# GS01 0163
# Analysis of Microarray Data

Keith Baggerly and Kevin Coombes
Section of Bioinformatics
Department of Biostatistics and Applied Mathematics
UT M. D. Anderson Cancer Center
kabagg@mdanderson.org
kcoombes@mdanderson.org

29 September 2005

# Lecture 8: Normalization, Affy, R, and Glass

- Revisiting Normalization in BioConductor
- R manipulations of AffyBatch
- Normalizing Project Normal

# A Bioconductor Adventure...

Our goal – to reproduce the study of Bolstad et al. (2003) using the data supplied with BioConductor.

First, pull in the Affy functions and get the data

```
> library(affy);
> library(affydata);
> data(Dilution);
> data(affybatch.example);
```

# What steps are we trying to follow?

Starting with an AffyBatch object, presumably assembled straight from CEL files, we want to test the effects of different normalization methods on the stability of probeset measurements of the same stuff.

The steps:

Background correction
Normalization
PM correction
Summary Quantification

Monitor as we go!

# Which data do we work with?

Eventually, we want to work with `Dilution`, as that's what they used, but there is a key argument for working with `affybatch.example` to begin with: the file is smaller. How much smaller?

```
> Dilution
AffyBatch object
size of arrays=640x640 features (12805 kb)
cdf=HG_U95Av2 (12625 affyids)
number of samples=4
number of genes=12625
annotation=hgu95av2
```

# Which data do we work with? (2)

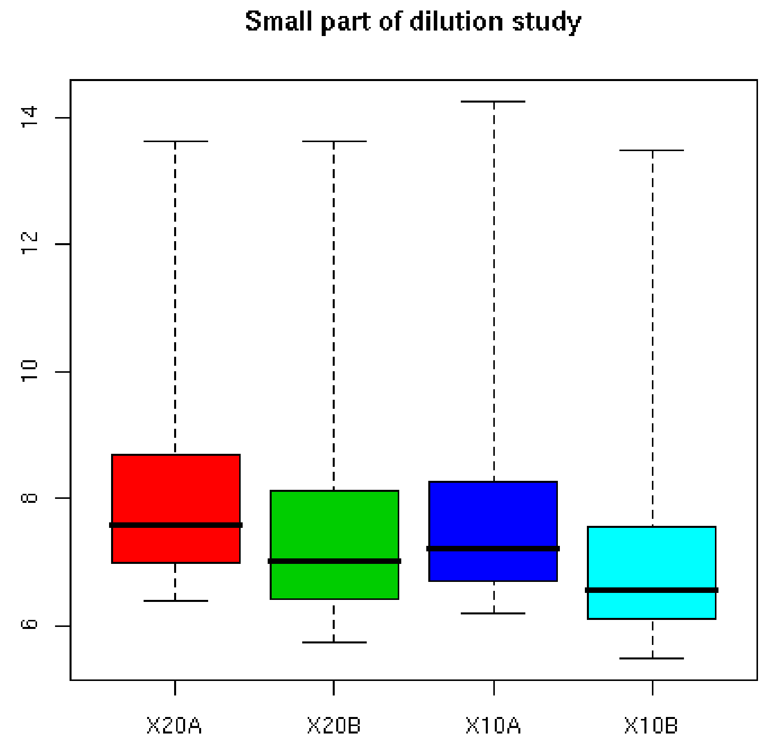
```
> affybatch.example
AffyBatch object
size of arrays=100x100 features (237 kb)
cdf=cdfenv.example (150 affyids)
number of samples=3
number of genes=150
annotation=
```

quick check: $237 * 6.4^2 * (4/3) = 12943$

We'll work with both from time to time.

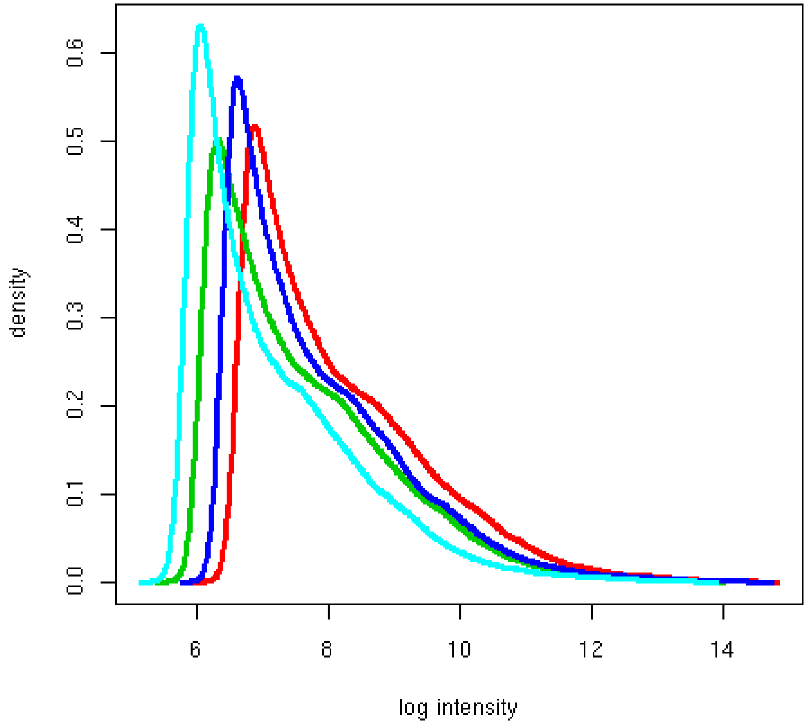
 

# Does this data need normalizing? (View 1)



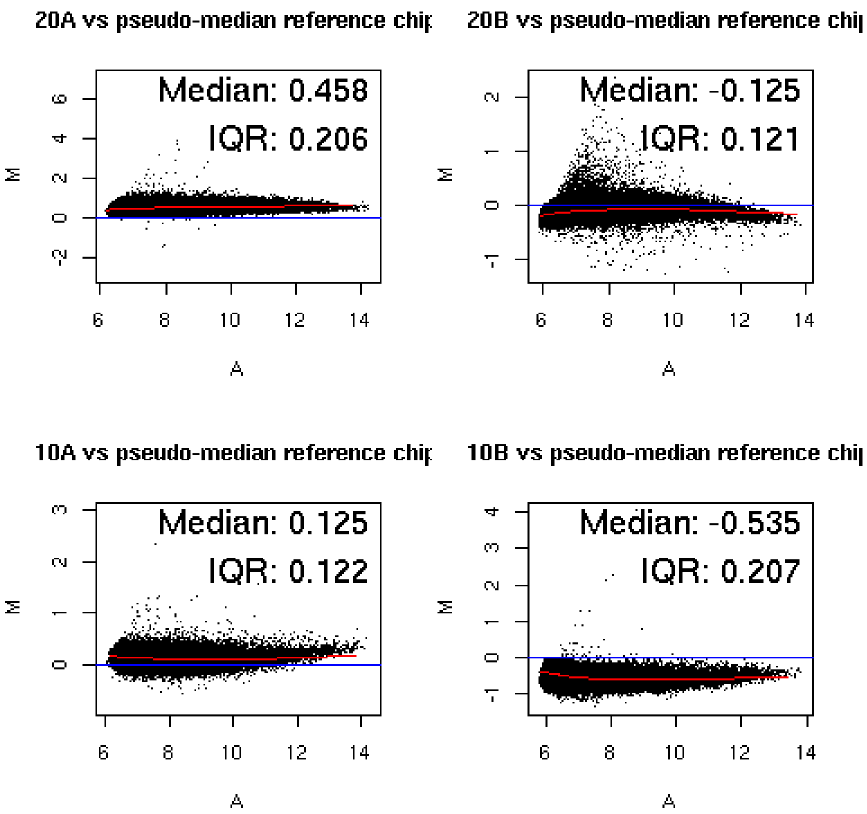

```
boxplot(Dilution); # shows log intensities!
dev.copy(png,file="boxplot1.png",col=2:5);
dev.off();
```

# What about the densities? (View 2)




```
hist(Dilution,lty=1,col=2:5,lwd=3);
dev.copy(png,file="hist1.png");
dev.off();
```

# and the MA plots?




```
par(mfrow=c(2,2));
MAplot(Dilution);
par(mfrow=c(1,1));
```

# Look at all pairs?

```
mva.pairs(Dilution);

Error in log(x, base) : Non-numeric
  argument to mathematical function
> help(mva.pairs)
```

want to feed this function a matrix, with columns corresponding to arrays. Where are these numbers?

# I can never remember...

Objects have slots!

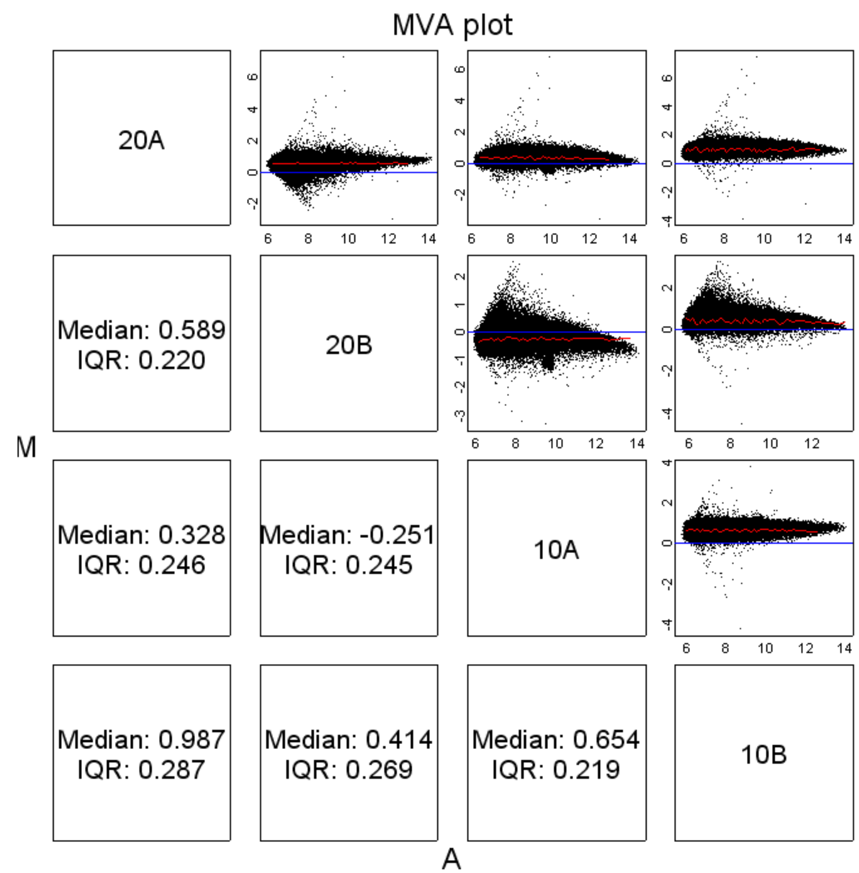
```
> slotNames(Dilution)
[1] "cdfName"     "nrow"          "ncol"
[4] "exprs"       "se.exprs"      "phenoData"
[7] "description" "annotation"    "notes"
```

I think we want `exprs`.

```
> length(exprs(Dilution))
[1] 1638400
> dim(exprs(Dilution))
[1] 409600 4
```

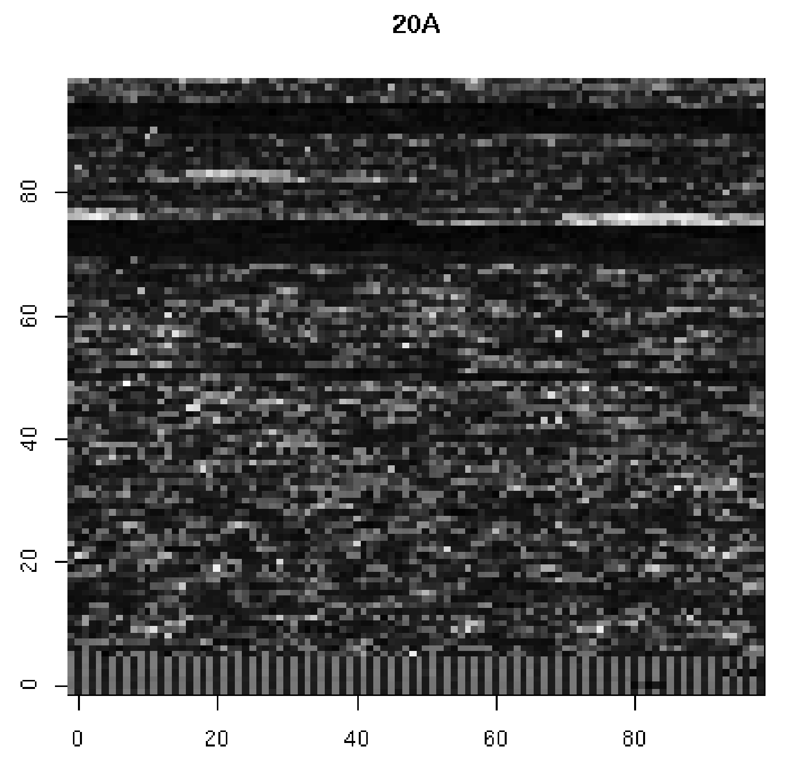
# Back to M vs A




```
mva.pairs(exprs(Dilution));
```

# Spatial Plots?




```
image(affybatch.example[,1],transfo=log2);
```

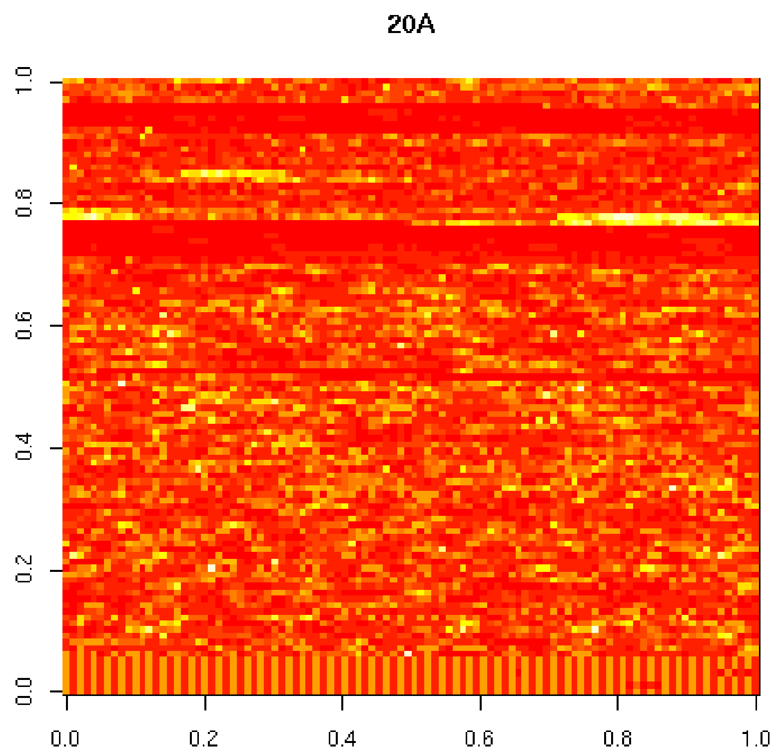
# Ratios of Spatial Plots?

```
image(matrix(exprs(affybatch.example[,1]),
             nrow=nrow(affybatch.example),
             ncol=ncol(affybatch.example)),
      transfo=log2);
```

parameter “transfo” can’t be set in high-level plot() function.

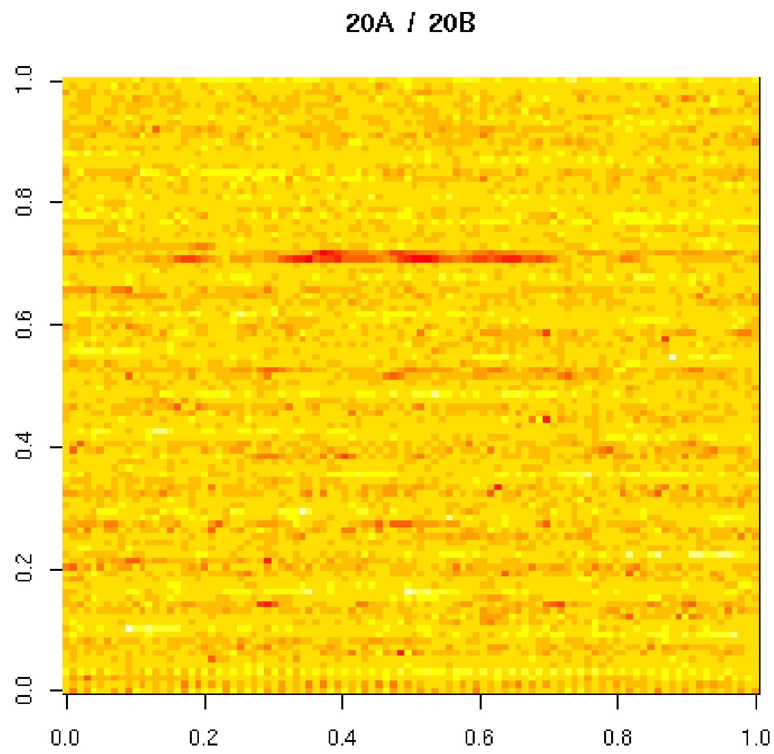
```
image(log2(matrix(
   exprs(affybatch.example[,1]),...
```

# Spatial Plot 1




```
image(log2(matrix(exprs(affybatch.example[,1]),..)),
      main=sampleNames(affybatch.example[,1]));
```

# Ratio Plot 1 (problem: fake geometry)




```
image(log2(matrix(exprs(affybatch.example[,1])/
                  exprs(affybatch.example[,2]),..)),
      main=paste(sampleNames(affybatch.example[,1]),
```

# Ok, start processing. BG first

```
Dilution.bg <- bg.correct.rma(Dilution);
```

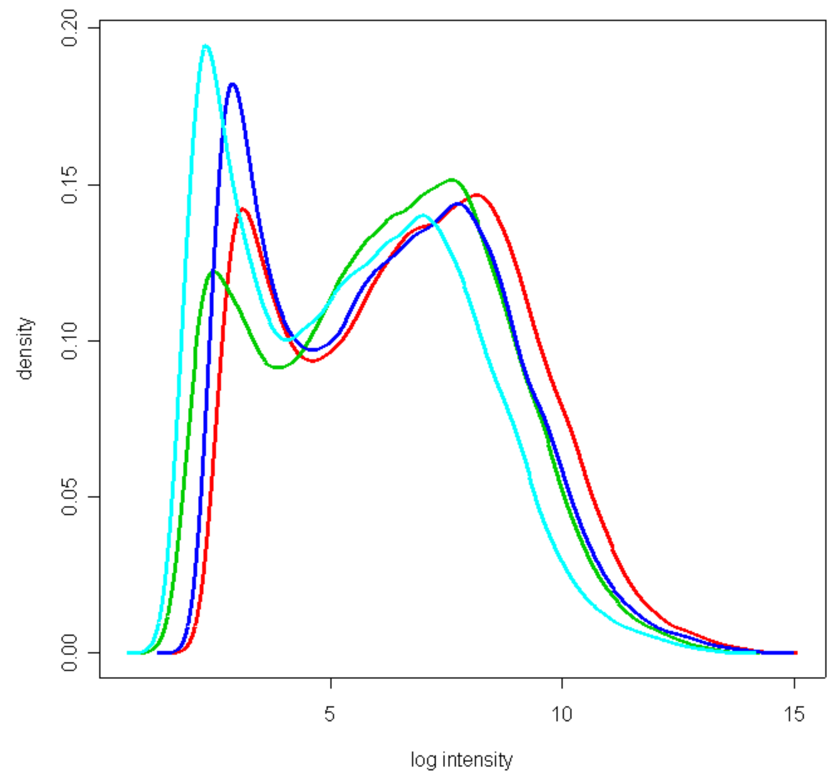
Did this change things?

```
hist(Dilution.bg, lty=1, col=2:5, lwd=3)
```

Let's also try it a different way to make sure...

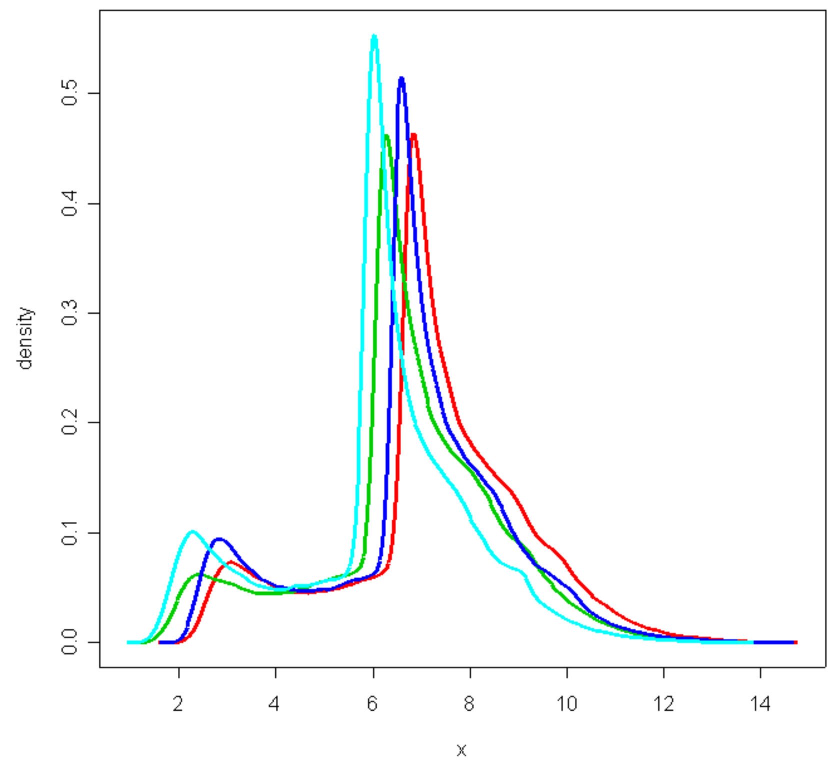
```
plotDensity(log2(exprs(Dilution.bg)),
            lty=1,col=2:5,lwd=3)
```

# Picture 1 After BG




```
hist(Dilution.bg, lty=1, col=2:5, lwd=3)
```

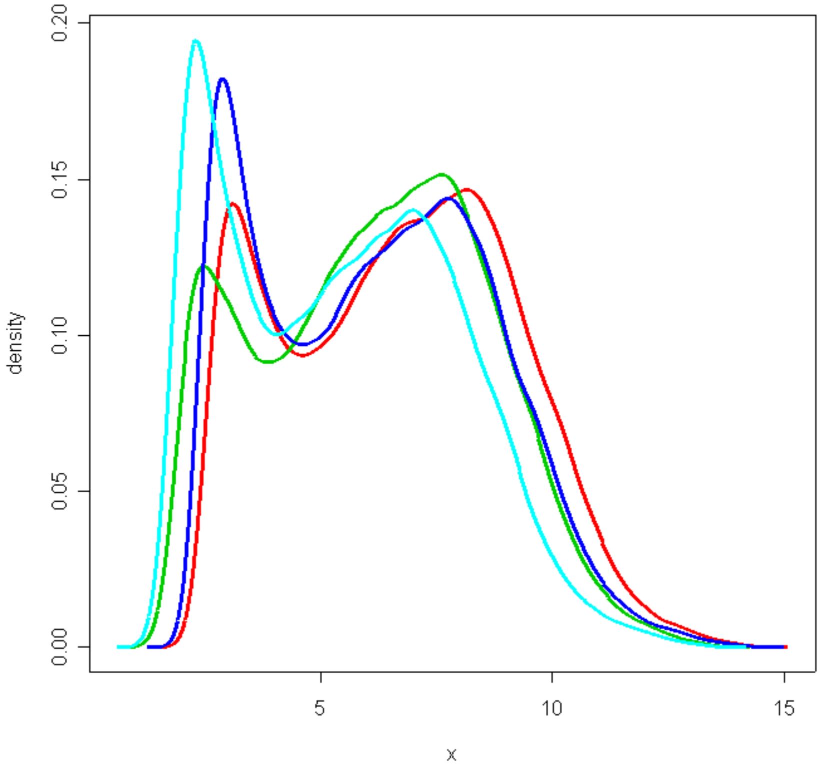
# Picture 2 After BG




```
plotDensity(log2(exprs(Dilution.bg)),
    lty=1,col=2:5,lwd=3)
```

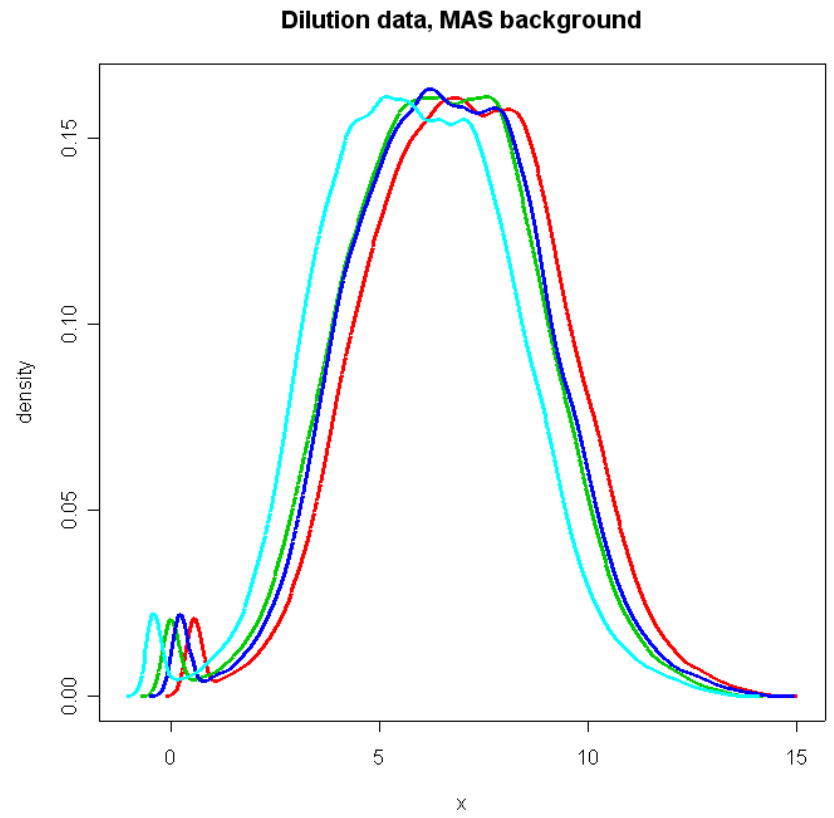
# Picture 2 (try 2) After BG




```
plotDensity(log2(pm(Dilution.bg)),
    lty=1,col=2:5,lwd=3)
```

# Is Background a Big Deal?




```
Dilution.bg <- bg.correct.mas(Dilution);
hist(Dilution.bg, lty=1, col=2:5, lwd=3);
title(main="Dilution data, MAS background");
```

# and now we normalize!

This is where the differences come in. We can invoke

normalize.AffyBatch.constant
normalize.AffyBatch.contrasts
normalize.AffyBatch.invariantset
normalize.AffyBatch.quantiles

or, of course, we can have `expresso`

# Expresso, no normalization

```
eset0 <- expresso(Dilution,
           bgcorrect.method="rma",
           normalize=FALSE,
           pmcorrect.method="pmonly",
           summary.method="medianpolish");
```

Now at this point, `eset0` is an exprSet object; while it still has slots for `exprs`, the dimensions have changed as we have shifted from features (probes) to probesets.

# What do we want?

The mean and variance of the probeset measurements gene by gene, to describe the behavior of this normalization method.

```
> dim(exprs(eset0))
[1] 12625 4
```

```
> eset0.mu <- apply(exprs(eset0),1,"mean");
> eset0.var <- apply(exprs(eset0),1,"var");
```

Now we want another method to compare to.

Actually, in order to explore things, I found it useful to tweak the parameters with a smaller sample just to be sure that things were working as desired. So, redo the above processing using `affybatch.example` instead of `Dilution`.

# Constant normalization: choosing baseline

find the “middle behavior” chip

```
> apply(exprs(affybatch.example),2,"median");
  20A     20B    10A
147.3   118.0  125.0

eset1 <- expresso(affybatch.example,
    bgcorrect.method = "rma",
    normalize.method = "constant",
    normalize.param  = list(refindex=3),
    pmcorrect.method = "pmonly",
    summary.method   = "medianpolish");

> eset1.mu <- apply(exprs(eset1),1,"mean");
> eset1.var <- apply(exprs(eset1),1,"var");
```

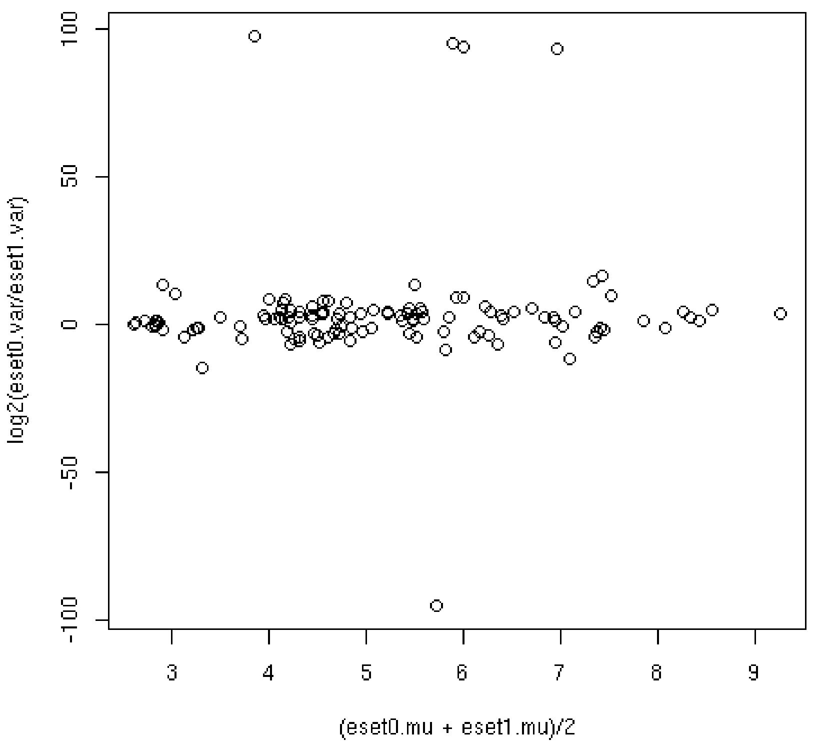
# So, how do we compute MA plots here?

Normally, we are plotting the results from one chip against that from another. Here, we are working with two sets of results from the same chips, just using different methods for quantification.

```
A1 <- (eset0.mu + eset1.mu)/2;
M1 <- (eset0.mu - eset1.mu)/2; # not quite.
M2 <- (eset0.var / eset1.var); # still not quite.
M3 <- log2(eset0.var / eset1.var);
```

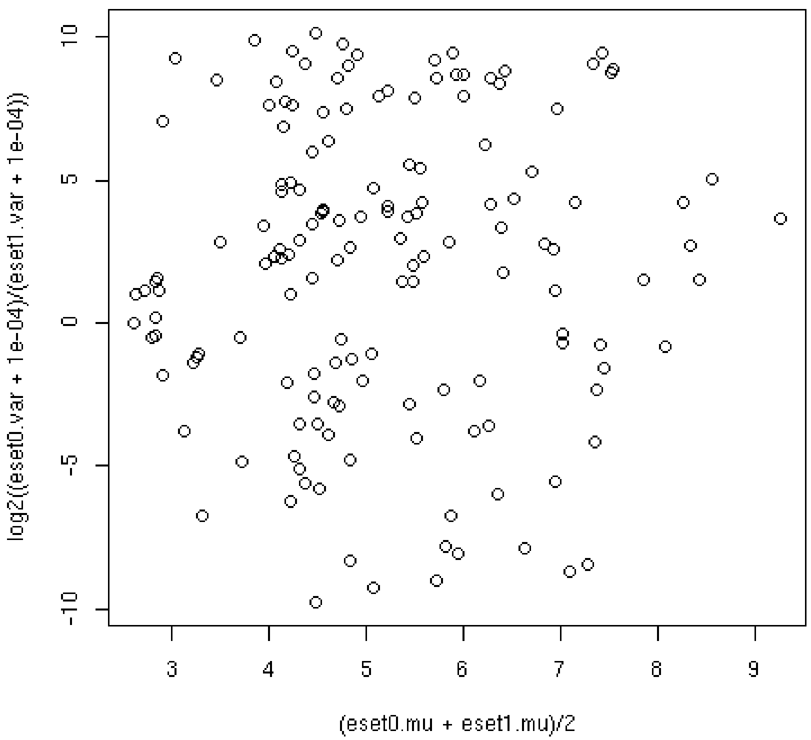
# Checking “none” against “scaling”




This initial plot is driven by outliers. Tweak.

```
d0 <- 0.0001;
M4 <- log2((eset0.var + d0)/(eset1.var + d0));
```

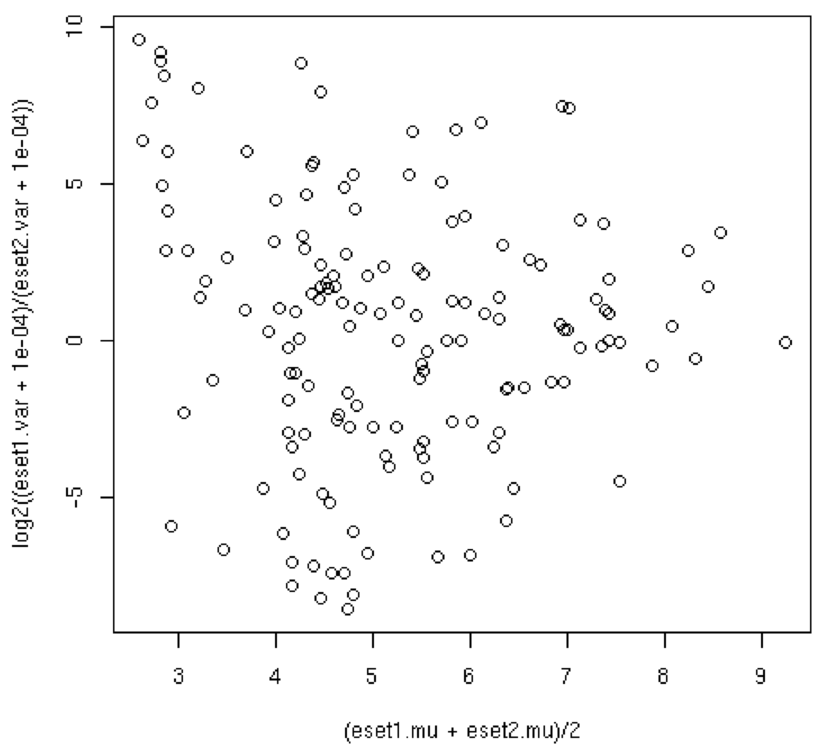
# Checking "none" against "scaling"




```
> sum(eset1.var < eset0.var)
```

Not that stark – 96 times out of 150, constant scaling gives lower variability. This is a small (fake) array.

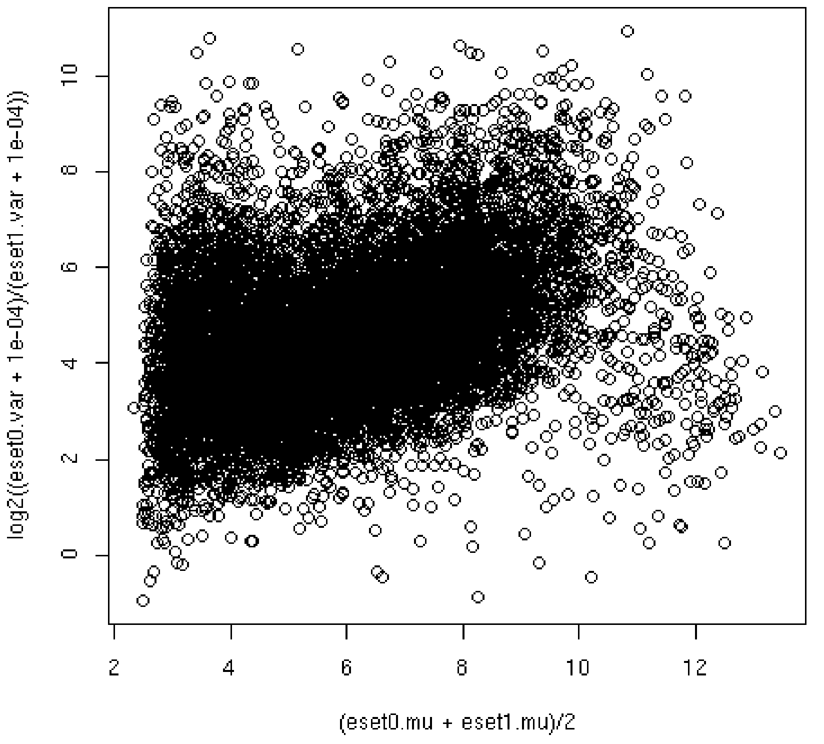
# Checking “scaling” against “quantiles”




Not that stark – 83 times out of 150, quantile scaling gives lower variability.

repeat with Dilution, now that we know what we want to do.

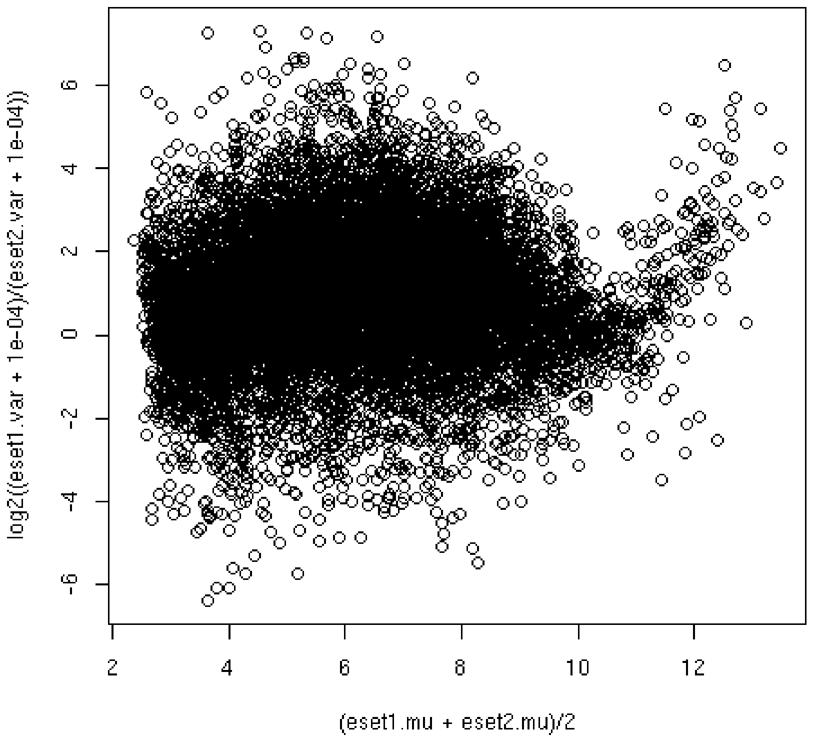
# Dilution: "none" against "scaling"




Here, 12615 times out of 12625, constant scaling gives lower variability. Mean log diff: 4.62

# Dilution: “scaling” against “quantiles”




Here, 9477 times out of 12625, quantile scaling gives lower variability. Mean log diff: 0.95

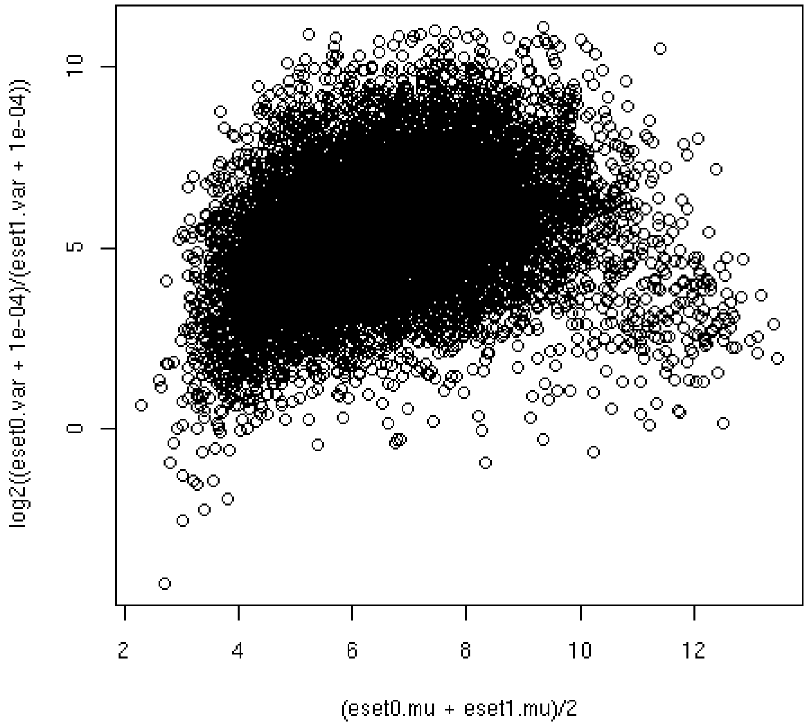
 

# What didn't they do?

Our comparison of normalization methods here focused on reducing variability, and it assumed that a particular type of background correction (rma) and summarization (median polish) had been employed.

But we saw that different background correction methods led to different shapes in the distributions of probe intensities. If we use "mas" as the background subtraction method, are the differences between the normalization methods still as stark?

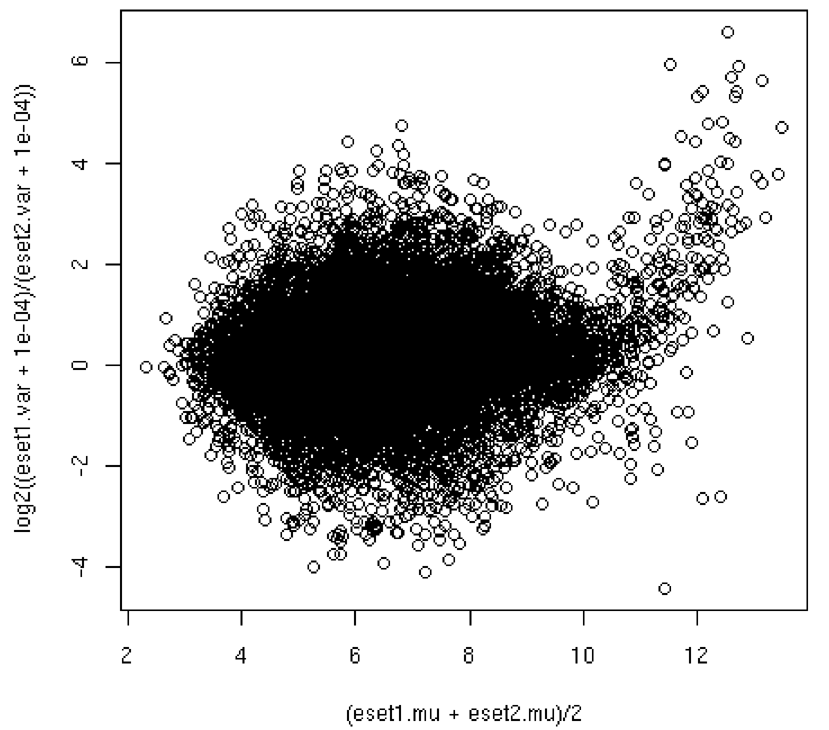
# Dilution: “none” against “scaling”, MAS BG




Here, 12600 times out of 12625, constant scaling gives lower variability. Mean log diff: 5.40

# Dilution: “scaling” against “quantiles”, MAS BG




Here, 7937 times out of 12625, quantile scaling gives lower variability. Mean log diff: 0.265

# Normalizing on Glass?

Main difference is two-color setup

Some general recommendations:

Normalize channels to each other first, then normalize log ratios across chips.

do dye swaps

MA plots, loess fits, and pictures

# Project Normal: A Cautionary Tale

Pritchard, Hsu, Delrow and Nelson
*Project Normal: Defining Normal Variance in Mouse Gene Expression*
PNAS **98** (2001), 13266-13271.

Data set used for the third annual Critical Analysis of Microarray Data (CAMDA 2002)

# Their Initial Goals

The goal of many microarray studies is to identify genes that are "differentially expressed".

Relative to what?

Differences larger in scale than those that would be encountered due to "normal" or technical variation.

Try to assess the fraction of genes exhibiting a large mouse-to-mouse heterogeneity in the absence of structure.

# Their Experimental Design

Eighteen Samples

- Six C57BL6 male mice
- Three organs: kidney, liver, testis

Reference Material

- Pool all eighteen mouse organs

Replicate microarray experiments using two-color fluorescence with common reference and dye swaps

- Four experiments per mouse organ, 2 each dye

# Their Analysis

Print-tip specific intensity dependent loess normalization

Perform F-tests on log(Exp/Ref) for each gene to see if mouse-to-mouse variance exceeds the array-to-array variance

# The Data Supplied

Images

One quantification file each for kidney, liver and testis.

```
CDNA ID, Cluster ID, Title,
Block, Column, Row

F635 Median M1K3_1, B635 Median M1K3_1
F532 Median M1K3_1, B532 Median M1K3_1
```

Mouse 1, Kidney Sample in Cy3 channel, first replicate.

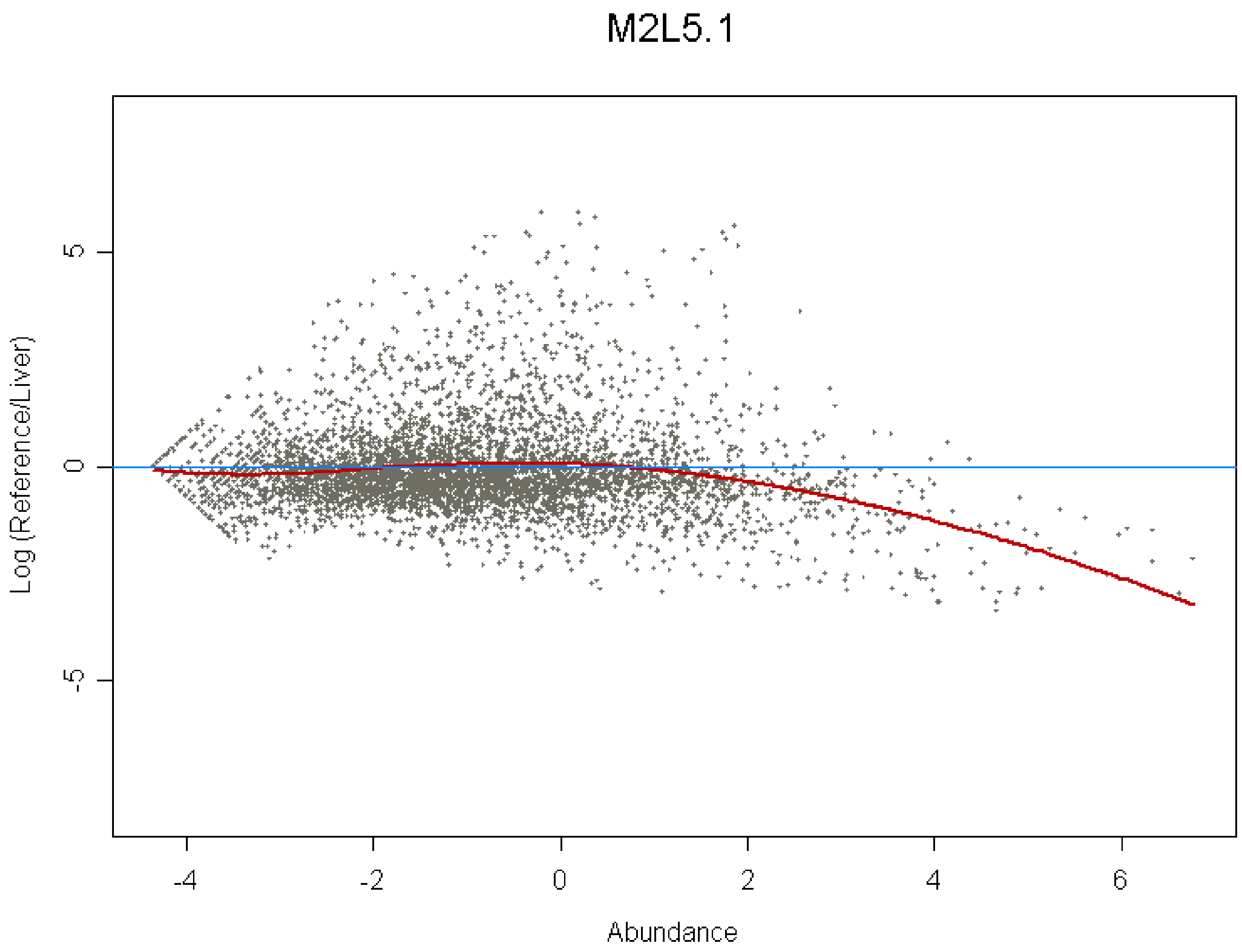
# Why We Got Involved

All in all, the analysis described looks pretty good. F-tests on log ratios seem reasonable, and the preprocessing steps they used are fairly standard. Furthermore, the images looked fairly clean.

"Fairly standard" $\neq$ correct

*For this data, we think that loess normalization is incorrect.*

# What Loess Looks Like for 1 Array



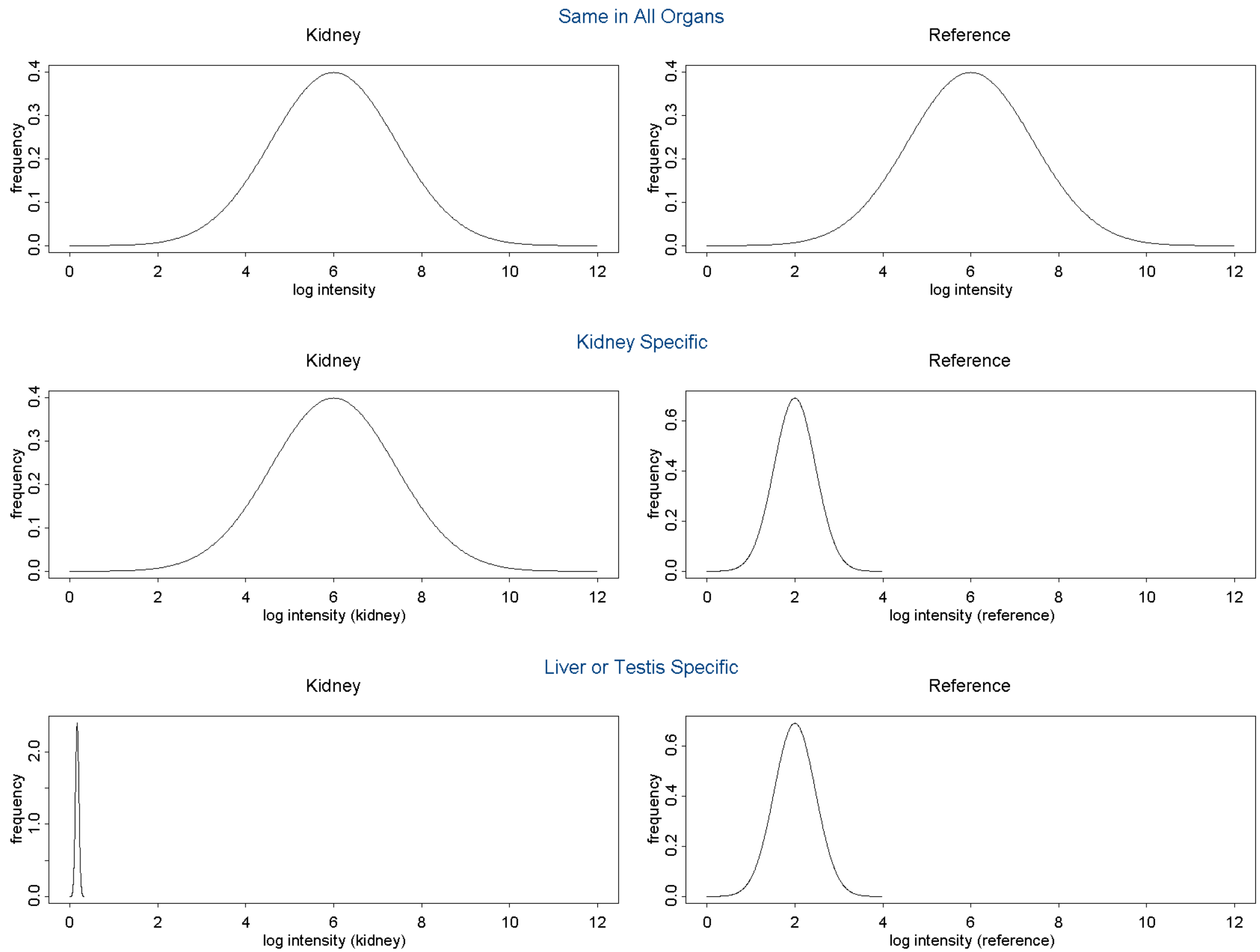
# Why Loess Normalization?

Most normalization methods assume:

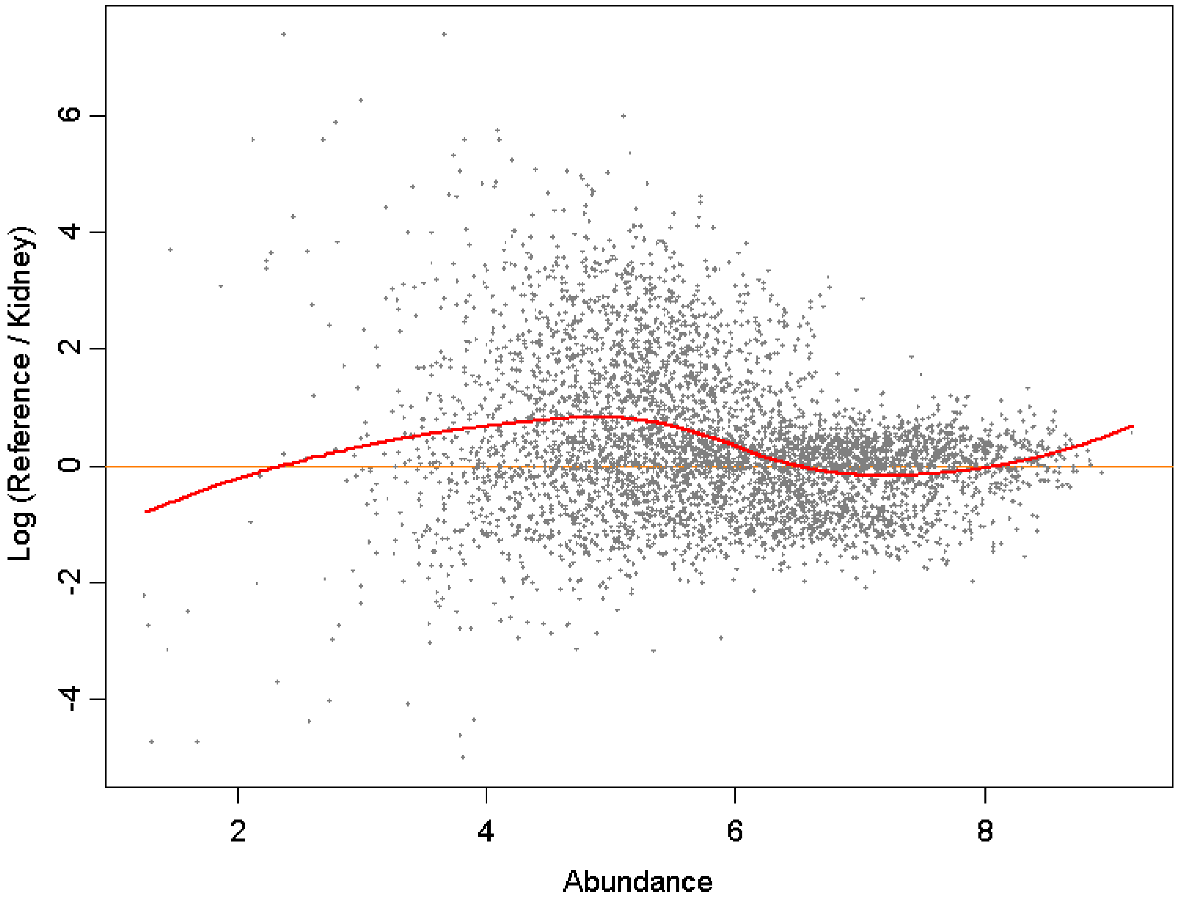
- Distributions of intensities are the same in the two channels
- Most genes do not change expression
- The number of overexpressed genes is about the same as the number of underexpressed genes

Loess normalization tries to force the distributions in the two channels to match, believing that differences are attributable to technology.

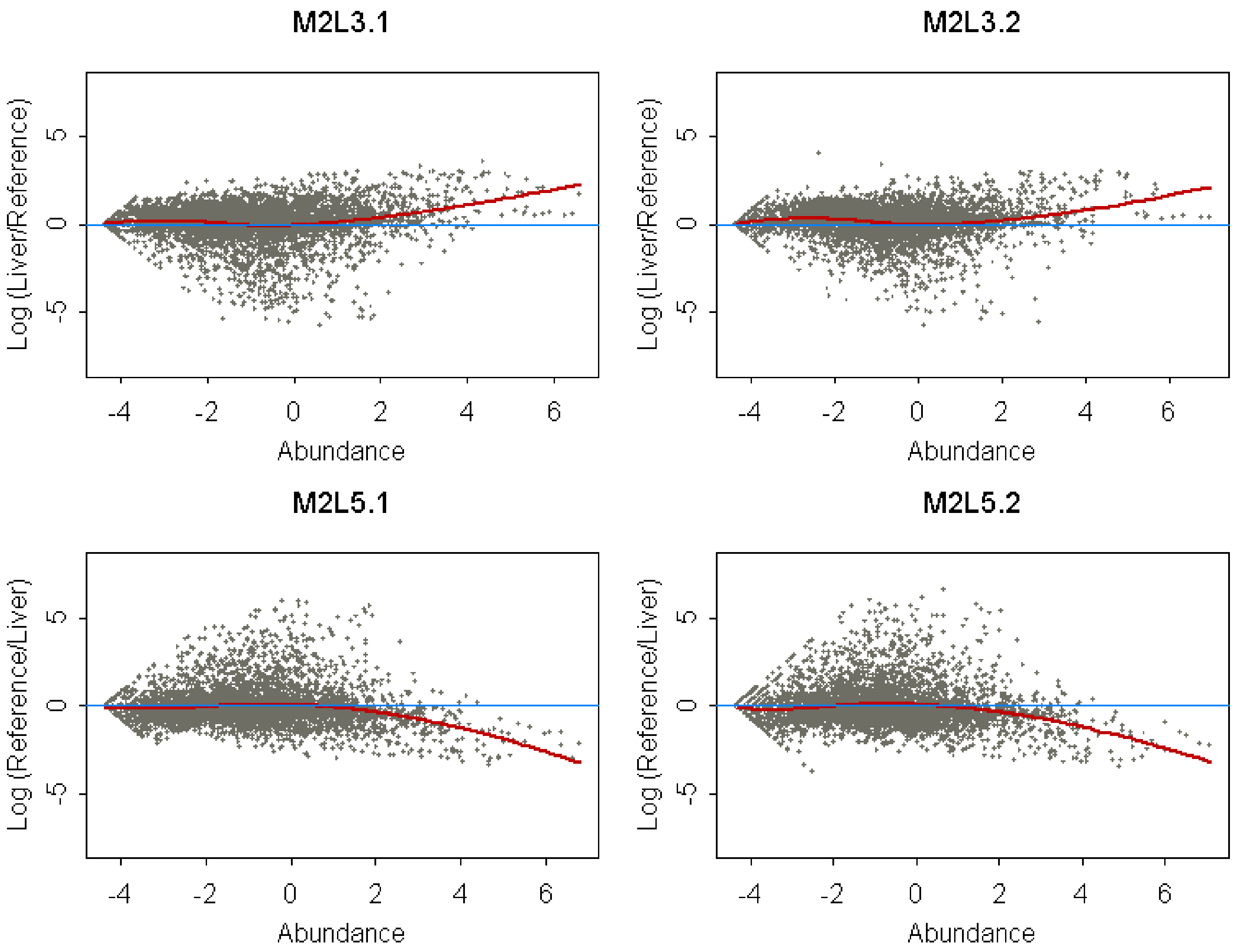
# Why We Think It's Wrong



# Simulated Data Using Our Approach



# Are We Right? Checking the Dye Swaps



# Interpretation

- Distributions of intensities are different in the two channels
- Difference is NOT caused by arrays, dyes, or technology
- Difference is inherent in the choice of reference material

# So, How Do We Normalize This Data?

*Normalize channels separately*

Divide by $75^{th}$ percentile (magic)

Multiply by 10 (arbitrary, equalizes scale)

Set threshold at 0.5 (more magic)

Log transform