Preprocessing Affymetrix Data

Educational Materials

Modified 21 November, 2009, M. Morgan

Input, Quality Assessment, and Pre-processing

Fast Track

- Identify CEL files.

  ```
  > fls <- list.files("my/directory", ".*CEL")
  ```

- Process CEL files using RMA (robust multi-array average).

  ```
  > eset <- justRMA(filenames = fls)
  ```

  End result: an *ExpressionSet* instance suitable for down-stream analysis.

A more relaxed journey

- Input raw data

  ```
  > library(affy)
  > aBatch <- ReadAffy(filenames = fls)
  ```

- Perform quality assessment

  ```
  > library(arrayQualityMetrics)
  ```

```
> outputDirectory <- tempfile()
> arrayQualityMetrics(aBatch, outputDirectory)
> browseURL(file.path(outputDirectory, "QMreport.html"))
```

Take appropriate actions based on the quality assessment.

- Custom background correct, normalize, and summarize probes.

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

End result: an *ExpressionSet* instance suitable for down-stream analysis.

Importing and Exploring Data with `ReadAffy`

A sample data set:

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

```
AffyBatch object
size of arrays=640x640 features (35221 kb)
cdf=HG_U95Av2 (12625 affyids)
number of samples=4
number of genes=12625
annotation=hgu95av2
notes=
```

- `CEL` files contain one observation per spot
- `CDF` files map from spot locations to probeset and ultimately to the identity of the gene being probed
- Bioconductor “annotation” packages map from probe sets to gene and other annotations.
- Tab-delimited, database, or other files provide phenotypic information.

Phenotypic Data

- Coordinated tracking of assay and phenotype data helps avoid mislabeling.
- `phenoData` reports phenotypic information; this can be added when the CEL files are input using `ReadAffy`.
- `pData` accesses phenotypic information.

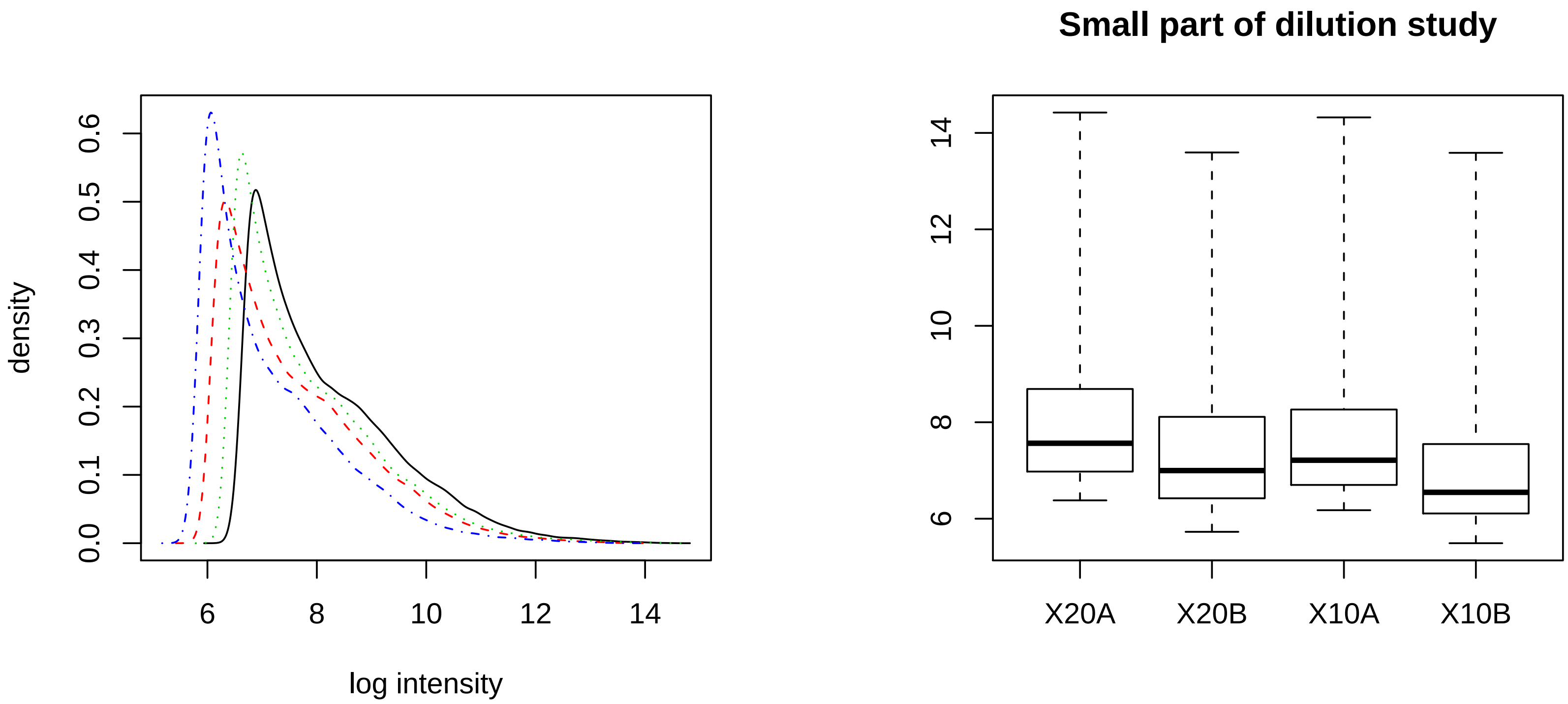
```
> pData(Dilution)
    liver sn19 scanner
20A    20    0       1
20B    20    0       2
10A    10    0       1
10B    10    0       2
```

- `varMetadata` describes what the columns of `pData` represent, in this case micrograms of liver (`liver`) or central nervous system (`sn19`) RNA hybridized to the array, and scanner ID number.

Probe Intensities

- `hist` and `boxplot` can be used to examine raw probe intensity behavior across 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 1: a) Density estimates of data from the dilution experiment. The x-axis is on a logarithmic scale (base 2). b) Box-plots.

Probe-level Data

`pm` and `mm` access probe level data.

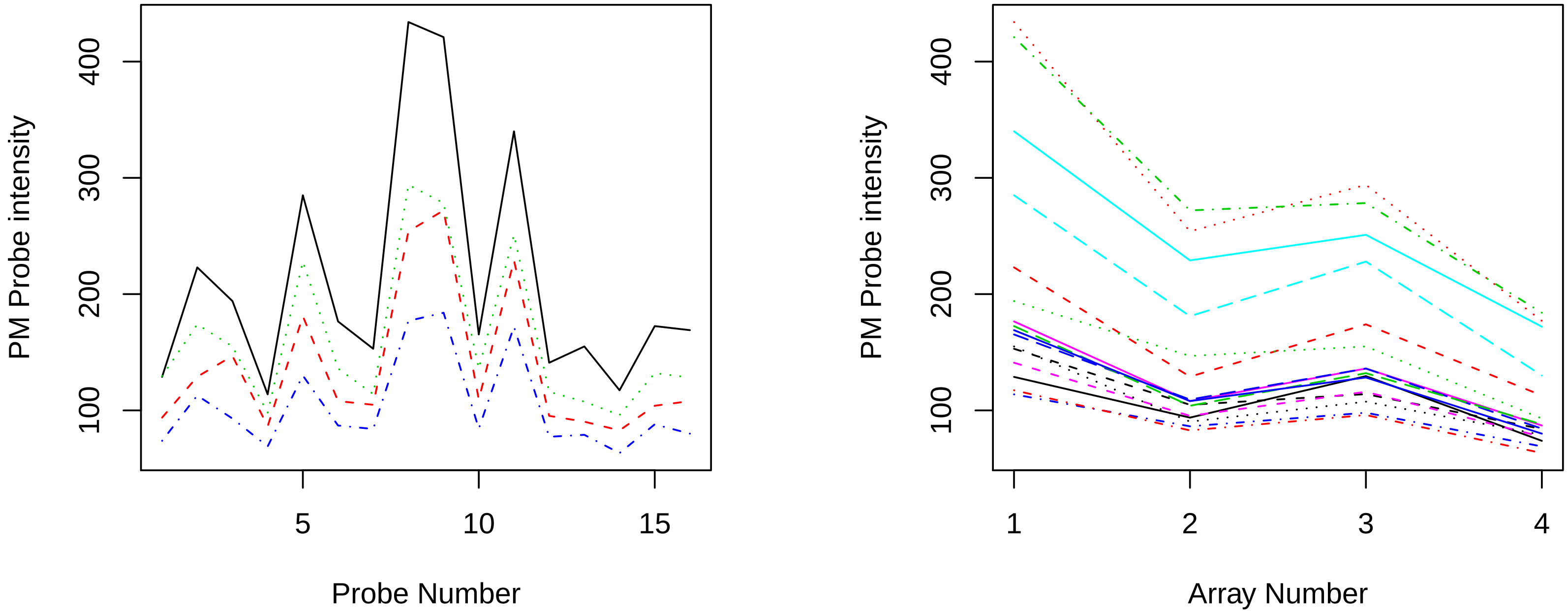
```
> pm(Dilution, "1001_at")[1:3, ]
```

```
            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
```

We can plot intensities of probe `1001_at` with

```
> matplot(pm(Dilution, "1001_at"), type = "l",
+     xlab = "Probe Number", ylab = "PM Probe intensity")
> matplot(t(pm(Dilution, "1001_at")), type = "l",
+     xlab = "Array Number", ylab = "PM Probe intensity")
```

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



a) b)

Figure 2: Examining the probe response pattern for a particular probeset a) across probe or b) across arrays.

MA Plots

- The MA plot is a rotated version of a scatter plot. The rotation helps to detect patterns as deviations from horizontal, rather than diagonal.
- Instead of ploting two vectors $Y_{2,j}$ versus $Y_{1,j}$, 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 represents fold change for gene j
  - A_j represents average log intensity for gene j.

```
> op = par(mfrow = c(2, 2))
> MAplot(Dilution, plot.method = "smoothScatter",
+     pch = 20)
> par(op)
```

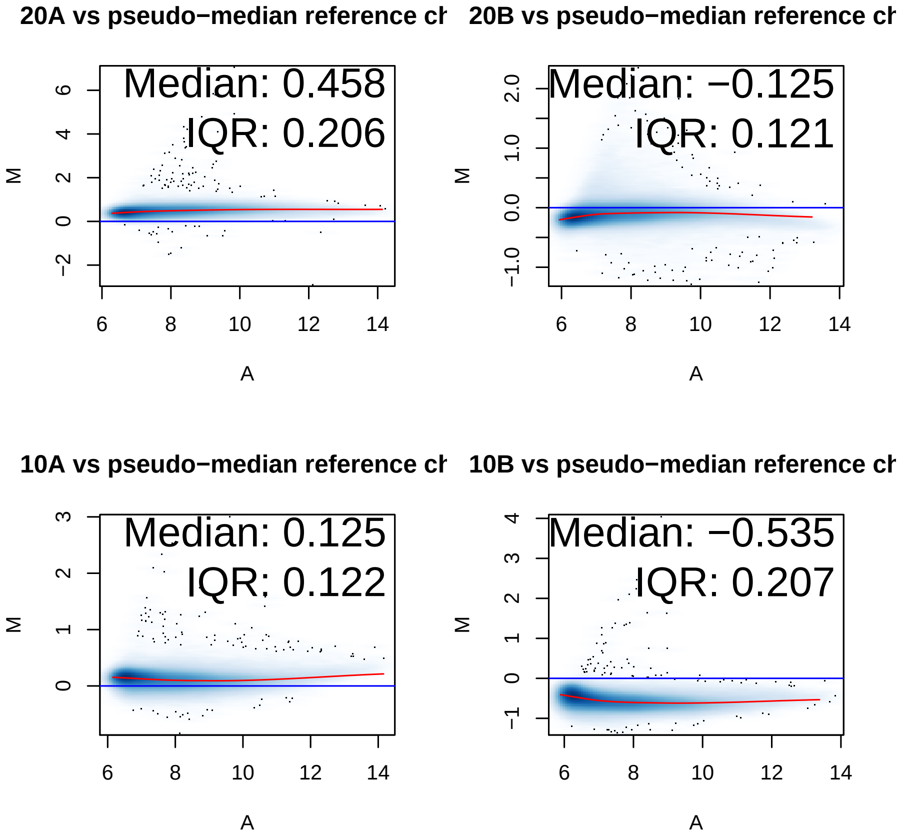


Figure 3: An MA plot for the Dilution data. Scatterplots were computed using the `smoothScatter` function.

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 differential expression (M) and average intensity (A).

Quality Assessment

- Packages providing quality assessment functionality: affyPLM (probe-level models); simpleaffy (implementing Affymetrix recommendations); arrayQualityMetrics (diverse arrays and data types).
- We load the ALLMLL example data (an `AffyBatch` instance) 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]
```

QA - Metrics

Affymetrix proposes a number of different quality metrics.

- Average Background: the average of the 16 background values.
- 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.
- 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 β-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.

QA - Example

We start by loading simpleaffy and calling the `qc` function

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

The average background for each array

```
> 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
```

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

QA - Example

The scale factors

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

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

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

```
> 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
```

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 should be less than 3:

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

  actin3/actin5 actin3/actinM
a     0.9697007    0.12291462
b     0.3235390   -0.19439139
c     0.4661537   -0.14331962
d     1.2567868    0.15861351
e     0.6036608    0.02095918
f     0.6715308    0.02916033
g     0.3798125   -0.15918419
h     0.4850414   -0.17911051
```

QA - 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.

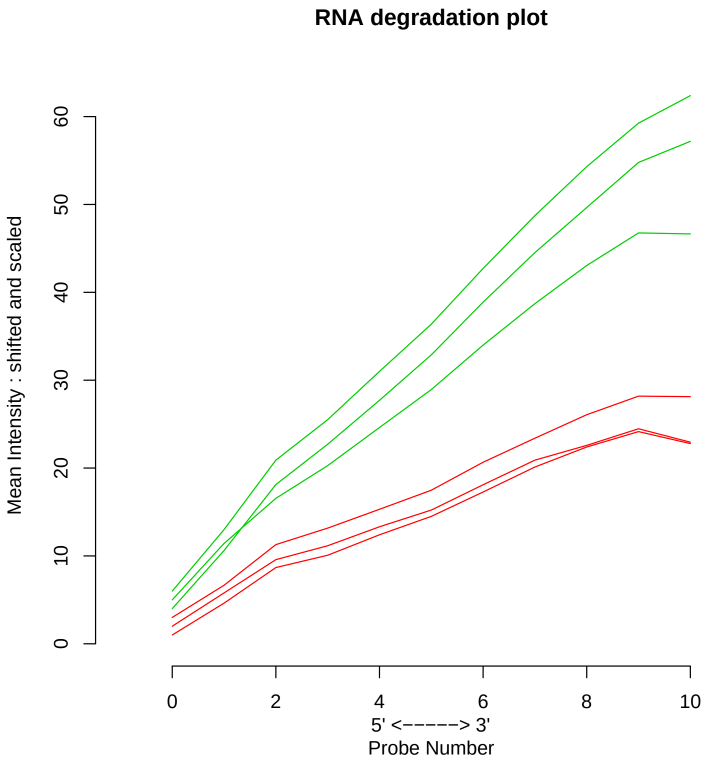


Figure 4: Each line represents one of 6 HG-U133A chips and shows the mean intensity by probeset position. Intensities have been shifted from original data for a clearer view, but slope is unchanged.

QA - Relative Log Expression (RLE)

- Compute the estimates (on a log scale) of expression $\hat{\theta}_{gi}$ for each gene g on each array i,
- Compute the median value across arrays for each gene, m_g,
- Define relative expression as $M_{gi} = \hat{\theta}_{gi} - m_g$.
- Display relative expressions as 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.

QA - Normalized Unscaled Standard Error (NUSE)

- Estimate the standard error for each gene on each array from the PLM fit.
- Account for variability between genes by adjusting the standard error estimates so that the median across arrays is 1 for each gene.

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

- 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

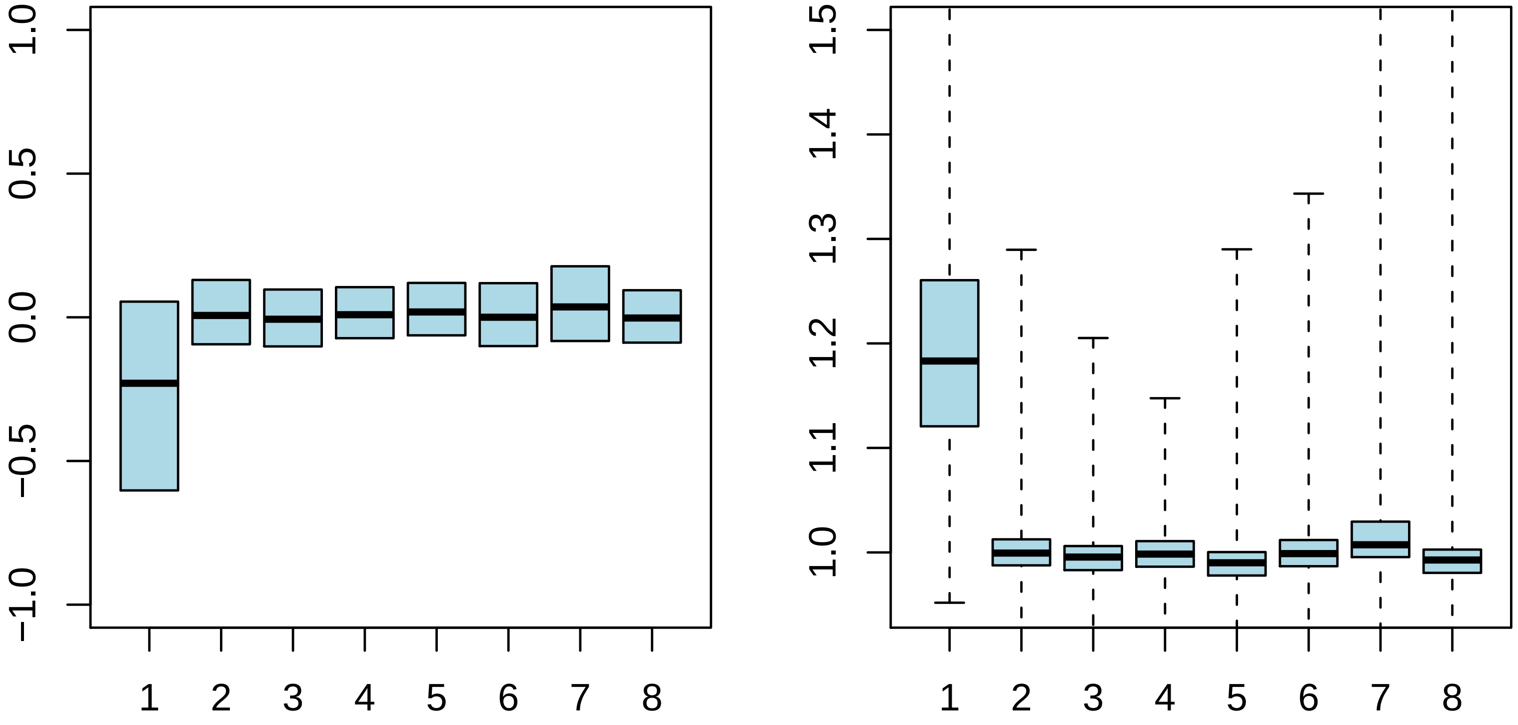


Figure 5: Interquartile ranges of RLE (a) and box-and-whiskers plots of NUSE values (b) for the ALLAML data.

QA - Interpretation of RLE and NUSE

- 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.

Pre-processing

- Preprocessing Affymetrix expression arrays usually involves three steps:
  1. Background correction.
  2. Normalization.
  3. Summarization: from probes to probesets.
- Bioconductor implements a wide variety of methods for each of these steps.
- Routines for background correction and normalization usually take an *AffyBatch* as input and return a processed *AffyBatch*.
- Routines for summarization produce *ExpressionSet* objects containing expression summary values.

Background Correction

- RMA convolution: detailed below.
- MAS 5.0: adjust probe intensities based on weighted average local background intensities.
- Ideal Mismatch: subtract mismatch intensities, adjusted to be smaller than perfect match intensities, from the PM intensities.
- Etc.

Background Correction - RMA Convolution

- MM probes are problematic (e.g., signal larger than PM), so…
- Correct PM values, array by array, using the empirical distribution of probe intensities.
- 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 values, truncate the Normal distribution at zero.
- `bg.correct` applies RMA to an *AffyBatch* object.

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

Normalization

- Goal: make measurements from different arrays comparable.
- Linear: scale normalization.
- Non-linear: cross-validated splines (Schadt et al 2001), running median lines (Li and Wong, 2001), loess smoothers (Bolstad et al.)
- Quantile: imposes the same empirical distribution of intensities to each array.
- `normalize.methods` displays available methods.
- `normalize` or specialized functions (e.g., in the `vsn` package) perform normalization.

Normalization - Linear

```
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 $\beta_i$.
end for
```

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

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

Normalization - Non-Linear

```
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")
```

Normalization - Quantile

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")
```

Normalization - VSN

- Variance Stabilizing Normalization (VSN) combines background correction and normalization.
- 'Shares' information across arrays to estimate background correction parameters.
- For a data matrix x_{ki}, with probe k and array i, fit

$$x_{ki} \mapsto h_i(x_{ki}) = \text{glog}\left(\frac{x_{ki} - a_i}{b_i}\right), \tag{1}$$

  where b_i is the scale parameter for array i, a_i is a background offset, and glog is the generalized (attenuated) logarithm.

- Normalizes data across arrays, makes variances across replicates approximately independent of the mean.

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

Summarization

- Goal: combine the multiple probe intensities for each probeset to produce an expression value
- A variety of methods available, usually integrated with background correction and normalization.

Combining background correction, normalization, and summarization.

- `rma` and `gcrma` are optimized for computing specific expression measures.
- `expresso` and `threestep` (in the affyPLM package) allow 'mix-and-match' pre-processing.

Combined - RMA and GCRMA

- RMA: convolution background correction, quantile normalization, and a summarization based on a multi-array model fit robustly using the median polish algorithm:

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

- `justRMA` processes `CEL` files directly and more efficiently.
- GCRMA computes an *affinity* measure using probe sequence information, and uses this for background correction.

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

- `justGCRMA` compute expression measures directly from `CEL` files.

Combined - `expresso` and `threestep`

- Most background adjustment, normalization and summarization methods can be combined.
- Use `bgcorrect.methods` `pmcorrect.methods` and `express.summary.stat.methods` to discover available methods available methods.
- `threestep` implemented in compiled code, so faster than `expresso`.

Example: compute expression measures where the ideal mismatch is subtracted from PM, quantile normalization occurs between arrays, and probesets are summarized using a robust average.

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

Summary

This presentation:

- Structured data types (e.g., *AffyBatch*, *ExpressionSet*) help keep complex data organized throughout analysis.
- Quality assessment and pre-processing tools provide extremely detailed and flexible access to data. These tools rely on the statistical, graphical, scripting, and interactive capabilities of R.

Elsewhere in Bioconductor:

- ≈ 350 packages available, organized into software views, with key packages highlighted for particular work flows.
- Support for many different platforms (e.g., one- and two channel, Affy, Agilent, Nimblegen, Illumina) and analyses (e.g., Expression, miRNA, tiling, SNP).