— Exploring Affymetrix Data —

Robert Gentleman, Wolfgang Huber

Practical DNA Microarray Analysis, Berlin, 2006 Feb 27 - Mar 02
http://compdiag.molgen.mpg.de/ngfn/pma2006feb.shtml

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

```
> 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] "/project/gene_expression/lib/bioconductor/release_1.7/data/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" "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. 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.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 Package 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 (25606 kb)
cdf=HG_U95Av2 (12625 affyids)
number of samples=8
number of genes=12625
annotation=hgu95av2
```

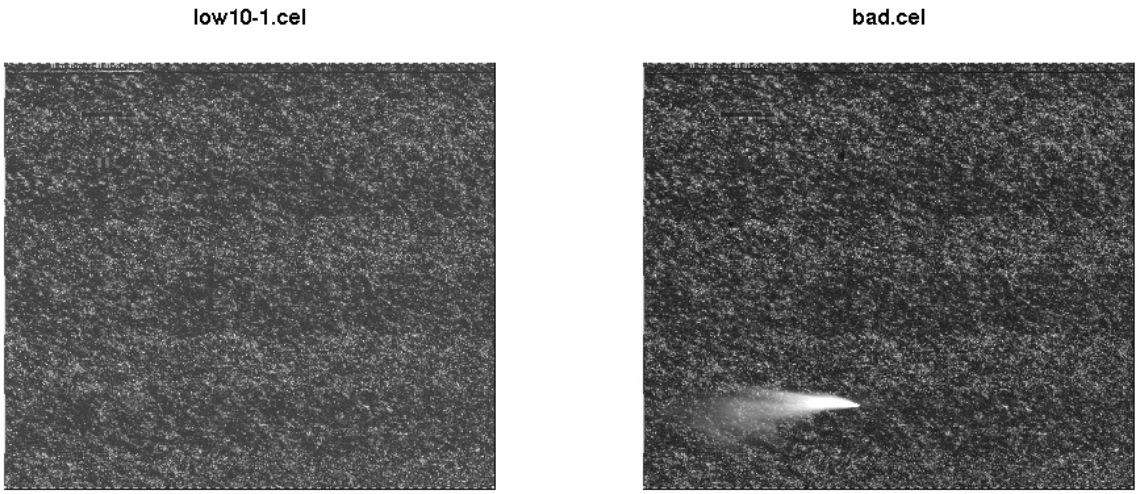

Figure 1: *see exercise 4.*

3.) 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.

4.) 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 dynamic 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")
```

5.) Normalization.

a. Before comparing data from different arrays the probe-level data has to be summarized to represent expression levels per gene and intensities have to be normalized between different arrays. We can use the function `expresso` to choose between different methodds to normalize the data and calculate expression values.

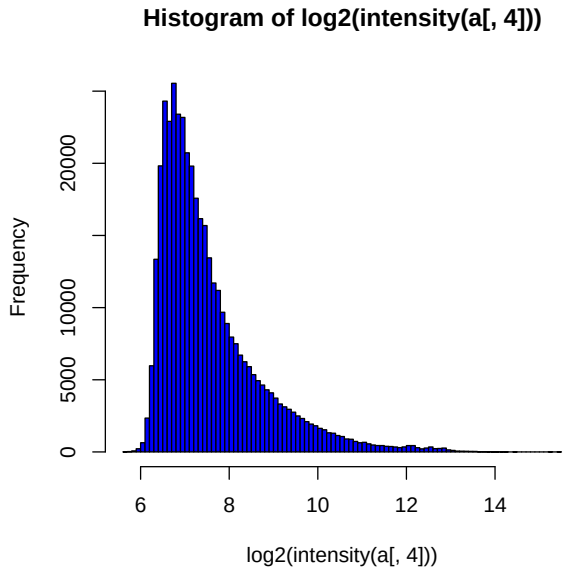

Figure 2: *see exercise 5.*

```
> x <- expresso(a,
                bg.correct = FALSE,
                normalize.method = "vsn",
                normalize.param = list(subsample = 1000),
                pmcorrect.method = "pmonly",
                summary.method = "medianpolish")
```

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). Do not bother, the variable `x` is already available in the current workspace of R, so you can simply continue with the next commands.
`workspace.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*? 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] "qspline"  "quantiles" "quantiles.robust" "vsn"
> express.summary.stat.methods
[1] "avgdiff" "liwong" "mas" "medianpolish" "playerout"
```

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")
```

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))`?

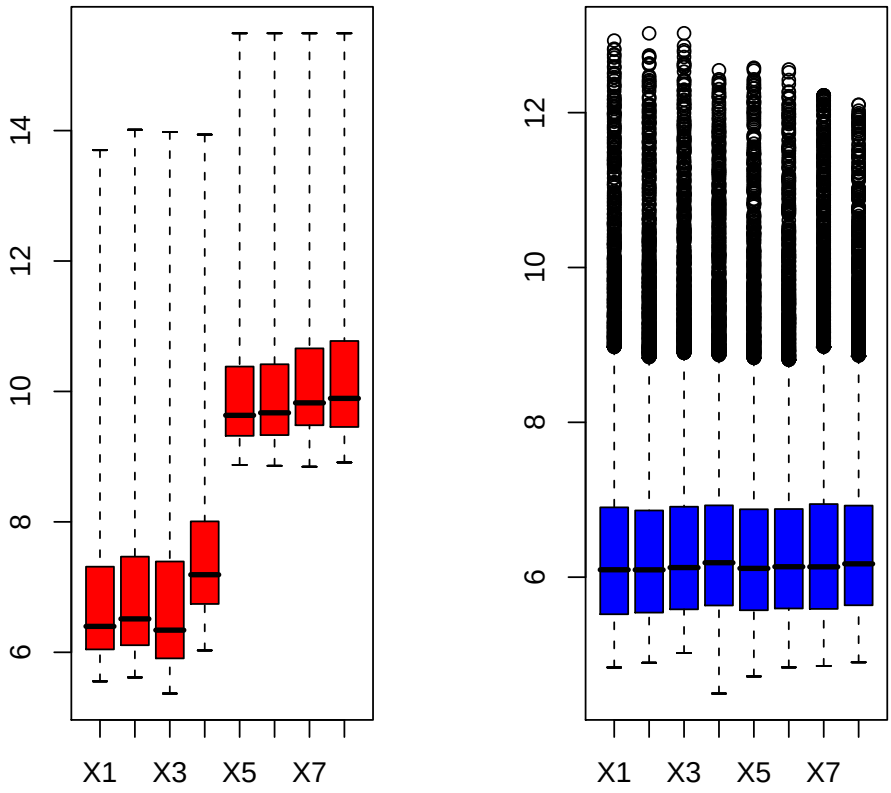

Figure 3: *see exercise 6.*

```
> 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): 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 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")
```

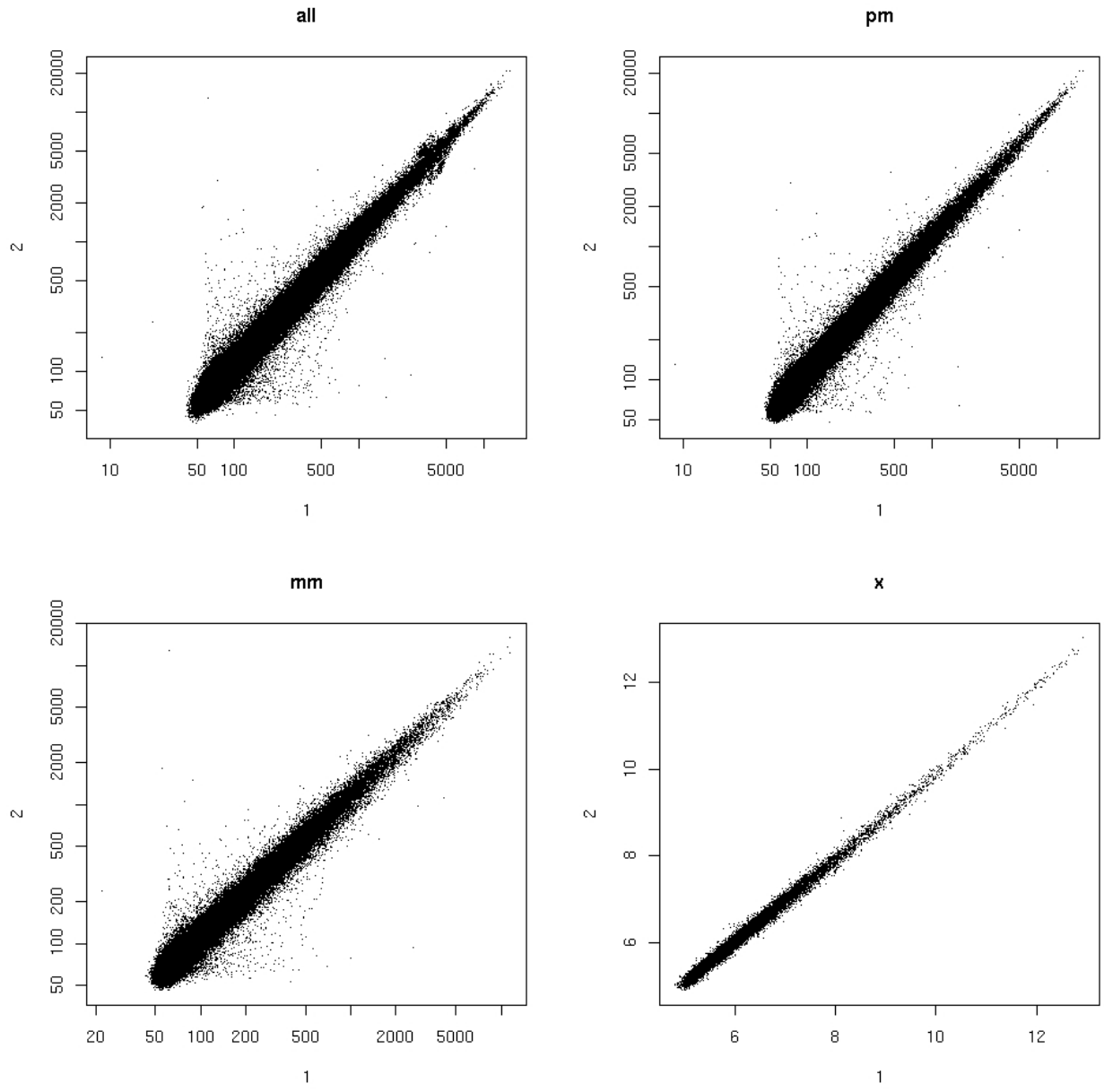

Figure 4: *see exercise 7.*

8.) Heatmap. Heatmaps are a quick and easy way to visualize and see structures in medium sized tables of data. Here we do a rough selection of the 50 genes with the highest variation (standard deviation) across chips. (see Fig. 5):

```
> rsd <- apply(exprs(x), 1, sd)
> sel <- order(rsd, decreasing = TRUE)[1:50]
```

```
> heatmap(exprs(x)[sel, ], col = gentlecol(256))
```

9.) ANOVA.

A better way to select genes is to estimate the variation between different types of experiments in the data. One way to do this is ANOVA. 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. Now we can start analysing our data for biological effects.

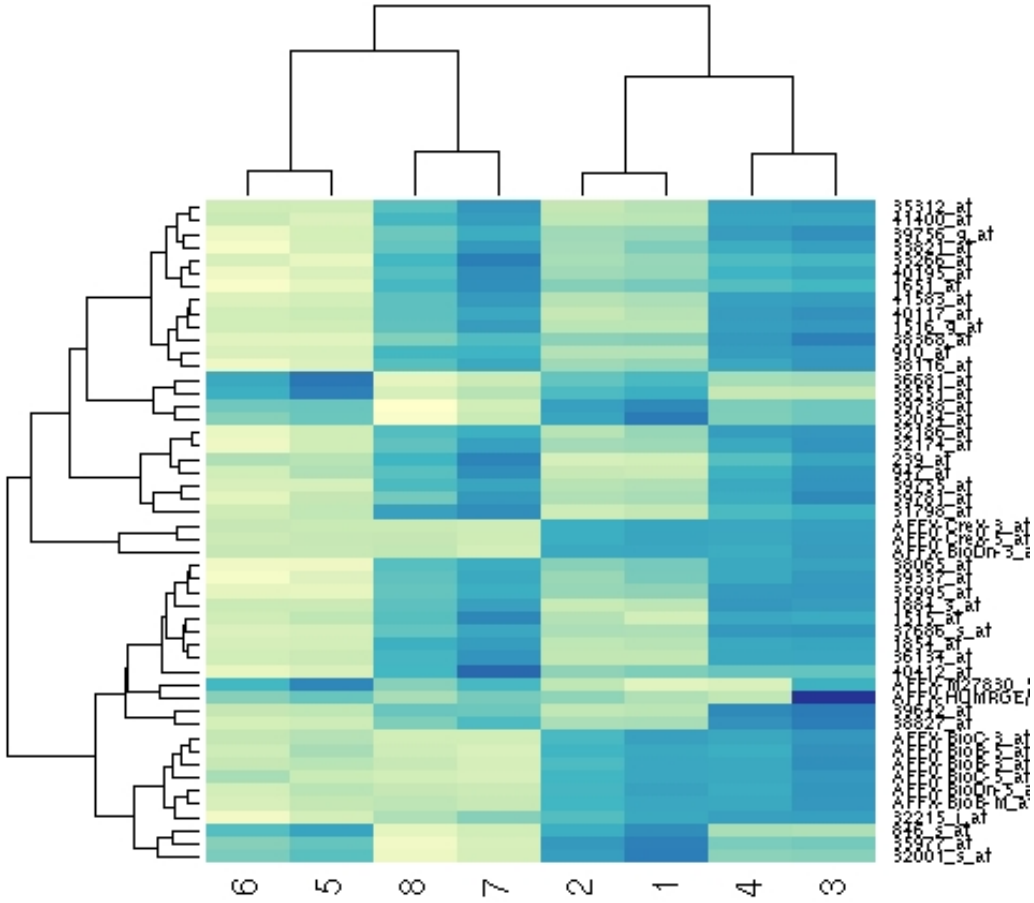

Figure 5: *see exercise 8.*

```
> 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" "estrogenpresent:time.h"
```

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

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"           "hgu95av2CHRLENGTHS"
 [5] "hgu95av2CHRLOC"        "hgu95av2ENZYME"        "hgu95av2ENZYME2PROBE" "hgu95av2GENENAME"
 [9] "hgu95av2GO"            "hgu95av2GO2ALLPROBES" "hgu95av2GO2PROBE"     "hgu95av2LOCUSID"
```

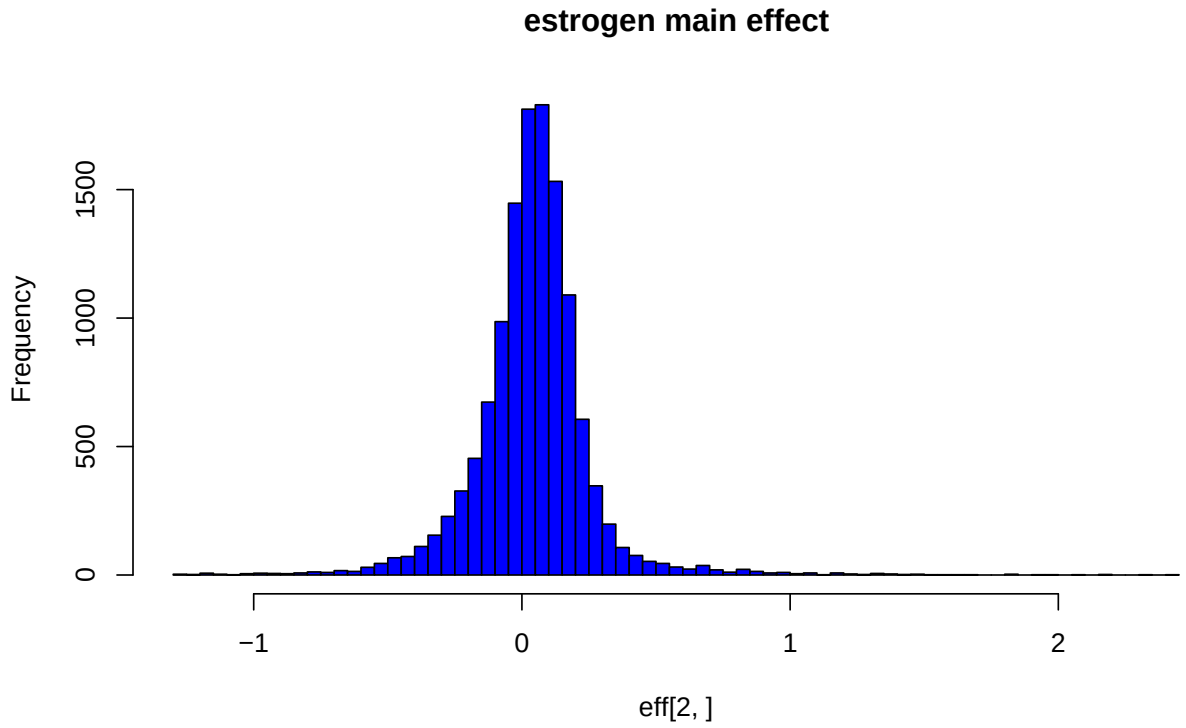

Figure 6: *see exercise 9.*

```
[13] "hgu95av2MAP"          "hgu95av2MAPCOUNTS"    "hgu95av2OMIM"          "hgu95av2ORGANISM"
[17] "hgu95av2PATH"         "hgu95av2PATH2PROBE"   "hgu95av2PFAM"          "hgu95av2PMID"
[21] "hgu95av2PMID2PROBE"   "hgu95av2PROSITE"      "hgu95av2QC"            "hgu95av2QCDATA"
[25] "hgu95av2REFSEQ"       "hgu95av2SUMFUNC"      "hgu95av2SYMBOL"        "hgu95av2UNIGENE"
```

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)

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

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

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

> 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 (breast cancer, estrogen-inducible sequence expressed in)"
```

```
$"40117_at"
[1] "MCM6 minichromosome maintenance deficient 6 (MIS5 homolog, S. pombe) (S. cerevisiae)"

> 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"

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