affy: Built-in Processing Methods

Ben Bolstad

October 17, 2016

Contents

1 Introduction

This document describes the preprocessing methods that have currently been built into the `affy` package. Hopefully it will clarify for the reader what each of the routines does. There is a separate vignette which describes how you might write your own routines and use them in combination with the built-in routines.

As usual, loading the package in your `R` session is required.

```
R> library(affy) ##load the affy package
```

2 Background methods

You can see the background correction methods that are built into the package by examining the variable `bgcorrect.method`.

```
> bgcorrect.methods()

[1] "bg.correct" "mas"        "none"       "rma"
```

2.1 none

Calling this method actually does nothing. It returns the object unchanged. May be used as a placeholder.

2.2 rma/rma2

These are background adjustment implementations for the rma method ?. They differ only in how they estimate a set of parameters (generally you should use `rma` in preference to `rma2`. In both cases PM probe intensities are corrected by using a global model for the distribution of probe intensities. The model is suggested by looking at plots of the empirical distribution of probe intensities. In particular the observed PM probes are modeled as the sum of a normal noise component N (Normal with mean μ and variance σ^2) and a exponential signal component S (exponential with mean α). To avoid any possibility of negatives, the normal is truncated at zero. Given we have O the observed intensity, this then leads to an adjustment.

$$E\left(s|O=o\right)=a+b\frac{\phi\left(\frac{a}{b}\right)-\phi\left(\frac{o-a}{b}\right)}{\Phi\left(\frac{a}{b}\right)+\Phi\left(\frac{o-a}{b}\right)-1}$$

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

Note that MM probe intensities are not corrected by either of these routines.

2.3 mas

This is an implementation of the background correction method outlined in the Statistical Algorithms Description Document ?. The chip is broken into a grid of 16 rectangular regions. For each region the lowest 2% of probe intensities are used to compute a background value for that grid. Each probe is then adjusted based upon a weighted average of the backgrounds for each of the regions. The weights are based on the distances between the location of the probe and the centriods of 16 different regions. Note this method corrects both PM and MM probes.

3 Normalization Methods

You can see the background correction methods that are built into the package by examining the variable `bgcorrect.method`.

```
> normalize.AffyBatch.methods()

[1] "constant"          "contrasts"         "invariantset"      "loess"
[5] "methods"           "qspline"           "quantiles"         "quantiles.robust"
```

The Quantile, Contrast and Loess normalizations have been discussed and compared in ?.

3.1 quantiles/quantiles.robust

The quantile method was introduced by **?**. The goal is to give each chip the same empirical distribution. To do this we use the following algorithm where X is a matrix of probe intensities (probes by arrays):

1. Given n array of length p, form X of dimension $p \times n$ where each array is a column
2. Sort each column of X to give X_{sort}
3. Take the means across rows of X_{sort} and assign this mean to each element in the row to get X'_{sort}
4. Get $X_{\text{normalized}}$ by rearranging each column of X'_{sort} to have the same ordering as original X

The quantile normalization method is a specific case of the transformation $x'_i = F^{-1}(G(x_i))$, where we estimate G by the empirical distribution of each array and F using the empirical distribution of the averaged sample quantiles. Quantile normalization is pretty fast.

The `quantiles` function performs the algorithm as above. The `quantile.robust` function allows you to exclude or down-weight arrays in the computation of $\hat{G}$ above. In most cases we have found that the `quantiles` method is sufficient for use and `quantiles.robust` not required.

3.2 loess

There is a discussion of this method in **?**. It generalizes the M vs A methodology proposed in **?** to multiple arrays. It works in a pairwise manner and is thus slow when used with a large number of arrays.

3.3 contrasts

This method was proposed by **?**. It is also a variation on the M vs A methodology, but the normalization is done by transforming the data to a set of contrasts, then normalizing.

3.4 constant

A scaling normalization. This means that all the arrays are scaled so that they have the same mean value. This would be typical of the approach taken by Affymetrix. However, the Affymetrix normalization is usually done after summarization (you can investigate `affy.scalevalue.exprSet` if you are interested) and this normalization is carried out before summarization.

3.5 invariantset

A normalization similar to that used in the dChip software **?**. Using a baseline array, arrays are normalized by selecting invariant sets of genes (or probes) then using them to fit a non-linear relationship between the "treatment" and "baseline" arrays. The non-linear relationship is used to carry out the normalization.

3.6 qspline

This method is documented in **?**. Using a target array (either one of the arrays or a synthetic target), arrays are normalized by fitting splines to the quantiles, then using the splines to perform the normalization.

4 PM correct methods

```
> pmcorrect.methods()
```

```
[1] "mas"        "methods"    "pmonly"     "subtractmm"
```

4.1 mas

An *ideal mismatch* is subtracted from PM. The ideal mismatch is documented by **?**. It has been designed so that you subtract MM when possible (ie MM is less than PM) or something else when it is not possible. The Ideal Mismatch will always be less than the corresponding PM and thus we can safely subtract it without risk of negative values.

4.2 pmonly

Make no adjustment to the pm values.

4.3 subtractmm

Subtract MM from PM. This would be the approach taken in MAS 4 **?**. It could also be used in conjunction with the Li-Wong model.

5 Summarization methods

```
> express.summary.stat.methods()
```

```
[1] "avgdiff"      "liwong"       "mas"          "medianpolish" "playerout"
```

5.1 avgdiff

Compute the average. This is the approach that was taken in **?**.

5.2 liwong

This is an implementation of the methods proposed in **?** and **?**. The Li-Wong MBEI is based upon fitting the following multi-chip model to each probeset

$$y_{ij} = \phi_i \theta_j + \epsilon_{ij} \tag{1}$$

where y_{ij} is PM_{ij} or the difference between $PM_{ij} - MM_{ij}$. The ϕ_i parameter is a probe response parameter and θ_j is the expression on array j.

5.3 mas

As documented in **?**, a robust average using 1-step Tukey biweight on $\log_2$ scale.

5.4 medianpolish

This is the summarization used in the RMA expression summary **?**. A multichip linear model is fit to data from each probeset. In particular for a probeset k with $i = 1, \ldots, I_k$ probes and data from $j = 1, \ldots, J$ arrays we fit the following model

$$\log_2 \left(PM_{ij}^{(k)} \right) = \alpha_i^{(k)} + \beta_j^{(k)} + \epsilon_{ij}^{(k)}$$

where α_i is a probe effect and β_j is the $\log_2$ expression value. The medianpolish is an algorithm (see **?**) for fitting this model robustly. Please note that expression values you get using this summary measure will be in $\log_2$ scale.

5.5 playerout

This method is detailed in **?**. A non-parametric method is used to determine weights. The expression value is then the weighted average.

6 Putting it altogether using `expresso`

The function that you should use is `expresso`. It is important to note that not every preprocessing method can be combined together. In particular the `rma` backgrounds adjust only PM probe intensities and so they should only be used in conjunction with the `pmonly` PM correction. Also remember that the `mas` and `medianpolish` summarization methods $\log_2$ transform the data, thus they should not be used in conjunction with any preprocessing steps that are likely to yield negatives like the `subtractmm` pm correction method. The following is a typical call to `expresso`.

```
library(affydata)
data(Dilution)
eset <- expresso(Dilution,bgcorrect.method="rma",
                 normalize.method="quantiles",
                 pmcorrect.method="pmonly",
                 summary.method="medianpolish")
```

This would give you the RMA expression measure, but of course there are other ways of computing RMA (chiefly `rma`). The true power of `expresso` becomes apparent when you start combining different methods. By choosing a method for each of the four steps (`bgcorrect.method`, `normalize.method`, `pmcorrect.method`, `summary.method`) you can create quite a variety of expression measures. For instance

```
eset <- expresso(Dilution,bgcorrect.method="mas",
                 normalize.method="qspline",
                 pmcorrect.method="subtractmm",
                 summary.method="playerout")
```

would be a valid way of computing an expression measure (it is up to the user to decide whether such a concoction is sensible or not).