Microarray data analysis using R/Bioconductor

Ståle Nygård
Bioinformatics core facility, OUS/UiO
stale.nygard@medisin.uio.no

September 08, 2010

Material partly adapted from Anja von Heydelbreck.

Contents

1 Before you start

...you need to install a few libraries from Bioconductor. Type the following commands

```
source("http://bioconductor.org/biocLite.R")
biocLite(c("affy","ALL","annotate","hgu95av2.db","multtest","topGO"))
```

This will take a few minutes.

2 Reading in data and pre-processing

The data used for this exercise come from a study of Chiaretti et al. (Blood 103:2771-8, 2004) on acute lymphoblastic leukemia (ALL), which was conducted with HG-U95Av2 Affymetrix arrays. The data package *ALL* contains an *ExpressionSet* object called `ALL`, which contains the expression data that were normalized with `rma` (intensities are on the log2scale), and annotations of the samples.

Load the ALL package. What is the dimension of the expression data matrix?

```
library(ALL)
data(ALL)
dim(exprs(ALL))
```

For this exercise you can now go directly to the next section: If you need to do the pre-processing yourself, a quick way of doing this is the following:

1. Create a directory, move all the relevant CEL files to that directory.
2. Start R in that directory.
3. If using the Rgui for Microsoft Windows make sure your working directory contains the CEL files (use ”File − > Change Dir” menu item).
4. Pre-process the data using the method gcrma.

   ```
   library (gcrma) #load the gcrma package
   ```

5. Make a tab-delimitted text file `target.txt` with the first column having the names of the CEL files, and the next columns describing the samples on each file (e.g. treatment, strain, sex etc.).
6. Read in the data and create an expression set.

   ```
   pd <- read.AnnotatedDataFrame("target.txt",header=TRUE)
   Data <- ReadAffy(filenames=rownames(pData(pd)),phenoData=pd)
   eset <- gcrma(Data)
   ```

3 Looking at a subset of samples

We want to look at the B-cell ALL samples (they can be identified by the column BT of the *ExpressionSet* `ALL`). Of particular interest is the comparison of samples with the BCR/ABL fusion gene resulting from a translocation of the chromosomes 9 and 22 (labelled BCR/ABL in the column mol.biol of the

ExpressionSet `ALL`), with samples that are cytogenetically normal (labelled NEG).

Define an *ExpressionSet* object containing only the data from the B-cell ALL samples. How many samples belong to the cytogenetically defined groups?

```
table(ALL$BT)
table(ALL$mol.biol)
subset <- intersect(grep("^B", as.character(ALL$BT)),
which(as.character(ALL$mol.biol) %in% c("BCR/ABL", "NEG")))
eset <- ALL[, subset]
eset$mol.biol <- factor(eset$mol.biol)
table(eset$mol.biol)
```

4 Data filtering

Many of the genes on the chip wont be expressed in the B-cell lymphocytes studied here, or might have only small variability across the samples. We try to remove these genes (more precisely: the corresponding probe sets) with an intensity filter (the intensity of a gene should be above 100 in at least 25 percent of the samples), and a variance filter (the interquartile range of log2-intensities should be at least 0.5). We create a new *ExpressionSet* containing only the probe sets which passed our filter. How many probe sets do we get?

```
library(genefilter)
f1 <- pOverA(0.25, log2(100))
f2 <- function(x) (IQR(x) > 0.5)
ff <- filterfun(f1, f2)
selected <- genefilter(eset, ff)
sum(selected)
esetSub <- eset[selected, ]
```

5 Differential expression

Now we are ready to examine the selected genes for differential expression between the BCR/ABL samples and the cytogenetically normal ones.

We use the two-sample t-test to identify genes that are differentially expressed between the two groups. The function `mt.teststat` from the *multtest* package allows to compute several commonly used test statistics for all rows of a data matrix (study its help page). First, we calculate the nominal p-values. The function `pt` gives the distribution function of the t-

distribution. We can get an impression of the amount of differential gene expression by looking at a histogram of the p-value distribution.

```
library(multtest)
cl <- as.numeric(esetSub$mol.biol == "BCR/ABL")
t <- mt.teststat(exprs(esetSub), classlabel = cl, test = "t.equalvar")
pt <- 2 * pt(-abs(t), df = ncol(exprs(esetSub)) - 2)
hist(pt, 50)
```

The function `mt.rawp2adjp` from the *multtest* package contains different multiple testing procedures. Look at the help page of this function. For p-value adjustment in terms of the FDR, we use the method of Benjamini and Hochberg. How many genes do you get when imposing an FDR of 0.1?

```
pAdjusted <- mt.rawp2adjp(pt, proc = c("BH"))
sum(pAdjusted$adjp[, "BH"] < 0.1)
```

Also, this function returns the adjusted p-values ordered from the smallest to the largest. To obtain the original ordering, we do:

```
pBH <- pAdjusted$adjp[order(pAdjusted$index), "BH"]
names(pBH)<-featureNames(esetSub)
```

6 Annotation

Now we want to see which genes are the most significant ones, and look at their raw and adjusted p-values from the different methods. Gene symbols are provided in the annotation package hgu95av2.

```
library(annotate)
library(hgu95av2.db)
diff <- pAdjusted$index[1:10]
genesymbolsDiff <- unlist(mget(featureNames(esetSub)[diff], hgu95av2SYMBOL))
genesymbolsDiff
```

The top 3 probe sets represent the ABL1 gene, which is affected by the translocation characterizing the BCR/ABL samples. Now we want to see whether there are further probe sets representing this gene, and whether these have been selected by our non-specific filtering.

```
geneSymbols = unlist(mget(featureNames(ALL), hgu95av2SYMBOL))
ABL1probes <- which(geneSymbols == "ABL1")
selected[ABL1probes]
```

So the other probe sets representing ABL1 have been filtered out because of low intensities or low variance. Now we want to see whether they also indicate differential expression of the ABL1 gene.

```
tABL1 <- mt.teststat(exprs(eset)[ABL1probes, ], classlabel = cl,
test = "t.equalvar")
ptABL1 <- 2 * pt(-abs(tABL1), df = ncol(exprs(esetSub)) - 2)
sort(ptABL1)
```

We see that only three out of the six ABL1 probe sets show evidence (in fact, very strong evidence) for differential expression! It might be interesting to further investigate the ABL1 probe sets regarding e.g. their location in the ABL1 transcript sequence - indeed the BCR/ABL fusion gene resulting from the translocation differs from the normal ABL1 gene.

7 Using Gene Ontology

Many of the effects due to the BCR/ABL translocation are mediated by tyrosine kinase activity. Lets look at the probe sets that are annotated at the GO term protein-tyrosine kinase activity, which has the identifier GO:0004713.

```
gN <- featureNames(esetSub)
tykin <- unique(unlist(lookUp("GO:0004713", "hgu95av2", "GO2ALLPROBES")))
str(tykin)
sel <- (gN %in% tykin)
```

We can now check whether there are more differentially expressed genes among the tyrosine kinases than among the other genes. Fishers exact test for contingency tables is used to check whether the proportions of differentially expressed genes are significantly different in the two gene groups.

```
tab <- table(pt < 0.05, sel, dnn = c("p < 0.05", "tykin"))
print(tab)
fisher.test(tab)
```

7.1 Gene set enrichment analysis using topGO

We now want to look for enrichment in differential expression in all GO categories, not just protein-tyrosine kinase activity. The `topGO` package is a nice tool for gene set enrichment analysis based on GO.

```
library(topGO) #load the package
```

We will need a function which selects differentially expressed genes. We define the following function:

```
topDiffGenes <- function(allScore) return(allScore < 0.05)
```

We are then ready to make a topGO object. We will look at the molecular function (MF) ontology.

```
GOdataMF <- new("topGOdata", ontology = "MF",
allGenes = pBH, geneSel = topDiffGenes, annot = annFUN.db,
affyLib = "hgu95av2.db")
```

We can then run a classic Fisher test on all biological processes:

```
resultFisher <- runTest(GOdataMF, algorithm = "classic", statistic = "fisher")
```

or using some other methods (see Alexa et al, Bioinformatics, 2006, 22(13):1600-1607):

```
resultKS<- runTest(GOdataMF, algorithm = "classic", statistic = "ks")
resultFisher.elim<- runTest(GOdataMF, algorithm = "elim", statistic = "fisher")
resultKS.elim<- runTest(GOdataMF, algorithm = "elim", statistic = "ks")
```

To display the results:

```
allRes <- GenTable(GOdataMF, classicFisher = resultFisher,
classicKS = resultKS,elimFisher = resultFisher.elim, elimKS = resultKS.elim,
orderBy = "elimKS", ranksOf = "classicFisher", topNodes = 20)
allRes
```

We want to look up the genes in one of the GO categories. The function `printGenes` is made for this task. Let's take protein tyrosine kinase activity (GO:0004713) as an example.

```
gt <- printGenes(GOdataMF, whichTerms = "GO:0004713", chip =  "hgu95av2.db",
numChar = 40)
gt
```

To read more about the topGO package, go to
`http://www.bioconductor.org/packages/2.6/bioc/html/topGO.html`