

# R/BioC Exercises: Affymetrix genechip analysis

Emiel Ver Loren van Themaat and Perry Moerland

April 19, 2010

☞ This document and information on how to log on to a PC in the exercise room and the UNIX server can be found here:

<http://amc-app1.amc.sara.nl/twiki/bin/view/BioLab/EducationBioinformaticsII>

## 1 Introduction

The first part of the lab introduces the basic tools provided for the low-level analysis of Affymetrix data. By the end you should be able to perform basic inspection of Affymetrix datasets, process raw data to produce expression measures, and assess the quality of the arrays. The last part of this lab once more uses linear models to extract biologically interesting information and ways to report results in HTML are presented. Before starting exercises 1 to 4 read Chapters 2 and 3. For exercise 5 read Chapters 23 and 25. The experimental data we will use is not (yet) public and may not be shared with others.

### 1.1 LBP/DBP mouse dataset

In the dataset we are about to explore, liver samples of mice are hybridized to Affymetrix ‘mouse4302’ arrays, containing about 45,000 probesets. In total we have 10 mice, 5 of them having a gene coding for LBP (also known as “Ehhadh”) knocked out and the 5 others have another gene knocked out, coding for DBP (also known as “Hsd17b4”). Both LBP and DBP are supposed to participate in the same biological process (beta-oxidation), however 50 % of the DBP knockout mice die, while all LBP mice survive. The LBP knocked out gene is represented on the array by the 1448382\_at probeset and DBP by 1417369\_at.

Copy the raw data to your directory:

```
mkdir Affymetrix
cd Affymetrix
cp /data/home/stud00/Affymetrix/* ~/Affymetrix
```

## 2 Affymetrix: Reading and exploring the data

Start R 2.10 and load the package affy.

## 2.1 The *affyBatch* object

**Exercise 1:** Try to read in CEL-files into an *affyBatch* object and play around with the data (see Chapter 2.2). Try to visualize the probe intensities for the LBP knocked out gene (as in Figure 2.1 of the book). Can you guess on which arrays the LBP knocked out mice are hybridized?

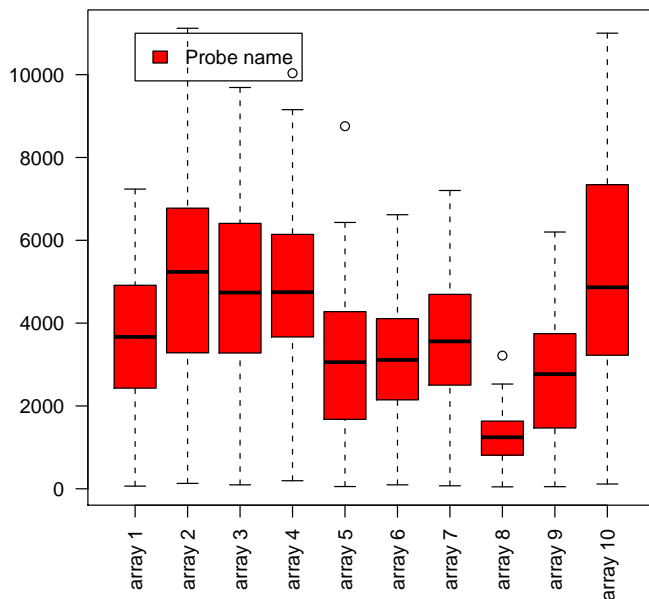
Control probes are printed on the arrays to inspect the quality of the hybridization process. The “CreX” probes are one of these control probes and are printed multiple times on the array. To all the mice samples the same amount of CreX strands are added. The CreX strand does not occur on the mouse genome but comes from *bacteriophage P1*. The intensity of the CreX probes on the array can therefore be a good indicator of hybridization quality.

**Exercise 2:** Two versions of the CreX strand are printed on the array. The names of both versions contain the string “CreX”. The *AffyBatch* class has a method *probeNames* which returns the names of all the probes on the array.

- Retrieve the full names of the two “CreX” probes. Tip: function *grep* might be useful.
- See Chapter 3.3 and 3.4 to find out what the 3 and 5 in the names mean.

**Exercise 3:** In the figure below, the perfect match values per array for the CreX probes representing the 3'-end are displayed in a boxplot. The order of the arrays has been permuted in this boxplot.

- Create such a boxplot with arrays in the right order and with the correct legend and sample labels. Tips: function *boxplot* creates boxplots and plots *data.frame* objects automatically by columns. To add a legend, use function *legend*.



- Which array behaves differently with respect to the others? What should we do with it?
- Try to plot the 5'-end probes of “CreX” in another boxplot. Can you explain the intensity differences between the 5- and 3'-end?

❏ Most plotting functions, like *boxplot* and *image* pass additional unmatched arguments to the function *par*. This function sets many graphical parameters. For example, to plot sample labels perpendicular to the axis in a boxplot, you can do:

```
> boxplot(data.frame, las = 2)
```

You can also set the parameters for all new plots:

```
> par(las = 2)
```

To restore settings, set

```
> par(las = 0)
```

Usually it is handy to store the default *par* setting in an object so you can restore the default settings:

```
> par.default <- par(no.readonly = TRUE)
> par(par.default)
```

## 2.2 Spatial quality inspection (Chapter 3.5)

In this section we will inspect if there are any spatial artifacts on the arrays itself (pieces of dust for example). The package *affyPLM* (that should now be loaded) makes a model for each probe on the array. In Chapter 3.5 methods contained in the package are described and some examples are given. It basically calculates a special type of median of the probe intensity over all arrays. These medians are used as a ‘reference array’ and by subtracting it from all individual arrays, areas with *relatively* high (or low) intensities can be visualized.

**Exercise 4:** Make a function which can automatically generate an image of the residuals (as in figure 3.5.C) for each array in the *AffyBatch* object. The images are written to a .png file. The function should be like:

```
> yourFunction1 <- function(AffyBatch) {
+   FIT_PROBE_LEVEL_MODEL
+   for (i in 1:length(AffyBatch)) {
+     png(file = MAKE_FILENAME, width = 1000, height = 1000)
+     MAKE_IMAGE_OF_RESIDUALS_FOR_ARRAY_I
+     dev.off()
+   }
+ }
```

where the capital written parts should be replaced with the right R code. Tip: Use function *paste* to generically create filenames. The editor ‘ConTEXT’ is handy to use to check syntax errors. When *yourFunction1* works correctly, use it to make the images and inspect these.

- Do you notice any spatial effects?
- How are spatial effects already partially taken care of in Affymetrix arrays? (this differs from spotted arrays)
- Any suggestions how to remove artificial effects?

### 2.3 Normalization and Summarization (Chapters 2.3 and 2.4)

There are three steps to be taken to normalize and summarize the data in such a way that we have one expression value per probeset per array. These are: background correction, normalization and summarization. Affymetrix developed methods themselves called ‘mas5’. These methods are not open source, but based on their documentation function *mas5* was developed. Another approach developed by Irizarry *et al.* is called RMA and is described in Chapter 2.4.3.

**Exercise 5:** Use the *threestep* function (Chapter 2.4.2) to compute expression measures in the RMA way. Use *mas5* to compute it the MAS5 way.

- Inspect and compare the computed expression values for the LBP probeset (1448382\_at).
- Why is there such a huge scale difference between mas5 and RMA (and try to solve this difference)?
- What is the class of the resulting objects?
- Store the resulting objects in a *.RData* file (use function *save*).

☞ In the 2.10 version of R, class *exprSet* has been changed into *ExpressionSet*. Note that this is different from the book where they use the *exprSet* class.

## 3 Linear modeling and reporting differentially expressed genes (Chapter 25)

In chapter 25, a few protocols to report differentially expressed genes are described. These methods can be applied to our data set when comparing the LBP versus the DBP mice. The libraries *limma*, *XML*, *annotate* and *mouse4302.db* are used.

### 3.1 Annotation of samples

We will first add sample information to the object of class *ExpressionSet* made in the previous exercise. It is handy to instantiate the *phenoData* attribute of the *ExpressionSet* object. In Chapter 25.2 an example is given for the *affyBatch* object. For an *ExpressionSet* object this works a bit differently and we will use method *pData*. The annotation of the LBP/DBP arrays is given in the file “DesignMatrix.txt” which you copied earlier.

**Exercise 6:** Write a function that automatically adds annotation data, stored in a file on your computer, to an *ExpressionSet* object:

```

> your_function2 <- function(exprSet, fileLocation) {
+   READ_IN_ANNOTATION_FILE_AS_DATA_FRAME
+   pData(exprSet) <- ANNOTATION_DATA_FRAME
+   return(exprSet)
+ }

```

where the function argument *fileLocation* is a variable pointing to “DesignMatrix.txt” in our case.

### 3.2 Finding differentially expressed genes

In yesterday’s lab you have worked with linear models to find differentially expressed genes. We will apply this knowledge (a reminder is given on page 436) in the next exercises.

**Exercise 7:** Use package *limma* (moderated t-test with adjusted *p*-values) to find differentially expressed genes between the LBP knockout mice and the DBP knockout mice, and answer the following questions.

- How many genes do you consider to be differentially expressed if we set adjusted *p*-value (FDR) < 0.005 for the RMA normalized data.
- Do you recognize some of the probesets popping up in the top 10?

Repeat the above for *mas5* (convert expression values to log<sub>2</sub>-scale) and check if it results in the same list of significant genes. One way to do this is by using function *consVenn* given in the R-file *consVenn.R*, which you copied to your local directory earlier this morning. This function plots a Venn diagram representing the overlap between two or three *character* vectors.

- What do you think of the overlap?
- Also try to check the overlap between the top 200 of both lists.

### 3.3 Annotating and reporting differentially expressed genes

Because there are so many genes, most researchers don’t know all the genes in the top list. When making reports or tables with the differentially expressed genes life can be made easier by hyperlinking the genes to databases with more detailed information on the genes. This is made possible by the package *annotate* and the package *mouse4302.db* which contains annotation for the probesets on our specific chip. In Chapter 25.5.2 a nice example is given.

**Exercise 8:** Select the significant genes (adjusted *p*-value < 0.005) of the RMA-normalized set and try to create a HTML file with the results as presented in Chapter 25.5.2. If you succeeded, try to add a column to this report containing Affymetrix probeset IDs (first convert the IDs to characters).

```

> sessionInfo()

```

```

R version 2.10.1 (2009-12-14)
i386-pc-mingw32

```

locale:

- [1] LC\_COLLATE=English\_United Kingdom.1252
- [2] LC\_CTYPE=English\_United Kingdom.1252
- [3] LC\_MONETARY=English\_United Kingdom.1252
- [4] LC\_NUMERIC=C
- [5] LC\_TIME=English\_United Kingdom.1252

attached base packages:

- [1] stats graphics grDevices utils datasets methods base

other attached packages:

- [1] mouse4302cdf\_2.5.0 affy\_1.24.2 Biobase\_2.6.1

loaded via a namespace (and not attached):

- [1] affyio\_1.14.0 preprocessCore\_1.8.0 tools\_2.10.1