Exploring cDNA data

Course in Practical Microarray Analysis, Heidelberg, October 2003

Wolfgang Huber and Andreas Buness

The following exercise will show you some possibilities to load data from spotted cDNA microarrays into R, and to explore it using the statistical and visualization facilities of R and Bioconductor.

1.) Preliminaries. To go through this exercise, you need to have installed R 1.7.1, recent versions of the Bioconductor libraries Biobase, vsn, multtest, marrayInput, marrayNorm, limma and the library lymphoma, which contains the excercises and the lymphoma data set (see http://llmpp.nih.gov/lymphoma/), as well as the library arrayMagic, which contains utility functions.

```
> library(vsn)
> library(multtest)
> library(marrayInput)
> library(marrayNorm)
> library(limma)
> library(lymphoma)
> library(arrayMagic)
```

2.) Reading and exploring data files.

a. First, you need to find the directory with the data on your harddisk. The path ends in `library/lymphoma/data`, and this is in the subdirectory where your `R` resides. On the command line, you can use the commands `dir()`, `getwd` and `setwd` to navigate around. In the GUI, you can use *File, Change dir* in the menu.

b. Open the file `lc7b048rex.DAT` in a text editor. This is the typical file format for the results from the image analysis on a cDNA slide. Different image analysis programs use slightly different conventions and column headings, but you can always adapt the input function (see below) to your needs.

c. Use the function `read.delim` to read the file into a *data frame* (that is a rectangular table of data) in R.

```
> x = read.delim("lc7b048rex.DAT")
```

d. The table is too large to print it out as a whole, but we can find out about its size (with the function `dim`) and look at individual rows of the table.

```
> dim(x)
> colnames(x)
> x[1:6, ]
```

3.) Simple plots.

a. Let us first look at the histogram of the values in the column CH1I, that is the channel 1 foreground intensity (see Fig. 1).

```
> hist(x$CH1I)
> hist(log2(x$CH1I), breaks = 100)
> hist(log2(x$CH1I), breaks = seq(5, 15, by = 0.25), col = "blue")
```

b. Visualization of the spatial homogeneity of the hybridisations may help to assess their quality. The logarithm of the background of channel 1 is shown in Fig. 2. Various transformations of the background intensity values, like the logarithm or rank, emphasize inhomogeneities more or less. Additionally, the foreground of channel 1 is visualized.

```
> bg1 = spatialLayout(value = x$CH1B, row = x$ROW, col = x$COL,
+     block = x$GRID)
> reverse = ncol(bg1):1
> plotDataMatrix(log(bg1[reverse, ]), yLabels = reverse, xLabels = 1:ncol(bg1),
+     title = "Log Background of Channel 1")
```

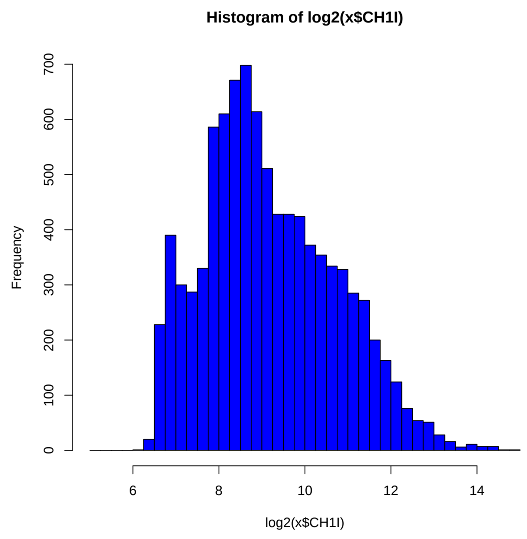


Figure 1:

```
> plotDataMatrix(bg1[reverse, ], yLabels = reverse, xLabels = 1:ncol(bg1))
> rankedbg1 = spatialLayout(value = rank(x$CH1B), row = x$ROW,
+     col = x$COL, block = x$GRID)
> plotDataMatrix(rankedbg1[reverse, ], yLabels = reverse, xLabels = 1:ncol(bg1))
> fg1 = spatialLayout(value = x$CH1I, row = x$ROW, col = x$COL,
+     block = x$GRID)
> plotDataMatrix(log(fg1[reverse, ]), yLabels = reverse, xLabels = 1:ncol(bg1))
```

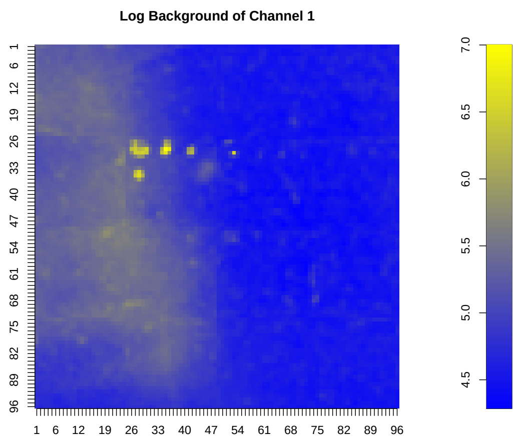


Figure 2:

c. Save one of the plots as PDF, and as Windows metafile. Copy and paste it into an MS-Office application.

4.) Calibration and variance stabilization.

a. Subtract the background intensities CH1B, CH2B from the foreground intensities CH1I, CH2I, and store the result in a 9216 x 2 matrix.

```
> y = cbind(x$CH1I - x$CH1B, x$CH2I - x$CH2B)
```

b. What does the function `cbind` do? Use the R online help to find out.

c. Now we can use the function `vsn` to calibrate and transform the data, and plot the result (Fig. 3).

```
> ny = vsn(y)
> plot(exprs(ny))
```

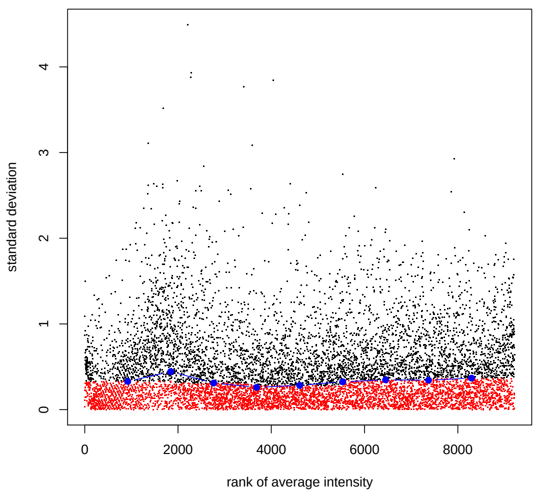


Figure 3:

d. Have a look at the "vignette" - try the command `openVignette("vsn")`.

a. **Reading a collection of files.**

b. The file `phenoData.txt` contains information on the samples that were hybridized onto the arrays. Look at it in a text editor. To load it into a `phenoData` object

```
> samples = read.phenoData("phenoData.txt", header = TRUE, as.is = TRUE)
> samples
```

```
         phenoData object with 5 variables and 8 cases
         varLabels
                fileName: read from file
                sampleid: read from file
                tumortype: read from file
                sex: read from file
                slideNumber: read from file
```

```
> pData(samples)
```

```
        fileName  sampleid tumortype  sex slideNumber
1 lc7b047rex.DAT    CLL-13       CLL    m           1
2 lc7b048rex.DAT    CLL-13       CLL    m           2
3 lc7b069rex.DAT    CLL-52       CLL    f           3
4 lc7b070rex.DAT    CLL-39       CLL    f           4
5 lc7b019rex.DAT DLCL-0032      DLCL    f           5
6 lc7b056rex.DAT DLCL-0024      DCLC    m           6
7 lc7b057rex.DAT DLCL-0029      DLCL    m           7
8 lc7b058rex.DAT DLCL-0023      DLCL <NA>           8
```

phenoData objects are where the Bioconductor stores information about samples, for example, treatment conditions in a cell line experiment or clinical or histopathological characteristics of tissue biopsies.

c. Now we can load the whole set of 8 slides into the data object a.

```
> files = samples$fileName
> files
> a = read.marrayRaw(files, name.Gf = "CH1I", name.Gb = "CH1B",
+     name.Rf = "CH2I", name.Rb = "CH2B")
```

d. . . . and try out different normalization methods:

1. vsn (affine normalization and variance stabilization)
2. maNorm with global median location normalization
3. maNorm with loess for intensity- or A-dependent location normalization using the 'loess' smoother

```
> na1 = vsn(a)
> na2 = maNorm(a, norm = "median", echo = T)
> na3 = maNorm(a, norm = "loess", echo = T)
```

e. These commands take their time! You can save the results into a file with the save function, and later restore them with the load function. You can use the GUI for the latter.

```
> save(na1, na2, na3, file = "lymphomanorm.rda")
> load(file = "lymphomanorm.rda")
```

f. Now we want to extract the normalized log-ratios. The first line in the following code creates a three-dimensional array M with space for 9216 genes, 8 samples and 3 different normalization methods. na1 is the result of vsn; the normalized intensities are accessed via the function exprs, and the log-ratios are obtained by subtracting the red intensities from the green ones. na2 and na3 are the output of marrayNorm, and the log-ratios are obtained through the *slot* maM using the *accessor* @.

```
> odd = seq(1, 15, by = 2)
> even = seq(2, 16, by = 2)
> M = array(NA, dim = c(9216, 8, 3))
> M[, , 1] = exprs(na1)[, odd] - exprs(na1)[, even]
> M[, , 2] = na2@maM
> M[, , 3] = na3@maM
> dim(M)
[1] 9216    8    3
```

5.) Compare the results. Look at scatterplots of the values of M from the same slide, calculated with different normalization methods. Do the values generally agree? How do they differ?

```
> plot(M[, 4, 1], M[, 4, 2], pch = ".", xlab = "vsn", ylab = "loess")
```

6.) Testing for differential transcription. Now we are ready to calculate test statistics and to select genes. *Note:* The number of replicates (4 versus 4) that we are considering here is too small to derive significant conclusions about individual genes. The full data set contains many more chips. Here we restrict ourselves to a few of them in order to keep things simple for the purpose of this course.

a. Look at the built-in function t.test, and at mt.teststat from the package multtest. Here, we use mt.teststat to calculate the t-test statistic for the comparison. The package multtest provides extensive functionality to calculate multiple-testing adjustments.

```
> classlabel = c(0, 0, 0, 0, 1, 1, 1, 1)
> tStat = mt.teststat(M[, , 1], classlabel)
> range(tStat)
```

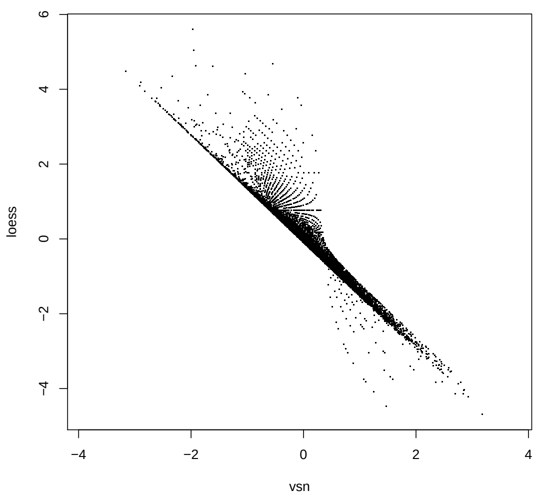


Figure 4:

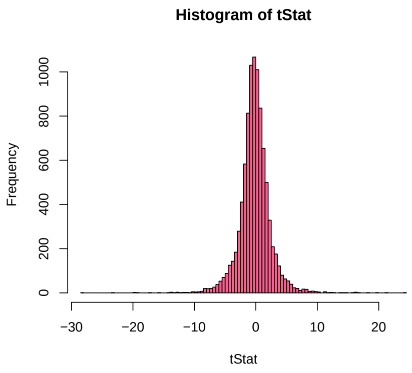


Figure 5:

```
[1] -28.04803  24.21349
> hist(tStat, breaks = 100, col = "#fb6090")
```

b. Similar to the FDR (false discovery rate) we estimate the significance of the most extreme t-values. The false discovery count is calculated for several lists of various length of the highest scored genes (Fig. 6).

```
> classlabel = c(0, 0, 0, 0, 1, 1, 1, 1)
> fc = fdc(M[, , 1], factor(classlabel), teststatfun = "rowttests")
> plot(fc$nrgenesel, fc$fdc, main = "False Discovery Count", xlab = "number of selected genes"
+     ylab = "false discovery count", type = "b")
```

c. Now we load the spot (gene) description table

```
> spotDescr = read.delim("annotationData.txt")
```

and print the 5 genes with the lowest values of the *t*-statistic

```
> selection = order(tStat)[1:5]
> selection
[1] 4323 4069 4331 2026 2143
```

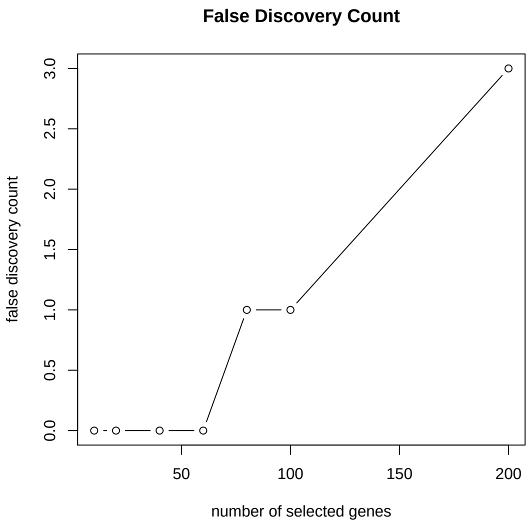


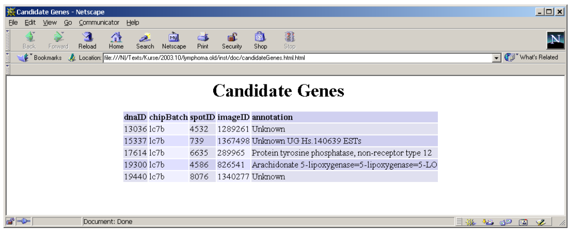
Figure 6:

as well as the 5 genes with the highest values of the *t*-statistic

```
> selection = order(tStat, decreasing = TRUE)[1:5]
> selection
[1] 4532 8076 6635 4586  739
```

In a following step we extract the annotation information for the 5 selected genes and generate a html-report (Fig. 7).

```
> spotIDs = x$SPOT[selection]
> geneAnno = spotDescr[spotDescr[, "spotID"] %in% spotIDs, ]
> write.htmltable(geneAnno, filename = "candidateGenes.html", title = "Candidate Genes")
```


Candidate Genes - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: file:///N|/Texts/Kurse/2003.10/lymphoma.old/inst/doc/candidateGenes.html.html
What's Related
Document: Done

Candidate Genes

dnaID	chipBatch	spotID	imageID	annotation
13036	lc7b	4532	1289261	Unknown
15337	lc7b	739	1367498	Unknown UG Hs.140639 ESTs
17614	lc7b	6635	289965	Protein tyrosine phosphatase, non-receptor type 12
19300	lc7b	4586	826541	Arachidonate 5-lipoxygenase=5-lipoxygenase=5-LO
19440	lc7b	8076	1340277	Unknown

Figure 7:

7.) A simplified route to preprocessing and quality control.

a. The package arrayMagic can be used to automatically process the image files, to create R-objects and to generate quality diagnostics. Fig. 8 characterizes the raw data distributions of all channels. Fig. 9 visualizes a distance measure calculated between all pairs of slides.

```
> resultList = processArrayData(slideDescriptionFile = "phenoData.txt",
+     loadPath = ".", savePath = ".", fileNameColumn = "fileName",
+     slideNumberColumn = "slideNumber", spotIdentifier = "SPOT",
+     imageFileType = "ScanAlyze", subtractBackground = TRUE, normalisationMethod = "vsn")
> qR = qualityParameters(arrayDataObject = resultList$arrayDataObject,
+     exprSetArrayObject = resultList$exprSetArrayObject, spotIdentifier = "SPOT",
+     hybNameColumn = "slideNumber")
> qualityDiagnostics(exprSetArrayObject = resultList$exprSetArrayObject,
+     arrayDataObject = resultList$arrayDataObject, qualityParametersList = qR,
+     savegraphicspath = ".")
> glratios = getExprSetCy3MinusCy5(resultList$exprSetArrayObject)
> phenoD = phenoDataSlide(resultList$exprSetArrayObject)
> pD = pDataSlide(resultList$exprSetArrayObject)
```

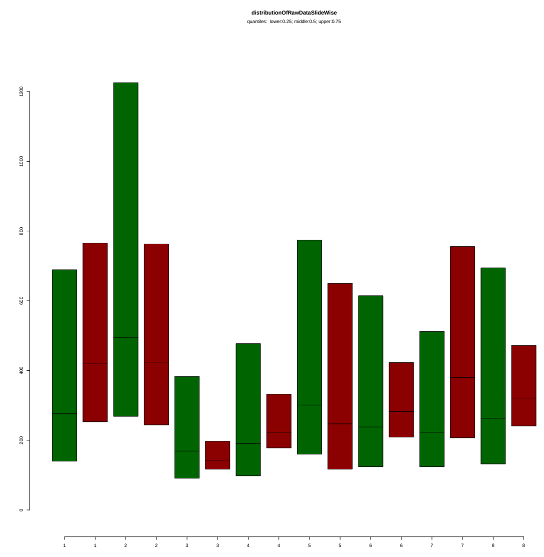


Figure 8:

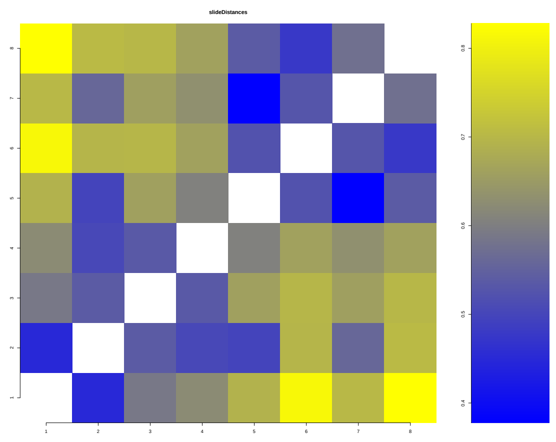


Figure 9: