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

June 2004

Robert Gentleman, Wolfgang Huber

1.) Preliminaries. To go through this exercise, you need to have installed R>=1.9.0, the libraries Biobase, affy, hgu95av2.db, hgu95av2cdf, and vsn from the Bioconductor release 1.4.

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

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] "/tmp/Rtmpc82ACa/Rinst51201af1a10f/estrogen/extdata"
> dir(datadir)
 [1] "bad.cel"       "estrogen.txt"  "high10-1.cel"  "high10-2.cel"
 [5] "high48-1.cel"  "high48-2.cel"  "low10-1.cel"   "low10-2.cel"
 [9] "low48-1.cel"   "low48-2.cel"   "phenoData.txt" "targLimma.txt"
> setwd(datadir)
```

The function `system.file` here is used to find the subdirectory `extdata` of the estrogen package on your computer's harddisk. To use your own data, set `datadir` to the appropriate path instead.

b. The file `estrogen.txt` contains information on the samples that were hybridized onto the arrays. Look at it in a text editor. Again, to use your own data, you need to preprare a similar file with the appropriate information on your arrays and samples. To load it into a `phenoData` object

```
> pd = read.AnnotatedDataFrame("estrogen.txt", header=TRUE, sep="", 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 as well as the phenotypic data into an `AffyBatch` object.

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

```
> a

AffyBatch object
size of arrays=640x640 features (19 kb)
cdf=HG_U95Av2 (12625 affyids)
number of samples=8
number of genes=12625
annotation=hgu95av2
notes=
```
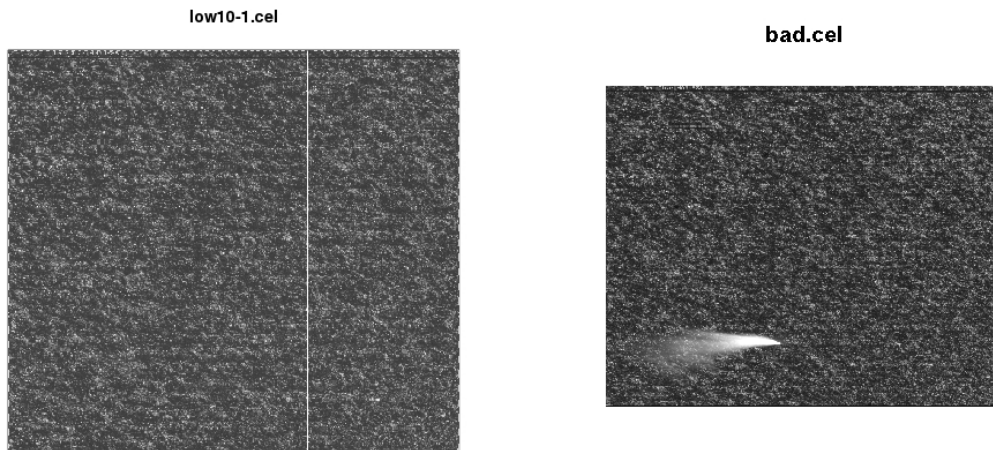

Figure 1: see exercise 4.

3.) Normalization.

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

```
> x <- expresso(a,
+               bg.correct       = FALSE,    ## bg correction is done by vsn
+               normalize.method = "vsn",
+               normalize.param  = list(subsample=1000),
+               pmcorrect.method = "pmonly",
+               summary.method   = "medianpolish")
> x
ExpressionSet (storageMode: lockedEnvironment)
assayData: 12625 features, 8 samples
  element names: exprs, se.exprs
protocolData
  sampleNames: low10-1.cel low10-2.cel ... high48-2.cel (8 total)
  varLabels: ScanDate
  varMetadata: labelDescription
phenoData
  sampleNames: low10-1.cel low10-2.cel ... high48-2.cel (8 total)
  varLabels: estrogen time.h
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation: hgu95av2
```

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.

b. What are other available methods for *normalization*, and *expression value calculation*? You can consult the vignettes for the `affy` package for this. Choose another method (for example, MAS5 or RMA) and compare the results. For example, look at scatterplots of the probe set summaries for the same arrays between different methods.

```
> normalize.methods(a)
[1] "constant"       "contrasts"       "invariantset"     "loess"
[5] "methods"        "qspline"         "quantiles"        "quantiles.robust"
[9] "vsn"
> express.summary.stat.methods()
[1] "avgdiff"        "liwong"        "mas"            "medianpolish" "playerout"
```

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)
```

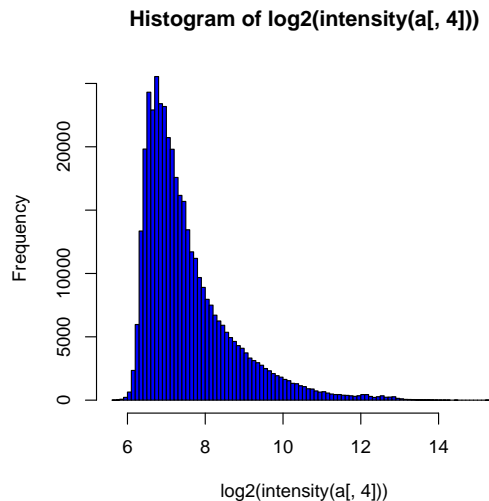Note that in these images, row 1 is at the bottom, and row 640 at the top.

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):
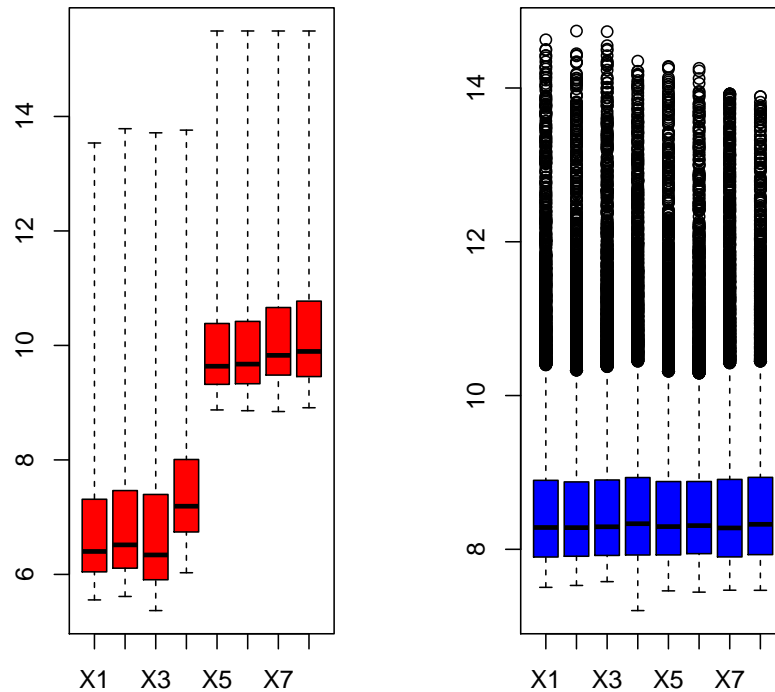
Figure 3: see exercise 6.

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

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] "ExpressionSet"
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): Why are the arrays that were made at $t = 48$h much brighter than those at $t = 10$h? Look at histograms and scatterplots of the probe intensities from chips at 10h and at48h to see whether you can find any evidence of saturation, changes in experimental
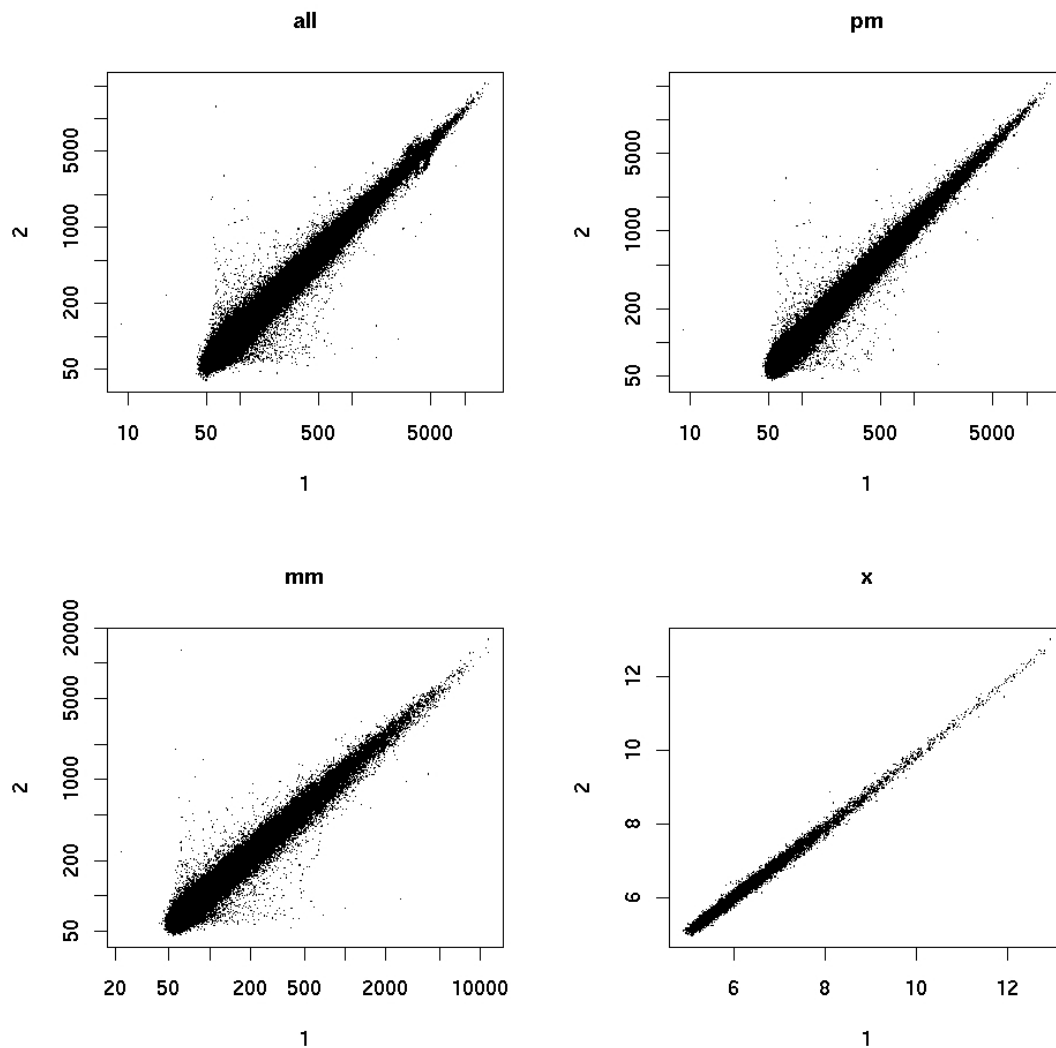
Figure 4: see exercise 7.

protocol, or other quality problems. Distinguish between probes that are supposed to represent genes (you can access these e.g. through the functions `pm()`) and control probes.

```
> 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))
```
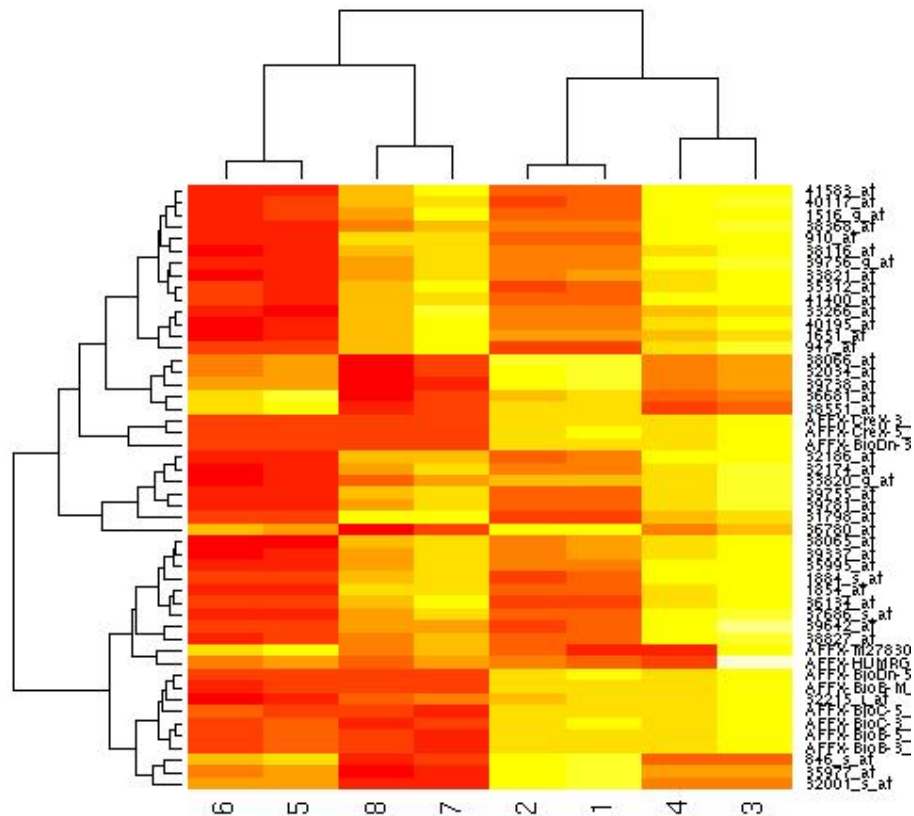
Figure 5: see exercise 8.

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)
```

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

```
> affyids <- colnames(eff)
```

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

estrogen main effect
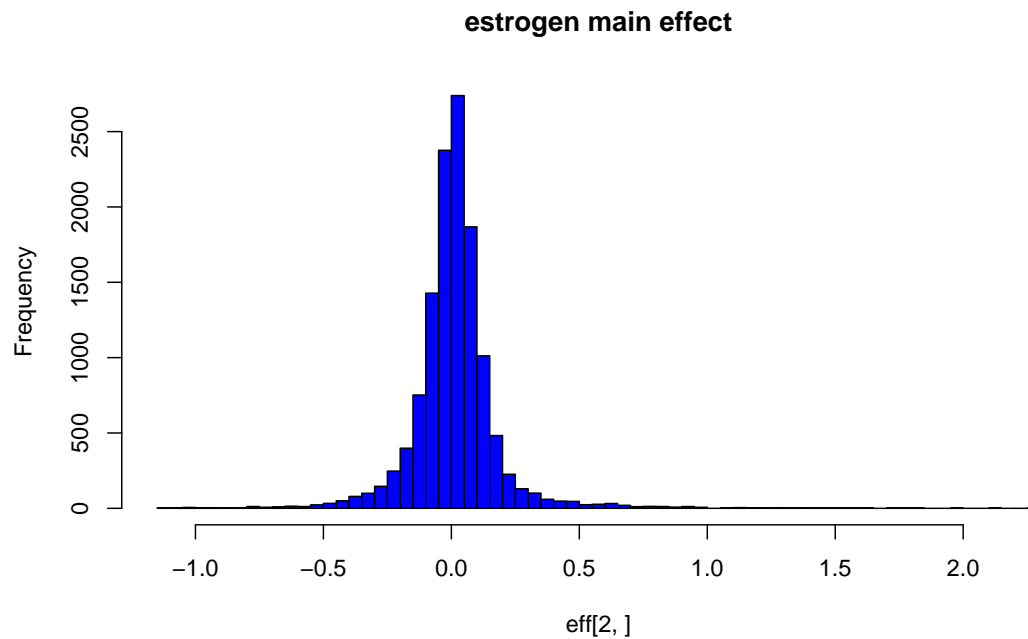

Figure 6: see exercise 9.

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

 [1] "hgu95av2"              "hgu95av2.db"           "hgu95av2ACCNUM"
 [4] "hgu95av2ALIAS2PROBE"   "hgu95av2CHR"           "hgu95av2CHRLENGTHS"
 [7] "hgu95av2CHRLOC"        "hgu95av2CHRLOCEND"     "hgu95av2ENSEMBL"
[10] "hgu95av2ENSEMBL2PROBE" "hgu95av2ENTREZID"      "hgu95av2ENZYME"
[13] "hgu95av2ENZYME2PROBE"  "hgu95av2GENENAME"      "hgu95av2GO"
[16] "hgu95av2GO2ALLPROBES"  "hgu95av2GO2PROBE"      "hgu95av2MAP"
[19] "hgu95av2MAPCOUNTS"     "hgu95av2OMIM"          "hgu95av2ORGANISM"
[22] "hgu95av2ORGPKG"        "hgu95av2PATH"          "hgu95av2PATH2PROBE"
[25] "hgu95av2PFAM"          "hgu95av2PMID"          "hgu95av2PMID2PROBE"
[28] "hgu95av2PROSITE"       "hgu95av2REFSEQ"        "hgu95av2SYMBOL"
[31] "hgu95av2UNIGENE"       "hgu95av2UNIPROT"       "hgu95av2_dbInfo"
[34] "hgu95av2_dbconn"       "hgu95av2_dbfile"       "hgu95av2_dbschema"
```

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**.

```
> lowest <- sort(eff[2,], decreasing=FALSE)[1:3]
> mget(names(lowest), hgu95av2GENENAME)

$`37405_at`
[1] "selenium binding protein 1"

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

```
$`37043_at`
[1] "inhibitor of DNA binding 3, dominant negative helix-loop-helix protein"

> highest <- sort(eff[2,], decreasing=TRUE)[1:3]
> mget(names(highest), hgu95av2GENENAME)

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

$`31798_at`
[1] "trefoil factor 1"

$`40117_at`
[1] "minichromosome maintenance complex component 6"

> hist(eff[4,], breaks=100, col="blue", main="estrogen:time interaction")
> highia <- sort(eff[4,], decreasing=TRUE)[1:3]
> mget(names(highia), hgu95av2GENENAME)

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

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

$`34852_g_at`
[1] "aurora kinase A"
```