Using ReportingTools in an Analysis of Microarray Data

Jason A. Hackney and Jessica L. Larson

April 16, 2015

Contents

1   Introduction

The `ReportingTools` package is particularly useful in displaying results from a microarray experiment. In this vignette we show how to display results from differential gene expression, Gene Ontology (GO), protein families (PFAM) and gene set enrichment analyses. In the final section, we create an index page where the user can easily access any of these results.

2   Differential expression analysis using limma

For this vignette we will examine the `ALL` dataset. First we load our `ReportingTools` package and the data. This dataset is from a clinical trial in acute lymphoblastic leukemia (ALL) and is available from Bioconductor.

```
> library(ReportingTools)
> library(ALL)
> library(hgu95av2.db)
> library(genefilter)
> data(ALL)
```

We will compare the gene expression between the `BCR/ABL` and `NEG` samples. We use `featureFilter` to remove most of the unexpressed genes.

```
> ALL <- ALL[, ALL$mol.biol %in% c('NEG','BCR/ABL') &
+            !is.na(ALL$sex)]
> ALL$mol.biol <- factor(ALL$mol.biol,
+                        levels = c('NEG', 'BCR/ABL'))
> ALL <- featureFilter(ALL)
```

Next we use `limma` to find statistical evidence of differentially expressed genes.

```
> library(limma)
> model <- model.matrix(~mol.biol+sex, ALL)
> fit <- eBayes(lmFit(ALL, model))
```

With the `limma` output we can make our differential analysis report. To publish `MArrayLM` objects, we supply the `eSet` and `factor` used in our analysis.

```
> library(lattice)
> rep.theme <- reporting.theme()
> lattice.options(default.theme = rep.theme)
> deReport <- HTMLReport(shortName = 'de_analysis',
+                        title = 'Analysis of BCR/ABL translocation differential expression',
+                        reportDirectory = "./reports")
> publish(fit, deReport, eSet=ALL, factor=ALL$mol.biol, coef=2, n=100)
> finish(deReport)
```

The resulting output is displayed on an .html page and includes several statistics of interest as well as an image of the data.

If we want to change our images, we can do so with `.modifyDF` (see the basic vignette for more examples of how to use this feature). In this example, we make lattice plots of the expression of each gene in our table stratified by `mol.biol` and `sex` . Note that `.modifyDF` uses the basic data frame (output from `topTable`) as its default object and then modifies it with the corresponding function. To modify the decorated `ReportingToolsdata.frame` , let `.modifyDF=list(modifyReportDF, makeNewImages)` .

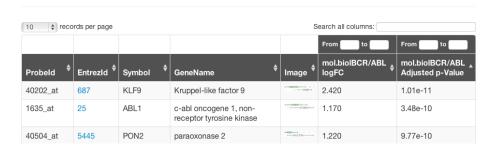## Analysis of BCR/ABL translocation differential expression

					mol.biolBCR/ABL logFC	mol.biolBCR/ABL Adjusted p-Value
ProbeId	**EntrezId**	**Symbol**	**GeneName**	**Image**		
40202_at	687	KLF9	Kruppel-like factor 9		2.420	1.01e-11
1635_at	25	ABL1	c-abl oncogene 1, non-receptor tyrosine kinase		1.170	3.48e-10
40504_at	5445	PON2	paraoxonase 2		1.220	9.77e-10

Figure 1: Resulting page created by `publish` for `fit` .

```
> library(hwriter)
> makeNewImages <- function(df,...){
+         imagename <- c()
+         for (i in 1:nrow(df)){
+                 probeId <- df$ProbeId[i]
+                 y_at <- pretty(exprs(ALL)[probeId,])
+                 y_labels <- formatC(y_at, digits = 1, format = 'f')
+                 imagename[i] <- paste0("plot", probeId, ".png")
+                 png(filename = paste0("./reports/figuresde_analysis/",
+                         imagename[i]))
+                 print(stripplot(exprs(ALL)[probeId,]~ALL$mol.biol|ALL$sex))
+                 dev.off()
+         }
+         df$Image <- hwriteImage(paste0("figuresde_analysis/", imagename),
+                             link=paste0("figuresde_analysis/", imagename),
+                             table=FALSE, width=100)
+         return(df)
+ }
> deReport2 <- HTMLReport(shortName='de_analysis2',
+                     title = 'Analysis of BCR/ABL translocation differential expression with new plo
+                     reportDirectory = "./reports")
> publish(fit, deReport2, eSet = ALL, factor = ALL$mol.biol, coef=2,
+         n=100,
+         ##.modifyDF=list(modifyReportDF, makeNewImages) ) ##to add new images to default RT output
+         .modifyDF=makeNewImages)
> finish(deReport2)
```

3   GO analysis using GOstats

In this section, we show how to use `ReportingTools` to `publish` a GO analysis to an html file. First we select the top 100 differential genes and then run the `hyperGTest` from the `GOstats` package.

```
> library(GOstats)
> tt <- topTable(fit, coef = 2, n = 100)
> selectedIDs <- unlist(mget(rownames(tt), hgu95av2ENTREZID))
```

3

Analysis of BCR/ABL translocation differential expression with new plots

					mol.bioBCR/ABL logFC	mol.bioBCR/ABL Adjusted p-Value
ProbeId	EntrezId	Symbol	GeneName	Image		
40202_at	687	KLF9	Kruppel-like factor 9		2.420	1.01e-11
1635_at	25	ABL1	c-abl oncogene 1, non-receptor tyrosine kinase		1.170	3.48e-10

Figure 2: Resulting page created by `makeNewImages`

```
> universeIDs <- unlist(mget(featureNames(ALL), hgu95av2ENTREZID))
> goParams <- new("GOHyperGParams",
+                 geneIds = selectedIDs,
+                 universeGeneIds = universeIDs,
+                 annotation = annotation(ALL),
+                 ontology = "BP",
+                 pvalueCutoff = 0.01,
+                 conditional = TRUE,
+                 testDirection = "over")
> goResults <- hyperGTest(goParams)
```

With these results, we can then make the GO report. We must supply `publish` with the genes of interest and the species annotation for this dataset. The default p-value cutoff is 0.01 and the minimum category size is 10 genes.

```
> goReport <- HTMLReport(shortName = 'go_analysis',
+                        title = 'GO analysis of BCR/ABL translocation',
+                        reportDirectory = "./reports")
> publish(goResults, goReport, selectedIDs = selectedIDs,
+     annotation.db = "org.Hs.eg")
> finish(goReport)
```

The resulting output is displayed on an .html page and includes several statistics of interest as well as an image of the overlap for each category.

4   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` from the `Category` package.

```
> library(Category)
> pfamParams <- new("PFAMHyperGParams",
+                 geneIds = selectedIDs,
+                 universeGeneIds = universeIDs,
+                 annotation = annotation(ALL),
```

4

GO analysis of BCR/ABL translocation

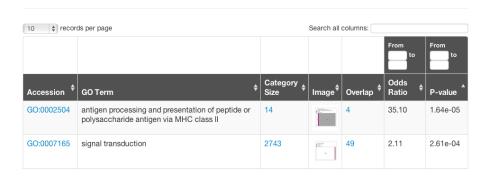| Accession	GO Term	Category Size	Image	Overlap	Odds Ratio	P-value
GO:0002504	antigen processing and presentation of peptide or polysaccharide antigen via MHC class II	14		4	35.10	1.64e-05
GO:0007165	signal transduction	2743		49	2.11	2.61e-04

Figure 3: Resulting page created by `publish` for `goResults`.

```
+                        pvalueCutoff = 0.01,
+                        testDirection = "over")
> PFAMResults <- hyperGTest(pfamParams)
```

Then we make the PFAM report. Again we supply `publish` with the genes of interest and the species annotation for this dataset. We set the minimum category size to 3 genes.

```
> PFAMReport <- HTMLReport(shortName = 'pfam_analysis',
+                          title = 'PFAM analysis of BCR/ABL translocation',
+                          reportDirectory = "./reports")
> publish(PFAMResults, PFAMReport, selectedIDs = selectedIDs,
+         annotation.db = "org.Hs.eg", categorySize = 3)
> finish(PFAMReport)
```

The resulting output is displayed on an .html page and includes several statistics of interest as well as an image of the overlap for each category.

5   GSEA analysis

In this section we show how to use `publish` to display `GeneSetCollection` objects and their corresponding gene set enrichment statistics. For this example, we will randomly select our gene sets and create our collection.

```
> library(GSEAlm)
> library(GSEABase)
> mapped_genes <- mappedkeys(org.Hs.egSYMBOL)
> eidsAndSymbols <- as.list(org.Hs.egSYMBOL[mapped_genes])
> geneEids <- names(eidsAndSymbols)
> set.seed(123)
> set1 <- GeneSet(geneIds=sample(geneEids, 100, replace=FALSE), setName="set1",
+                 shortDescription = "This is set1")
> set2 <- GeneSet(geneIds=sample(geneEids, 10, replace=FALSE), setName="set2",
+                 shortDescription = "This is set2")
> set3 <- GeneSet(geneIds=sample(geneEids, 37, replace=FALSE), setName="set3",
+                 shortDescription = "This is set3")
```

```
> set4 <- GeneSet(geneIds=sample(geneEids, 300, replace=FALSE), setName="set4",
+                 shortDescription = "This is set4")
> geneSets <- GeneSetCollection(c(set1, set2, set3, set4))
```

We can now make a very simple `GeneSetCollection` html table with `ReportingTools` .

```
> geneSetsReport <- HTMLReport(shortName = "gene_sets",
+                              title = "Gene Sets",
+                              reportDirectory = "./reports")
> publish(geneSets, geneSetsReport, annotation.db = "org.Hs.eg")
> finish(geneSetsReport)
```

The resulting output is displayed on an .html page and includes the gene sets and links to pages listing the genes within the corresponding set.

Often, investigators would like more information about the enrichment of certain gene sets. Thus, we will proceed with gene set enrichment analysis (GSEA). To begin, we determine the overlap between our sets and our genes of interest by creating an incidence matrix.

```
> mat <- matrix(data=0, ncol=length(universeIDs),nrow=length(geneSets))
> for(i in 1:length(geneSets)){
+   geneIdEntrez <- unlist(geneIds(geneSets[[i]]))
+   mat[i,match(geneIdEntrez, universeIDs)] <- 1
+ }
> colnames(mat) <- universeIDs
> rownames(mat) <- sapply(geneSets, function(x) x@setName)
```

Now we can run the GSEA and obtain set-specific statistics and p-values.

```
> lm <- lmPerGene(ALL, ~mol.biol+sex, na.rm=TRUE)
> GSNorm <- GSNormalize(lm$tstat[2,], mat)
> #one-sided p-values
> pVals <- gsealmPerm(ALL,~mol.biol+sex, mat, nperm=100)
> bestPval <- apply(pVals, 1, min)
```

We can add these statistics to our report page.

```
> gseaReport <- HTMLReport(shortName = "gsea_analysis",
+                          title = "GSEA analysis",
+                          reportDirectory = "./reports")
> publish(geneSets, gseaReport, annotation.db = "org.Hs.eg",
+         setStats = GSNorm, setPValues = 2*bestPval)
> finish(gseaReport)
```

The resulting output is displayed on an .html page and includes our set statistics and p-values. Links to set-specific pages are also created.

We can also add the same statistics via `.modifyDF` . As demonstrated in the basic vignette, .modifyDF allows us to manipulate the output published to our html pages.

```
> runGSEA <- function(df,...){
+   mat <- matrix(data = 0, ncol = length(universeIDs), nrow = length(geneSets))
+   for(i in 1:length(geneSets)){
+     geneIdEntrez <- unlist(geneIds(geneSets[[i]]))
+     mat[i,match(geneIdEntrez, universeIDs)] <- 1
+   }
```

GSEA analysis

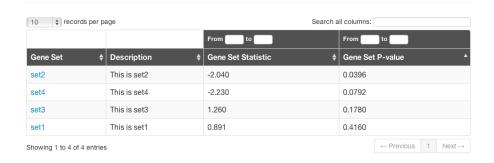| Gene Set	Description	Gene Set Statistic	Gene Set P-value
set2	This is set2	-2.040	0.0396
set4	This is set4	-2.230	0.0792
set3	This is set3	1.260	0.1780
set1	This is set1	0.891	0.4160

Showing 1 to 4 of 4 entries

Figure 4: Resulting updated page created by `publish` for `geneSets` after we include the set statistics and p-values.

Genes in set1 -- This is set1

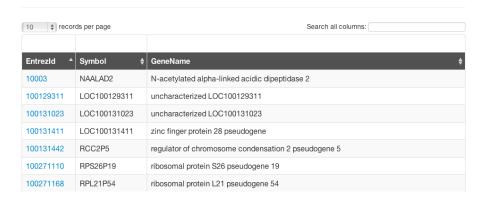| EntrezId	Symbol	GeneName
10003	NAALAD2	N-acetylated alpha-linked acidic dipeptidase 2
100129311	LOC100129311	uncharacterized LOC100129311
100131023	LOC100131023	uncharacterized LOC100131023
100131411	LOC100131411	zinc finger protein 28 pseudogene
100131442	RCC2P5	regulator of chromosome condensation 2 pseudogene 5
100271110	RPS26P19	ribosomal protein S26 pseudogene 19
100271168	RPL21P54	ribosomal protein L21 pseudogene 54

Figure 5: The set-specific page.

Analysis of ALL Gene Expression

Analysis of BCR/ABL translocation differential expression
GO analysis of BCR/ABL translocation
PFAM analysis of BCR/ABL translocation
GSEA report has a new title

Figure 6: The page created from calling `publish` on all of our previous pages.

```
+    colnames(mat) <- universeIDs
+    rownames(mat) <- sapply(geneSets, function(x) x@setName)
+    lm <- lmPerGene(ALL, ~mol.biol+sex, na.rm=TRUE)
+    GSNorm <- GSNormalize(lm$tstat[2,], mat)
+    pVals <- gsealmPerm(ALL,~mol.biol+sex, mat, nperm = 100)
+    bestPval <- apply(pVals,1, min)
+    df <- cbind(df, GSNorm, bestPval)
+    return(df)
+ }
> gseaReport2 <- HTMLReport(shortName = "gsea_analysis2",
+                           title = "GSEA analysis",
+                           reportDirectory = "./reports")
> publish(geneSets, gseaReport2, annotation.db = "org.Hs.eg",
+         .modifyDF = runGSEA)
> finish(gseaReport2)
```

6 Putting it all together

We now make an index page to put all the output together.

```
> indexPage <- HTMLReport(shortName = "index",
+                         title = "Analysis of ALL Gene Expression",
+                         reportDirectory = "./reports")
> publish(Link(list(deReport, goReport), report = indexPage), indexPage)
> publish(Link(PFAMReport, report = indexPage), indexPage)
> publish(Link("GSEA report has a new title", "gsea_analysis.html"), indexPage)
> finish(indexPage)
```

7 References

Huntley, M.A., Larson, J.L., Chaivorapol, C., Becker, G., Lawrence, M., Hackney, J.A., and J.S. Kaminker. (2013). ReportingTools: an automated results processing and presentation toolkit for high throughput genomic analyses. *Bioinformatics.* **29**(24): 3220-3221.