)
> finish(goReport)
```

5 PFAM analysis

In this section, we show how to use `ReportingTools` to write a table of PFAM analysis results to an html file. First we run the `hyperGTest` using our genes of interest from the previous section.

```
> library(Category)
> params <- new("PFAMHyperGParams",
+         geneIds= selectedIDs,
+         universeGeneIds=universeIDs,
+         annotation="org.Mm.eg",
+         pvalueCutoff= 0.01,
+         testDirection="over")
> PFAMResults <- hyperGTest(params)
```

Then we make the PFAM report.

```
> PFAMReport <- HTMLReport(shortName = 'pfam_analysis_rnaseq',
+         title = "PFAM analysis of mockRnaSeqData",
+         reportDirectory = "./reports")
> publish(PFAMResults, PFAMReport, selectedIDs=selectedIDs, annotation.db="org.Mm.eg",categorySize=5)
> finish(PFAMReport)
```

6 Putting it all together

Here, we make an index page that puts all three analyses together for easy navigation.

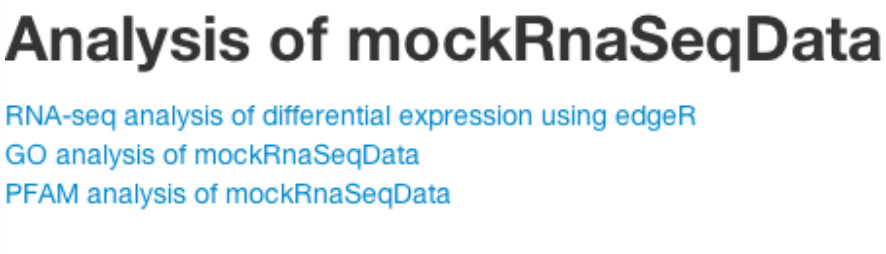

Figure 4: Resulting page created by calling `publish` on all our analysis pages

```
> indexPage <- HTMLReport(shortName = "indexRNASeq",
+     title = "Analysis of mockRnaSeqData",
+     reportDirectory = "./reports")
> publish(Link(list(deReport,goReport, PFAMReport), report = indexPage),
+     indexPage)
> finish(indexPage)
```