affyPLM: Advanced use of the MAplot function

Ben Bolstad
bmb@bmbolstad.com
http://bmbolstad.com

October 26, 2021

Contents

1 Introduction

This document is a basic users guide to the `MAplot` facilities of the affyPLM package. Other vignettes for this package describe other functionalities. Note that the `MAplot` generic function supports dealing with the *AffyBatch* (actually supplied by the affy package), *ExpressionSet* and *PLMset* objects.

To begin, load the package using

```
> library(affyPLM)
> options(width=60)
```

While the `MAplot` function can be applied to all of the objects discussed this document will illustrate the general usage of this function with an *ExpressionSet* object. However, the same `MAplot` function calls will give the same results on the *AffyBatch* and *PLMset* objects.

The *Dilution* dataset which is built into the affydata package will be used. Now `Dilution` is an *AffyBatch* which can be turned into an *ExpressionSet* object by using one of the many functions for producing expression values. In this case the `rma` function will be used.

```
> require(affydata)
> data(Dilution)
> eset.Dilution <- rma(Dilution)

Background correcting
Normalizing
Calculating Expression
```

Now `eset.Dilution` is an *ExpressionSet* containing RMA expression values.

2 Basic Usage of the MAplot function

The initial way that most users use the `MAplot` function is to simply call it by supplying only the input object. To do this with the RMA expression values in `eset.Dilution` use:

```
> par(mfrow=c(2,2))
> MAplot(eset.Dilution)
```

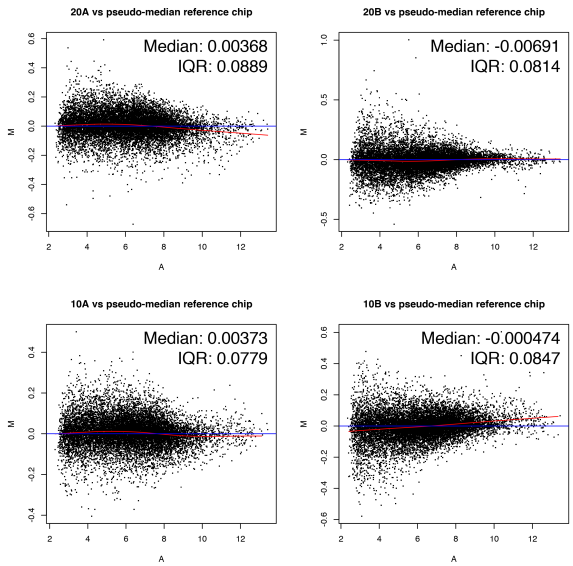

Figure 1: MA-plots comparing the expression values for each array with a synthetic probeset-wise median array.

This produces the set of MA-plots shown in Figure 1. The MA-plots produced by default compare the expression values on each array in the dataset with a synthetic array created using probeset-wise median expression values. The MA-plots have loess lines in red and the $M = 0$ horizontal axis in blue. In situations where it is believed that there should be little change in expression between arrays these can be used to assess the effectiveness of the normalization. Situations where an array has a clearly abberant loess line on these MA-plots often are indicative of potential quality problems. The median and IQR values appearing on each plot relate to the center and vertical spread of the M values. These statistics can be turned off by supplying the `show.statistics=FALSE` argument to `MAplot`.

3 Gaining greater control over the function

There are a number of optional parameters that can be provided to a call to `MAplot`. The first is `plot.method="smoothScatter"` which gives an alternative method of drawing the MA-plot. It internally uses the `smoothScatter` function to do the plotting. MA-plots can be created in this manner for the `eset.Dilution` dataset using:

```
> par(mfrow=c(2,2))
> MAplot(eset.Dilution,plot.method="smoothScatter")
```

with the resulting set of MA-plots shown in Figure 2.

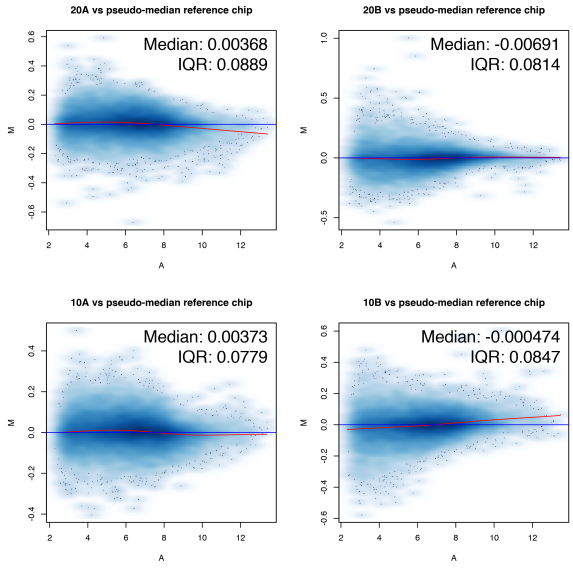

Figure 2: MA-plots created using smoothScatter

Sometimes it is more desirable to create MA-plots which compare each array against one of the arrays in the dataset rather than against a synthetic array created using

probeset-wise median expression values. The array which is used as the reference is controlled using the `ref` argument. For instance to create MA-plots using the first array as the reference array and then comparing it to each of the other 3 arrays in the dataset the following code can be used:

```
> par(mfrow=c(2,2))
> MAplot(eset.Dilution,plot.method="smoothScatter",ref=1)
```

with the results shown in Figure 3.

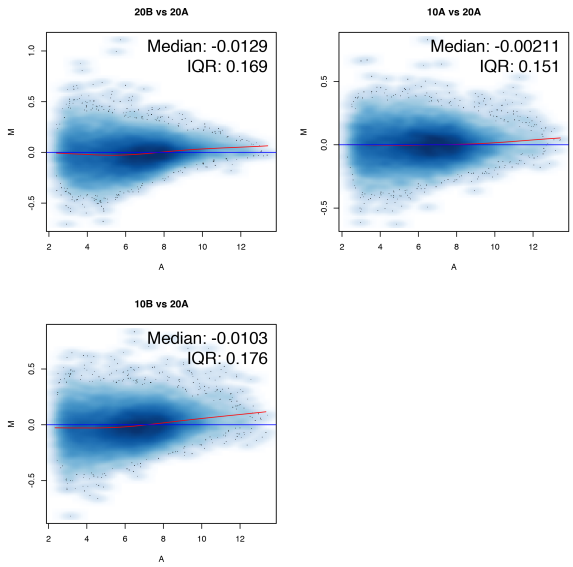

Figure 3: MA-plots created with the `ref` argument.

Additionally, rather than all possible pairwise MA-plots sometimes it desirable to look only at subset of the possible arrays. This can be done using the `which` function argument. For instance to examine MA-plots comparing the second array to the first array and also the second array to the first array the following code can be used:

```
> par(mfrow=c(2,1))
> MAplot(eset.Dilution,which=c(2,4),ref=1,plot.method="smoothScatter")
```

with the resultant plots shown in Figure 4. Notice that these are identical to the corresponding plots in Figure 3.

If it is desirable to compute the median reference array using a subset of arrays then the `ref` can also be used. For instance the following code:

```
> par(mfrow=c(2,1))
> MAplot(eset.Dilution,which=c(1,2),ref=c(1,2),plot.method="smoothScatter")
```

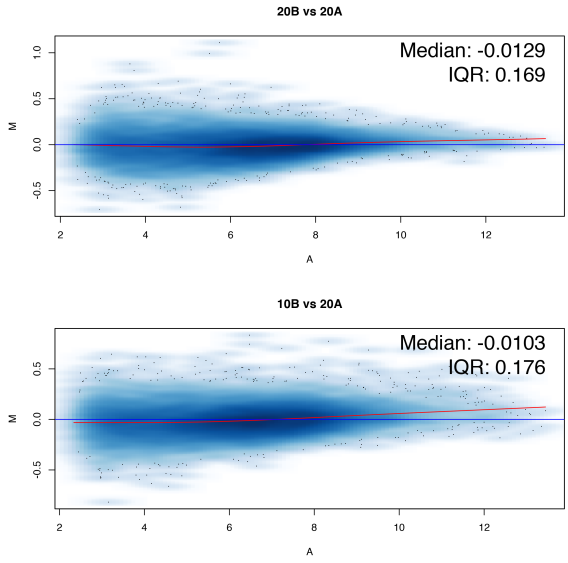

Figure 4: MA-plots created with the `which` argument.

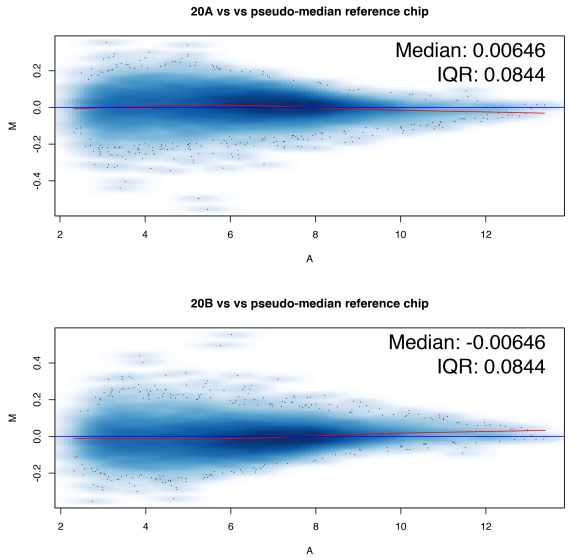

Figure 5: MA-plots created with the `ref` argument.

creates the MA-plots for the first and second arrays each against a median reference array created using these two arrays. The resulting plots are shown in Figure 5.

It should be noted that it is perfectly acceptable to supply sample names rather than integer indices for both `ref` and `which`. For example,

```
> MAplot(eset.Dilution,which=c("20A","20B"),ref=c("20A","20B"),plot.method="smoothSca
```

could be used in place of the above.

4 Advanced Usage

To demonstrate how the `MAplot` function can be used to build up an MA-plot by adding additional points Presence/Absence calls will be used stratify probesets. Use the following code to compute P/A calls and then count how many times a probeset is present in the dataset:

```
> PA.calls <- mas5calls(Dilution)
```

```
Getting probe level data...
Computing p-values
Making P/M/A Calls
```

```
> Is.Present <- exprs(PA.calls) == "P"
> Number.Present <- apply(Is.Present,1,sum)
```

The `plot.method="add"` function argument can be used to add additional points to an existing MA-plot. Note that `add.loess=FALSE` prevents the loess smoother from being added to the MA-plot. The following code produces an MA-plot comparing the first array in the dataset with the probesetwise median array. However, different color points are used to identify the number of times that probeset is called Present in the dataset.

```
> MAplot(eset.Dilution[Number.Present ==4,],show.statistics=FALSE,which=1,pch=20,cex=
> MAplot(eset.Dilution[Number.Present ==0,],plot.method="add",col="red",which=1,pch=2
> MAplot(eset.Dilution[Number.Present ==3,],plot.method="add",show.statistics=FALSE,w
> MAplot(eset.Dilution[Number.Present ==2,],plot.method="add",show.statistics=FALSE,w
> MAplot(eset.Dilution[Number.Present ==1,],plot.method="add",show.statistics=FALSE,w
```

which results in Figure 6.

Rather than just MA-plots of an individual array versus another array or combination of arrays, it is also possible to combine arrays into groups (by averaging for instance) and then compare groups. This can be done using the `groups` argument. If the groups argument is used then `ref` and `which` are refer to groups rather than individual arrays. Suppose that we want to compare the arrays which were liver dilution 20 to those that were liver dilution 10. This can be done using:

```
> MAplot(eset.Dilution,groups=c("Liver 20","Liver 20","Liver 10","Liver 10"),ref="Liv
```

with the resulting plot in Figure 7.

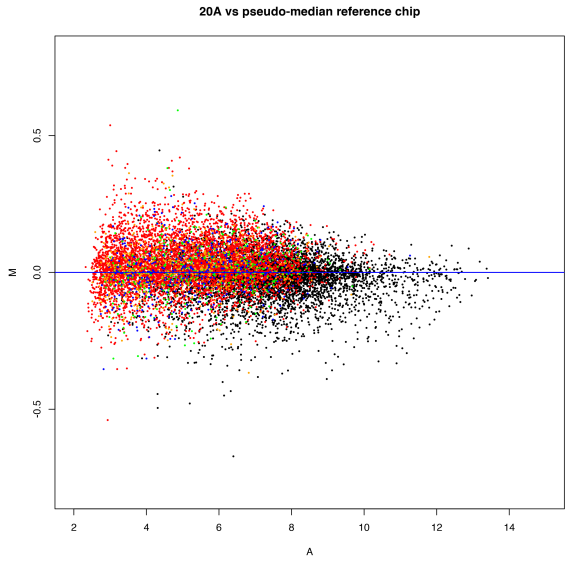

Figure 6: MA-plots created with the `add` argument.

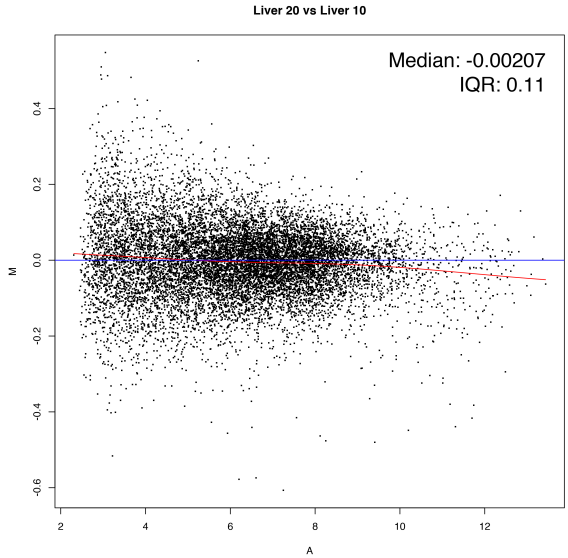

Figure 7: MA-plots created with the `groups` argument.

5 Special notes about differences in `MAplot` for *AffyBatch*, *ExpressionSet* and *PLMset* objects.

The main difference is in the value of the `log` argument. For an *AffyBatch* by default `log=TRUE`, whilst for both *ExpressionSet* and *PLMset* this argument is `log=FALSE` by default. This argument tells the function whether or the data needs to be logarithmically transformed before computing the M and A values by differencing and averaging. Note that when `log=TRUE` $\log_2$ is used. Since *AffyBatch* objects often store raw cel file data they typically need to be logged first. If `rma` or `threestep` is used then the resulting *ExpressionSet* contains $\log_2$ scale expression values. If `expresso`, or another function, is used which produces expression values in the natural scale then `log=TRUE` should be supplied to MAplot.

For *AffyBatch* objects there is also a special function argument `type` that can be used to control which probe type is plotted. The default is `type="both"`, but `type="pm"` and `type="mm"` can also be used.

6 Final Comments

The `MAplot` function has many options and is more powerful than it might first appear. This document serves to highlight some of its more advanced features and demonstrates their usage.