Working with Affymetrix data: estrogen, a 2x2 factorial design example

Practical Microarray Course, Heidelberg/Berlin Spring 2004

Robert Gentleman, Wolfgang Huber

1.) **Preliminaries.** To go through this exercise, you need to have installed R>=1.8.1, the libraries `Biobase`, `affy`, `hgu95av2`, `hgu95av2cdf` from the Bioconductor release 1.3, `vsn` >= 1.4.6, and the library `estrogen`, which contains the data.

```
> library(affy)
> library(estrogen)
> library(vsn)
```

2.) **Load the data.**

a. Find the directory where the example cel files are. The directory path should end in `.../R/library/estrogen/extdata`.

```
> datadir = system.file("extdata", package = "estrogen")
> datadir
[1] "/home/whuber/R-1.8.1.linux/library/estrogen/extdata"
> dir(datadir)
 [1] "bad.cel"          "estrogen.tex"     "estrogen.txt"     "high10-1.cel"
 [5] "high10-2.cel"     "high48-1.cel"     "high48-2.cel"     "low10-1.cel"
 [9] "low10-2.cel"      "low48-1.cel"      "low48-2.cel"      "phenoData.txt"
[13] "workspace.RData"
> setwd(datadir)
```

The function `system.file` here is used to find the subdirectory `extdata` of the estrogen package on your computer's harddisk. If you had your own data, you would have to specify the appropriate path.

b. The file `estrogen.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

```
> pd = read.phenoData("estrogen.txt", header = TRUE, row.names = 1)
> pData(pd)
             estrogen time.h
low10-1.cel    absent     10
low10-2.cel    absent     10
high10-1.cel  present     10
high10-2.cel  present     10
low48-1.cel    absent     48
low48-2.cel    absent     48
high48-1.cel  present     48
high48-2.cel  present     48
```

`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. The `header` option lets the `read.phenoData` function know that the first line in the file contains column headings, and the `row.names` option indicates that the first column of the file contains the row names.

c. Load the data from the CEL files into an `AffyBatch`. An `AffyBatch` is an object in which the Bioconductor can store the raw data from an Affymetrix genechip experiment (i.,e., the CEL file data), as well as accompanying experiment annotation.

```
> a = ReadAffy(filenames = rownames(pData(pd)), phenoData = pd,
+     verbose = TRUE)
```

```
> a
AffyBatch object
size of arrays=640x640 features (25604 kb)
cdf=HG_U95Av2 (12625 affyids)
number of samples=8
number of genes=12625
annotation=hgu95av2
```

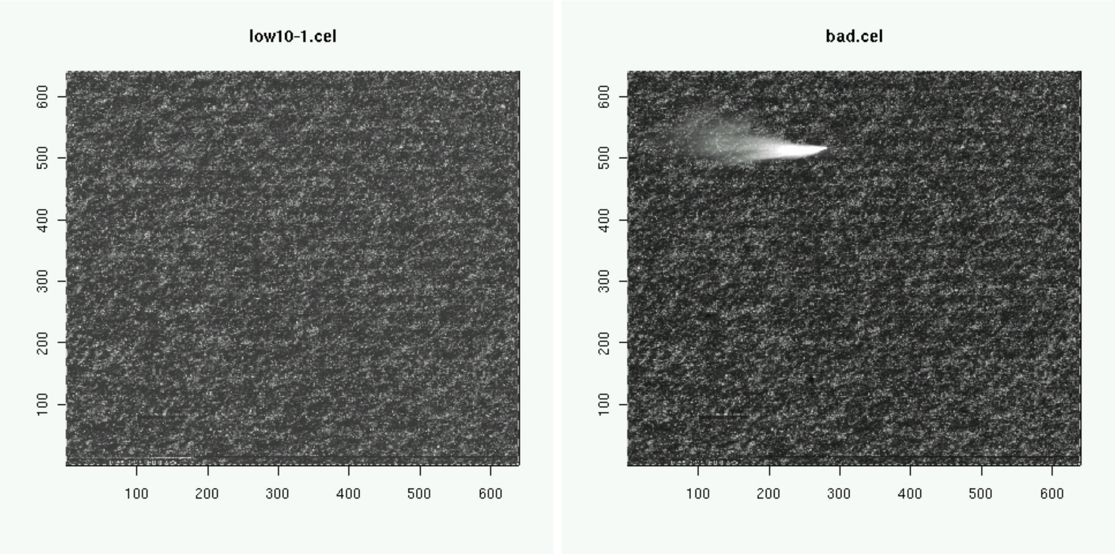

Figure 1: see exercise 4.

3.) Normalization.

a. Now we can use the function `expresso` to normalize the data and calculate expression values.

```
> x <- expresso(a, bg.correct = FALSE, normalize.method = "vsn",
+     normalize.param = list(subsample = 1000), pmcorrect.method = "pmonly",
+     summary.method = "medianpolish")
> x
Expression Set (exprSet) with
        12625 genes
        8 samples
                 phenoData object with 2 variables and 8 cases
         varLabels
                estrogen: read from file
                time.h: read from file
```

The parameter `subsample` determines the time consumption, as well as the precision of the calibration. The default (if you leave away the parameter `normalize.param = list(subsample=1000)`) is 20000; here we chose a smaller value for the sake of demonstration. There is the possibility that `expresso` is not working properly due to memory problems (normally it should work with 384 MB). Then you should end this session, start a new R session, and load the libraries and data by typing

```
> library(affy)
> library(estrogen)
> library(vsn)
> datadir = system.file("extdata", package = "estrogen")
> setwd(datadir)
> load("workspace.RData")
```

`image.RData` includes the expression set `x` and the affybatch `a`. Then you can continue with the next paragraph.

b. What are other available methods for normalization, and expression value calculation?

```
> normalize.methods(a)
[1] "constant"         "contrasts"         "invariantset"     "loess"
[5] "qspline"          "quantiles"         "quantiles.robust" "vsn"
```

4.) Looking at the CEL file images. The `image` function allows us to look at the spatial distribution of the intensities on a chip. This can be useful for quality control. Fortunately, all of the 8 celfiles that we have just loaded do not show any remarkable spatial artifacts (see Fig. 1).

```
> image(a[1])
```

But we have another example:

```
> badc = ReadAffy("bad.cel")
> image(badc)
```

Note that in these images, row 1 is at the bottom, and row 640 at the top.

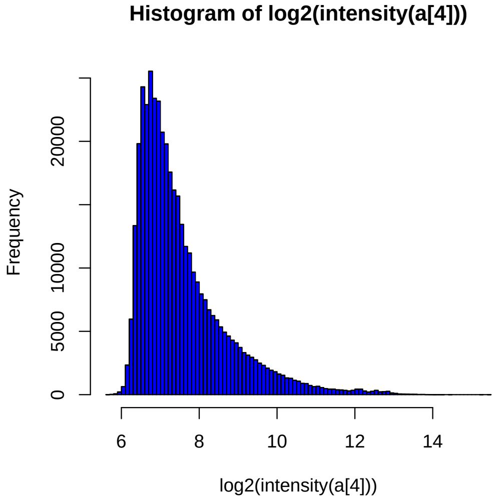

Figure 2: see exercise 5.

5.) Histograms. Another way to visualize what is going on on a chip is to look at the histogram of its intensity distribution. Because of the large dynamical range ($O(10^4)$), it is useful to look at the log-transformed values (see Fig. 2):

```
> hist(log2(intensity(a[4])), breaks = 100, col = "blue")
```

6.) Boxplot. To compare the intensity distribution across several chips, we can look at the boxplots, both of the raw intensities `a` and the normalized probe set values `x` (see Fig. 3):

```
> boxplot(a, col = "red")
> boxplot(data.frame(exprs(x)), col = "blue")
```

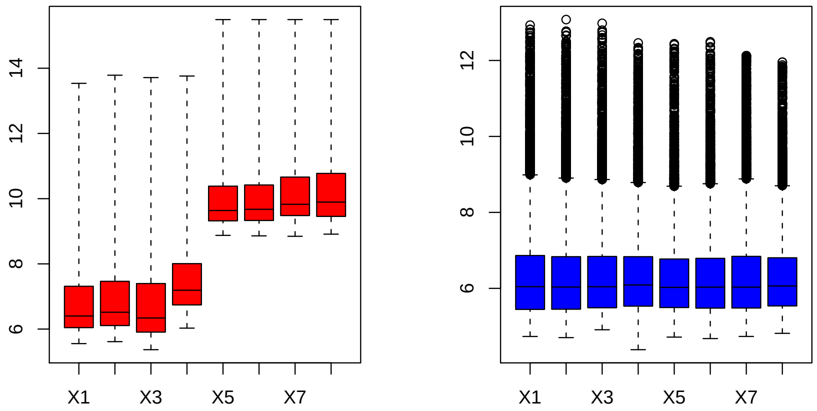

Figure 3: see exercise 6.

The boxplot for the rawdata `a` shows a strong time effect: the arrays made at $t = 48$h are generally much brighter than those at $t = 10$h. This is probably not biological effect, but an experimental artifact.

In the commands above, note the different syntax: `a` is an object of type `AffyBatch`, and the `boxplot` function has been programmed to know automatically what to do with it. `exprs(x)` is an object of type `matrix`. What happens if you do `boxplot(x)` or `boxplot(exprs(x))`?

```
> class(x)
```

```
[1] "exprSet"
attr(,"package")
[1] "Biobase"
```

```
> class(exprs(x))
```

```
[1] "matrix"
```

7.) **Scatterplot.** The scatterplot is a visualization that is useful for assessing the variation (or reproducibility, depending on how you look at it) between chips. We can look at all probes, the perfect match probes only, the mismatch probes only, and of course also at the normalized, probe-set-summarized data: (see Fig. 4):

```
> plot(exprs(a)[, 1:2], log = "xy", pch = ".", main = "all")
> plot(pm(a)[, 1:2], log = "xy", pch = ".", main = "pm")
> plot(mm(a)[, 1:2], log = "xy", pch = ".", main = "mm")
> plot(exprs(x)[, 1:2], pch = ".", main = "x")
```

8.) **Heatmap.** Select the 50 genes with the highest variation (standard deviation) across chips. (see Fig. 5):

```
> rsd <- rowSds(exprs(x))
> sel <- order(rsd, decreasing = TRUE)[1:50]
> heatmap(exprs(x)[sel, ], col = gentlecol(256))
```

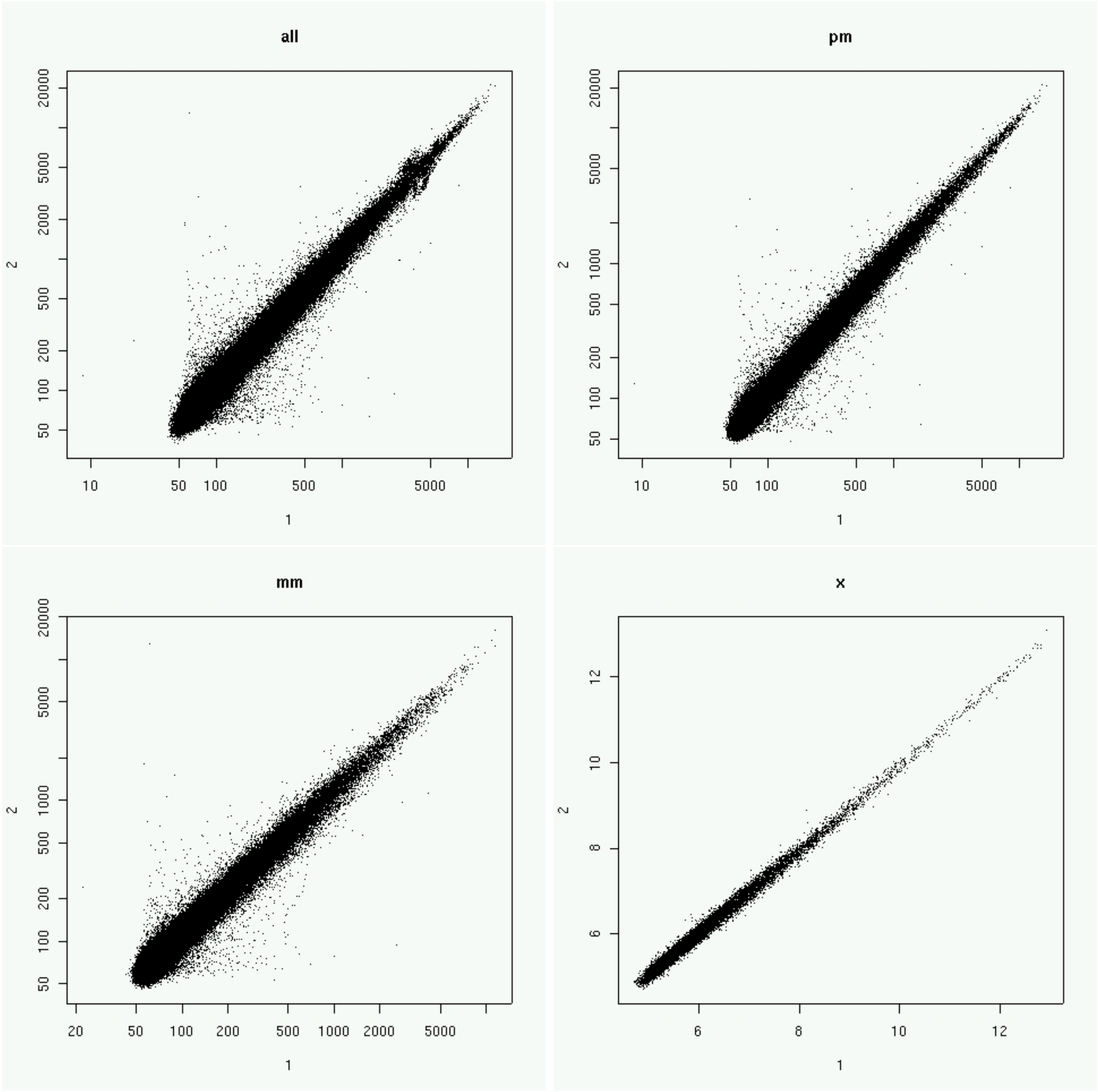

Figure 4: see exercise 7.

```
> savejpg("estrogen-heatmap.jpg")
```

9.) **ANOVA.** Now we can start analysing our data for biological effects. We set up a linear model with main effects for the level of estrogen (`estrogen`) and the time (`time.h`). Both are factors with 2 levels.

```
> lm.coef = function(y) lm(y ~ estrogen * time.h)$coefficients
> eff = esApply(x, 1, lm.coef)
```

For each gene, we obtain the fitted coefficients for main effects and interaction:

```
> dim(eff)
```

```
[1]     4 12625
```

```
> rownames(eff)
```

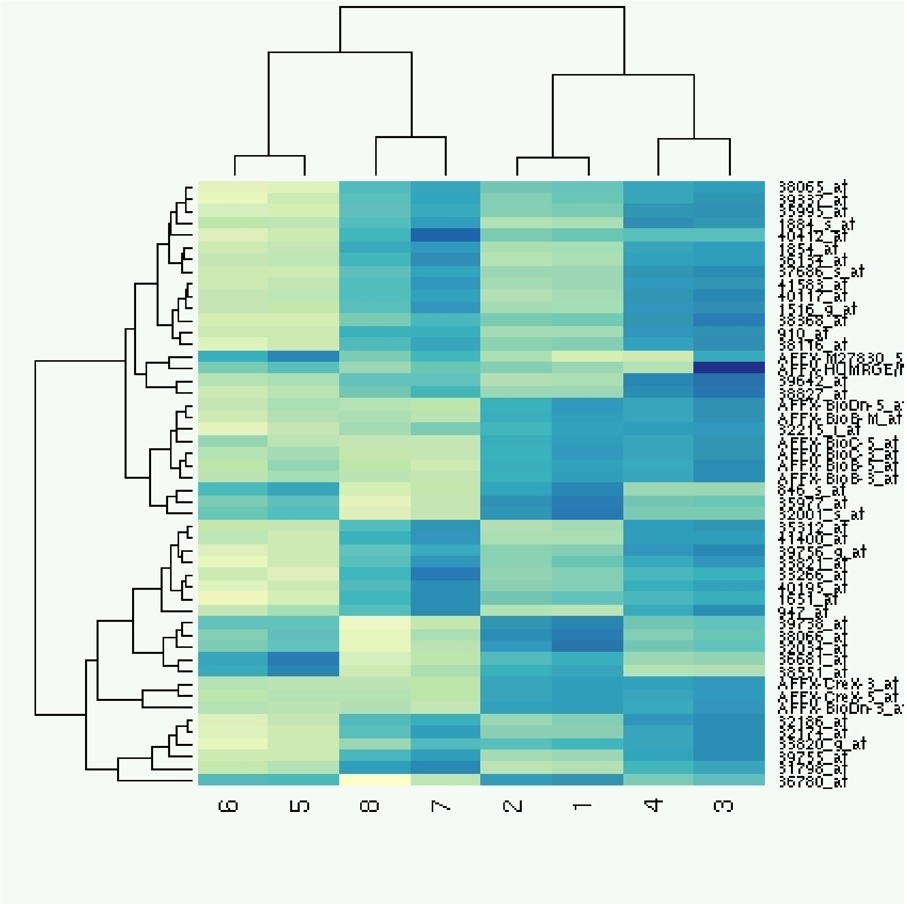

Figure 5: see exercise 8.

```
[1] "(Intercept)"             "estrogenpresent"        "time.h"
[4] "estrogenpresent:time.h"
```

Let's bring up the mapping from the vendor's probe set identifier to gene names.

```
> library(hgu95av2)
> ls("package:hgu95av2")
```

```
 [1] "hgu95av2"              "hgu95av2ACCNUM"       "hgu95av2CHR"
 [4] "hgu95av2CHRLENGTHS"    "hgu95av2CHRLOC"       "hgu95av2ENZYME"
 [7] "hgu95av2ENZYME2PROBE" "hgu95av2GENENAME"     "hgu95av2GO"
[10] "hgu95av2GO2ALLPROBES" "hgu95av2GO2PROBE"     "hgu95av2GRIF"
[13] "hgu95av2HGID"          "hgu95av2LOCUSID"      "hgu95av2MAP"
[16] "hgu95av2NM"            "hgu95av2NP"           "hgu95av2OMIM"
[19] "hgu95av2ORGANISM"      "hgu95av2PATH"         "hgu95av2PATH2PROBE"
[22] "hgu95av2PMID"          "hgu95av2PMID2PROBE"   "hgu95av2QC"
[25] "hgu95av2SUMFUNC"       "hgu95av2SYMBOL"       "hgu95av2UNIGENE"
```

```
> genename = env2list(hgu95av2GENENAME)
```

Let's now first look at the **estrogen main effect**, and print the top 3 genes with largest effect in one direction, as well as in the other direction. Then, look at the **estrogen:time interaction**.

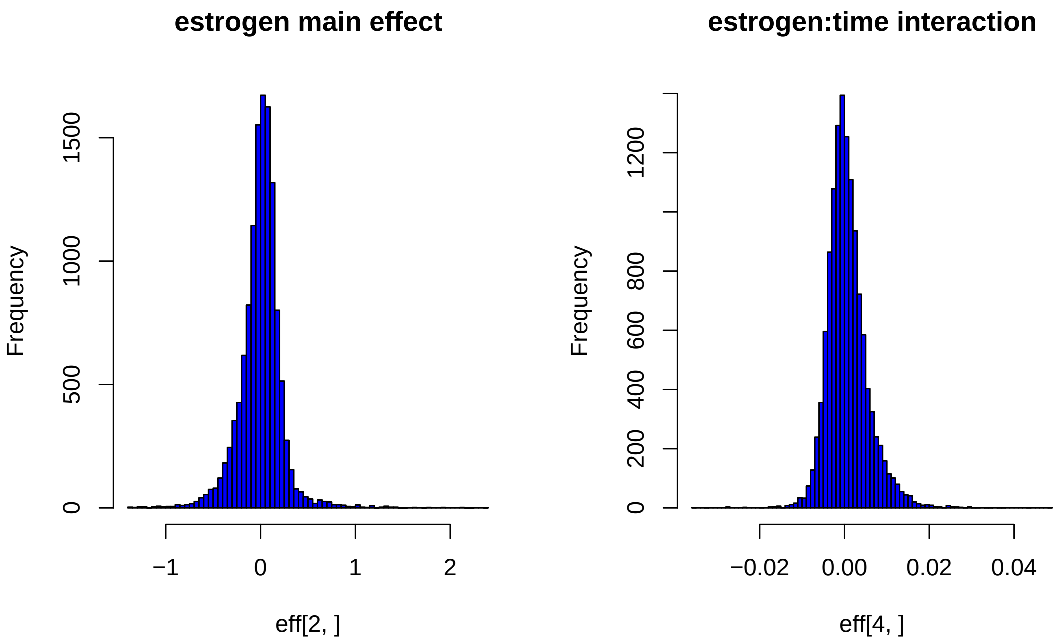

Figure 6: see exercise 9.

```
> hist(eff[2, ], breaks = 100, col = "blue", main = "estrogen main effect")
> sel = order(eff[2, ], decreasing = FALSE)[1:3]
> genename[sel]

$"846_s_at"
[1] " BCL2-antagonist/killer 1"

$"36617_at"
[1] " inhibitor of DNA binding 1, dominant negative helix-loop-helix protein"

$"37294_at"
[1] " B-cell translocation gene 1, anti-proliferative"

> sel = order(eff[2, ], decreasing = TRUE)[1:3]
> genename[sel]

$"910_at"
[1] " thymidine kinase 1, soluble"

$"31798_at"
[1] " trefoil factor 1 (breast cancer, estrogen-inducible sequence expressed in)"

$"1884_s_at"
[1] " proliferating cell nuclear antigen"

> hist(eff[4, ], breaks = 100, col = "blue", main = "estrogen:time interaction")
> sel <- order(eff[4, ], decreasing = TRUE)[1:3]
> genename[sel]
```

```
$"1651_at"
[1] " ubiquitin-conjugating enzyme E2C"

$"40412_at"
[1] " pituitary tumor-transforming 1"

$"1945_at"
[1] " cyclin B1"
```