— Exploring cDNA Data—

Achim Tresch, Andreas Buness, Wolfgang Huber, Tim Beißbarth

Practical DNA Microarray Analysis
http://compdiag.molgen.mpg.de/ngfn/pma2005nov.shtml

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. 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 $>=$ 2.6, the release versions $>=$2.1 of the Bioconductor libraries Biobase, cluster, genefilter, marray, multtest, limma, RColorBrewer, vsn, and arrayMagic.

```
> library("vsn")
> library("marray")
> library("limma")
> library("arrayMagic")
> library("cluster")
> library("RColorBrewer")
```

2.) Reading data files.

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>/arrayMagic/extdata/`. Set the working directory path to that location. With a little luck,

```
> setwd(system.file("extdata", package = "arrayMagic"))
```

does the job. The file names are `lc7b047rex.DAT`, `lc7b048rex.DAT`, ... 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 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 `phenoDataLymphoma.txt`. You may examine its structure with any text or spreadsheet editor. There is a convenient method for converting such a table into a `phenoData` object. Ignore all the warnings, which are due to a recent change in the Bioconductor base classes.

```
> lymphenoData = read.phenoData("phenoDataLymphoma.txt", header = TRUE)
```

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

```
> lymphRaw = readIntensities(pData(lymphenoData), fileNameColumn = "fileName",
+     slideNameColumn = "slideNumber", type = "ScanAlyze")
```

```
> lymphNormvsn = normalise(lymphRaw, subtractBackground = T, method = "vsn",
+      spotIdentifier = "SPOT")
> lymphoma = as(as.exprSet(lymphNormvsn), "ExpressionSet")
> setwd(homedir)
```

3.) The Bioconductor class `ExpressionSet`.

a. The object `lymphoma` is now of class `ExpressionSet`. This class is the standart representation of a microarray experiment in Bioconductor. It consists of the objects ("slots"). You can get an overview of the structure of an object using the function `str(lymphoma)`. Slots can be accessed directly with "@" (e.g. `lymphoma@phenoData`), but one should use the accessor methods for the class `ExpressionSet`. See `help(ExpressionSet)` 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. In older Bioconductor Versions the standart class for microarray data was `exprSet`, which still works similarly as `ExpressionSet`, but is depreciated now. We will use it though, in parts of this excercise as some packages as e.g. `arrayMagic` do not support the new class yet.

```
> dim(exprs(lymphoma)) # genes × samples
[1] 9216    16
> exprs(lymphoma)[1:3, 1:6]
                1        2        3        4        5        6
gene #1 4.705011 5.157140 5.315628 5.970322 6.187813 4.736314
gene #2 6.542351 6.561461 7.373095 6.251734 7.219573 6.471994
gene #3 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 = rep(c("red", "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  green
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. Produce a histogram and a density plot of the log intensities in channel 1 of slide 1. The command `x11()` opens a new graphics window.
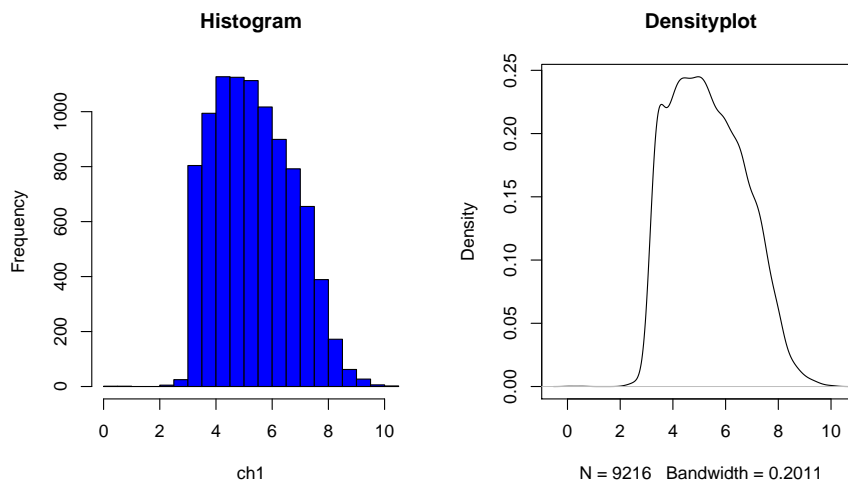
```
> ch1=logexpr[,1]; ch2=logexpr[,2] # we will need the second channel later
```

```
> x11()
> plot(hist(ch1), main = "Histogram")
> plot(density(ch1), main = "Densityplot")
```


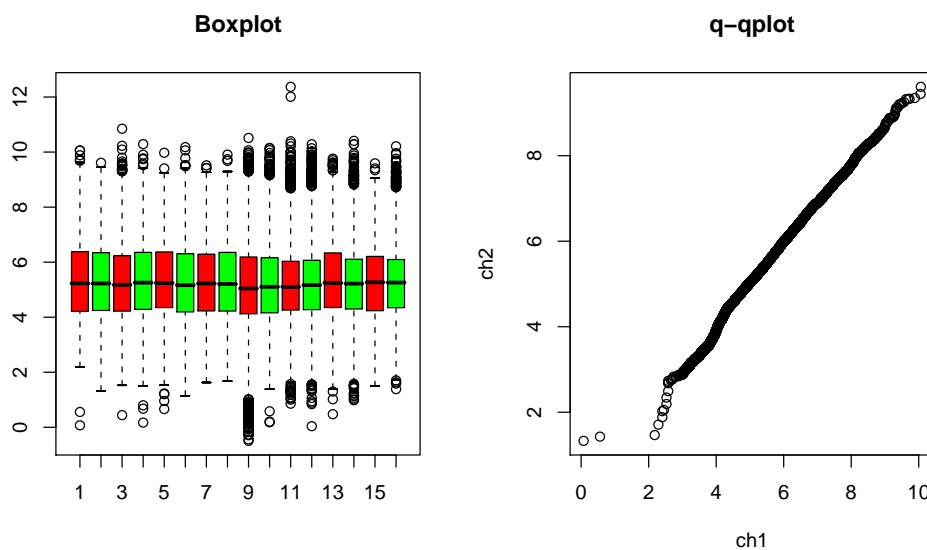**Histogram**   **Densityplot**

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")
```


Boxplot   **q-qplot**

A convenient way to compare the expression distributions between two samples is a quantile-quantile plot

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

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!

Scatterplot   **M–A plot**
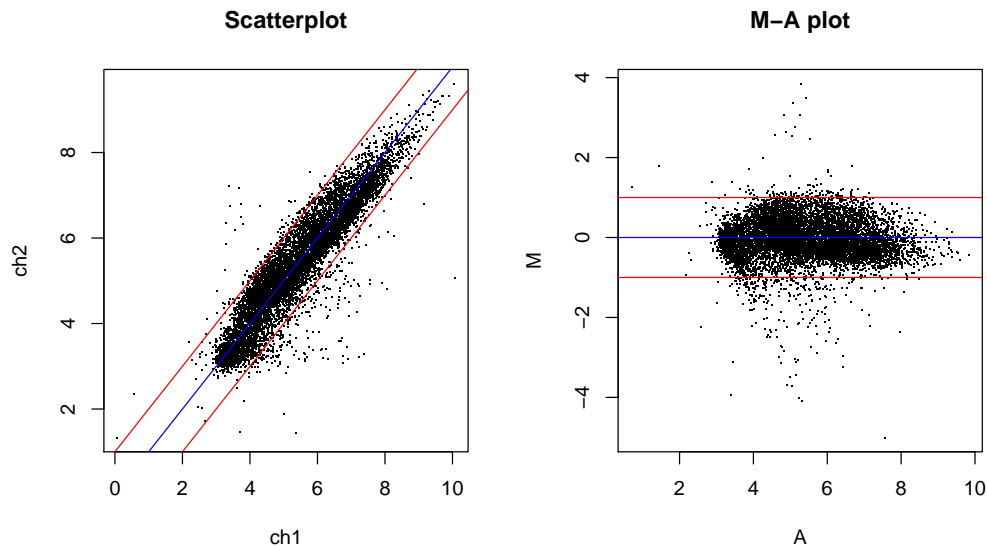
5.) Normalization

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.) Further data exploration

a. Another explorative tool is clustering of genes, which can be done in many ways. We use k-means clustering to obtain 50 gene clusters, say.

```
> set.seed(0)
> palette(rainbow(9))
> result = kmeans(logexpr, centers = 50, iter.max = 30)
> x11()
> par(mfrow = c(3, 3))
> for (j in 1:9) {
+     selection = (1:nrow(logexpr))[result$cluster %in% j]
+     plot(c(0, 0), xlim = c(1, 16), ylim = c(min(logexpr[selection,
+         ]), max(logexpr[selection, ])), type = "n", xlab = "",
+         ylab = "")
+     apply(logexpr[selection, ], 1, points, type = "l", col = j)
+ }
```
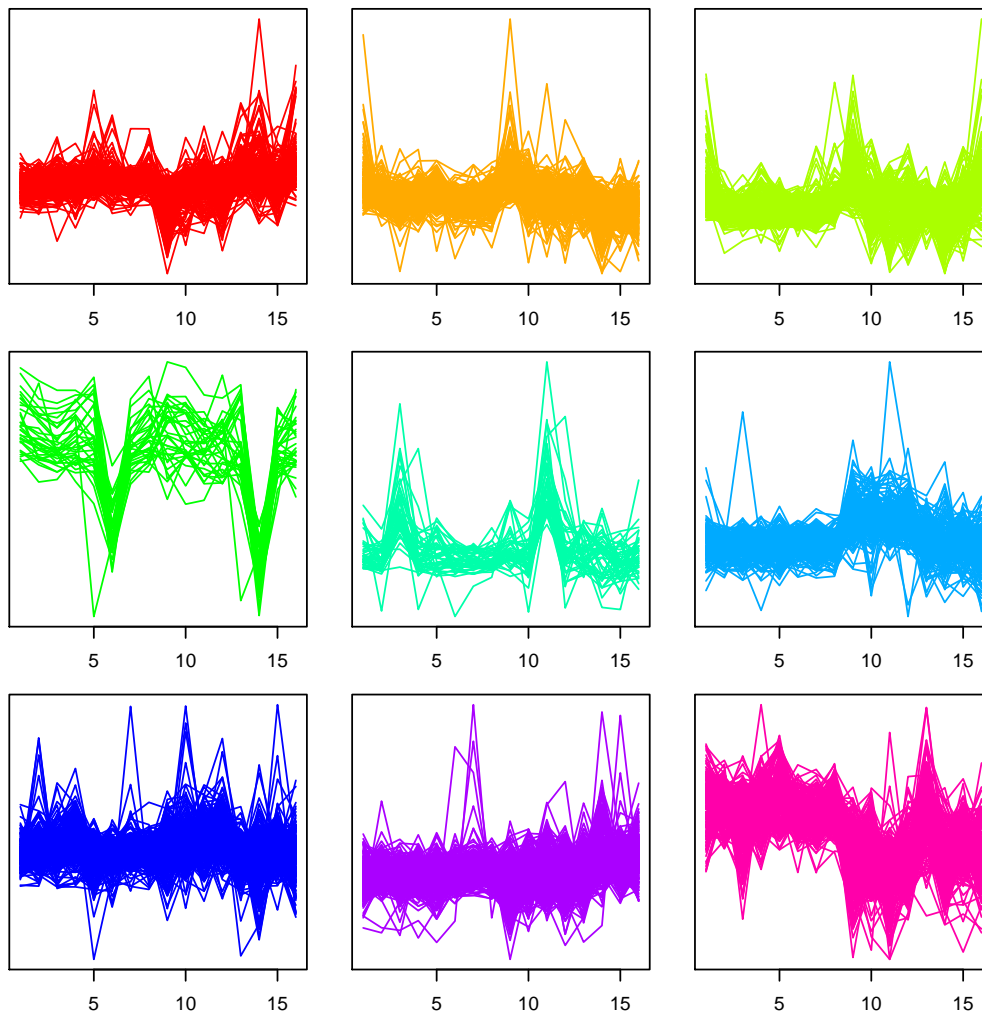
b. Due to their pervasive power, heatmaps enjoy high popularity (although they hardly prove anything). We can produce them in a few lines of R code.

```
> selection1 = (1:nrow(logexpr))[result$cluster %in% 11]
> selection2 = (1:nrow(logexpr))[result$cluster %in% 20]
> heatmap(logexpr[c(selection1[1:20], selection2[1:20]), ], col = brewer.pal(10,
+       "RdBu"))
```
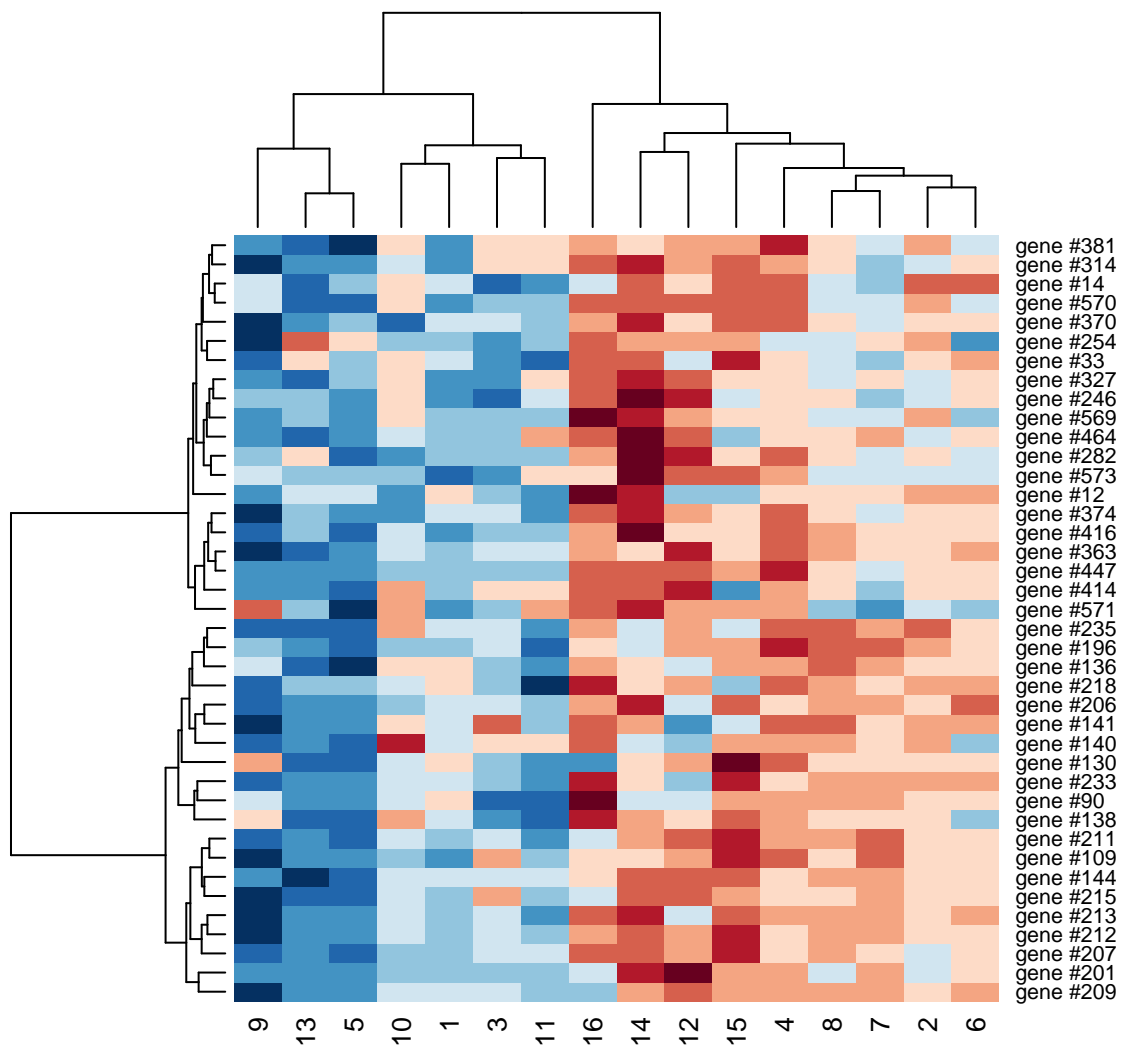
7.) Further quality assessment

The package `arrayMagic` provides additional measures for quality control. It produces a bunch of graphics which are saved as pdf files in your current working directory - you might want to create a new directory and change into it using setwd() first. Take the time to examine some of them. Have a look at the vignette `arrayMagicVignette` for details before proceeding.

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


X11
   2


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

here are some samples of the output:

slideDistances

lc7b069rex.DAT
lc7b070rex.DAT
lc7b048rex.DAT
lc7b047rex.DAT
lc7b058rex.DAT
lc7b056rex.DAT
lc7b057rex.DAT
lc7b019rex.DAT

lc7b019rex.DAT
lc7b057rex.DAT
lc7b056rex.DAT
lc7b058rex.DAT
lc7b047rex.DAT
lc7b048rex.DAT
lc7b070rex.DAT
lc7b069rex.DAT

green foreground   **green background**   **red foreground**   **red background**