Gene Set Enrichment Analysis

R. Gentleman and Seth Falcon

January 10, 2007

1 Introduction

In this case study we will see how to use gene set enrichment analysis (**??**). We will make use of the data from an investigation into acute lymphoblastic leukemia (ALL) reported in **?** for our examples. We will primarily concentrate on the two sample problem, where the data can be divided into two distinct groups, and we want to understand differences in gene expression between the two groups. For the ALL data we will compare those samples with BCR/ABL to those that have no observed cytogenetic abnormalities (NEG).

The basic idea behind gene set enrichment analysis is that we want to use predefined sets of genes, perhaps based on function, in order to better interpret the observed gene expression data. In some ways the ideas here are quite similar to those that the usual Hypergeomtric testing is based on.

Preprocessing

We begin by loading up the appropriate libraries.

```
> library("Biobase")
> library("annotate")
> library("Category")
> library("hgu95av2")
> library("genefilter")
```

As for all analyses, we must first load the data and process it. In the code chunk below we load the data and then select the subset we are interested in.

Exercise 1
How many samples are in our subset? How many are BCR/ABL and how many NEG?

Next, we remove from our data set those probes that have no mapping to EntrezGene, since we will not be able to find any metadata for these probes.

```
> entrezIds <- mget(featureNames(eset), envir=hgu95av2ENTREZID)
> haveEntrezId <- names(entrezIds)[sapply(entrezIds, function(x) !is.na(x))]
> numNoEntrezId <- length(featureNames(eset)) - length(haveEntrezId)
> eset <- eset[haveEntrezId, ]
```

Next we do some basic prefiltering. My preference is to filter genes according to their variability across samples. In the code below we compute the IQR (approximately) and then select for our gene set those genes that have an IQR above the median value.

```
> ## Non-specific filtering based on IQR
> lowQ = rowQ(eset, floor(0.25 * numBN))
> upQ = rowQ(eset, ceiling(0.75 * numBN))
> iqrs = upQ - lowQ
> selected <- iqrs > median(iqrs)
> nsFiltered <- eset[selected, ]
```

In the next code chunk, we find all probes that map to a single gene. We want only one probe set to represent each gene (otherwise we have to do a lot of downstream adjustments) and our decision here is to chose the one that shows the most variation, as measured by the IQR, across samples.

```
> ## Reduce to unique probe <--> gene mapping by keeping largest IQR
> ## This gives us "unique genes" in the non-specific filtered gene
> ## set which simplifies further calculations.
> nsFilteredIqr <- iqrs[selected]
> uniqGenes <- findLargest(featureNames(nsFiltered), nsFilteredIqr, "hgu95av2")
> nsFiltered <- nsFiltered[uniqGenes, ]
> ## basic stats on our non-specific filter result
> numSelected <- length(featureNames(nsFiltered))
> numBcrAbl <- sum(nsFiltered$mol.biol == "BCR/ABL")
> numNeg <- sum(nsFiltered$mol.biol == "NEG")
```

Exercise 2
How many genes have been selected for our analysis?

Since we have done this selection without regard to phenotype, or the type of gene sets we are going to use we will be able to use this same subset of the data in all examples.

Using KEGG

We now want to use KEGG to assign genes to pathways, and to then use those pathways for our gene sets.

```
> ## Remove genes with no PATH mapping
> havePATH <- sapply(mget(featureNames(nsFiltered), hgu95av2PATH),
+                  function(x) if (length(x) == 1 && is.na(x)) FALSE else TRUE)
> numNoPATH<- sum(!havePATH)
> nsF <- nsFiltered[havePATH, ]
```

Exercise 3
How many genes are we left with?

Now we must compute the incidence matrix, that is the matrix that maps between probes and the pathways. This matrix has zero's and ones in it. The last two commands rearrange the rows of the A matrix so that they are in the same order as the genes in our *exprSet* object.

```
> Am = PWAmat("hgu95av2")
> egN = unlist(mget(featureNames(nsF), hgu95av2ENTREZID))
> sub1 = match(egN, row.names(Am))
> Am = Am[sub1,]
> dim(Am)
```

```
[1] 1661  181
```

```
> table(colSums(Am))
```

```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  15  16  17  18  19  20
  4  11   7   6   6   6  11   8   7   5   5   4   5   9   3   7   4   3   4   2
 21  22  23  24  25  26  27  28  29  30  31  32  34  37  39  41  42  44  46  47
  3   1   2   4   1   1   4   2   2   3   1   2   2   1   3   1   3   1   2   1
 48  51  52  54  55  56  59  65  68  69  70  72  76  80  83  84  95 103 148
  2   1   3   1   1   2   1   1   2   1   1   1   1   1   1   1   1   1   1
```

Exercise 4
How many categories and how many genes are represented by the A matrix? How many categories have fewer than 10 genes in them? What is the largest number of categories a gene is in?

Next we will compute the per gene test statistics using the `rowttests` function from the genefilter package. There are several other fast test statistic computations that you can do as well (e.g. `rowFtests`).

```
> rtt = rowttests(nsF, "mol.biol")
> rttStat = rtt$statistic
```

Exercise 5
How many test statistics are positive? How many are negative? How many have a p-value less than 0.01*?*

Next we further reduce the A matrix, by removing all categories that have fewer than 10 genes in them. When carrying out your own analyses you should select a value you are comfortable with.

```
> Amat = t(Am)
> rs = rowSums(Amat)
> Amat2 = Amat[rs>10,]
> rs2 = rs[rs>10]
> nCats = length(rs2)
```

And now it is fairly easy to compute the per category test statistics and to produce a qq-plot, Figure 1. We see that there is one pathway that has a remarkably low observed value (less than -5) so we will take a closer look at this pathway.

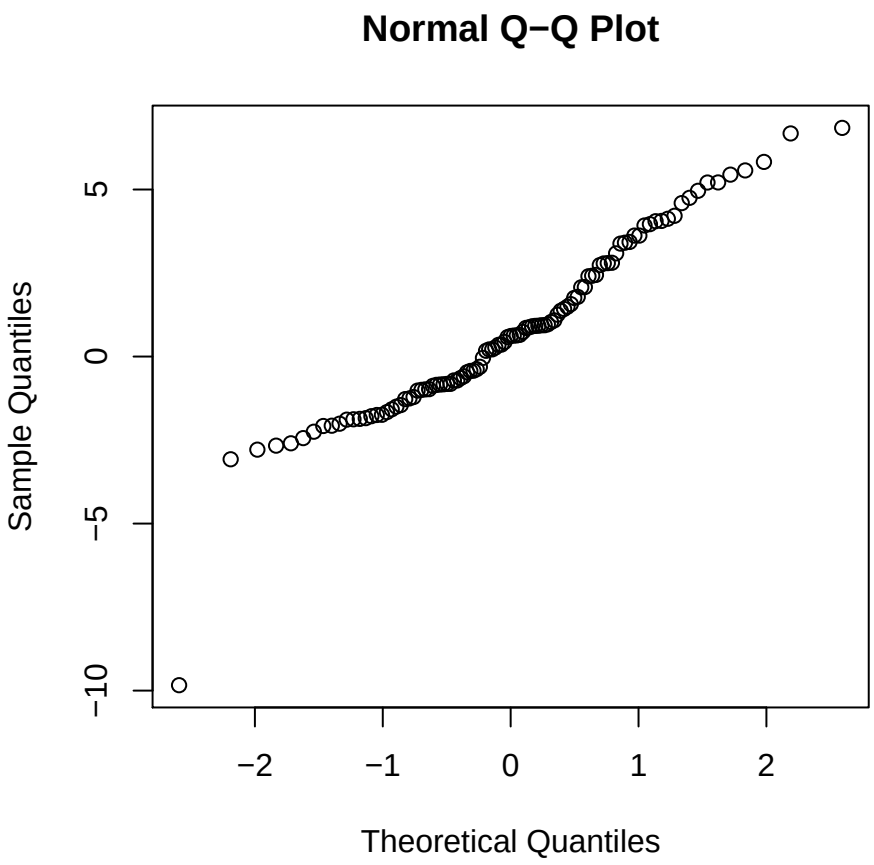

Figure 1: The per category qq-plot.

```
> tA = as.vector(Amat2 %*% rttStat)
> tAadj = tA/sqrt(rs2)
> names(tA) = names(tAadj) = row.names(Amat2)
```

To find the pathway, we first find the value, and then use

```
> smPW = tAadj[tAadj < -5]
> pwName = KEGGPATHID2NAME[[names(smPW)]]
> pwName
```

```
[1] "Ribosome"
```

Now we can produce some summary plots based on the genes annotated at this pathway. The mean plot presents a comparison of the average expression value for each of our two groups, for each gene in the specified pathway. That is, each point in Figure 2 represents one gene and the value on the x-axis is the mean in the BCR/ABL samples while the value on the y-axis is the mean value in the NEG samples.

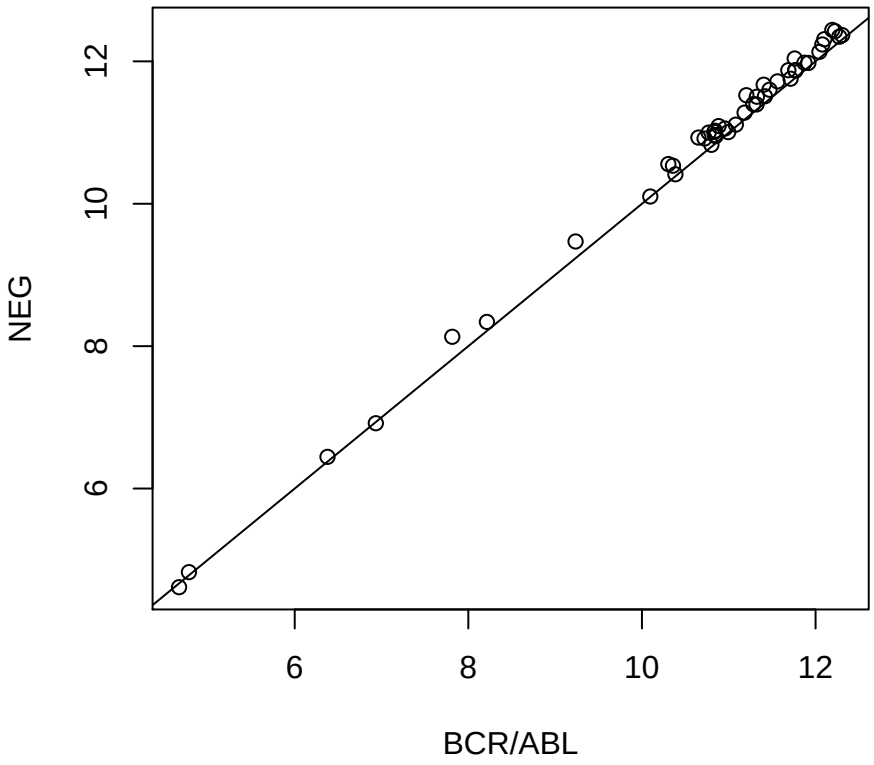

Figure 2: The mean plot for the Ribosome pathway.

Exercise 6
What do you notice in this plot?

And finally we can produce a heatmap, Figure 3, for the genes in the Ribosome pathway. We use the `KEGG2heatmap` function to do most of the hard work.

Exercise 7
What sorts of things do you notice in the heatmap? The gene labeled `41214_at` *has a very unusual pattern of expression. Can you guess what is happening? Hint: look at which chromosome it is on.*

Exercise 8
Repeat these plots for the pathway with the largest observed average t-statistic.

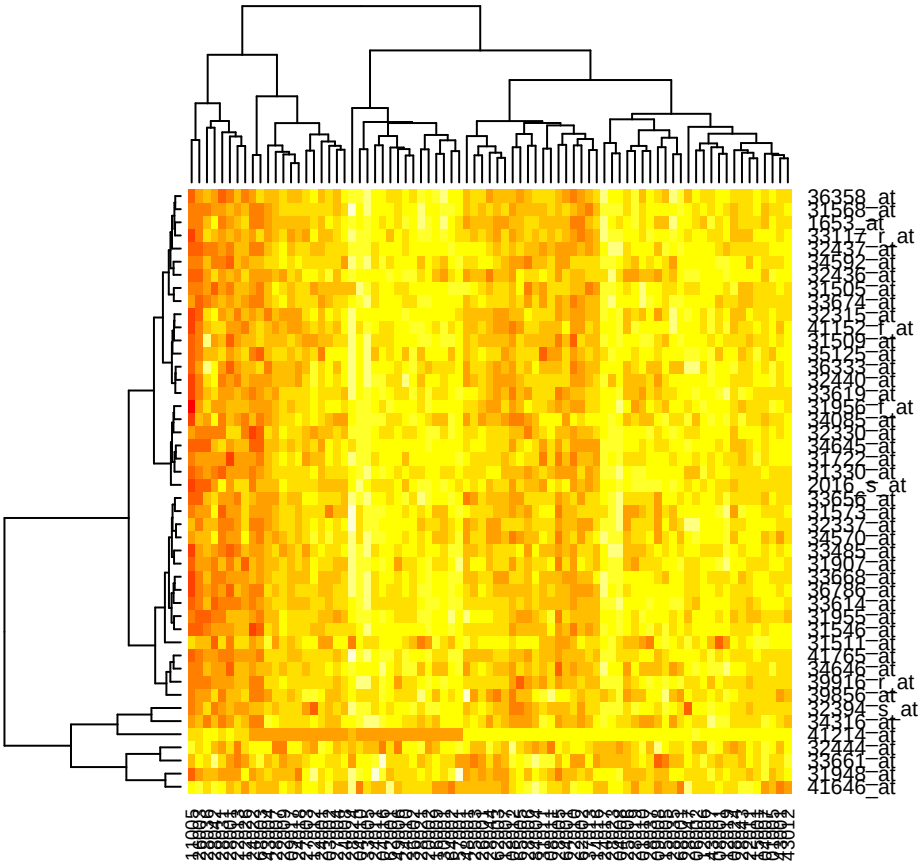

Figure 3: A heatmap for the Ribosome pathway.

Permutation testing

The analysis above was based on a presumption that the data are approximately Normally distributed. However, this is sometimes viewed as a relatively strong assumption and there is some interest in using another approach. For GSEA it is relatively straightforward to compute a permutation t-test. The `ttperm` in the Category package can be used for this purpose.

In the next code chunk we compute the permutation distribution for this same problem. The value returned by `ttperm` is a list, the first entry is the observed t-statistic (the return value of a call to `rowttests`) while the second element is itself a list of length B, the number of permutations.

In the code chunk below we compute the permutation distribution based on 100 permutations, in practice you would typically use a much larger value.

```
> NPERM = 100
> ttp = ttperm(exprs(nsF), nsF$mol.biol, NPERM)
> permDm = do.call("cbind", lapply(ttp$perms, function(x) x$statistic))
> permD = Amat2 %*% permDm
> pvals = matrix(NA, nr=nCats, ncol=2)
> dimnames(pvals) = list(row.names(Amat2), c("Lower", "Upper"))
> for(i in 1:nCats) {
+     pvals[i,1] = sum(permD[i,] < tA[i])/NPERM
+     pvals[i,2] = sum(permD[i,] > tA[i])/NPERM
+ }
> ord1 = order(pvals[,1])
> lowC = (row.names(pvals)[ord1])[pvals[ord1,1]< 0.05]
> highC = row.names(pvals)[pvals[,2] < 0.05]
```

In the next code chunk we print out the pathway names for all the ones that have low values, but only a few of those that have high values, since there are 24 of them.

```
>  getPathNames(lowC)

$`03010`
[1] "Ribosome"

$`03320`
[1] "PPAR signaling pathway"

$`01030`
[1] "Glycan structures - biosynthesis 1"

$`05030`
[1] "Amyotrophic lateral sclerosis (ALS)"

$`00260`
[1] "Glycine, serine and threonine metabolism"
```

```
> getPathNames(highC)[1:5]

$`04510`
[1] "Focal adhesion"

$`04512`
[1] "ECM-receptor interaction"

$`04514`
[1] "Cell adhesion molecules (CAMs)"

$`04520`
[1] "Adherens junction"

$`04670`
[1] "Leukocyte transendothelial migration"

> lnhC = length(highC)
```

Exercise 9
How many pathways have low t-statistics (those where the observed t-statistic was lower than that for the permutation distribution)? How many have high? How do you interpret these? What p-value is the most extreme? What does the heatmap look like for this gene set?

Chromosome Bands

Another interesting application is to use chromosome band information as the gene set. In doing this, you are studying whether there are anomolies in the pattern of gene expression that relate to chromosomal location. Unfortunately this analysis is incomplete in the sense that we have not adjusted for nesting of bands, but software for doing this will be available in the 1.9 release of Bioconductor.

The chromosome band information is contained in the metadata variable with the suffix `MAP`, so for us it is in `hgu95av2MAP`.

Exercise 10
What does the manual page say the interpretation of the MAP position `17p33.2` *is?*

A preliminary approach to finding relevant MAP positions is available in the `MAPAmat` function from the Category package (note this function will be improved over the next few months). Since we want to focus on relatively few categories we are going to restrict attention to MAP locations with at least 10 genes.

```
> AmChr = MAPAmat("hgu95av2", minCount=5)
```

Exercise 11
How many genes were selected? How many map positions?

Next we must reduce this matrix to those genes we have selected for analysis. So in the next code chunk we repeat the steps performed above, for KEGG, to match row names to those selected and then to reduce the incidence matrix accordingly.

```
> subC = row.names(AmChr) %in% egN
> AmChr = AmChr[subC,]
> dim(AmChr)

[1] 1147  564

> table(colSums(AmChr))

  0   1   2   3   4   5   6   7   8   9  10  11  13  19  24  31
134 173 115  48  38  21   9   5   3   5   5   2   3   1   1   1
```

Exercise 12
Further reduce `AmChr` *so that only bands with at least 5 genes are retained. Produce a qq-plot and identify the interesting bands. Use Google, or some other search engine to determine whether others have identified these bands as interesting.*

Repeat the analysis performed on the KEGG pathways. Produce mean plots, heatmaps etc. Try to identify a set of interesting bands.

The version number of R and the packages and their versions that were used to generate this document are listed below.

```
R version 2.5.0 Under development (unstable) (2006-11-27 r40032)
i386-apple-darwin8.8.1

locale:
C

attached base packages:
[1] "splines"   "tools"     "stats"     "graphics"  "grDevices" "utils"
[7] "datasets"  "methods"   "base"

other attached packages:
       ALL   hgu95av2   Category genefilter   survival      graph         GO
   "1.4.1"   "1.15.0"    "2.1.8"   "1.13.7"     "2.30"   "1.13.2"   "1.15.1"
      KEGG   annotate    Biobase
  "1.15.1"   "1.13.3"  "1.13.30"
```