MBCB Usage

Jeffrey D. Allen

October 26, 2021

This file will demonstrate the usage of the MBCB package. The goal of this package is to use model-based techniques to background-correct Illumina datasets. This method makes use of the negative control probes offered on more recent Illumina microarrays.

1 Sample Data

Sample data is included in the MBCB package. You can access this data with the following command:

```
> library(MBCB);
> data(MBCBExpressionData);
```

which creates two new variables: *expressionSignal* and *negativeControl*. These two sets of data are the foundation for all of the analysis we'll do with MBCB.

Because most users will have such data stored in files, we offer a function to read the data in from files on disk and format them properly. However, in order to demonstrate these, we'll first need to write the data to files. We'll write them as tab-delimited files by using the following commands:

```
> write.table(expressionSignal, 'signal.txt', sep="\t");
> write.table(negativeControl, 'negative.control.txt', sep="\t");
```

If we wanted to read in two such files into an MBCB-friendly format, we could use the following command:

```
> data <- mbcb.parseFile('signal.txt', 'negative.control.txt');
> signal <- data$sig;
> negCon <- data$con;
```

which creates two separate matrices, one of which represents our signal data, the other of which represents our negative control data.

2 Analysis

Next, we can begin to do some analysis on the data. MBCB offers five background correction methods:

- Non-Parametric
- MLE
- GMLE
- Bayesian/MCMC
- RMA

RMA is a method commonly used on Affymetrix data which may not have negative control beads. Note that this is the only method which does not require a negative control files, but can be run solely on the signal data.

The *mbcb.correct* method is the heart of most of the data analysis. It can be used by first setting the Boolean values representing which methods we are interested in (or just using the defaults):

```
> nonparametric <- TRUE;
> RMA<- TRUE;
> MLE <- TRUE;
> GMLE <- FALSE;
> MCMC <- FALSE;
```

and then running the correction using the assigned methods.

```
>  cor <- mbcb.correct(expressionSignal, negativeControl, nonparametric, RMA, MLE, MCMC, GMLE);
```

3 mbcb.main

A simpler way to generate output files using similar techniques is to use the *mbcb.main* method. With it, we call it in the same way as above but also specify the format for the output files. Unlike the *mbcb.correct* function which returns the data as objects within R, in this method, the data is written out to files specified as follows:

```
> mbcb.main(expressionSignal, negativeControl, nonparametric, RMA, MLE, MCMC, GMLE, "param-est", "bgCor
```

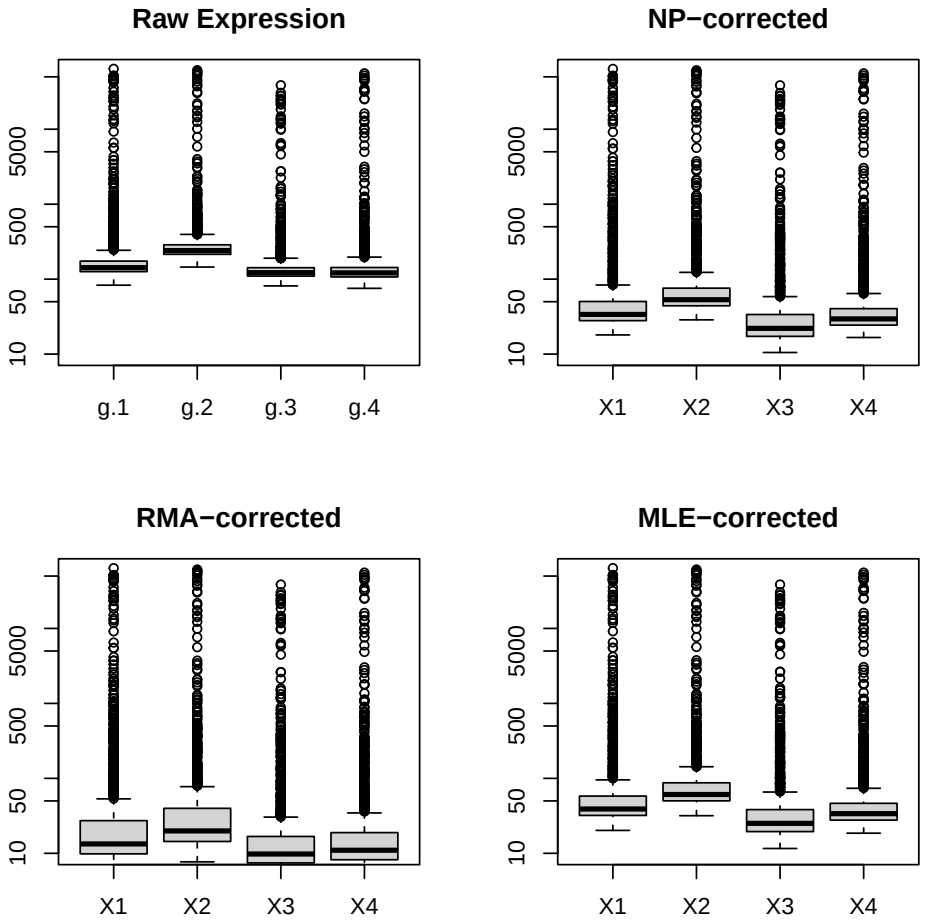
or, more simply,

```
> mbcb.main(expressionSignal, negativeControl, paramEstFile="param-est", bgCorrectedFile="bgCorrected"),
```

which will prefix all output files with the above strings.

Boxplots demonstrating the ouput of each background correction method can also be generated:

```
> ylimits <- c(10,60000);
> par(mfrow=c(2,2), mar=c(4,4,3,1))
> boxplot(expressionSignal, log="y", ylim=ylimits, main="Raw Expression")
> boxplot(cor$NP, log="y", ylim=ylimits, main="NP-corrected")
> boxplot(cor$RMA, log="y", ylim=ylimits, main="RMA-corrected")
> boxplot(cor$MLE, log="y", ylim=ylimits, main="MLE-corrected")
```



4 Normalization

We can also make use of normalization after we background correct the data using *mbcb.main*. Our options for normalization are:

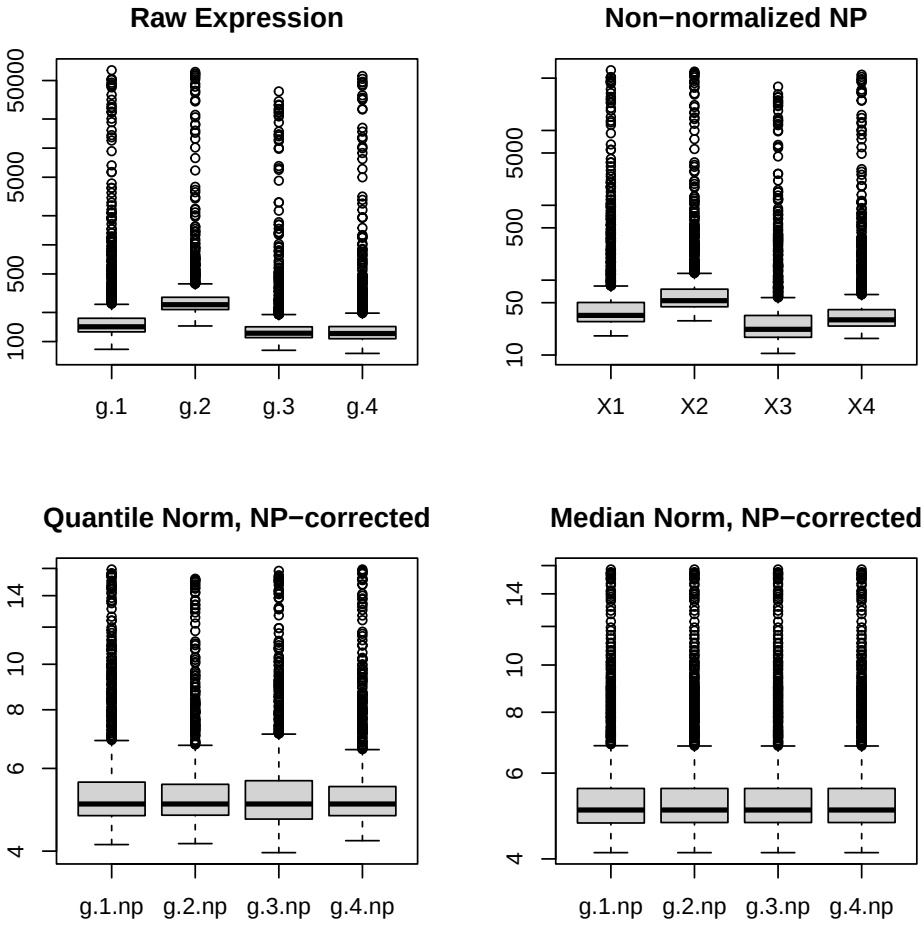
- "none" - no normalization will be applied
- "quant" - Quantile-Quantile normalization (requires the affy/affyio packages)
- "median" - Median or "global" normalization

We can use these as follows:

```
> mbcb.main(expressionSignal, negativeControl, normMethod="quant");
> quant_np <- read.csv("bgCorrected-NP.csv", row.names=1)
> mbcb.main(expressionSignal, negativeControl, normMethod="median");
> median_np <- read.csv("bgCorrected-NP.csv", row.names=1)
```

Boxplots showing the impact of normalization can be created as follows. Note that the Y-axes vary between each plot.

```
> par(mfrow=c(2,2), mar=c(4,4,3,1))
> boxplot(expressionSignal, log="y", main="Raw Expression")
> boxplot(cor$NP, log="y", main="Non-normalized NP")
> boxplot(median_np, log="y", main="Quantile Norm, NP-corrected")
> boxplot(quant_np, log="y", main="Median Norm, NP-corrected")
```



5 GUI

A Graphical User Interface (GUI) is provided which simplifies the workflow for using all of these methods.The following function opens the GUI:

```
> mbcb.gui();

<Tcl>
```

The GUI will walk you through the process and configuration of the package in a more user-friendly format. You'll begin by importing your files, then will select which method you want to use to perform the background correction, and (optionally) select a normalization method. You will then be asked where to save the output files.

Enjoy!

6 Session Information

The resulting sesion information is as follows:

```
R version 4.1.1 (2021-08-10)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 20.04.3 LTS

Matrix products: default
BLAS:   /home/biocbuild/bbs-3.14-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.14-bioc/R/lib/libRlapack.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_GB              LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] tcltk     stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] MBCB_1.48.0   tcltk2_1.2-11

loaded via a namespace (and not attached):
[1] compiler_4.1.1        tools_4.1.1           preprocessCore_1.56.0
```