Analysis of Microarray Data

Keith Baggerly and Kevin Coombes
Section of Bioinformatics
Department of Biostatistics and Applied Mathematics
UT M. D. Anderson Cancer Center
kabagg@mdanderson.org
kcoombes@mdanderson.org

14 September 2004
Lecture 5: BioConductor And Affymetrix Arrays

- Bioconductor Packages
- Microarray Data Structures
- Affymetrix Data in BioConductor
- Processing Affymetrix data
- Quantification = summarization
- More about reading Affymetrix data
Bioconductor Packages

You will need the following packages from the Bioconductor website. Use the menu item “Packages” —> “Install package(s) from BioConductor...” to get them.

reposTools : Repository tools for R

Biobase : Base functions for BioConductor

affy : Methods for Affymetrix oligonucleotide arrays

affydata : Affymetrix data for demonstration purposes

affypdnn : Probe dependent nearest neighbor (PDNN) for the affy package
Bioconductor Widget Packages

In order to use some of the graphical tools that make it easier to read Affymetrix microarray data and construct sensible objects describing the experiments, you will also need the following packages from the Bioconductor web site.

**tkWidgets**: R based Tk widgets

**widgetTools**: Creates an niteractive tcltk widget

**DynDoc**: Dynamic document tools
Microarray Data Structures

What information do we need in order to analyze a collection of microarray experiments?
Experiment/Sample Information

In even the simplest experimental designs, where we want to find out which genes are differentially expressed between two types of samples, we at least have to be told which samples are of which type. In more complicated experimental designs, we may be interested in a number of additional factors. For example, in a study comparing cancer patients to healthy individuals, we may want to record the age and sex of the study subjects. In animal experiments, there may be a variety of different treatments that have to be recorded.

The R object that holds this kind of information is a `data.frame`. Conceptually, a `data.frame` is just a two-dimensional table. By convention, they are arranged so that each row corresponds to an experimental sample and each column corresponds to one of the interesting factors.
Example of a data.frame

<table>
<thead>
<tr>
<th>Array</th>
<th>Age</th>
<th>Sex</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>41</td>
<td>M</td>
<td>cancer</td>
</tr>
<tr>
<td>a2</td>
<td>64</td>
<td>F</td>
<td>cancer</td>
</tr>
<tr>
<td>a3</td>
<td>56</td>
<td>M</td>
<td>healthy</td>
</tr>
<tr>
<td>a4</td>
<td>48</td>
<td>F</td>
<td>healthy</td>
</tr>
</tbody>
</table>

Data frames are particularly useful for this purpose in R, because they can hold textual factors as well as numeric ones. For most array studies, it is best to create a table of the interesting information and store it in a separate file. If you create the table in a spreadsheet program (like Excel), you should store it as a text file in “tab-separated-value” format. That is, each row holds the information from one experiment, and column entries are separated by tab characters.
Phenotypes

You can create a data frame in R from a file in tab-separate-value format using the `read.table` command. (You can also create them directly, as illustrated below.)

The `Biobase` package in BioConductor views the sample information as an extension of the notion of a data frame, which they call a `phenoData` object. In their conception, this object contains the “phenotype” information about the samples used in the experiment. The extra information in a `phenoData` object consist of optional “long” labels that can be used to identify the covariates (or factors) in the columns.
Mock data

Let’s create a fake data set. We pretend we have measured 200 genes in 8 experimental samples, the first four of which are healthy and the last four are cancer patients.

```r
> fake.data <- matrix(rnorm(8*200), ncol=8)
> sample.info <- data.frame(
+   spl=paste('A', 1:8, sep=''),
+   stat=rep(c('healthy', 'cancer'), each=4))
```

At this point, we have a matrix containing fake expression data and a data frame containing two columns ("spl" and "stat"). Let’s create a `phenoData` object with more intelligible labels:

```r
> pheno <- new("phenoData", pData=sample.info,
+   varLabels=list('Sample Name', 'Cancer Status'))
```
intro

> pheno

phenoData object with 2 variables and 8 cases
varLabels
  : Sample Name
  : Cancer Status

> pData(pheno)

  spl  stat
  1   A1  cancer
  2   A2  cancer
  3   A3  cancer
  4   A4  cancer
  5   A5  healthy
  6   A6  healthy
  7   A7  healthy
  8   A8  healthy
**ExprSets**

The object in BioConductor that links together a collection of expression data and its associated sample information is called an `exprSet`.

```r
> my.experiments <- new("exprSet",
+   exprs=fake.data, phenoData=pheno)
> my.experiments
Expression Set (exprSet) with
   200 genes
   8 samples
   phenoData object with 2 variables and 8 cases
   varLabels
     : Sample Name
     : Cancer Status
```
Warning

If you create a real exprSet this way, you should ensure that the columns of the data matrix are in exactly the same order as the rows of the sample information data frame; the software has no way of verifying this property without your help.

You’ll also need to put together something that describes the genes used on the microarrays.
**Where is the gene information?**

The `exprSet` object we have created so far lacks an essential piece of information: there is nothing to describe the genes. One flaw in the design of BioConductor is that it allows you to completely separate the biological information about the genes from the expression data. (This blithe acceptance of the separation is surprisingly common among analysts.)

Each `exprSet` includes a slot called `annotation`, which is a character string containing the name of the environment that holds the gene annotations.

We’ll return to this topic later to discuss how to create these annotation environments.
Optional parts of an `exprSet`

In addition to the expression data (`exprs`) and the sample information (`phenoData`), each `exprSet` includes several optional pieces of information:

- **annotation** name of the gene annotation environment

- **se.exprs** matrix containing standard errors of the expression estimates

- **notes** character string describing the experiment

- **description** object of class MIAME describing the experiment
Affymetrix Data in BioConductor

For working with Affymetrix data, BioConductor includes a specialized kind of exprSet called an AffyBatch. To create an AffyBatch object from the CEL files in the current directory, do the following:

> library(affy)  # load the affy library
> my.data <- ReadAffy()  # read CEL data

You may have to start by telling R to use a different working directory to find the CEL files; the command to do this is setwd.

> setwd("/my/celfiles")  # point to the CEL files

Paths in R are separated by forward slashes (/) not backslashes (\); this is a common source of confusion.
Demonstration data

Note: If you are trying to follow along and have not yet obtained some CEL files, the `affydata` package includes demonstration data from a dilution experiment. You can load it by typing

```r
> library(affydata)
> data(Dilution)
```

These commands will create an AffyBatch object called `Dilution` that you can explore.
Peeking at what’s inside

BioConductor will automatically build an object with the correct gene annotations for the kind of array you are using the first time you access the data; this may take a while, since it downloads all the information from the internet. So, don’t be surprised if it takes a few minutes to display the response to the command

> Dilution
AffyBatch object
size of arrays=640x640 features (12805 kb)
cdf=HG_U95Av2 (12625 affyids)
number of samples=4
annotation=hgu95av2
Looking at the experimental design

You can see what the experiments are by looking at the phenotype information.

```r
> phenoData(Dilution)

phenoData object with 3 variables and 4 cases

varLabels

liver: amount of liver RNA hybridized to array
sn19: amount of central nervous system RNA hybridized to array
scanner: ID number of scanner used

> pData(Dilution)

<table>
<thead>
<tr>
<th>liver</th>
<th>sn19</th>
<th>scanner</th>
</tr>
</thead>
<tbody>
<tr>
<td>20A</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>20B</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>10A</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>10B</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>
```
A first look at an array

```r
> image(Dilution[,1])
```
A summary view of four images

> boxplot(Dilution, col=1:4)
The distribution of feature intensities

> hist(Dilution, col=1:4, lty=1)
Examining individual probesets

The **affy** package in BioConductor includes tools for extracting individual probe sets from a complete **AffyBatch** object. To get at the probe sets, however, you need to be able to refer to them by their “name”, which at present means their Affymetrix ID.

```r
> geneNames(Dilution)[1:3]
[1] "100_g_at" "1000_at" "1001_at"
> random.affyid <- sample(geneNames(Dilution), 1)
> # random.affyid <- '34803_at'
> ps <- probeset(Dilution, random.affyid)[[1]]
```

The **probeset** function returns a list of probe sets; the mysterious stuff with the brackets takes the first element from the list (which only had one...).
A probeset profile in four arrays

> plot(c(1,16), c(50, 900), type='n',
+  xlab='Probe', ylab='Intensity')
> for (i in 1:4) lines(pm(ps)[,i], col=i)
Examining individual probesets

Let's add the mismatch probes to the graph:

```r
> for (i in 1:4) lines(pm(ps)[,i], col=i)
```
> plot(c(1,16), c(-80, 350), type='n',
+   xlab='Probe Pair', ylab='PM - MM')
> temp <- pm(ps) - mm(ps)
> for (i in 1:4) lines(temp[,i], col=i)
RNA degradation

Individual (perfect match) probes in each probe set are ordered by location relative to the 5’ end of the targeted mRNA molecule. We also know that RNA degradation typically starts at the 5’ end, so we would expect probe intensities to be lower near the 5’ end than near the 3’ end.

The `affy` package of BioConductor includes functions to summarize and plot the degree of RNA degradation in a series of Affymetrix experiments. These methods pretend that something like “the fifth probe in an Affymetrix probe set” is a meaningful notion, and they average these things over all probe sets on the array.
Visualizing RNA degradation

> degrade <- AffyRNAdeg(Dilution)
> plotAffyRNAdeg(degrade)
BioConductor breaks down the low-level processing of Affymetrix data into four steps. The design is highly modular, so you can choose different algorithms at each step. It is highly likely that the results of later (high-level) analyses will change depending on your choices at these steps.

- Background correction
- Normalization (on features)
- PM-correction
- Summarization
Background correction

The list of available background correction methods is stored in a variable:

```r
> bgcorrect.methods
[1] "mas"  "none"  "rma"  "rma2"
```

So there are four methods:

- **none**  Do nothing
- **mas**  Use the algorithm from MAS 5.0
- **rma**  Use the algorithm from the current version of RMA
- **rma2**  Use the algorithm from an older version of RMA
Background correction in MAS 5.0

MAS 5.0 divides the microarray (more precisely, the CEL file) into 16 regions. In each region, the intensity of the dimmest 2% of features is used to define the background level. Each probe is then adjusted by a weighted average of these 16 values, with the weights depending on the distance to the centroids of the 16 regions.
Background correction in RMA

RMA takes a very different approach to background correction. First, only PM values are adjusted, the MM values are not changed at all. Second, they try to model the distribution of PM intensities statistically as a sum of

- exponential signal with mean $\lambda$
- normal noise with mean $\mu$ and variance $\sigma^2$ (truncated at 0 to avoid negatives).

If we observe a signal $X = x$ at a PM feature, we adjust it by

$$E(s|X = x) = a + b \frac{\phi(a/b) - \phi((x - a)/b)}{\Phi(a/b) + \Phi((x - a)/b) - 1}$$

where $b = \sigma$ and $a = s - \mu - \lambda \sigma^2$. 
**Comparing background methods**

```r
> d.mas <- bg.correct(Dilution[,1], "mas")
> d.rma <- bg.correct(Dilution[,1], "rma")
> bg.with.mas <- pm(Dilution[,1]) - pm(d.mas)
> bg.with.rma <- pm(Dilution[,1]) - pm(d.rma)
```
> summary(bg.with.mas)
Min.   :74.53
1st Qu.:93.14
Median :94.35
Mean   :94.27
3rd Qu.:95.80
Max.   :97.67

> summary(bg.with.rma)
Min.   : 72.4
1st Qu.:113.7
Median :114.9
Mean   :112.1
3rd Qu.:114.9
Max.   :114.9
Difference in background estimates

On this array, RMA gives slightly larger background estimates, and gives estimates that are more nearly constant across the array. The overall differences can be displayed in a histogram.
**Quantification = summarization**

I’m going to avoid talking about normalization and PM correction for the moment, and jump ahead to summarization. As we have explained previously, this step is the critical final component in analyzing Affymetrix arrays, since it’s the one that combines all the numbers from the PM and MM probe pairs in a probe set into a single number that represents our best guess at the expression level of the targeted gene.

The available summarization methods, like the other available methods, can be obtained from a variable.

```r
c> express.summary.stat.methods
[1] "avgdiff"    "liwong"    "mas"
    "medianpolish" "playerout"
```
Including the PDNN method

The implementation of the PDNN method is contained in a separate package. When you load the package library, it updates the list of available methods.

```r
> library(affypdnn)
registering new summary method 'pdnn'.
registering new pmcorrect method 'pdnn'
and 'pdannpredict'.
> express.summary.stat.methods
[1] "avgdiff" "liwong" "mas"
[4] "medianpolish" "playerout" "pdnn"
```
The recommended way to put together all the steps for processing Affymetrix arrays in BioConductor is with the function `expresso`. Here’s an example that blocks everything except the summarization:

```r
> tempfun <- function(method) {
+   expresso(Dilution, bg.correct=FALSE,
+            normalize=FALSE, pmcorrect.method="pmonly",
+            summary.method=method)
+ }
> ad <- tempfun("avgdiff") # MAS4.0
> al <- tempfun("liwong") # dChip
> am <- tempfun("mas") # MAS5.0
> ap <- tempfun("pdnn") # PDNN
> ar <- tempfun("medianpolish") # RMA
```
We have mentioned M-versus-A plots before. (Statisticians knew these as “Bland-Altman” plots before anyone started studying microarrays.) Instead of plotting two similar things on the usual $x$ and $y$ axes, they plot the average ($\frac{(x + y)}{2}$) along the horizontal axis and the difference ($y - x$) along the vertical axis. The affy package includes a function called `mva.pairs` to make it easier to generate these plots. We’re going to use this to compare the different quantification/summary methods.

```r
> temp <- data.frame(exprs(ad)[,1], exprs(al)[,1], 
+ exprs(am)[,1], 2^exprs(ar)[,1])
> dimnames(temp)[[2]] <- c('Mas4', 'dChip', 
+ 'Mas5', 'RMA')
```
**MVA plot**

- **Mas4**
  - Median: -0.47
  - IQR: 0.595

- **dChip**
  - Median: 0.0110
  - IQR: 0.0943

- **Mas5**
  - Median: 0.473
  - IQR: 0.567

- **RMA**
  - Median: 0.0035
  - IQR: 0.0727

---

@copyright 2004, Kevin R. Coombes and Keith A. Baggerly

GS01 0163: Analysis of Microarray Data
More about reading Affymetrix data

The BioConductor `affy` package includes a graphical interface to make it easier to read in Affymetrix data and construct `AffyBatch` objects.
Affy Widgets

Choose CEL files
Directory: h:/affymetrix/Abruzzo/Lynn_Barron_09_18_03
Files in directory
U133A_CL33A_09_18_03.CEL
U133A_CL35A_09_18_03.CEL
U133A_CL42A_09_18_03.CEL
U133A_CL48A_09_18_03.CEL
Files selected

Up Select >> << Remove Clear Finish
Affy Widgets

Choose CEL files

Directory: h:/affymetrix/Abruzzo/Lynn_Barron_09_18_03

Files in directory

<table>
<thead>
<tr>
<th>U133A_CLL30A_09_18_03.CEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>U133A_CLL33A_09_18_03.CEL</td>
</tr>
<tr>
<td>U133A_CLL35A_09_18_03.CEL</td>
</tr>
<tr>
<td>U133A_CLL37A_09_18_03.CEL</td>
</tr>
<tr>
<td>U133A_CLL38A_09_18_03.CEL</td>
</tr>
<tr>
<td>U133A_CLL39A_09_18_03.CEL</td>
</tr>
<tr>
<td>U133A_CLL42A_09_18_03.CEL</td>
</tr>
<tr>
<td>U133A_CLL48A_09_18_03.CEL</td>
</tr>
</tbody>
</table>

Files selected

<table>
<thead>
<tr>
<th>U133A_CLL30A_09_18_03.CEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>U133A_CLL33A_09_18_03.CEL</td>
</tr>
<tr>
<td>U133A_CLL35A_09_18_03.CEL</td>
</tr>
<tr>
<td>U133A_CLL37A_09_18_03.CEL</td>
</tr>
<tr>
<td>U133A_CLL38A_09_18_03.CEL</td>
</tr>
<tr>
<td>U133A_CLL39A_09_18_03.CEL</td>
</tr>
<tr>
<td>U133A_CLL42A_09_18_03.CEL</td>
</tr>
<tr>
<td>U133A_CLL48A_09_18_03.CEL</td>
</tr>
</tbody>
</table>

Up  Select >>  << Remove  Clear  Finish
# Affy Widgets

<table>
<thead>
<tr>
<th>File Path/Names</th>
<th>Sample Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CL13A_09_18_03.CEL</td>
<td>U133A_CL13A_09_18_03</td>
<td></td>
</tr>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CL13A_09_18_03.CEL</td>
<td>U133A_CL13A_09_18_03</td>
<td></td>
</tr>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CL13A_09_18_03.CEL</td>
<td>U133A_CL13A_09_18_03</td>
<td></td>
</tr>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CL13A_09_18_03.CEL</td>
<td>U133A_CL13A_09_18_03</td>
<td></td>
</tr>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CL13A_09_18_03.CEL</td>
<td>U133A_CL13A_09_18_03</td>
<td></td>
</tr>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CL13A_09_18_03.CEL</td>
<td>U133A_CL13A_09_18_03</td>
<td></td>
</tr>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CL13A_09_18_03.CEL</td>
<td>U133A_CL13A_09_18_03</td>
<td></td>
</tr>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CL13A_09_18_03.CEL</td>
<td>U133A_CL13A_09_18_03</td>
<td></td>
</tr>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CL13A_09_18_03.CEL</td>
<td>U133A_CL13A_09_18_03</td>
<td></td>
</tr>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CL13A_09_18_03.CEL</td>
<td>U133A_CL13A_09_18_03</td>
<td></td>
</tr>
</tbody>
</table>

© Copyright 2004, Kevin R. Coombes and Keith A. Baggerly
## Affy Widgets

### Sample Information

<table>
<thead>
<tr>
<th>File PathNames</th>
<th>Sample Names</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CLL30A_09_18_03.CEL</td>
<td>U133A_CLL30A_09_18_03</td>
<td>CLL30</td>
</tr>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CLL33A_09_18_03.CEL</td>
<td>U133A_CLL33A_09_18_03</td>
<td>CLL33</td>
</tr>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CLL35A_09_18_03.CEL</td>
<td>U133A_CLL35A_09_18_03</td>
<td>CLL35</td>
</tr>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CLL37A_09_18_03.CEL</td>
<td>U133A_CLL37A_09_18_03</td>
<td>CLL37</td>
</tr>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CLL38A_09_18_03.CEL</td>
<td>U133A_CLL38A_09_18_03</td>
<td>CLL38</td>
</tr>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CLL39A_09_18_03.CEL</td>
<td>U133A_CLL39A_09_18_03</td>
<td>CLL39</td>
</tr>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CLL42A_09_18_03.CEL</td>
<td>U133A_CLL42A_09_18_03</td>
<td>CLL42</td>
</tr>
<tr>
<td>h:/affymetrix/Abruzzo/Lynn_Baron_09_18_03/U133A_CLL48A_09_18_03.CEL</td>
<td>U133A_CLL48A_09_18_03</td>
<td>CLL48</td>
</tr>
</tbody>
</table>

© Copyright 2004, Kevin R. Coombes and Keith A. Baggerly
Affy Widgets

Enter the Number Of Covariates: 2

Continue
## Affy Widgets

### Covariate Names

<table>
<thead>
<tr>
<th>Covariate Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cov 1</td>
<td>Mutation Status</td>
</tr>
<tr>
<td>Cov 2</td>
<td>survival</td>
</tr>
</tbody>
</table>

- Cov 1: Mutation Status
  - Is the IgV region somatically mutated?
- Cov 2: survival
  - Survival time in months

[Diagram of Affy Widgets with example covariates]
# Affy Widgets

<table>
<thead>
<tr>
<th>Sample Names</th>
<th>Mutation Status</th>
<th>survival</th>
</tr>
</thead>
<tbody>
<tr>
<td>J133A_CL136A_08_18_03</td>
<td>Mut</td>
<td>125</td>
</tr>
<tr>
<td>J133A_CL133A_08_18_03</td>
<td>Mut</td>
<td>45</td>
</tr>
<tr>
<td>J133A_CL135A_08_18_03</td>
<td>Mut</td>
<td>23</td>
</tr>
<tr>
<td>J133A_CL137A_08_18_03</td>
<td>Mut</td>
<td>18</td>
</tr>
<tr>
<td>J133A_CL138A_08_18_03</td>
<td>Un</td>
<td>213</td>
</tr>
<tr>
<td>J133A_CL139A_08_18_03</td>
<td>Un</td>
<td>123</td>
</tr>
<tr>
<td>J133A_CL142A_08_18_03</td>
<td>Un</td>
<td>55</td>
</tr>
<tr>
<td>J133A_CL149A_08_18_03</td>
<td>Un</td>
<td>46</td>
</tr>
</tbody>
</table>

Save phenoData as

[Back]  [Continue]
Affy Widgets

![MIAME Information Window]

- Experimenter's Name: Abruzzo
- Laboratory: Abruzzo
- Contact Information: 
- Experiment Title: mutation status
- Experiment Description: 
- URL: 

Finish