Estrogen 2x2 factorial design

Bioconductor Workshop DSC/Vienna 19 Mar 2003

Robert Gentleman, Wolfgang Huber

1.) Preliminaries. To go through this exercise, you need to have installed R>=1.6, recent versions of the Bioconductor libraries Biobase, affy, hgu95av2, ..., and the library estrogen, which contains the data.

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

2.) Load the data.

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

```
> datadir = file.path(.path.package("estrogen"), "data")
> datadir
[1] "/scratch/homes/whuber/R/library/estrogen/data"
> dir(datadir)
 [1] "00Index"       "bad.cel"       "e10-1.cel"     "E10-1.cel"
 [5] "e10-2.cel"     "E10-2.cel"     "e48-1.cel"     "E48-1.cel"
 [9] "e48-2.cel"     "E48-2.cel"     "phenoData.txt"
```

The function `.path.package` here is used to find the directory of the estrogen example package on your computer's harddisk. If you had your own data, you would have to specify the appropriate path. The function `file.path` combines this path prefix with the name of subdirectory `data`. Finally, the function `dir` lists the content of this directory.

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` objects

```
> pd = read.phenoData(file.path(datadir, "phenoData.txt"), header = TRUE,
+     row.names = 1)
> pData(pd)
          estrogen time.h
e10-1.cel   absent     10
e10-2.cel   absent     10
E10-1.cel  present     10
E10-2.cel  present     10
e48-1.cel   absent     48
e48-2.cel   absent     48
E48-1.cel  present     48
E48-2.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 = file.path(datadir, 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
notes=
```

d. The slot `annotation` has been set to a default, the name of the genechips' CDF file. The actual name of the annotation package is `hgu95a`, so let us change it.

```
> a@annotation = "hgu95a"
```

3.) Normalization.

a. Now we can use the function `express` to normalize the data and calculate expression values. By default, it uses a combination of methods called *RMA*. A number variations are possible, depending on the type of your data and the goals of your subsequent analysis.

```
> x = express(a)
> 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
```

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

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

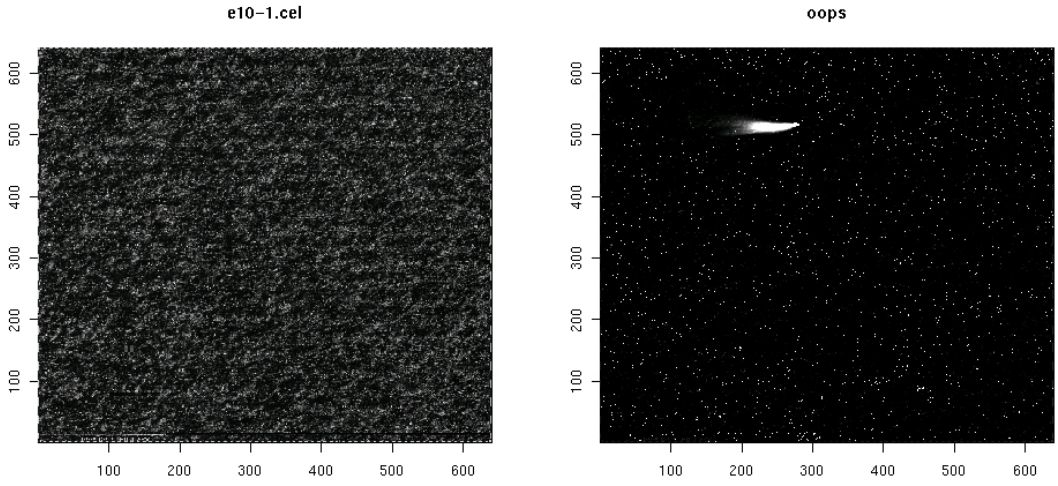

Figure 1:

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)
```

But we have another example:

```
> badc = read.celfile(file.path(datadir, "bad.cel"))
> image(badc, main = "oops")
```

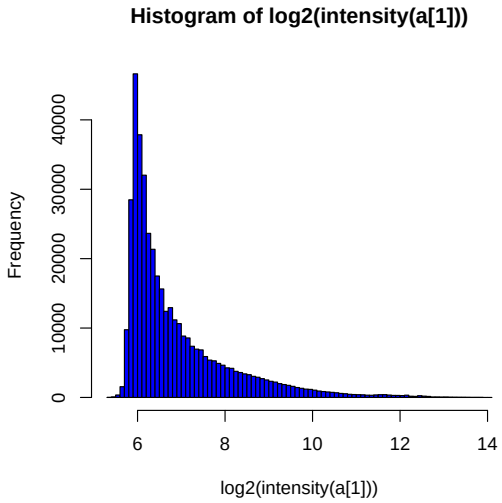

Figure 2:

5.) Histograms. Another way to visualize what is going on on a chip is to look at the histogram 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[1])), breaks = 100, col = "blue")
```

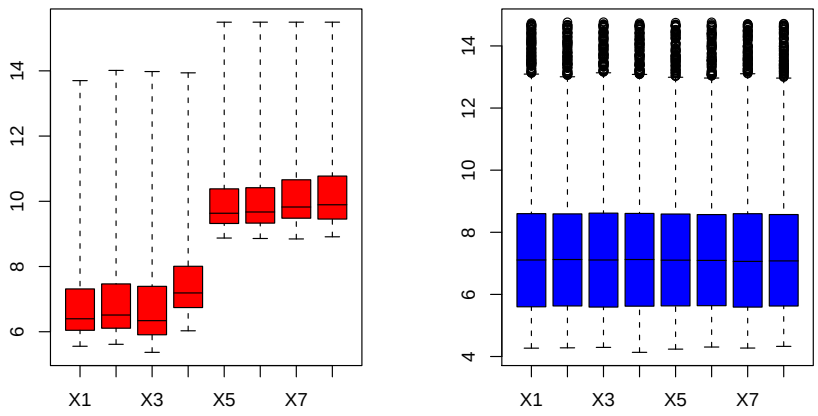

Figure 3:

```
> pData(pd)

          estrogen time.h
e10-1.cel   absent     10
e10-2.cel   absent     10
E10-1.cel  present     10
E10-2.cel  present     10
e48-1.cel   absent     48
e48-2.cel   absent     48
E48-1.cel  present     48
E48-2.cel  present     48
```

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")
```

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(intensity(a)[, 1:2], log = "xy", pch = ".", main = "all")
> pm = unlist(pmindex(a))
> plot(intensity(a)[pm, 1:2], log = "xy", pch = ".", main = "pm")
> mm = unlist(mmindex(a))
> plot(intensity(a)[mm, 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):

```
> library(mva)
> rsd = apply(exprs(x), 1, sd)
> sel = order(rsd, decreasing = TRUE)[1:50]
> heatmap(exprs(x)[sel, ], col = gentlecol(256))
```

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

> coef = esApply(x, 1, lm.coef)
```

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

```
> dim(coef)

[1]     4 12625

> rownames(coef)

[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.

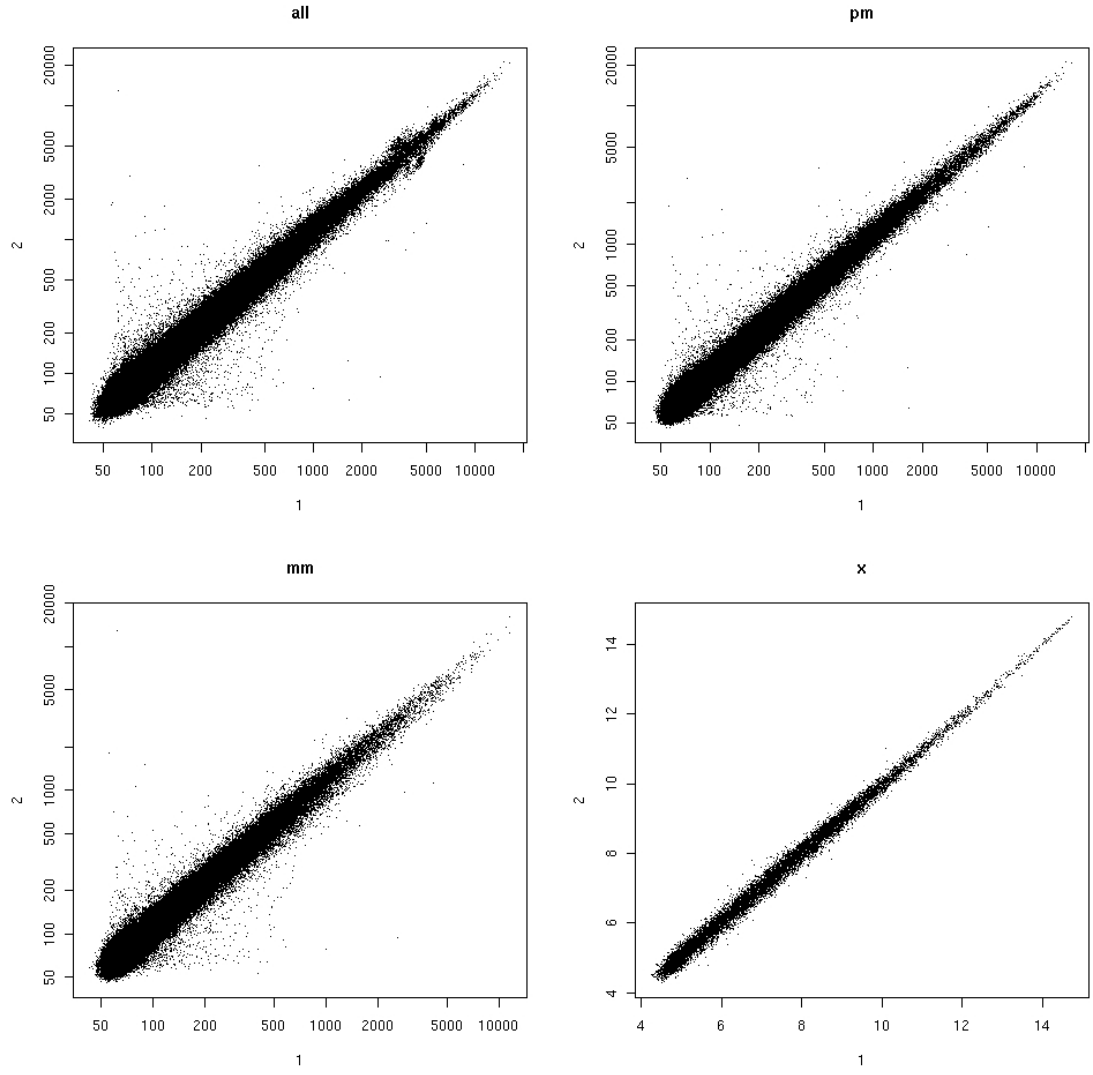
all
pm
mm
x
1
2
50
100
200
500
1000
2000
5000
10000
20000
4
6
8
10
12
14

Figure 4:

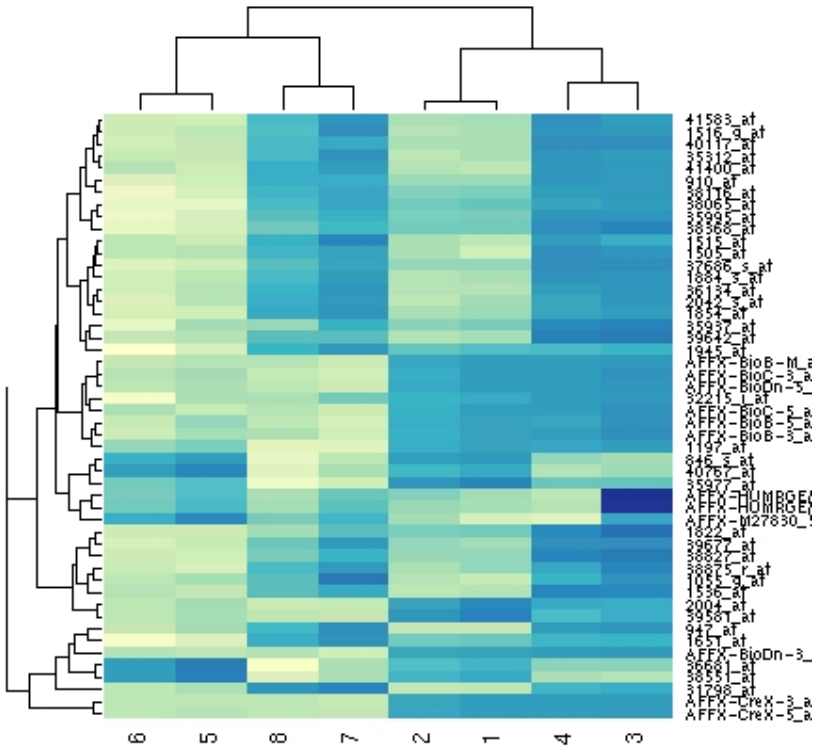

Figure 5:

```
> library(hgu95a)
> vendorids = ls(env = hgu95aGENENAME)
> genename = unlist(multiget(vendorids, env = hgu95aGENENAME))
```

Let's now first look at the **estrogen main effect**, and print the top 5 genes with largest effect in one direction, as well as in the other direction. Then, look at the **estrogen:time interaction**.

```
> par(mfrow = c(1, 2))
> hist(coef[2, ], breaks = 100, col = "blue", main = "coef. estrogen main effect")
> sel = order(coef[2, ], decreasing = FALSE)[1:3]
> paste(genename[sel])

[1] "selenoprotein P, plasma, 1"
[2] "cyclin G2"
[3] "B-cell translocation gene 1, anti-proliferative"

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

[1] "elongation of very long chain fatty acids (FEN1/Elo2, SUR4/Elo3, yeast)-like 2"
[2] "anterior gradient 2 homolog (Xenepus laevis)"
[3] "thymidine kinase 1, soluble"
```

```
> hist(coef[4, ], breaks = 100, col = "blue", main = "coef. estrogen:time interaction")
> paste(genename[order(coef[4, ], decreasing = TRUE)[1:3]])
```

```
[1] "cyclin B1"                          "ubiquitin-conjugating enzyme E2C"
[3] "chromosome 20 open reading frame 1"
```