Preprocessing Affymetrix Data

Educational Materials

Introduction

- This lecture is based on Chapter 2, *Preprocessing High-density Oligonucleotide Arrays* by B. M. Bolstad, R. A. Irizarry, L. Gautier,Z. Wu
- It describes some of the basic preprocessing steps and includes code for carrying out some of these tasks.

Importing Data

- to import `CEL` file data use `ReadAffy`:

```
> library("affy")
> Data <- ReadAffy()
```

- the functions `list.celfiles` can be used to list the `CEL` files in the current working directory, and you can specify which ones to use in the call to `ReadAffy`.
- the data are stored in an *AffyBatch* object, which can be used as input to other processing functions.

Importing Data

- data from `CEL` files represent fairly raw data, they are one observation per spot
- the mapping from the spot location to the probeset and ultimately to the identity of the gene being probed is handled by the `CDF` file.
- the affy package will automatically find and load the correct `CDF` package (if one is available).

Examining probe-level data

- We first load some example data.

```
> library("affydata")
Loading required package: affy
Loading required package: Biobase

Attaching package: 'Biobase'


        The following object(s) are masked _by_ .GlobalEnv :

         cache


Loading required package: affyio
> data(Dilution)
```

- Two functions, `pm` and `mm`, provide access to the probe level data.

```
> pm(Dilution, "1001_at")[1:3, ]
           20A   20B   10A   10B
```

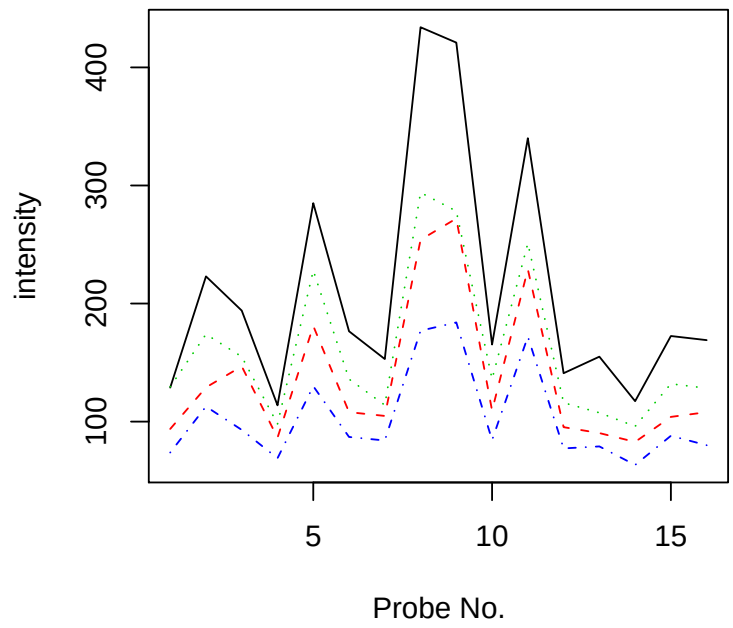
```
1001_at1 128.8  93.8 129.5  73.8
1001_at2 223.0 129.0 174.0 112.8
1001_at3 194.0 146.8 155.0  93.0
```

Probe intensity plots

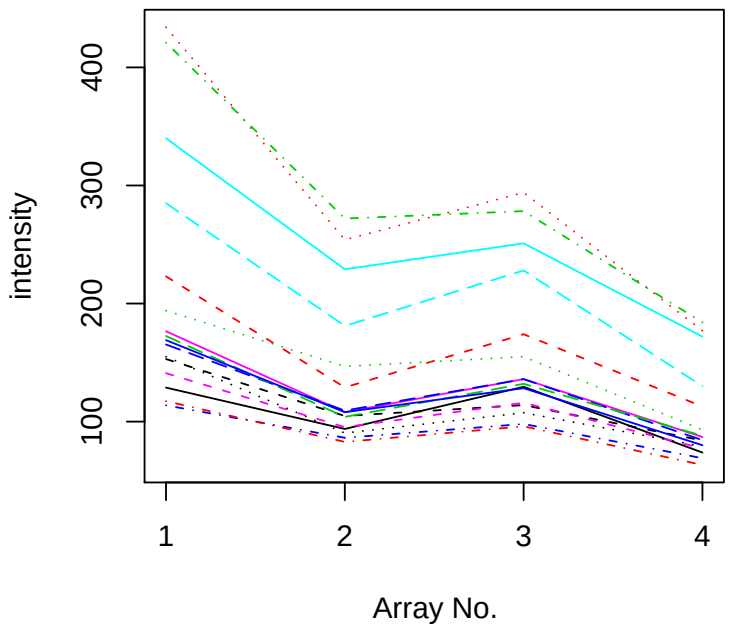
We can plot probe intensities using the following commands.

```
> matplot(pm(Dilution, "1001_at"), type = "l", xlab = "Probe No.",
+     ylab = "PM Probe intensity")
> matplot(t(pm(Dilution, "1001_at")), type = "l", xlab = "Array No.",
+     ylab = "PM Probe intensity")
```

Notice the large probe effects. The variability between probes is larger than the variability between arrays.



a)



b)

Figure 1: Examining the probe response pattern for a particular probeset a) across probe or b) across arrays.

Phenotypic data

- the `phenoData` slot is where phenotypic data is stored.
- the function `pData` can be used to access this information.

```
> pData(Dilution)
```

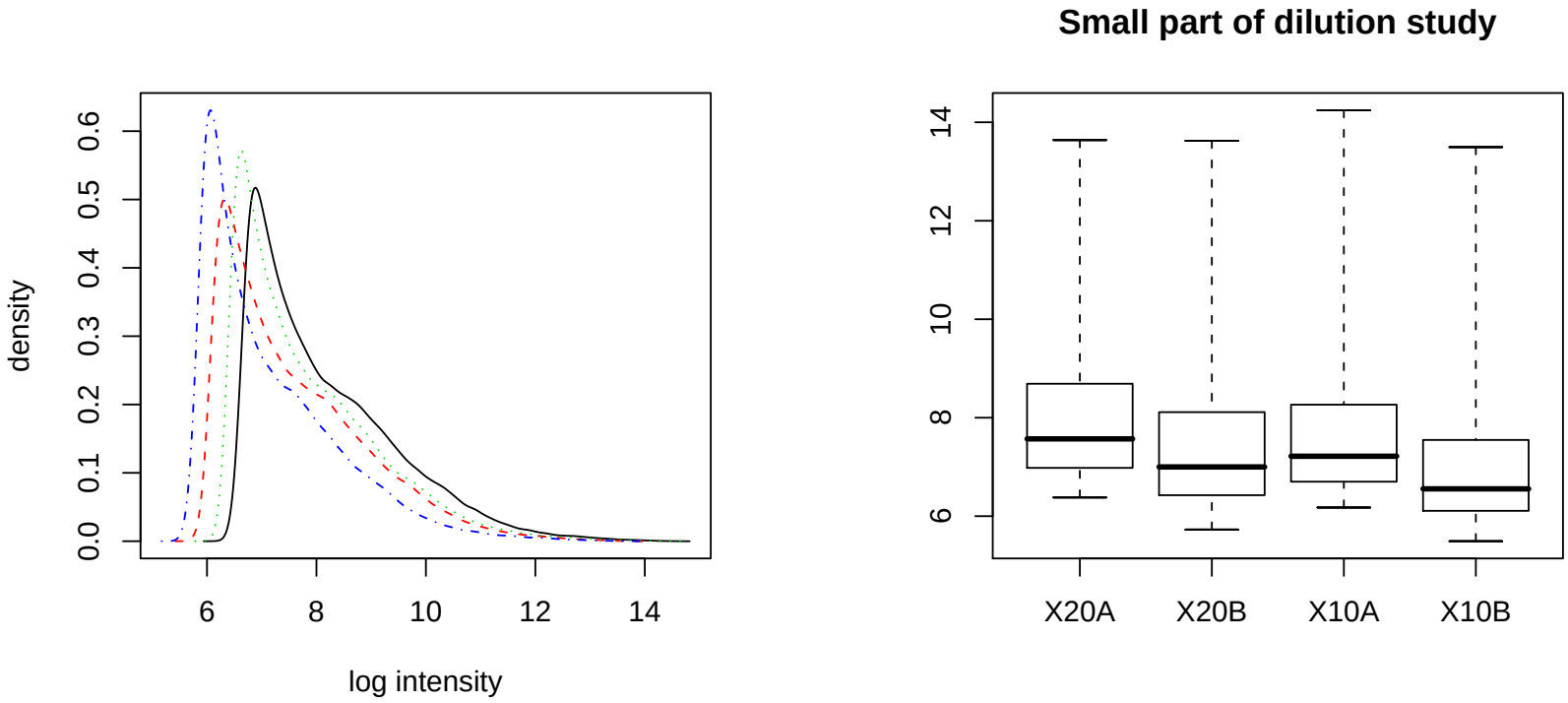
```
    liver sn19 scanner
20A    20    0       1
20B    20    0       2
10A    10    0       1
10B    10    0       2
```

- phenotypic data consists of the concentrations of RNA from two different samples, obtained from liver and central nervous system total RNA, along with the ID of the scanner

Probe Intensity Behavior

- To examine probe intensity behavior for a number of different arrays we can use the `hist` and `boxplot` methods.
- these boxplots are useful for identifying differences in the level of raw probe-intensities between arrays
- differences between arrays in the shape or center of the distribution often highlight the need for normalization

```
> hist(Dilution)
> boxplot(Dilution)
```



a) b)

Figure 2: a) Density estimates of data from the dilution experiment. The x-axis is on a logarithmic scale (base 2). b) Box-plots.

MA-Plots

- The MA-plot of two vectors, Y_1 and Y_2, is a 45 degree rotation and axis scaling of their scatter plot.
- instead of $Y_{2,j}$ versus $Y_{1,j}$ for $j = 1, \ldots, J$ we plot $M_j = Y_{2,j} - Y_{1,j}$ versus $A_j = (Y_{2,j} + Y_{1,j})/2$.
- if Y_1 and Y_2 are logarithmic expression values, then M_j will represent the log fold change for gene j
- and A_j will represent average log intensity for that gene

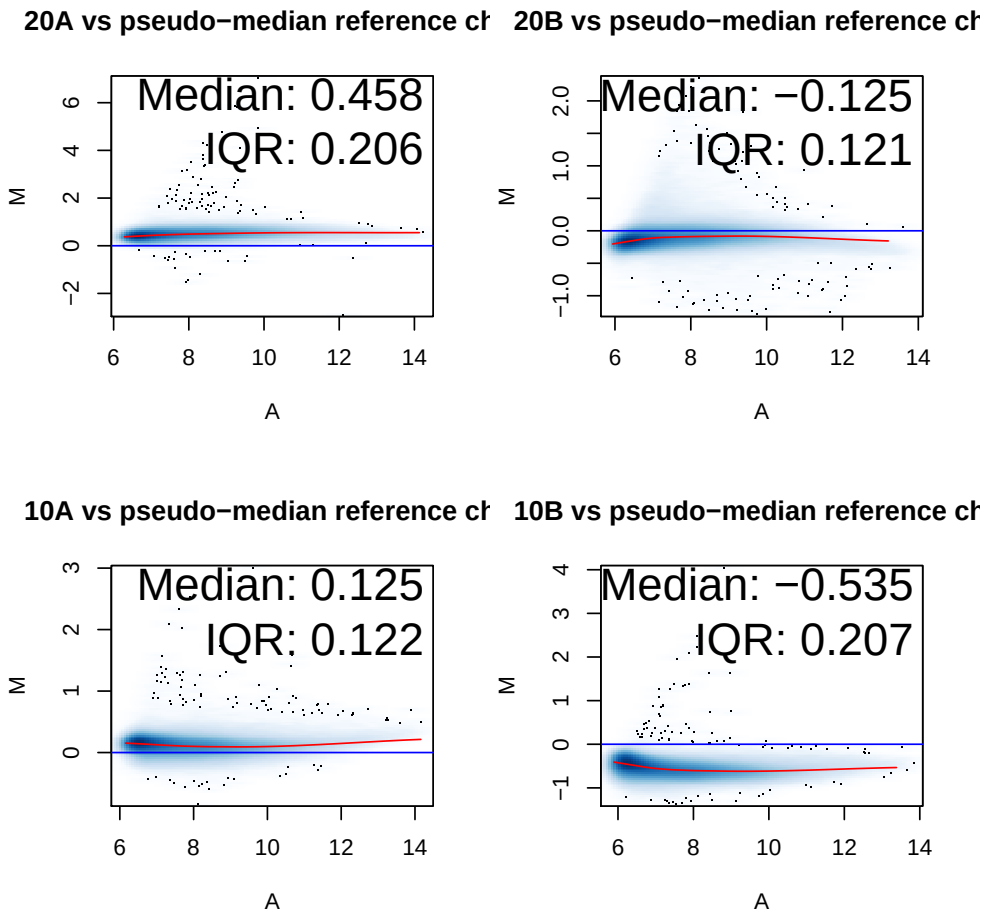


Figure 3: A MA pairs plot for the Dilution data. Scatterplots were computed using the `smoothScatter` function.

MA Plots

- if most genes are not differentially expressed the loess curves should be close to the horizontal line $M = 0$
- non-linearity in the loess curve indicates a relationship between M and A; or in being differentially expressed and the average intensity
- the plots are rotated versions of the usual scatter plot, the rotation is helpful since we are better able to detect patterns from horizontal lines than from angled lines

Background adjustment and normalization

- Preprocessing Affymetrix expression arrays usually involves three steps:
  1. background adjustment
  2. normalization
  3. summarization
- Bioconductor software implements a wide variety of methods for each of these steps.
- Self-contained routines for background correction and normalization usually take an *AffyBatch* as input and return a processed *AffyBatch*.
- Routines for summarization produce *exprSet* objects containing expression summary values.

Background adjustment

- RMA convolution
- MAS 5.0 background
- Ideal Mismatch

RMA convolution

- when developing RMA the authors found the MM probes problematic and proposed a method that used only the PM probes.
- the PM values are corrected, array by array, using a model for the probe intensities motivated by the empirical distribution of probe intensities.
- the observed PM probes are modeled as the sum of a noise component, $B \sim N(\mu, \sigma^2)$ and a signal component, $S \sim Exp(\alpha)$.
- To avoid the possibility of negatives expression values, the Normal distribution is truncated at zero.

- Letting Y denote the observed intensity:

$$E\left(S|Y=y\right)=a+b\frac{\phi\left(\frac{a}{b}\right)-\phi\left(\frac{y-a}{b}\right)}{\Phi\left(\frac{a}{b}\right)+\Phi\left(\frac{y-a}{b}\right)-1}, \tag{1}$$

  where $a=y-\mu-\sigma^2\alpha$ and $b=\sigma$. Note that ϕ and Φ are the standard Normal density and distribution functions, respectively.

- To produce a background adjusted *AffyBatch* for the `Dilution` dataset the following code can be used.

```
> Dilution.bg.rma <- bg.correct(Dilution, method = "rma")
```

MAS 5.0 background

- Proposed in the Statistical Algorithms Description Document and used in the MAS 5.0 software.
- The chip is divided into a grid of k (default $k = 16$) rectangular regions.
- For each region the lowest 2% of probe intensities are used to compute a background value for that grid.
- Then each probe intensity is adjusted based upon a weighted average of each of the background values.

MAS 5.0 background con't

- The weights are dependent on the distance between the probe and the centroid of the grid. In particular, the weights are:

$$w_k(x,y) = \frac{1}{d_k^2(x,y) + s_0}$$

  where $d_k(x,y)$ is the Euclidean distance from location (x,y) to the centroid of region k. The default value for the smoothing coefficient s_0 is 100.

- Special care is taken to avoid negative values or other numerical problems for low intensity regions.

- this method corrects both PM and MM probes.

- a background adjusted *AffyBatch* could be produced using the following code:

```
> Dilution.bg.mas <- bg.correct(Dilution, method = "mas")
```

Ideal Mismatch

- the suggested purpose of the MM probes was that they could be used to adjust the PM probes for probe-specific non-specific binding by subtracting the intensity of the MM probe from the intensity of the corresponding PM probe.
- this becomes problematic because, for data from a typical array, as many as 30% of MM probes have intensities higher than their corresponding PM probes
- thus, when raw MM intensities are subtracted from the PM intensities many negative expression values result, which makes little sense since
- to remedy the negative impact of using raw MM values, Affymetrix introduced the concept of an *Ideal Mismatch* (IM); which was guaranteed, by design, to be smaller than the corresponding PM intensity.

Ideal Mismatch

- the goal is to use MM when it is physically possible and a quantity smaller than the PM in other cases.
- If i is the probe and k is the probeset then for the probe pair indexed by i and k the ideal mismatch IM is given by

$$IM_i^{(k)} = \begin{cases} MM_i^{(k)} & \text{when } MM_i^{(k)} < PM_i^{(k)} \\ \frac{PM_i^{(k)}}{2^{SB_k}} & \text{when } MM_i^{(k)} \geq PM_i^{(k)} \text{ and } SB_k > \tau_c \\ \frac{PM_i^{(k)}}{2^{\tau_c/(1+(\tau_c - SB_k)/\tau_s)}} & \text{when } MM_i^{(k)} ij \geq PM_i^{(k)} \text{ and } SB_i \leq \tau_c \end{cases}$$

where τ_c and τ_s are tuning constants, referred to as the *contrast* τ (with a default value of 0.03) and the *scaling* τ (with a default value of 10), respectively.

Ideal Mismatch

- the adjusted PM intensity is obtained by subtracting the corresponding IM from the observed PM intensity
- to use this background correction you will either need to write your own code (for use with functions in the affy package) or use the `threestep` function from the affyPLM package.

Normalization

- normalization refers to the task of manipulating data to make measurements from different arrays comparable, some are linear and some non-linear
- scale normalization (linear)
- non-linear normalization such as: cross-validated splines (Schadt et al 2001), running median lines (Li and Wong. Genome Biology 2001), loess smoothers (Bolstad et al)
- quantile normalization, which imposes the same empirical distribution of intensities to each array.
- to perform a normalization procedure on an *AffyBatch* the generic function `normalize` may be used

Scale Normalization

Pick a column of X to serve as baseline array, say column j.
Compute the (trimmed) mean of column j. Call this $\tilde{X}_j$.
for $i = 1$ to n, $i \neq j$ **do**
  Compute the (trimmed) mean of column i. Call this $\tilde{X}_i$.
  Compute $\beta_i = \tilde{X}_j / \tilde{X}_i$.
  Multiply elements of column i by β_i.
end for

An *AffyBatch* can be scale normalized using the following code:

```
> Dilution.norm.scale <- normalize(Dilution, method = "constant")
```

Non-linear methods

Pick a column of X to serve as the baseline array, say column j.

for $i = 1$ to n, $i \neq j$ **do**

Fit a smooth non-linear relationship mapping column i to the baseline j. Call this $\hat{f}_i$

Normalized values for column j are given by $\hat{f}_i\left(X_j\right)$

end for

Non-linear normalization can be performed using the code below.

```
> Dilution.norm.nonlinear <- normalize(Dilution, method = "invariantset")
```

Quantile Normalization

Given n vectors of length p, form X, of dimension $p \times n$, where each array is a column.
Sort each column of X separately to give X_s.
Take the mean, across rows, of X_s and create X'_s, an array of the same dimension as X, but where all values in each row are equal to the row means of X_s.
Get X_n by rearranging each column of X'_s to have the same ordering as the corresponding input vector.

To apply this procedure use the code below.

```
> Dilution.norm.quantile <- normalize(Dilution, method = "quantiles")
```

Cyclic loess

Let X be a $p \times n$ matrix with columns representing arrays and rows probes or probesets.
log transform the data: $X \leftarrow \log X$
repeat
  for $i = 1$ to $n - 1$ **do**
    for $j = i + 1$ to n **do**
      for $k = 1$ to p **do**
        Compute $M_k = x_{ki} - x_{kj}$ and $A_k = \frac{1}{2}\left(x_{ki} + x_{kj}\right)$
      end for
      fit a loess curve for M on A. Call this $\hat{f}$.
      for $k = 1$ to p **do**
        $\hat{M}_k = \hat{f}(A_K)$
        set $a_k = (M_k - \hat{M}_k)/n$
        $x_{ki} = x_{ki} + a_k$ and $x_{kj} = x_{ki} - a_k$
      end for
    end for
  end for
until convergence or the maximum number of iterations is reached
Revert to the original scale $X \leftarrow \exp(X)$

Cyclic Loess

This procedure can be carried out using the affy package by specify the method to be `loess` in the call to the `normalize` function.

```
> Dilution.norm.loess <- normalize(Dilution, method = "loess")
```

Variance Stabilizing Normalization (vsn)

- the vsn method combines background correction and normalization into one single procedure
- a possible advantage of the combined approach is that information across arrays can be shared to estimate the background correction parameters, which are otherwise estimated separately for each array.
- for a data matrix x_{ki}, with k counting over the probes and i over the arrays, it fits a normalization transformation

$$x_{ki} \mapsto h_i(x_{ki}) = \mathrm{glog}\left(\frac{x_{ki} - a_i}{b_i}\right), \tag{2}$$

where b_i is the scale parameter for array i, a_i is a background offset, and glog is the so-called generalized logarithm or attenuated logarithm

VSN

- one of the nice properties of the glog function is that with appropriate values of a_i and b_i, the data from the different arrays are not just adjusted to each other, but also the variances across replicates are approximately independent of the mean.
- software for fitting the model and applying the transformation is provided in the vsn package. We can use the following code to normalize an *AffyBatch*.

```
> library("vsn")
> Dil.vsn <- normalize(Dilution, method = "vsn")
```

- the transformation parameters are returned in the `preprocessing` slot of the `description` slot of the returned *AffyBatch* object

Summarization

- summarization is the final stage in preprocessing Affymetrix GeneChip data
- it is the process of combining the multiple probe intensities for each probeset to produce an expression value
- Bioconductor packages provide a number of functions that carry out summarization to produce gene expression values: `expresso` and `threestep`, provide the ability to produce expression measures using a wide variety of user specified preprocessing methods.
- other functions are optimized for computing specific expression measures such as `rma`, `gcrma`, and `expressopdnn` are also available.

expresso

- `expresso` provides quite general facilities for computing expression summary values
- it allows most background adjustment, normalization and summarization methods to be combined
- The trade-off is that `expresso` is often considerably slower than the functions that have been optimized for producing specific expression measures.
- the names of background correction, PM correction and summarization methods available to `expresso` can be found by typing `bgcorrect.methods` `pmcorrect.methods` and `express.summary.stat.methods` respectively.

Normalization Methods

- for example, calling the function `normalize.methods` on an *AffyBatch* will list the available normalization methods:

```
> normalize.methods(Dilution)
[1] "constant"          "contrasts"         "invariantset"     "loess"
[5] "qspline"           "quantiles"         "quantiles.robust"
```

expresso

- A call to **expresso** setting most of the parameters:

```
> options(width = 50)
```

```
> eset <- expresso(Dilution, bgcorrect.method = "rma",
+     normalize.method = "constant", pmcorrect.method = "pmonly",
+     summary.method = "avgdiff")
```

- Or the PM-only model based the expression index model of Li and Wong can be performed by:

```
> eset <- expresso(Dilution, normalize.method = "invariantset",
+     bg.correct = FALSE, pmcorrect.method = "pmonly",
+     summary.method = "liwong")
```

- To obtain MAS 5 estimates

```
> eset <- mas5(Dilution)
```

threestep

- The affyPLM package provides the `threestep` function that can be used to compute very general expression measures.
- Because `threestep` is primarily implemented in compiled code it is typically faster than `expresso`
- `threestep` always returns expression measures in $\log_2$ scale
- The `background.method`, `normalize.method` and `summary.method` arguments of `threestep` control which preprocessing methods are used at each stage.

threestep - Example

To compute expression measures where the ideal mismatch is subtracted from PM, then quantile normalization between arrays is carried out, and probesets are summarized by using a robust average, one can use the following code:

```
> library("affyPLM")
> eset <- threestep(Dilution, background.method = "IdealMM",
+     normalize.method = "quantile", summary.method = "tukey.biweight")
```

RMA

- is an expression measure consisting of three particular preprocessing steps: convolution background correction, quantile normalization and a summarization based on a multi-array model fit robustly using the median polish algorithm

```
> eset <- rma(Dilution)
```

- The function `justRMA` can be used instead of `rma` in cases where there are a large number of `CEL` files to process and no other low-level analysis is desired

GCRMA

- since the global background adjustment in RMA ignores the different propensities of probes to undergo non-specific binding (NSB), the background is often underestimated
- The characteristics of each probe are determined by its sequence
- Using the sequence information an *affinity* measure is computed.
- A background adjustment method motivated by this model has been implemented and together with quantile normalization and the median polish procedures, used by RMA, define a new expression measure called GCRMA.

GRMA

- the function `gcrma` computes GCRMA expression measures from *AffyBatch* objects and returns them in `exprSet` objects

```
> library("gcrma")
> Dil.expr <- gcrma(Dilution)
```

- the affinity information can be computed once and saved for future analysis in the following way:

```
> ai <- compute.affinities(cdfName(Dilution))
> Dil.expr <- gcrma(Dilution, affinity.info = ai)
```

GCRMA

- the default background correction method in GCRMA uses both probe affinity information and the observed MM intensities(`type="fullmodel"`).
- Users can choose to use only affinity information by setting `type="affinities"` or to use only MM intensities with `type="mm"`.

```
> Dil.expr2 <- gcrma(Dilution, affinity.info = ai,
+     type = "affinities")
```

- `justGCRMA` can be used to compute expression measures directly from `CEL` files.

Assessing preprocessing methods

- the existing alternatives for background correction, normalization and summarization yield a great number of different possible methods for preprocessing probe level data
- however, we seem to have passed the point where it makes sense to simply say: *here is a new method*, and rather we should start to expect *here is a better method*, and to have better actually mean something.

affycomp

- affycomp package provides a *graphical tool* for the assessment of preprocessing procedures for Affymetrix probe level data
- it is based on publicly available benchmark data
- affycomp contains most of the software needed to carry out the comparison
- check out the results at
  `http://affycomp.biostat.jhsph.edu/`

Quality Assessment

- You will find more details on this part of the lecture in Chapter 3, *Quality Assessment of Affymetrix GeneChip Data*.
- the affyPLM package provides one set of tools for quality assessment of Affymetrix microarrays.
- there are also tools for quality assessment in the simpleaffy package
- those in simpleaffy are largely based on Affymetrix recommendations and make use of different provided control spots on the arrays.

Example Data

- We load the ALLMLL example data and take a subset of it.

```
> library("ALLMLL")
> data(MLL.B)
> Data <- MLL.B[, c(2, 1, 3:5, 14, 6, 13)]
> sampleNames(Data) <- letters[1:8]
```

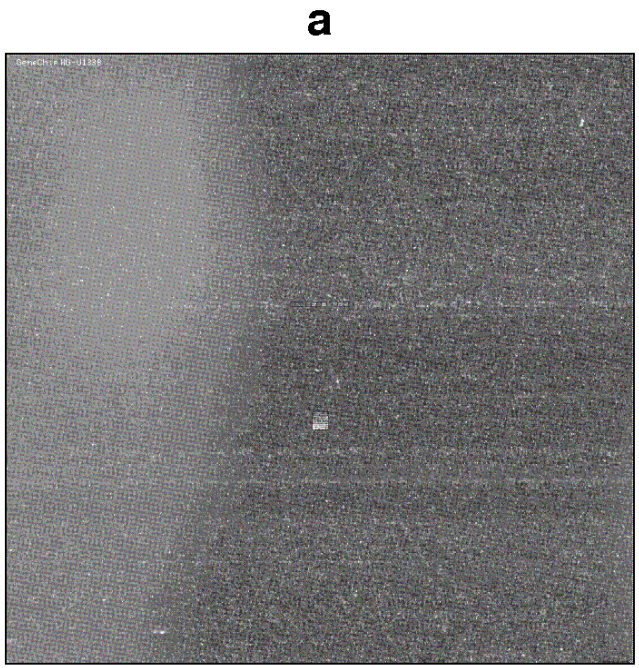


Figure 4: Image plot of intensities on a log scale.

Affymetrix quality assessment metrics

Affymetrix has proposed a number of different quality metrics.

- Average Background: the average of the 16 background values (see Section 2.3.1).
- Scale Factor: The constant β_i which is the ratio of the trimmed mean for array i to the trimmed mean of the reference array (Table 2.1).
- Percent Present: the percentage of spots that are present according to Affymetrix detection algorithm.
- $3'/5'$ ratios: for different quality control probe sets, such as $\beta-$Actin and GAPDH, each represented by 3 probesets, one from the $5'$ end, one from the middle and one from the $3'$ end of the targeted transcript. The ratio of the $3'$ expression to the $5'$ expression for these genes serves as a measure of RNA quality.

Affymetrix QC

First we load simpleAffy and call the qc function to do the computations.

```
> library("simpleaffy")
> Data.qc <- qc(Data)
```

Average background for each array is computed by:

```
> avbg(Data.qc)
```

```
       a         b         c         d         e
 68.18425  67.34494  42.12819  61.31731  53.64844
       f         g         h
128.41264  49.39112  49.25758
```

The average background values should be comparable to each other. Notice the large background value for array `f`. This might be indicative of a problem.

Affymetrix QC

The scale factors can be computed using:

```
> sfs(Data.qc)
```

```
[1]  9.765986  4.905489 10.489529  7.053323
[5]  7.561613  2.475224 13.531238  8.089458
```

These values should be within 3-fold of each other. In this example there appears to be a problem with, for example, arrays `f` and `g`.

The percentage of present calls can be obtained with the following code:

```
> percent.present(Data.qc)
```

```
a.present b.present c.present d.present e.present
 21.65158  26.53124  25.58181  23.53279  23.35615
f.present g.present h.present
 25.25061  17.96423  24.40274
```

These should be similar for replicate samples with extremely low values being a possible indication of poor quality.

Finally, the $3'/5'$ ratios for the first two quality control probesets are computed with:

```
> ratios(Data.qc)[, 1:2]

  AFFX-HSAC07/X00351.3'/5'
a                0.9697007
b                0.3235390
c                0.4661537
d                1.2567868
e                0.6036608
f                0.6715308
g                0.3798125
h                0.4850414
  AFFX-HUMGAPDH/M33197.3'/5'
a                 0.16387418
b                 0.05796629
c                -0.15570382
d                 0.57552773
e                -0.14019396
f                 0.24674941
g                -0.01830517
h                 0.27684843
```

These should be less than 3.

RNA degradation

- RNA degradation plots inform us as to whether there are big differences in RNA degradation between arrays.
- The amount of degradation (slope of the lines) is not that important, but rather whether one (or more) lines have very different slopes, or other features, than the others
- these differences can manifest themselves in altered estimates of expression.
- For any single probeset the probe effects dominate even the most dramatic signs of degradation; a $3'/5'$ trend only becomes apparent on the average over large numbers of probesets.

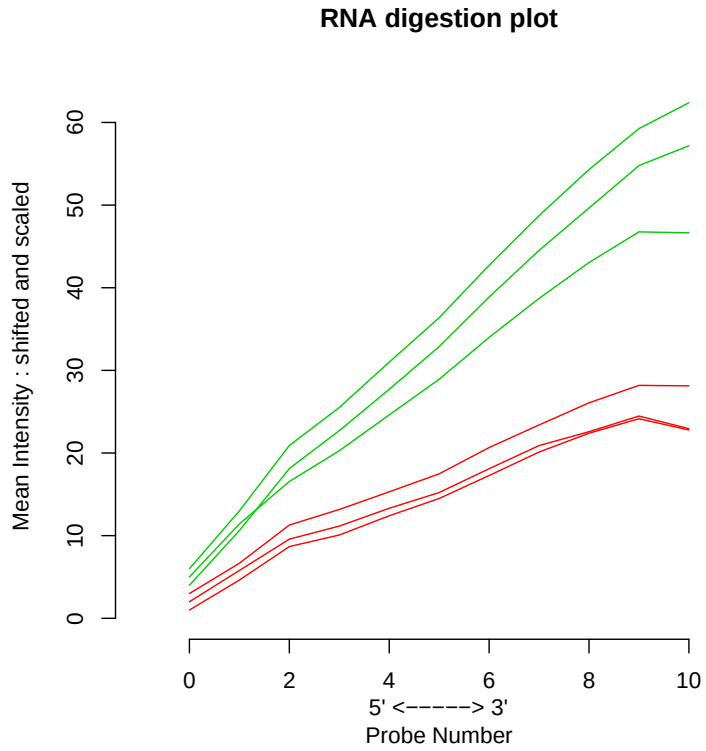


Figure 5: Each line represents one of 6 HG-U133A chips. Plotted on the Y axis is mean intensity by probeset position. Intensities have been shifted from original data for a clearer view, but slope is unchanged.

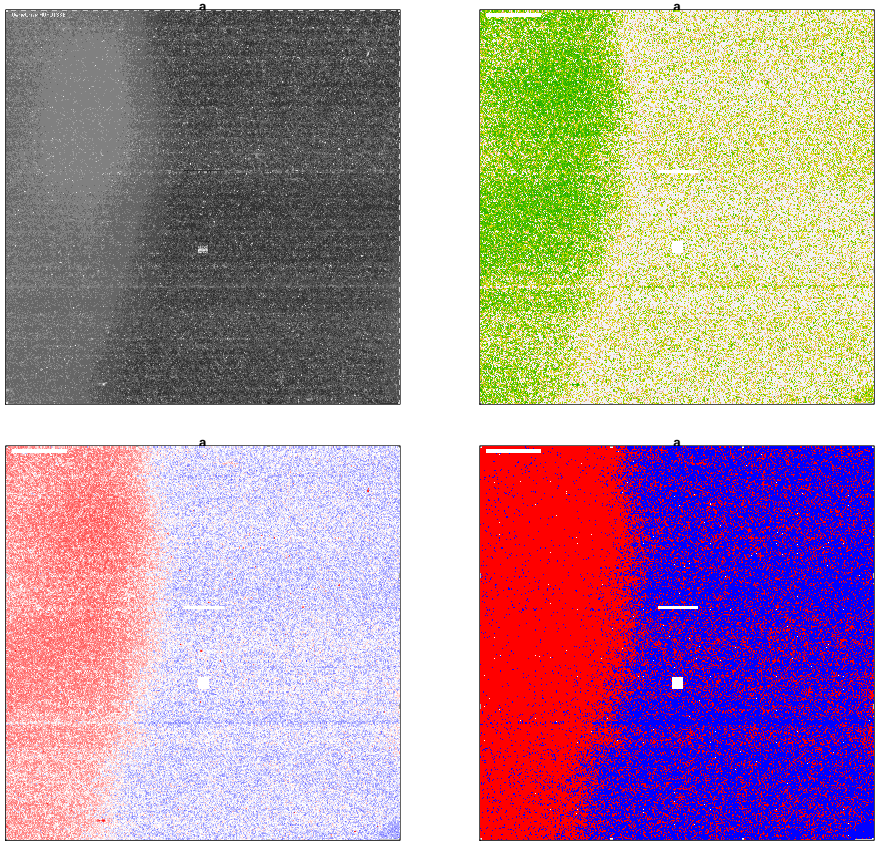


Figure 6: *a*: raw data. *b*: weights used in fitting the model, low weights are dark green *c*: residuals *d*: signs of the residuals: red positive, blue negative

Quality Assessment

- It is not uncommon to have some artifacts and small blemishes on microarrays. And in general these will have a small effect on the estimates of gene expression.
- But, some slides can be of very poor quality, or substantially different from other slides. And we would like to identify and potentially adjust for, or remove, such slides.
- B. Bolstad has done a lot of work in this area and the affyPLM has tools for carrying out some diagnostic proceedures.

Relative Log Expression (RLE)

- Compute the log scale estimates of expression $\hat{\theta}_{gi}$ for each gene g on each array i,
- next compute the median value across arrays for each gene, m_g,
- define the relative expression as $M_{gi} = \hat{\theta}_{gi} - m_g$.
- these relative expressions are then displayed with a boxplot for each array,
- an array that has problems will either have larger spread, or will not be centered at $M = 0$, or both

Normalized Unscaled Standard Error (NUSE)

- we estimate the standard error for each gene on each array from the PLM fit,
- to account for the fact that variability differs considerably between genes standardize these standard error estimates so that the median standard error across arrays is 1 for each gene
- specifically, NUSE values are computed using

$$\text{NUSE}\left(\hat{\theta}_{gi}\right) = \frac{\text{SE}\left(\hat{\theta}_{gi}\right)}{\text{med}_i\left(\text{SE}\left(\hat{\theta}_{gi}\right)\right)}.$$

- plot the NUSE values using boxplots
- low quality arrays are those that are significantly elevated or more spread out, relative to the other arrays
- NUSE values are not comparable across data sets

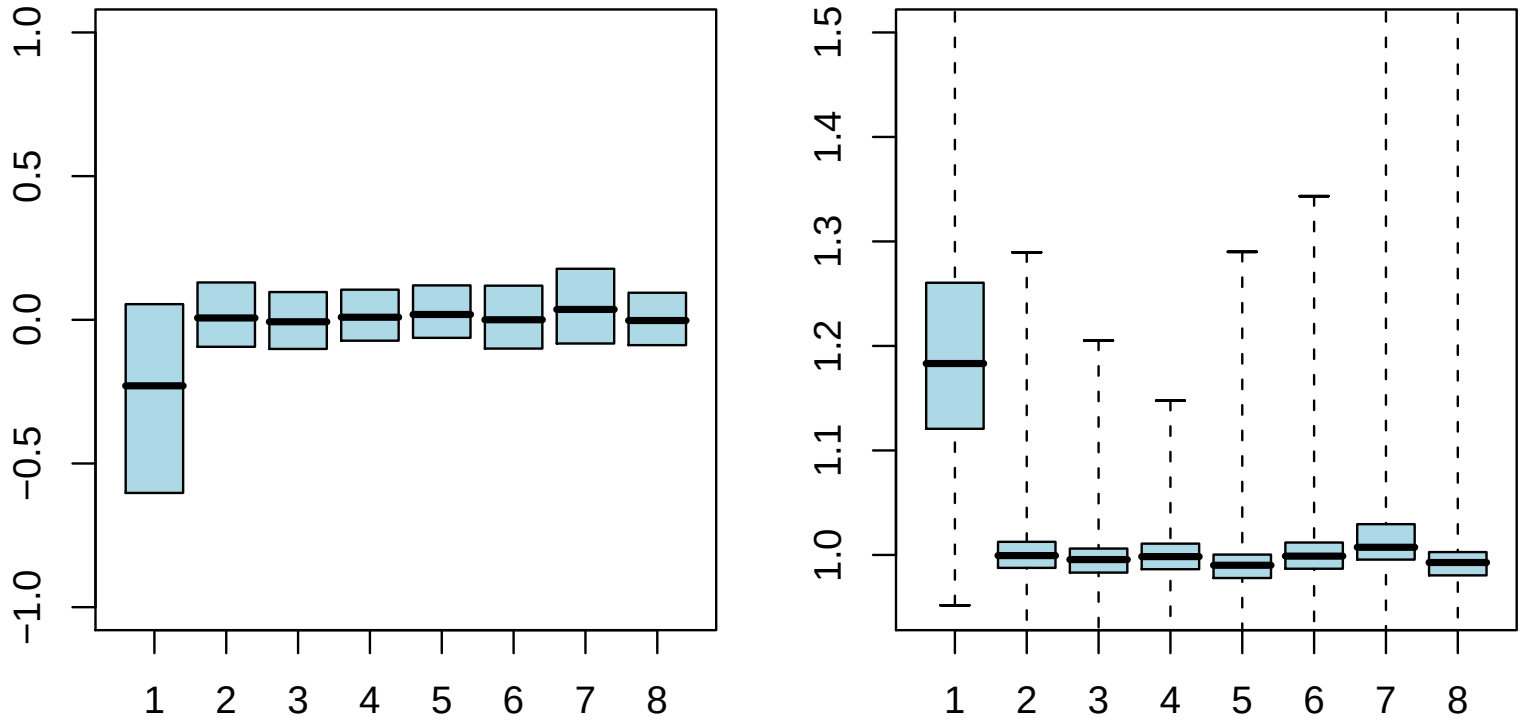


Figure 7: Interquartile ranges of RLE (a) and box-and-whiskers plots of NUSE values (b) for the ALLAML data.

Interpretation of RLE

- notice that array 1 shows fairly substantial problems in both the NUSE and RLE plots
- this array seems to be enough different from the others that its use in the analysis is suspect