— Exploring cDNA Data—

Achim Tresch, Andreas Buness, Tim Beißbarth, Florian Hahne, Wolfgang Huber

December 20, 2006

The following exercise will guide you through the first steps of a spotted cDNA microarray analysis. These steps comprise loading data into R/Bioconductor, quality control of the measurements, and preprocessing of the raw data via normalization up to the generation of very simple gene lists. Make extensive use of the `help(<object>)` command to find information about particular objects. Use the `vignette(<package>)` command to get introductory material for a certain package.

1.) Preliminaries. To go through this exercise, you need to have installed R >= 2.1.0, the release 1.6 versions of the Bioconductor libraries Biobase, marray, multtest, limma, vsn, and arrayMagic and the library lymphoma, which contains the excercises and part of the lymphoma data set.

```
> library("vsn")
> library("multtest")
> library("marray")
> library("arrayMagic")
> library("RColorBrewer")
```

2.) Reading data files. Considerable attention should be payed to data import. Experience tells that this is one of the most error prone steps.

a. Your data has to be stored in one folder, with one file corresponding to one sample. Our example folder is located at `<R library path>/lymphoma/extdata/`. You can find the path to that location using `system.file`.

```
> path = system.file("extdata", package = "lymphoma")
> path
[1] "/tmp/Rinst2845311948/lymphoma/extdata"
```

The file names are `lc7b047rex.DAT, lc7b048rex.DAT, ....`

```
> dir(path, pattern = ".DAT$")
[1] "lc7b019rex.DAT" "lc7b047rex.DAT" "lc7b048rex.DAT" "lc7b056rex.DAT"
[5] "lc7b057rex.DAT" "lc7b058rex.DAT" "lc7b069rex.DAT" "lc7b070rex.DAT"
```

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 we will describe an import method which is suited to the most common software (Genepix, Spot, ...??).

c. For a both easy and flexible import of the data, there has to be a description file. The description file contains a table with all the hybridization data file names in one column and possibly additional sample information in further columns. Create a tab-delimited text file of this kind. We have done this for you, the description file is named `phenoData.txt`. You may examine its structure with any text editor. There is a convenient method for converting such a table into a `phenoData` object.

```
> lymphenoData = read.phenoData(file.path(path, "phenoData.txt"))
```

d. The `phenoData` object is used to simulatneously import all data files.

```
> lymphRaw = readIntensities(pData(lymphenoData), loadPath = path,
+     fileNameColumn = "fileName", slideNameColumn = "slideNumber",
+     type = "ScanAlyze")
> lymphNormvsn = normalise(lymphRaw, subtractBackground = T, method = "vsn",
+     spotIdentifier = "SPOT")
> lymphoma = as.exprSet(lymphNormvsn)
```

3.) The Bioconductor class `exprSet`.

a. The object `lymphoma` is of class `exprSet`. This class is the standard representation of a microarray experiment in Bioconductor. It consists of the objects ("slots")

exprs	:	A spots × samples matrix containing the expression levels
se.exprs	:	A spots × samples matrix containing an estimate of the standard error of each single spot measurement
phenoData	:	An object of class `phenoData`, essentially a data frame containing phenotypical information about the samples that were hybridized
annotation	:	Textual annotation
description	:	Object of class `MIAME` which incorporates those MIAME-entries that are not covered by other objects of the `exprSet` class
notes	:	Text containing additional remarks

Slots can be accessed directly with "@" (e.g. `lymphoma@phenoData`), but one should use the accessor methods for the class `exprSets`. See `help(exprSets)` for details. The most interesting objects to us are the expression matrix, given by `exprs(lymphoma)` and the data frame with the phenotype data, `pData(lymphoma)`. Have a look at them.

```
> dim(exprs(lymphoma)) # genes × samples
[1] 9216   16
> exprs(lymphoma)[1:3, 1:6]
          [,1]     [,2]     [,3]     [,4]     [,5]     [,6]
gene1 4.705011 5.157140 5.315628 5.970322 6.187813 4.736314
gene2 6.542351 6.561461 7.373095 6.251734 7.219573 6.471994
gene3 7.117775 7.051510 7.457064 6.635266 7.733980 7.020954
> dim(pData(lymphoma)) # samples × descriptors
[1] 16  5
> pData(lymphoma)
        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
⋮
```

b. You might want to add a column containing the hybridization colour.

```
> colour = c(rep("red", 8), rep("green", 8))
> pData(lymphoma) = cbind(pData(lymphoma), colour)
        fileName sampleid tumortype sex slideNumber colour
1 lc7b047rex.DAT   CLL-13       CLL   m           1    red
2 lc7b048rex.DAT   CLL-13       CLL   m           2    red
3 lc7b069rex.DAT   CLL-52       CLL   f           3    red
⋮
```

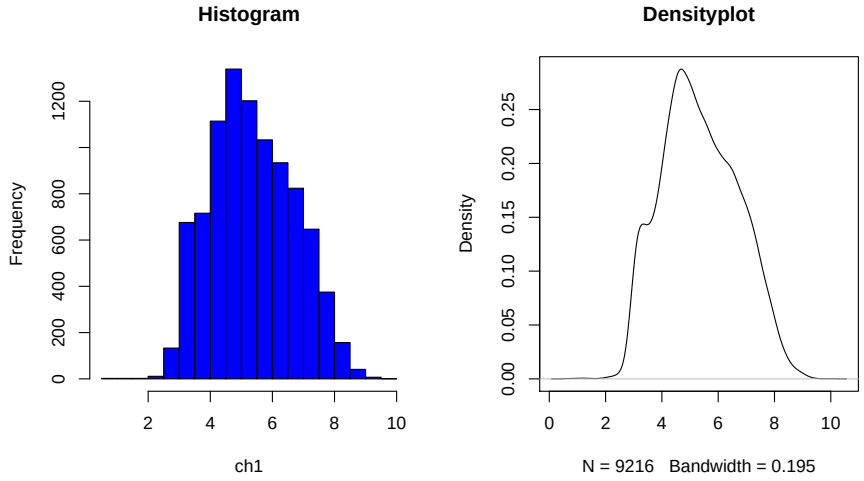
4.) Simple plots.

a. We will perform some elementary diagnostic plots for quality control. Most analyses are carried out on the log transformed data, so `lymphoma` contains (generalized) log transformed expression values. For convenience, we extract these values into another variable.

```
> logexpr = exprs(lymphoma)
```

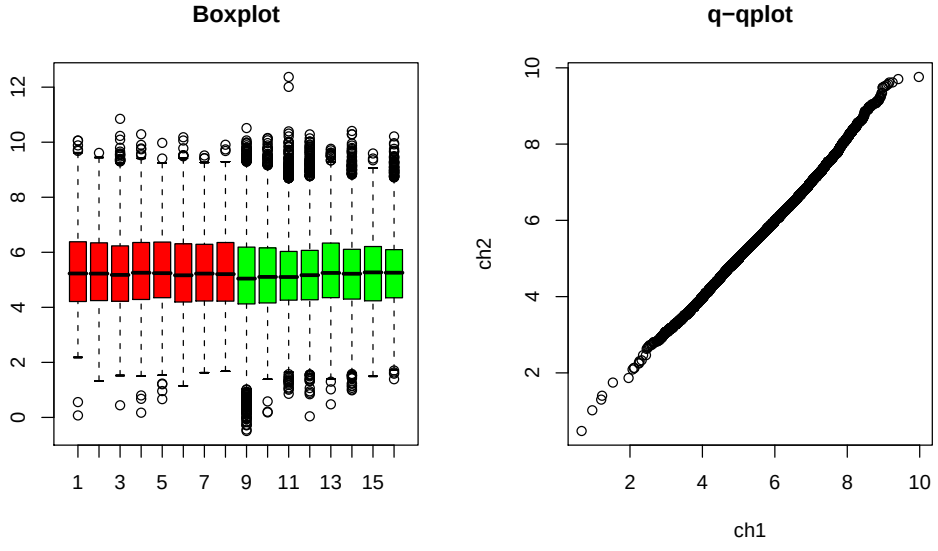
It is possible to examine each single channel or to focus on log ratios or difference between them. Notice that in our expSet the data for the red channel are in columns 1 to 8 and data for the green channel in columns 9 to 16. For the following plots we will first use the two channels of slide number 5. Later we will also look at the log ratios.
Now produce a histogram and a density plot of the log intensities in channel 1 of slide 5.

```
> slidenr = 5
> ch1 = logexpr[, slidenr]
> ch2 = logexpr[, slidenr + 8]
> plot(hist(ch1), main = "Histogram", col = "blue")
> plot(density(ch1), main = "Densityplot")
```



Here are some plots that help to detect bad hybridizations. Compare the log intensity boxplots of the slides

```
> boxplot(split(t(logexpr), 1:ncol(logexpr)), col = as.vector(pData(lymphoma)[,
+     "colour"]), main = "boxplot")
```



A convenient way to compare the expression distributions between two samples is a quantile-quantile plot. For equal distributions it should be a diagonal line.

```
> qqplot(ch1, ch2, main = "q-q plot")
```

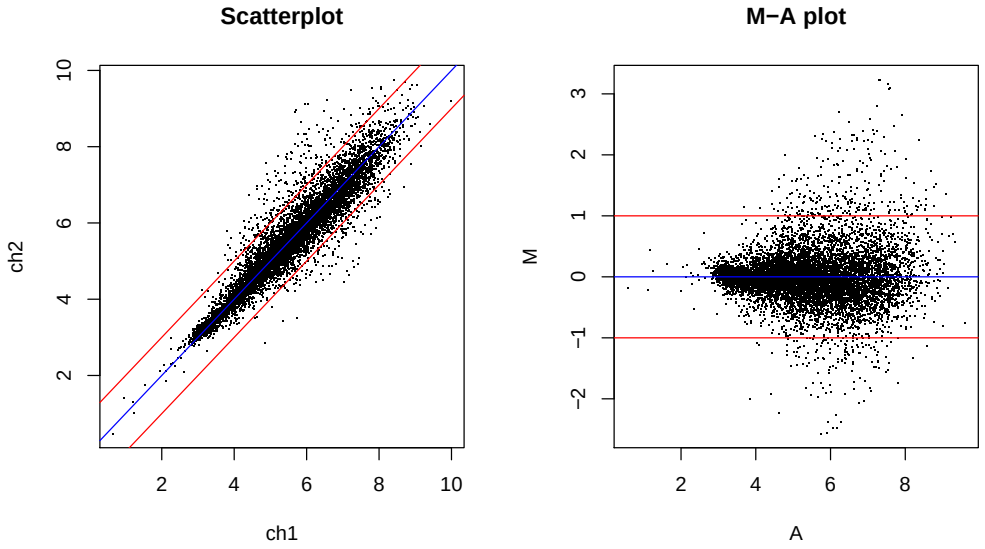
Do you think the q-qplot of slide number 5 looks OK?

b. Save one of the plots as a PDF. Copy and paste it into an MS-Office application.

```
> pdf(file = "savedplot")
> qqplot(ch1, ch2, main = "q-q plot")
> dev.off()
```

c. A more detailed view is provided by a scatter plot and its corresponding M-A plot

```
> plot(ch1,ch2,pch=".",main="Scatterplot")
> abline(a=0,b=1,col="blue"); abline(a=1,b=1,col="red"); abline(a=-1,b=1,col="red")
> plot((ch1+ch2)/2,ch2-ch1,pch=".",xlab="A",ylab="M",main="M-A plot")
> abline(h=0,col="blue"); abline(h=1,col="red"); abline(h=-1,col="red")
```



d. We hid one "outlier slide" among the original lymphoma data. Find it using diagnostic plots!

5.) Normalization

a. Before we started analysis, we tacitly normalized our data (cf. `normalise(lymphraw...)`). Try two other commonly used normalization methods:

```
> lymphNormloess = normalise(lymphRaw, subtractBackground = T,
+     method = "loess", spotIdentifier = "SPOT")
> logexpr2 = as.exprSet(lymphNormloess)
> lymphNormquantile = normalise(lymphRaw, subtractBackground = T,
+     method = "quantile", spotIdentifier = "SPOT")
> logexpr3 = as.exprSet(lymphNormquantile)
```

b. 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. In MS-Windows, you can use the GUI for the latter.

c. Compare the results of the variance stabilization method to the loess method! Which Plots are appropriate for that?

6.) Testing for differential expression

a. Let us now get the log-ratios of the 2 channels for all 8 slides in our data set. This can be done quite easily using function `getExprSetLogRatio`. Look into its manual and also find out about `exprSetRG` objects, which extend regular `exprSets` to include information about the two different channels.

```
> lymphRatio = getExprSetLogRatio(lymphNormvsn)
> logRatios = exprs(lymphRatio)
```

Finally 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(logRatios, classlabel)
> summary(tStat)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-28.1000  -1.3670  -0.1861  -0.1832   1.0330  24.3400
> hist(tStat, breaks = 100, col = "#fb6090")
```

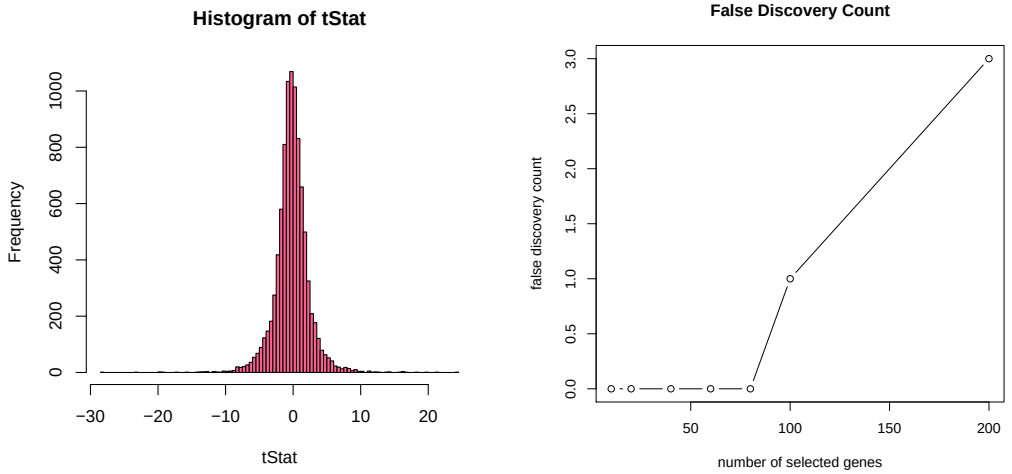


Figure 1: *a) left: histogram of t-statistic, b) right: false disocvery count.*

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. 1 right).

```
> fc = fdc(logRatios, factor(classlabel), teststatfun = "rowttests")
> plot(fc$nrgenesel, fc$fdc, main = "False Discovery Count", xlab = "number of selected g
+     ylab = "false discovery count", type = "b")
```

c. Now we load the spot (gene) description table with function `read.delim`. You can find it in the same folder as the raw data in the file annotationData.txt.

What does `read.delim` do?

```
> spotDescr = read.delim(system.file("extdata", "annotationData.txt",
+     package = "lymphoma"))
```

Let's print the 5 genes with the lowest values of the t-statistic

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

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. 2).

```
> spotIDs = getSpotAttr(lymphRaw)$SPOT[selection]
```

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

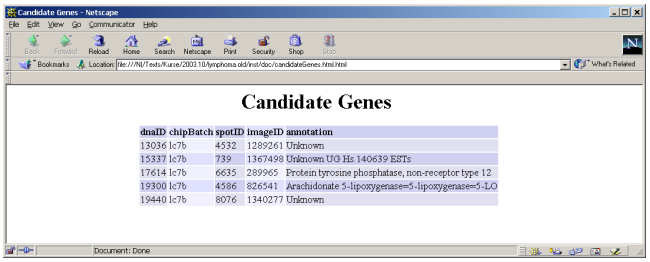
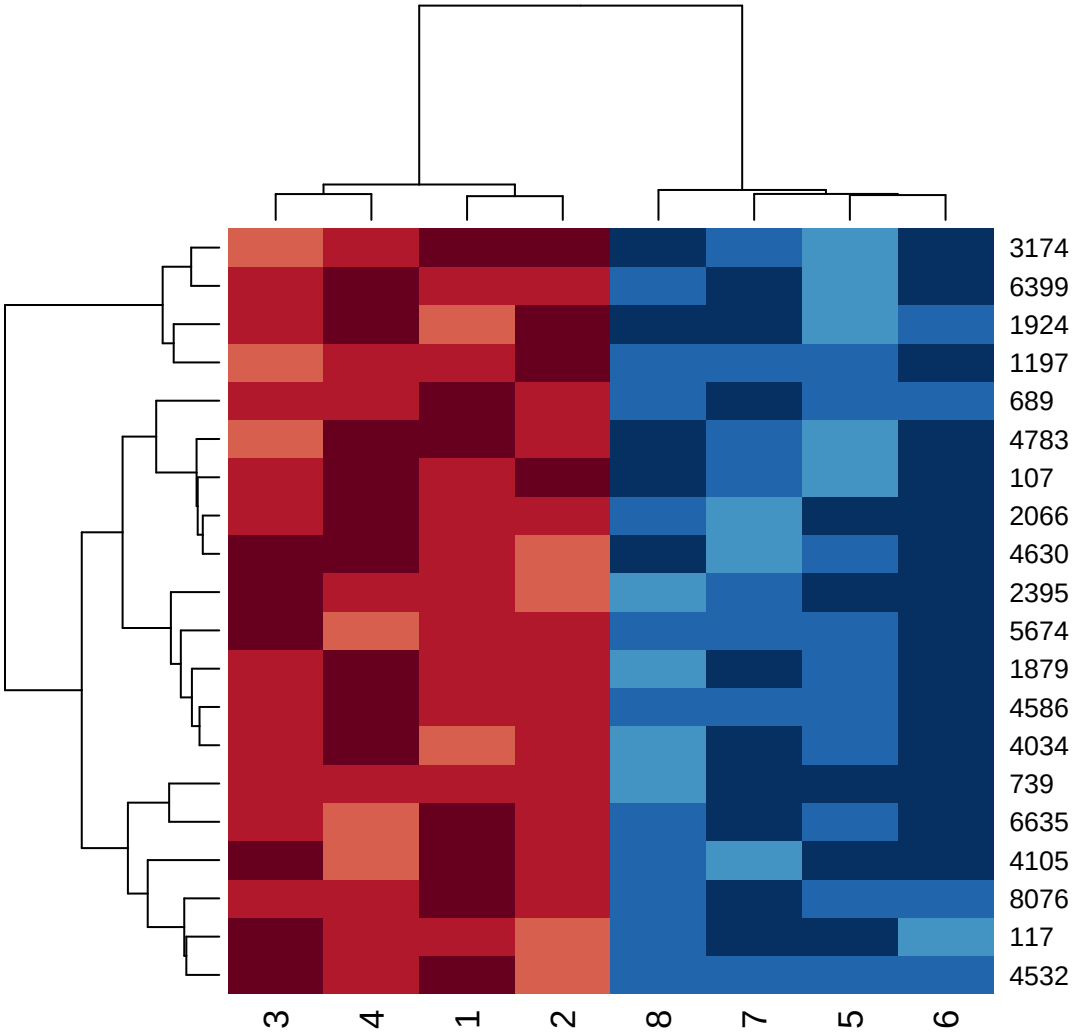


Figure 2:

d. Due to their pervasive power, heatmaps enjoy high popularity (although they hardly prove anything). We can produce one that shows the expression levels of the 20 genes with highest values of the t-statistc in a few lines of R code.

```
> selection = order(tStat, decreasing = TRUE)[1:20]
> heatmap(logRatios[selection, ], col = brewer.pal(10, "RdBu"))
```



7.) Further quality assessment

The package `arrayMagic` provides additional measures for quality control. It produces a bunch of graphics which are saved in your current working directory. Take the time to examine some of them.

Have a look at the vignette arrrayMagicVignette for details before proceeding.

```
> vignette("arrayMagicVignette")
```

```
> qP <- qualityParameters(lymphRaw, lymphNormvsn, resultFileName = "qP.txt",
+     spotIdentifier = "SPOT", slideNameColumn = "fileName")
> qualityDiagnostics(lymphRaw, lymphNormvsn, qP)
```

```
postscript
        2
```

```
> visualiseHybridisations(lymphRaw[, 1], mappingColumns = list(Block = "GRID",
+     Column = "COL", Row = "ROW"))
```

here are some samples of the output: