Data Analysis and Modeling Methods

# Microarray Technology (Affymetrix©) and Analysis

# Practicals

B. Haibe-Kains[1,2] and G. Bontempi[2]

[1]Unité Microarray, Institut Jules Bordet
[2]Machine Learning Group, Université Libre de Bruxelles

November 28, 2006

## 1 R and Bioconductor

R is available from `http://www.r-project.org` and the Bioconductor packages are available from `http://www.bioconductor.org`. For the further exercises, you need the *affy*, *Heatplus*, *gplot*, *scatterplot3d*, *amap* and *survival* libraries. If you have problem about basic R functions (matrix manipulations, function writing, etc), you can consult the R manuals available from `http://cran.r-project.org/manuals.html`.
Launch R and load the affy library (this command will load automatically the dependencies).

```
> library(affy)
```

### 1.1 Help

Here are three ways to access help under R :

- you can search a specific keyword with the function *help.search(pattern)*.

  ```
  > help.search("rma")
  ```

- you can access to the help page of a specific function by the function *?*.

  ```
  > `?`(just.rma)
  ```

- you can access to help (in pdf format) for all Bioconductor loaded libraries by typing

  ```
  > openVignette()
  ```

## 2 Read the CEL Files

You can download the CEL and the demographics files from `http://www.ulb.ac.be/di/map/bhaibeka/bioinfo_courses/celfiles_test.zip`. It is a ZIP archive. Decompress it and put all the CEL files in a directory (referred as `path_to_cel_files`) and the demographics file in another directory (referred as `path_to_demo`).

```
> fn <- list.celfiles(path = "path_to_cel_files", full.names = TRUE)
> fn2 <- list.celfiles(path = "path_to_cel_files", full.names = FALSE)
> phenod <- read.csv("path_to_demo/demo.csv")
> dimnames(phenod)[[1]] <- as.character(phenod[, 1])
> if (!all(phenod[, "filename"] == fn2)) {
+     stop("the demographics and the file names are not in the same order!")
+ }
> phenod <- new("phenoData", pData = phenod, varLabels = list("samplename",
+     "filename", "grade", "t.rfs", "e.rfs", "tumor.size"))
> abatch <- read.affybatch(filenames = fn, phenoData = phenod,
+     verbose = TRUE)

1 reading path_to_cel_files/OXFU_HG1_1183 A.CEL ...instantiating an AffyBatch (intensity a
Reading in : path_to_cel_files/OXFU_HG1_1183 A.CEL
Reading in : path_to_cel_files/OXFU_HG1_1210 A.CEL
Reading in : path_to_cel_files/OXFU_HG1_1248 A.CEL
Reading in : path_to_cel_files/OXFU_HG3_138 A.CEL
Reading in : path_to_cel_files/OXFU_HG3_157 A.CEL
Reading in : path_to_cel_files/OXFU_HG3_223 A.CEL
```

You can print information about your `AffyBatch` object.

```
> print(abatch)

AffyBatch object
size of arrays=712x712 features (23772 kb)
cdf=HG-U133A (22283 affyids)
number of samples=6
number of genes=22283
annotation=hgu133a

> print(sampleNames(abatch))

[1] "OXFU_HG1_1183" "OXFU_HG1_1210" "OXFU_HG1_1248" "OXFU_HG3_138"
[5] "OXFU_HG3_157"  "OXFU_HG3_223"

> print(geneNames(abatch)[1:10])

 [1] "1007_s_at" "1053_at"   "117_at"    "121_at"    "1255_g_at" "1294_at"
 [7] "1316_at"   "1320_at"   "1405_i_at" "1431_at"
```

You can see that your CEL files coming from hgu133a chips which contain $712*712$ probes. These probes will be summarized in 22283 probesets.

## 2.1 Probe Intensities

The AffyBatch class is just an extension of the ExprSet class (see the presentation[1] for details). So, you can access to the matrix containing all the intensities at the probe level with the function `intensity(object)`. The matrix contains one probe per row and one chip per column. You can access at the PM or the MM intensities for one or several probe sets with the function `probeset(object, genenames)`.

```
> my.probesets <- c("1007_s_at", "1053_at")
> r <- probeset(abatch, genenames = my.probesets)
> print(r)

$`1007_s_at`
ProbeSet object:
  id=1007_s_at
  pm= 16 probes x  6  chips

$`1053_at`
ProbeSet object:
  id=1053_at
  pm= 16 probes x  6  chips

> r[[1]]@id

[1] "1007_s_at"

> r[[1]]@pm[1:3, 1:3]

     OXFU_HG1_1183 OXFU_HG1_1210 OXFU_HG1_1248
[1,]        1213.3         764.5         912.0
[2,]        2203.3        2164.0        3369.5
[3,]        1983.3        1516.5        2391.8

> r[[1]]@mm[1:3, 1:3]

     OXFU_HG1_1183 OXFU_HG1_1210 OXFU_HG1_1248
[1,]         334.5         382.3         321.5
[2,]        1772.0        1903.3        2153.3
[3,]        2586.8        2310.0        3020.5

> r[[1]]@pm[1:3, 1:3] - r[[1]]@mm[1:3, 1:3]

     OXFU_HG1_1183 OXFU_HG1_1210 OXFU_HG1_1248
[1,]         878.8         382.2         590.5
[2,]         431.3         260.7        1216.2
[3,]        -603.5        -793.5        -628.7
```

[1]The presentation is available from `http://www.ulb.ac.be/di/map/bhaibeka/bioinfo_courses/microarray_hkb.pdf`.

### 2.2 Phenotype Information

Like the exprSet class, the AffyBatch class has a phenoData slot containing the information of the microarray experiments. You can access and modify this information with the function *pData(object)*. This information is a matrix.

```
> pData(abatch)
```

```
                samplename            filename grade t.rfs e.rfs tumor.size
OXFU_HG1_1183 OXFU_HG1_1183 OXFU_HG1_1183 A.CEL     1  1591     0        0.8
OXFU_HG1_1210 OXFU_HG1_1210 OXFU_HG1_1210 A.CEL     1  1920     0        0.8
OXFU_HG1_1248 OXFU_HG1_1248 OXFU_HG1_1248 A.CEL     1  3262     0        2.0
OXFU_HG3_138   OXFU_HG3_138  OXFU_HG3_138 A.CEL     3  5029     0        3.0
OXFU_HG3_157   OXFU_HG3_157  OXFU_HG3_157 A.CEL     3  4926     1        2.0
OXFU_HG3_223   OXFU_HG3_223  OXFU_HG3_223 A.CEL     3  1875     1        2.1
```

## 3 RMA procedure

You can perform the RMA procedure with :

- only background correction and gene expression summarization (log2)

```
> eset.bkg <- rma(abatch, background = TRUE, normalize = FALSE,
+     verbose = TRUE)

Background correcting
Calculating Expression
```

- background correction, normalization and gene expression summarization (log2)

```
> eset.bkg.norm <- rma(abatch, background = TRUE, normalize = TRUE,
+     verbose = TRUE)

Background correcting
Normalizing
Calculating Expression
```

**Remark** Because of the high memory consumption of an AffyBatch object, you can use the function `just.rma()`.

- only background correction and gene expression summarization (log2)

```
> eset.bkg <- just.rma(filenames = fn, phenoData = phenod, background = TRUE,
+     normalize = FALSE, verbose = TRUE)

Background correcting
Calculating Expression
```

- background correction, normalization and gene expression summarization (log2)

```
> eset.bkg.norm <- just.rma(filenames = fn, phenoData = phenod,
+     background = TRUE, normalize = TRUE, verbose = TRUE)

Background correcting
Normalizing
Calculating Expression
```

### 3.1 Gene Expression

As previously mentioned for the AffyBatch class, you can access to the gene expressions with the function `exprs(object)`. The matrix contains the observed expression levels. The columns represent patients or cases and rows represent genes.

```
> exprs(eset.bkg.norm)[1:3, 1:3]

          OXFU_HG1_1183 OXFU_HG1_1210 OXFU_HG1_1248
1007_s_at     11.034252     11.341137     11.742761
1053_at        7.495097      7.054914      7.182206
117_at         7.495658      6.989012      7.150119

> my.probesets <- c("212581_x_at", "213453_x_at", "217398_x_at")
> gapdh <- exprs(eset.bkg.norm)[my.probesets, ]
> apply(gapdh, 1, mean)

212581_x_at 213453_x_at 217398_x_at
   13.81652    13.65501    13.72684

> apply(gapdh, 1, var)

212581_x_at 213453_x_at 217398_x_at
 0.05962283  0.05597977  0.13816952
```

## 4 Unsupervised Analysis

Once we have normalized the raw microarray data, we can perform some high-level analyses. Firstly we can do an unsupervised analysis to assess if there is a "structure" in the data.

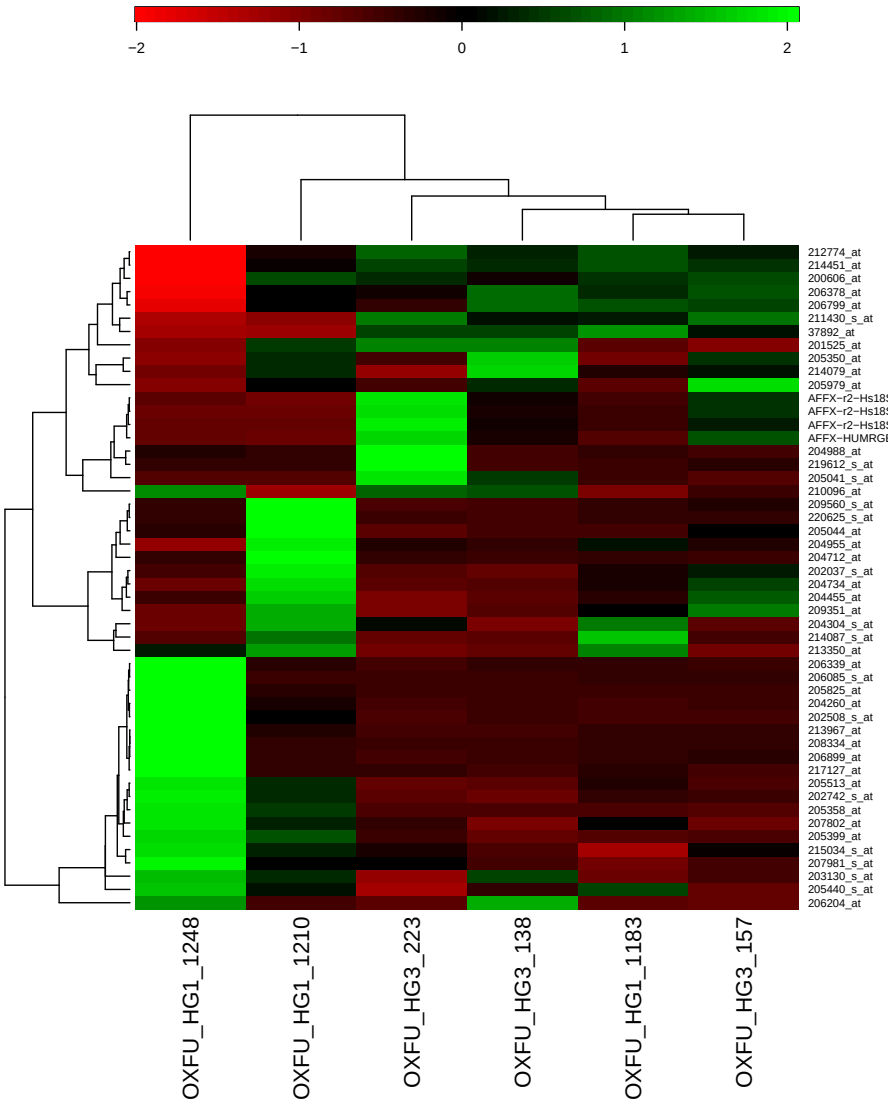
### 4.1 Hierarchical Clustering

The following code performs a hierarchical clustering with two parameters :

- the distance between two cases is the Pearson's correlation coefficient
- the linkage, i.e. the way to compute the distance between a case and a cluster is the average method.

```
> library(amap)
> library(Heatplus)
> grade <- pData(phenod)[, "grade"]
```

```
> tumor.size <- pData(phenod)[, "tumor.size"]
> myeset <- exprs(eset.bkg.norm)
> filt.ix <- order(apply(myeset, 1, var), decreasing = TRUE)[1:50]
> myeset.filt <- myeset[filt.ix, ]
> myhclust <- function(x) {
+     return(hclust(x, method = "average"))
+ }
> mydist <- function(x) {
+     return(Dist(x, method = "pearson"))
+ }
> heatmap_2(myeset.filt, distfun = mydist, hclustfun = myhclust,
+     col = RGBColVec(64), legend = 3, legfrac = 12)
```




You should note that the gene expressions are scaled to see down-regulation (negative gene expression) and up-regulation (positive gene expression) of a gene. The scaling is performed in subtracting the mean of each row.

## 4.2 Principal Components Analysis

Using the PCA method, we can take the first three principal components, i.e. the components explaining the maximum of gene expression variance, and plot the cases in this space.

Th code performs the following steps:

1. computes all the principal components,
2. takes the first three
3. generates a pdf file (called `mypca.pdf`) in the current directory to scatter the cases in 3D.

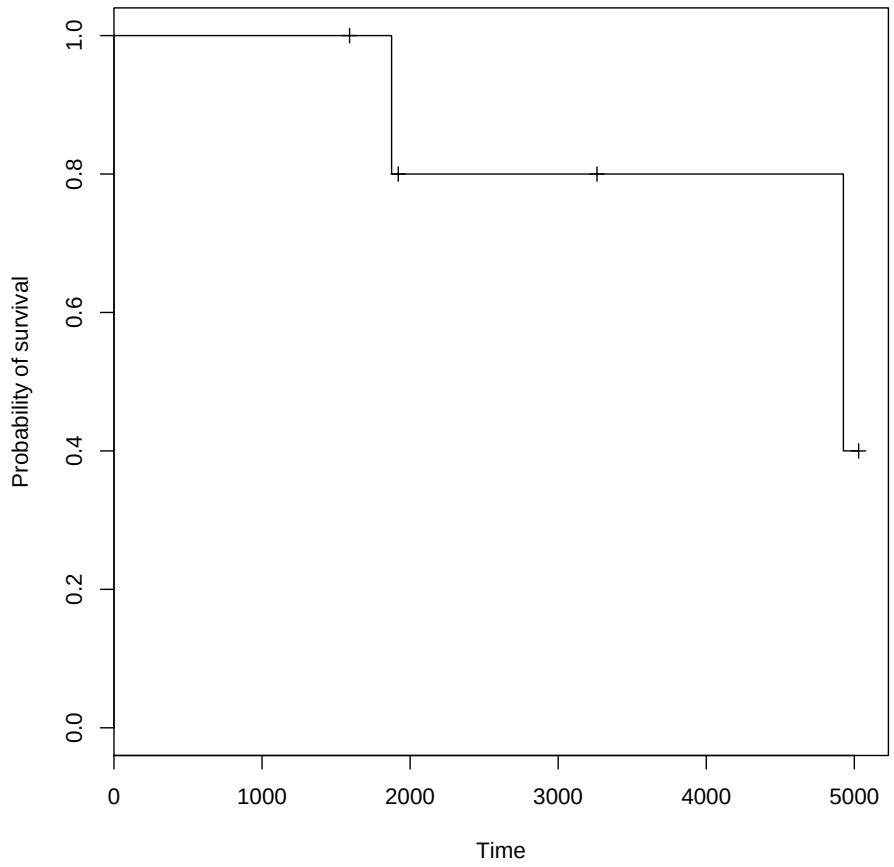
```
> library(scatterplot3d)
> library(gplots)
> mypc <- prcomp(x = t(myeset.filt), center = TRUE, scale = TRUE)
> summary(mypc)

Importance of components:
                        PC1   PC2   PC3    PC4    PC5      PC6
Standard deviation     5.0 3.689 2.289 1.9079 1.5762 1.40e-15
Proportion of Variance 0.5 0.272 0.105 0.0728 0.0497 0.00e+00
Cumulative Proportion  0.5 0.773 0.878 0.9503 1.0000 1.00e+00

> cc2 <- grade + 2
> pdf("mypca.pdf", width = 7, height = 7, version = "1.4")
> for (i in 5:175) {
+     s3d <- scatterplot3d(mypc$x[, 1:3], color = cc2, pch = 19,
+         angle = i, grid = F)
+     smartlegend(x = "right", y = "center", unique(grade), col = unique(cc2),
+         pch = c(19, 19))
+ }
> dev.off()

postscript
         2
```

# 5 Survival Analysis

You can download an R file containing the definition of some functions for survival analysis from `http://www.ulb.ac.be/di/map/bhaibeka/bioinfo_courses/survival_foo.R`. Put it in a directory (referred as `path_to_fun`).

The survival curve of the survival data from the `phenoData` object can be generated using the following code :

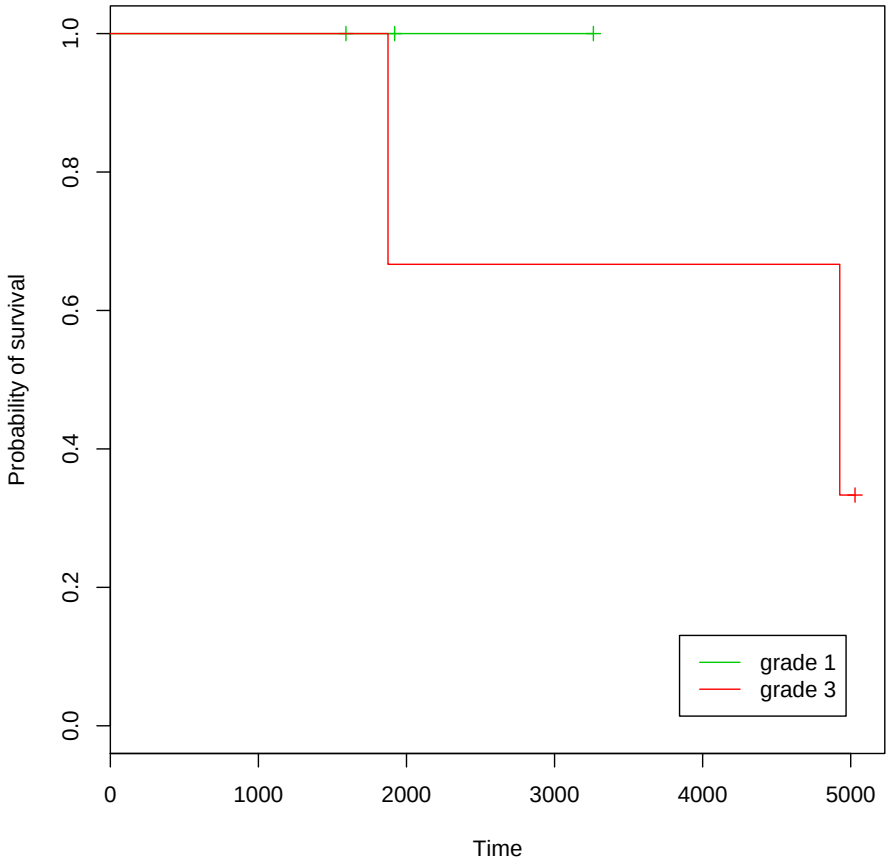
```
> library(survival)
> source("path_to_fun/survival_foo.R")
> surv.time <- pData(phenod)[, "t.rfs"]
> surv.event <- pData(phenod)[, "e.rfs"]
> fit <- survfit(Surv(surv.time, surv.event), conf.type = "none")
> par(mar = c(5, 4, 2, 2) + 0.1)
> plot(fit, xlab = "Time", ylab = "Probability of survival")
```




If we consider the two classes specified by the "grade" information (1 vs 3), we can study the difference in survival between these two groups

- in using the logrank test :

```
> grade.logrank <- survdiff(Surv(surv.time, surv.event) ~ grade)
> print(grade.logrank)

Call:
survdiff(formula = Surv(surv.time, surv.event) ~ grade)

        N Observed Expected (O-E)^2/E (O-E)^2/V
grade=1 3        0      0.4       0.4     0.667
grade=3 3        2      1.6       0.1     0.667

 Chisq= 0.7  on 1 degrees of freedom, p= 0.414

> par(mar = c(5, 4, 2, 2) + 0.1)
> mysurvivalplot(group = grade, surv.time = surv.time, surv.event = surv.event,
+     group.name = c("grade 1", "grade 3"))
```




- in computing the hazard ratio of a Cox model :

```
> gradebin <- ifelse(grade == 3, 1, 0)
> coxph(Surv(surv.time, surv.event) ~ gradebin)

Call:
coxph(formula = Surv(surv.time, surv.event) ~ gradebin)


          coef exp(coef) se(coef)        z p
gradebin 20.9  1.14e+09    41390 0.000504 1

Likelihood ratio test=1.02  on 1 df, p=0.312  n= 6
```

**Good Work !**