Preprocessing and Barcoding of Single Microarrays and Microarray Batches (frma)

Matthew N. McCall

October 18, 2010

Contents

1 Frozen Robust Multiarray Analysis (fRMA)

Frozen RMA (fRMA) is a microarray preprocessing algorithm that allows one to analyze microarrays individually or in small batches and then combine the data for analysis. This is accomplished by utilizing information from the large publicly available microarray databases. In particular, estimates of probe-specific effects and variances are precomputed and frozen. Then, with new data sets, these are used in concert with information from the new array(s) to normalize and summarize the data.

This section describes `frma`, which implements the methods proposed in the paper, *Matthew N. McCall, Benjamin M. Bolstad, and Rafael A. Irizarry, "FROZEN ROBUST MULTI-ARRAY ANALYSIS (fRMA)" (May 2009). Johns Hopkins University, Dept. of Biostatistics Working Papers. Working Paper 189.* Ideally someone interested in using this package would first read that paper and then proceed to the sections below.

1.1 Getting Started

If all you want is to go from probe level data (*CEL* files) to expression measures here are some quick ways.

1.1.1 Single Array: Median Polish

If you want a quick single array version of RMA using median polish, the quickest way of reading in data and getting expression measures is the following:

1. Download and install frma and the appropriate data file (i.e. hgu133afrmavecs, hgu133plus2frmavecs, etc.).
2. Create a directory and move all the relevant *CEL* files to that directory.
3. If using linux/unix, start R in that directory.
4. If using the Rgui for Microsoft Windows make sure your working directory contains the *CEL* files (use "File -> Change Dir" menu item).
5. Load the library.

   ```
   > library(frma)
   ```

6. Read in the data and preprocess using the "median_polish" option.

```
> library(frmaExampleData)
> data(AffyBatchExample)
> object <- frma(AffyBatchExample, summarize = "median_polish")
```

1.1.2 Single Array: Robust Weighted Average

If you want the robust single array method described in the fRMA paper, the quickest way of reading in data and getting expression measures is the following:

1. Download and install frma and the appropriate data file (i.e. hgu133afrmavecs, hgu133plus2frmavecs, etc.).
2. Create a directory and move all the relevant *CEL* files to that directory.
3. If using linux/unix, start R in that directory.
4. If using the Rgui for Microsoft Windows make sure your working directory contains the *CEL* files (use "File -> Change Dir" menu item).
5. Load the library.

   ```
   > library(frma)
   ```

6. Read in the data and preprocess using the "robust_weighted_average" option.

   ```
   > library(frmaExampleData)
   > data(AffyBatchExample)
   > object <- frma(AffyBatchExample, summarize = "robust_weighted_average")
   ```

1.1.3 Multiple Arrays: Batch Effect Adjusted

If you want the batch effect adjustment method described in the fRMA paper, the quickest way of reading in data and getting expression measures is the following:

1. Download and install frma and the appropriate data file (i.e. hgu133afrmavecs, hgu133plus2frmavecs, etc.).
2. Create a directory and move all the relevant *CEL* files to that directory.
3. If using linux/unix, start R in that directory.
4. If using the Rgui for Microsoft Windows make sure your working directory contains the *CEL* files (use "File -> Change Dir" menu item).

5. Load the library.

6. Read in the data and preprocess using the "batch" option.

In each of the cases above, the final `object` will be a PLMset. To obtain a matrix of gene-level expression values, enter the following command:

```
> e <- coefs(object)
```

Depending on the size of your dataset and on the memory available to your system, you might experience errors like 'Cannot allocate vector . . . '. An obvious option is to increase the memory available to your R process (by adding memory and/or closing external applications). You might also consider analyzing the data in smaller batches.

1.2 Advanced Options

1.2.1 Summarization Methods

Summarization refers to the method used to combine probe-level expression values to obtain gene-level expression estimates. There are several summarization methods that one can choose from when running frma. A brief description of each of the methods follow:

- **average:** compute the mean of the probes in each probeset
- **median:** compute the median of the probes in each probeset
- **median polish:** subtract the probe-effect and then compute the median of the probes in each probeset
- **weighted average:** compute a weighted average of the probes in each probeset with weights equal to the inverse of the sum of the precomputed within and between batch variance estimates.
- **robust weighted average:** compute a weighted average of the probes in each probeset with weights equal to the weights returned by an M-estimation procedure divided by the sum of the precomputed within and between batch variance estimates.
- **batch:** the robust weighted average method adapted for a batch of new arrays.

For a more in depth description of the final two methods, see *Matthew N. McCall, Benjamin M. Bolstad, and Rafael A. Irizarry, "FROZEN ROBUST MULTI-ARRAY ANALYSIS (fRMA)" (May 2009). Johns Hopkins University, Dept. of Biostatistics Working Papers. Working Paper 189.*

1.2.2 Input Vectors

While the vast majority of users will want to stick to the precomputed vectors provided in the frmavecs packages, the frma function will accept user-supplied vectors. These should be given in a list with elements normVec, probeVec, probeVarBetween, probeVarWithin, and probesetSD. A description of each of these elements follows:

- **normVec:** a vector containing values of the reference distribution to which samples will be quantile normalized.
- **probeVec:** a vector of probe-effect estimates.
- **probeVarBetween:** a vector of the between batch variance for each probe.
- **probeVarWithin:** a vector of the within batch variance for each probe.
- **probesetSD:** a vector of average within probeset standard deviations.

For further information on how these vectors are computed in the frmavecs packages, see the frmaTools package.

1.2.3 Output Parameters

While the default is to only return the gene-level expression estimates, the frma function can also return additional information about the estimates depending on the summarization method chosen. A description of the arguments that can be included in the output.param argument follows:

- **stderr:** the standard errors of the gene-level expression estimates.
- **weights:** the weights from the M-estimation procedure.
- **residuals:** the residuals from fitting the probe-level model.

Not all of these outputs are available for all of the summarization methods; however, all three are available for the default summarization procedure – robust weighted average.

2 Creation of Gene Expression Barcodes

The barcode algorithm is designed to estimate which genes are expressed and which are unexpressed in a given microarray hybridization. This is accopmlished by: (1) using the distribution of observed log_2 intensities across a wide variety of tissues to estimate an expressed and an unexpressed distribution for each gene, and (2) for each gene in a sample, denoting it as expressed if its observed log_2 intensity is more likely under the expressed distribution than under the unexpressed distribution and as unexpressed otherwise. The first step is accomplished by fitting a hierarchical mixture model to the plethora of publicly available data. The second step is accomplished by determining where the observed intensities from the new array fall in these distributions. The output of our algorithm is a vector of ones and zeros denoting which genes are estimated to be expressed (ones) and unexpressed (zeros). We call this a *gene expression barcode.*

This section describes `barcode`, which implements the methods proposed in the paper, *Matthew N. McCall, Michael J. Zilliox, and Rafael A. Irizarry, "Gene Expression Barcodes Based on Data from 8,277 Microarrays" (October 2009). Johns Hopkins University, Dept. of Biostatistics Working Papers. Working Paper 200.* Ideally someone interested in using this package would first read that paper and then proceed to the sections below.

2.1 Getting Started

To create a gene expression barcode, one needs estimates of the gene expression distributions – specifically the mean and variance of the unexpressed distribution for each gene. Fortunately, we have precomputed these for 3 popular Affymetrix microarray platforms – hgu133a, hgu133plus2, and mouse4302. To use one of these 3, simply preprocess your data using the default options of `frma` and then run `barcode` on the resulting object.

2.1.1 Example

1. Download and install the frma package and the appropriate data packages (i.e. hgu133afrmavecs and hgu133abarcodevecs).

2. Create a directory and move all the relevant *CEL* files to that directory.

3. If using linux/unix, start R in that directory.

4. If using the Rgui for Microsoft Windows make sure your working directory contains the *CEL* files (use "File -> Change Dir" menu item).

5. Load the libraries.

```
> library(frma)
```

6. Read in the data and preprocess using the default options.

   ```
   > library(frmaExampleData)
   > data(AffyBatchExample)
   > object <- frma(AffyBatchExample)
   ```

7. Create a gene expression barcode.

   ```
   > bc <- barcode(object)
   ```

2.2 Output Options

The default output of the `barcode` function is to return a vector of ones (expressed) and zeros (unexpressed); however, there are alternative output options. A brief description of each of these follows:

- **weight:** a vector of weights which roughly correspond to the probability of expression for each gene.

- **z-score:** a vector of z-scores under the unexpressed normal distribution for each gene.

- **p-value:** a vector of p-values under the unexpressed normal distributionfor each gene.