# 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

10 November 2005

# Lecture 20: A Two-Color Case Study

- Using limma
- Fitting models and contrasts
- Making tables
- Normalization and Glossing
- Digressions on Lysate Arrays

# Where We Left Off...

```
> library("marray");
> library("mclust");
> library("convert");
> library("arrayQuality");
> library("colorspace");
> library("grid");
> library("hexbin");
> TargetInfo <- read.marrayInfo("TargetBeta7.txt")
> mraw <- read.GenePix(targets = TargetInfo);
> normdata <- maNorm(mraw);
```

# Let's try Fitting Data

```
> library("limma");
> LMres <- lmFit(normdata,design =
    c(1,-1,-1,1,1,-1), weights = NULL);
```

Ok, what did we just do?

lmFit is the main workhorse function of the limma package, and it fits LInear Models to MicroArrays. But what is a linear model?

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \epsilon$$

# Let's try Fitting Data

What numbers are being played with here?

```
> slotNames(LMres)
[1] ".Data"
```

Surprise! While this is an object of type "MArrayLM", its contents are contained in a simple data frame. So, what things do we have here?

# Fitted Numbers, Part 1

```
> LMres[1,]
An object of class "MArrayLM"
$coefficients
          [,1]
[1,] -1.954552
$stdev.unscaled
          [,1]
[1,] 0.5773503
$sigma
[1] 2.89688
$df.residual
[1] 2
```

# Fitted Numbers, Part 2

```
$cov.coefficients
          [,1]
[1,] 0.1666667
$pivot
[1] 1
$method
[1] "ls"
$design
  1 -1 -1  1  1 -1
$genes
                    ID Name
H200000297 H200000297 OVGP1 - Oviductal glycoprotein
$Amean
[1] 5.84333
```

# Behind the Curtain, 1

```
> normdata@maM[1,]
6Hs.195.1.gpr   6Hs.168.gpr   6Hs.166.gpr
   -0.4520881     0.1175666            NA
6Hs.187.1.gpr   6Hs.194.gpr   6Hs.243.1.gpr
           NA            NA     5.2940001
```

These are the log ratios that we have available for this gene.

```
> mean(normdata@maM[1,],na.rm=T)
[1] 1.653160  # not quite...
> mean(normdata@maM[1,]*LMres[1,]$design,na.rm=T)
[1] -1.954552  # this is the coeff value!
```

So, why are there NAs in the maM field? Why do we need to multiply by the design vector? What do the 1's and -1s indicate?

# Behind the Curtain, 2

```
> sqrt(var(normdata@maM[1,]*LMres[1,]$design,na.rm=T)
         [,1]
[1,] 2.89688 # this is "sigma"
> LMres[1,]$stdev.unscaled^2
           [,1]
[1,] 0.3333333 # 1/number of valid reads
```

The lmFit call is summarizing the individual M values according in order to highlight a specified contrast. By changing the design matrix, different contrasts can be seen, and tested for.

# Testing Significance

```
> LMresEB <- eBayes(LMres);
```

So, what do we get? (long list)

```
> slotNames(LMresEB)
[1] ".Data"
> summary(LMresEB)
                  Length Class      Mode
coefficients      23184  -none-     numeric
stdev.unscaled    23184  -none-     numeric
sigma             23184  -none-     numeric
df.residual       23184  -none-     numeric
cov.coefficients      1  -none-     numeric
pivot                 1  -none-     numeric
method                1  -none-     character
```

```
design          6  -none-     numeric
genes           2  data.frame list
Amean       23184  -none-     numeric
df.prior        1  -none-     numeric
s2.prior        1  -none-     numeric
var.prior       1  -none-     numeric
proportion      1  -none-     numeric
s2.post     23184  -none-     numeric
t           23184  -none-     numeric
p.value     23184  -none-     numeric
lods        23184  -none-     numeric
F           23184  -none-     numeric
F.p.value   23184  -none-     numeric
```

# Reporting Significance

Ok, at this point we have some test statistic values and associated p-values. The test-stat values were computed by borrowing strength across the genes available on the array to get more stable estimates of “null variation”, so we have “moderated” t-tests as opposed to the plain vanilla variety.

Still, given these, we would like to extract a small number of them and report them in a fairly illustrative fashion.

```
> shortTable <- topTable(LMresEB, number = 10,
    resort.by = "M");
```

# So, What Do We Get?

```
> shortTable[1:2,]
             ID  Name
3152 H200012024  ITGA1 - Integrin, alpha 1
1755 H200014446  P2Y5 - Purinergic receptor (family A
     M           A           t
3152  1.2673703   6.979029   8.390349
1755 -0.9152066  12.226345  -6.147830
     P.Value   B
3152 1          0.5947600
1755 1         -0.3437579
```

Hey, integrin made it to the list!

Most of this I recognize, but why are the p-values 1? And what's B? Look at the full table...

# The Full Table...

## BioConductor Gene Listing

| ID | Name | M | A | t | P.Value | B |
|---|---|---|---|---|---|---|
| H200012024 | ITGA1 - Integrin, alpha 1 | 1.27 | 6.98 | 8.39 | 1 | 0.59 |
| H200014446 | P2Y5 - Purinergic receptor (family A group 5) | -0.92 | 12.23 | -6.15 | 1 | -0.34 |
| H200001079 | EGFL5 - EGF-like-domain, multiple 5 | -0.96 | 7.74 | -6.27 | 1 | -0.28 |
| H200001929 | EPLIN - Epithelial protein lost in neoplasm beta | -1.1 | 8.62 | -6.29 | 1 | -0.26 |
| H200003977 | F5 - Coagulation factor V (proaccelerin, labile factor) | -1.1 | 7.5 | -6.55 | 1 | -0.14 |
| H200007427 | CENTG2 - Centaurin, gamma 2 | -1.19 | 6.09 | -8.29 | 1 | 0.56 |
| H200004937 | Homo sapiens cDNA FLJ12815 fis, clone NT2RP2002546 | -1.24 | 6.3 | -6.85 | 1 | 0.01 |
| H200003784 | SEMA5A - Sema domain, seven thrombospondin repeats (type 1 and type 1-like), transmembrane domain (TM) and sh | -1.35 | 6.81 | -6.89 | 1 | 0.03 |
| H200018884 | Homo sapiens cDNA FLJ11375 fis, clone HEMBA1000411, weakly similar to ANKYRIN | -1.6 | 6.62 | -8.76 | 0.77 | 0.71 |
| H200017286 | GPR2 - G protein-coupled receptor 2 | -2.45 | 7.79 | -10.77 | 0.18 | 1.17 |

```
> table2html(shortTable, disp = "file");
```

Note the live links!

# What's Going On?

The p-values are so big because they adjust for multiple testing (how?). They hit 1 because with this data set and the number of samples involved, the differences aren't clear enough for us to see and use.

The value B is the "log odds" that the gene is differentially expressed – if the value of B is 0.59, then the odds that the gene is differentially expressed are $\exp(0.59) = 1.803$ to 1, and the probability of differential expression is $1.803/(1.803 + 1) = 0.643$. Not great.

# Is this the Right Contrast?

Not quite. Leading question – why were there at least two arrays run for each patient?

Dye Swaps.

There may be a dye effect present, and it may be possible to account for this. This requires adjusting the design matrix.

```
> design <- cbind(Dye = 1, c(1,-1,-1,1,1,-1));
> LMres2 <- lmFit(normdata, design, weights = NULL);
> LMres2EB <- eBayes(LMres2);
> shortTable2 <- topTable(LMres2EB,adjust="fdr",
    number=10,resort.by="M")
```

# Accounting for Dye...

## BioConductor Gene Listing

| ID | Name | M | A | t | P.Value | B |
|---|---|---|---|---|---|---|
| H200006376 | ARHA - Ras homolog gene family, member A | 1.88 | 13.68 | 11.63 | 0.03 | 3.25 |
| H200013376 | SLC8A2 - Solute carrier family 8 (sodium-calcium exchanger), member 2 | 1.74 | 4.5 | 11.67 | 0.03 | 3.27 |
| H200006035 | TYRP1 - Tyrosinase-related protein 1 | 1.66 | 12.16 | 13.01 | 0.03 | 3.73 |
| H200007318 | HAP1 - Huntingtin-associated protein 1 (neuroan 1) | 1.62 | 11.54 | 11.67 | 0.03 | 3.27 |
| H200005892 | HLA-DQB1 - Major histocompatibility complex, class II, DQ beta 1 | 1.57 | 12 | 13.24 | 0.03 | 3.81 |
| H200006019 | PPP5C - Protein phosphatase 5, catalytic subunit | 1.49 | 10.13 | 11.66 | 0.03 | 3.27 |
| H200006025 | DAP - Death-associated protein | 1.44 | 12.64 | 12.48 | 0.03 | 3.56 |
| H200011083 | KIAA0692 - KIAA0692 protein | 1.36 | 10.88 | 12.14 | 0.03 | 3.44 |
| H200005780 | C19orf7 - Chromosome 19 open reading frame 7 | 1.35 | 13.1 | 12.4 | 0.03 | 3.53 |
| H200013564 | RBMX - RNA binding motif protein, X chromosome | 1.29 | 12.52 | 11.79 | 0.03 | 3.32 |

```
> table2html(shortTable2, disp = "file");
```

The table changes a lot...

# Is it what they used in the paper?

Not quite. There, they also decided to not subtract background, with the result that they didn't have to deal with those pesky NA values.

They also dealt with several replicates per person. This can be accomodated in lmFit by defining a more extensive model matrix.

The general lesson here is that the answers that we get change rather drastically as we change the nature of the question being asked.

This is addressed in considerable detail in Chapter 23 of Gentleman et al on "limma" by Gordon Smyth. We'll revisit this in later lectures.

# More Comments on Replication

Why do we need to treat replicates differently than other samples?

They're not measuring "independent" quantities. If we measure 10 replicates from a sick person and 10 replicates from a healthy person, then contrasting these 20 arrays, we're still contrasting just one person with another. Replications in the form of dye-swaps, however, is still useful in that it allows us to preclude certain biases from affecting our results.

# An Example

(adapted from Smyth's chapter)

Let's say that we have two patients that we want to compare with two controls. How many possible pairwise combinations are there?

Here, there are 8: 2 controls * 2 patients * 2 dye orderings.

Now, what we really want to say something about is the difference between disease states: avg disease - avg control.

This in turn is given by

$$(D1 + D2)/2 - (C1 + C2)/2$$

But how do we get these?

# Levels of Contrasts

These are average levels for each individual, so that the results for one individual do not dominate the results by simply being present in more of the samples.

These average levels can be estimated, using an appropriately defined design matrix.

Let’s lay things out more precisely.

# What Goes Where

```
> fakeSamples <- read.table("fakeSamples.txt",header=
> fakeSamples
  FileName Cy3 Cy5
1 F1.gpr   D1  C1
2 F2.gpr   D1  C2
3 F3.gpr   D2  C1
4 F4.gpr   D2  C2
5 F5.gpr   C1  D1
6 F6.gpr   C2  D1
7 F7.gpr   C1  D2
8 F8.gpr   C2  D2
```

Coming up with the design is fairly easy, for one relative to all of the others

# The Model Matrix

```
> design <- modelMatrix(fakeSamples, ref="D1")
> design
  C1 C2 D2
1  1  0  0
2  0  1  0
3  1  0 -1
4  0  1 -1
5 -1  0  0
6  0 -1  0
7 -1  0  1
8  0 -1  1
```

+1 if the sample is in Cy5, -1 if it is in Cy3.

# Fit the Model and Contrast

```
> design <- cbind(Dye = 1, design);
> fakeFit <- lmFit(fakeMA, design);
```

Given things that are directly estimable, define the contrast of interest in terms of the values found, and fit the contrast.

```
> contrast.matrix <- makeContrasts(
    DvsC = (D2/2) - ((C1+C2)/2),
    levels = design)
> fakeFitCont <- contrasts.fit(fakeFit,
    contrast.matrix)
> fakeFitContEB <- eBayes(fakeFitCont)
```

# More on Using Replicates

We've talked about looking at replicate runs of samples, but we haven't looked explicitly at the use of replicate spots of the same gene on a given slide.

While these can be summarized (eg, use the mean value if a single log ratio is needed for a gene), replicate spots are useful for far more things. Checking the correlations of pairs of replicates can suggest whether the quantification was good, and finding a way to look at replication in spatial context can point out other problems.

# Spatial Replicates

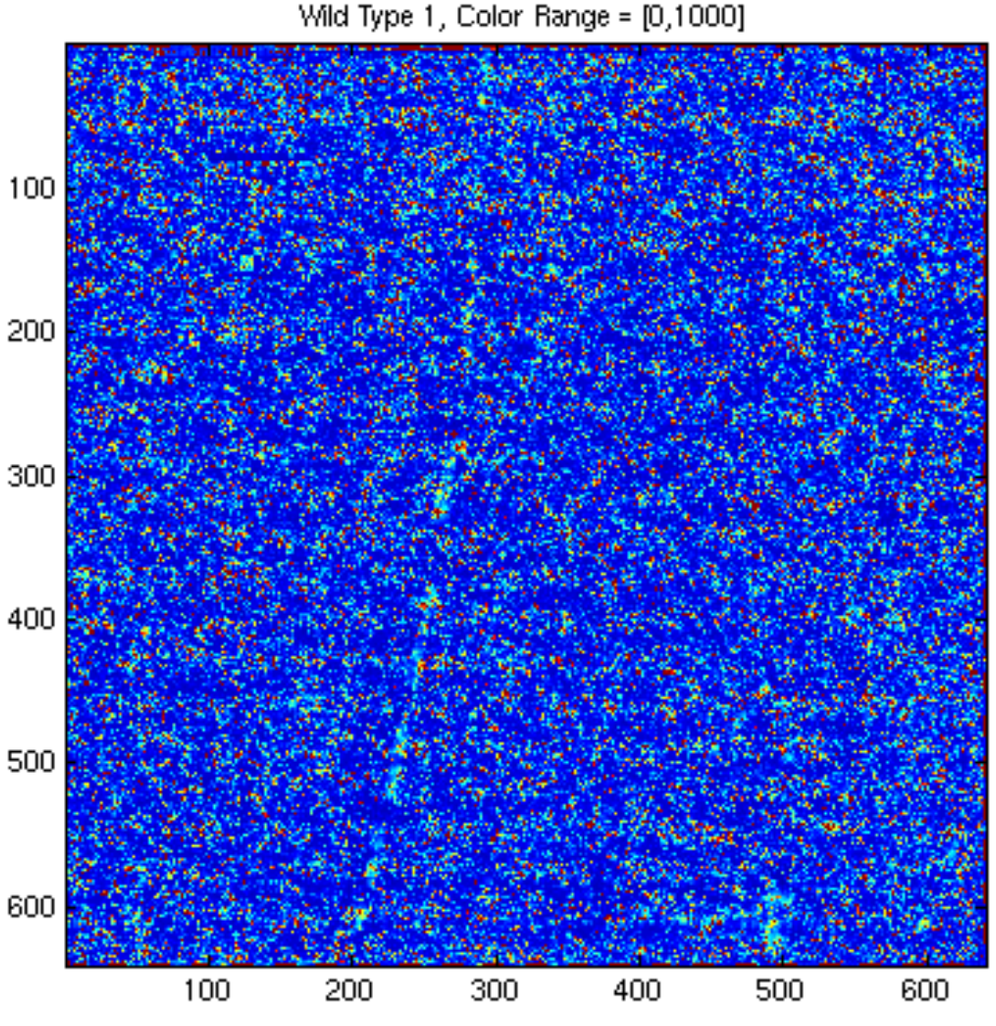
Step 1: form a spatial plot of the log ratios for array 1.

Step 2: repeat step 1 for a second array

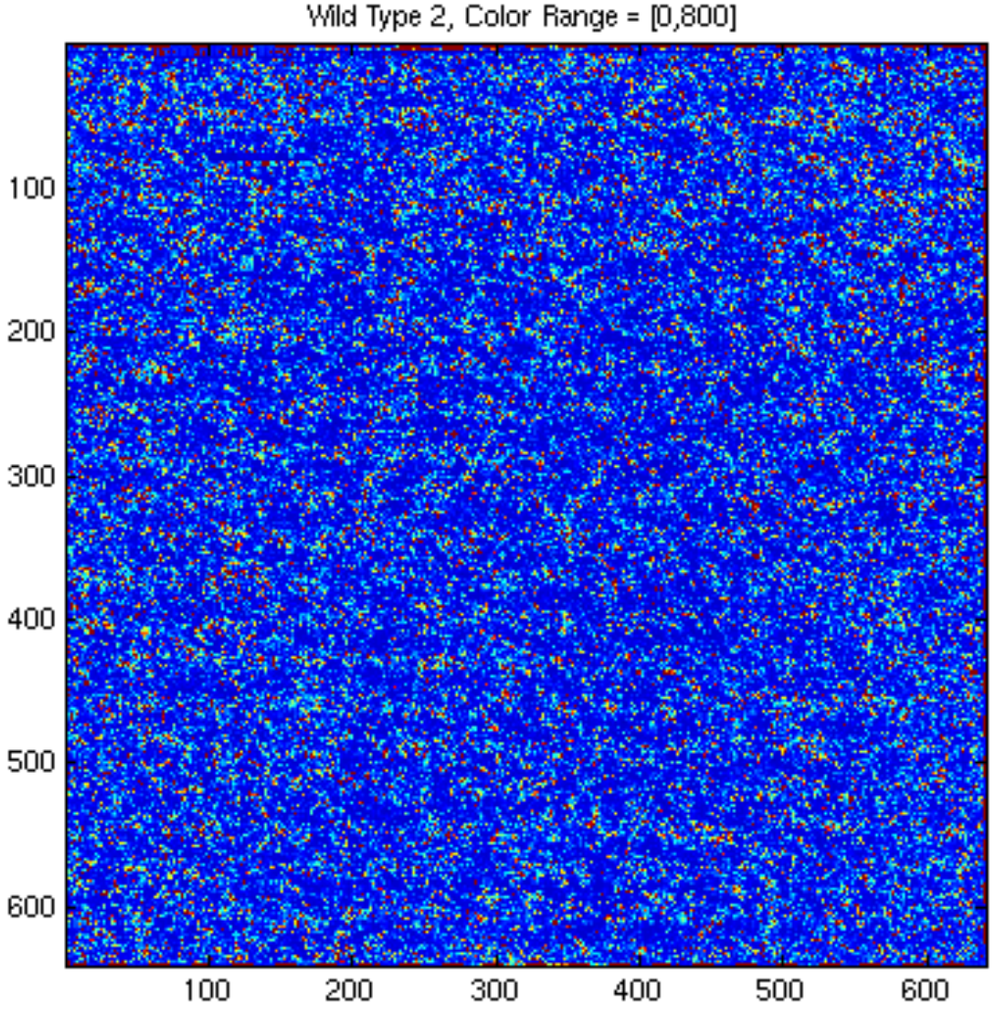
Step 3: form an image of the differences, point by point. If the printing is dense, then we will see big trends if these remain.

Try this out with Affy first

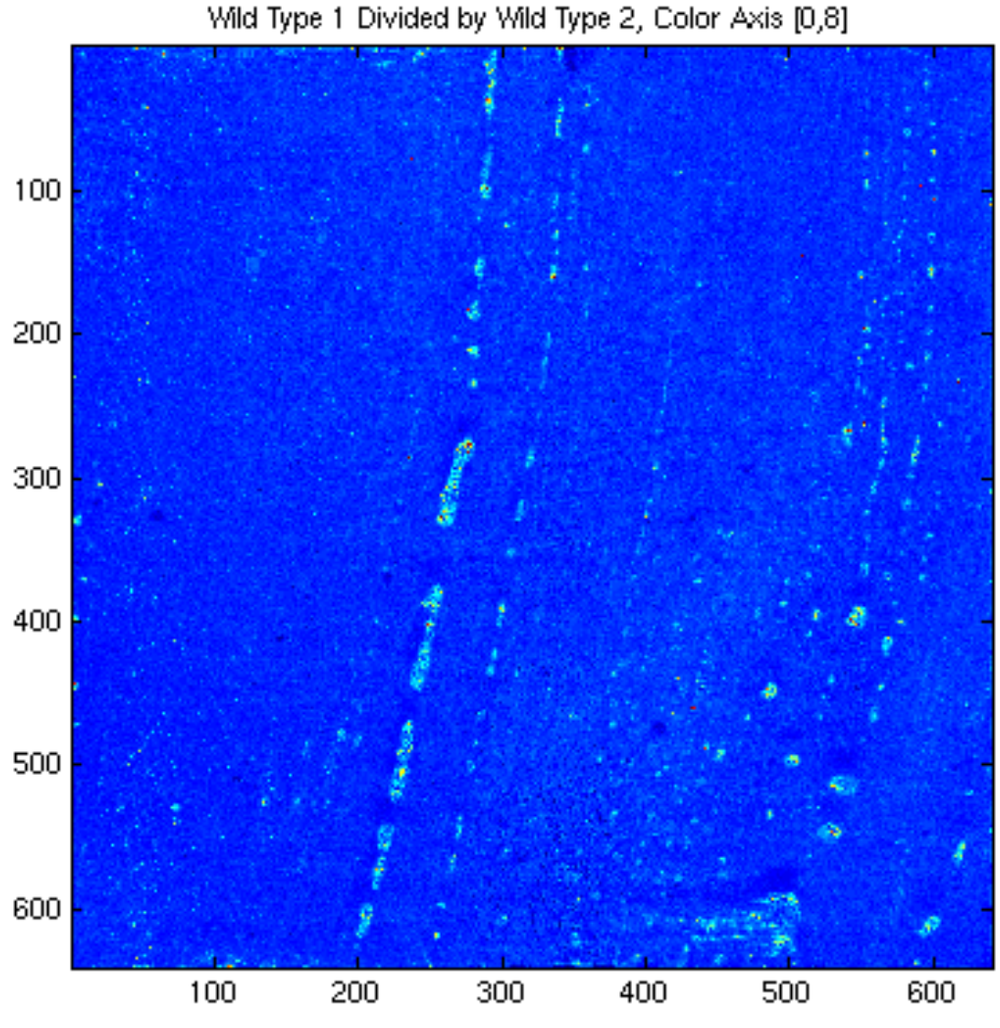
# Chip 1



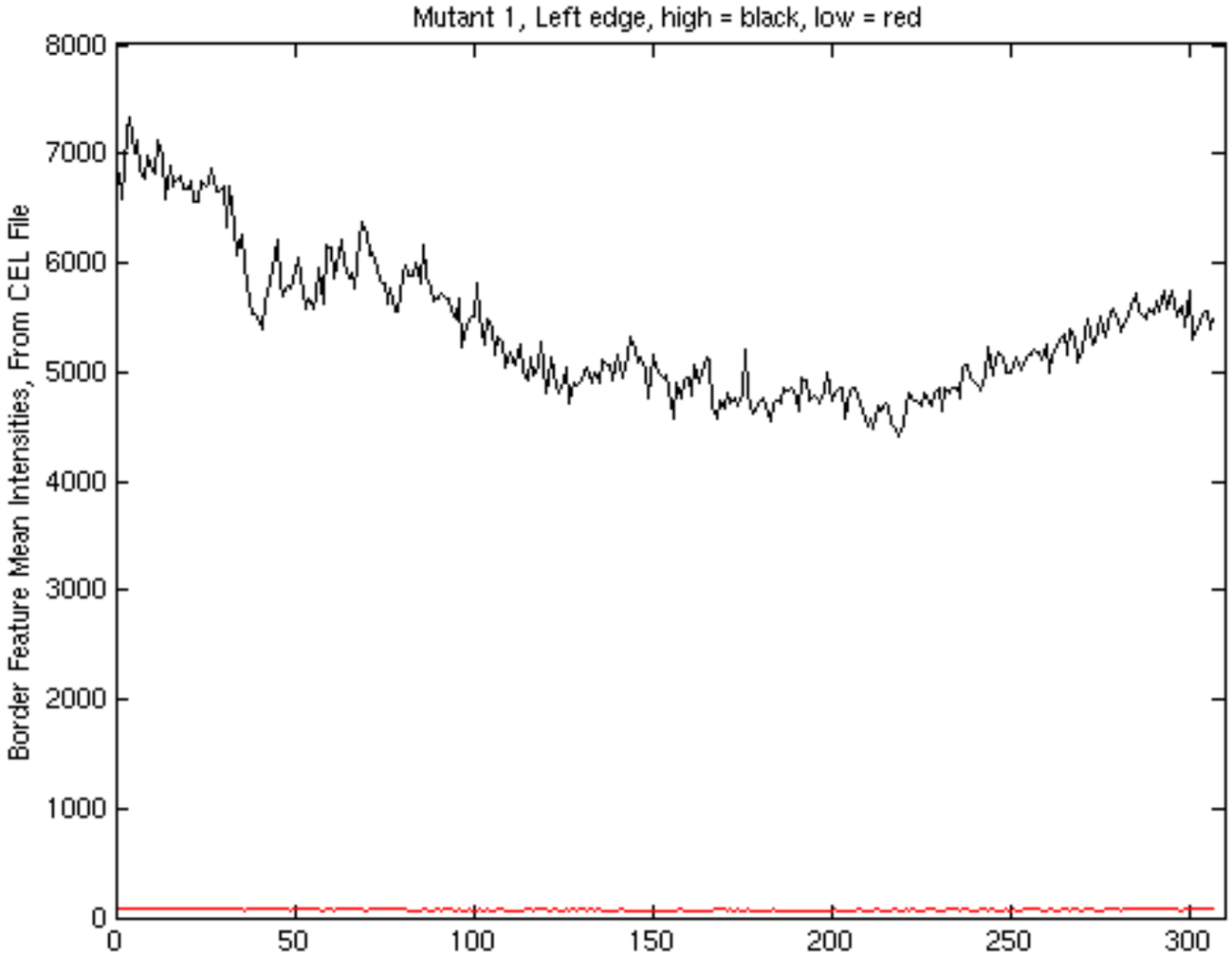
# Chip 2



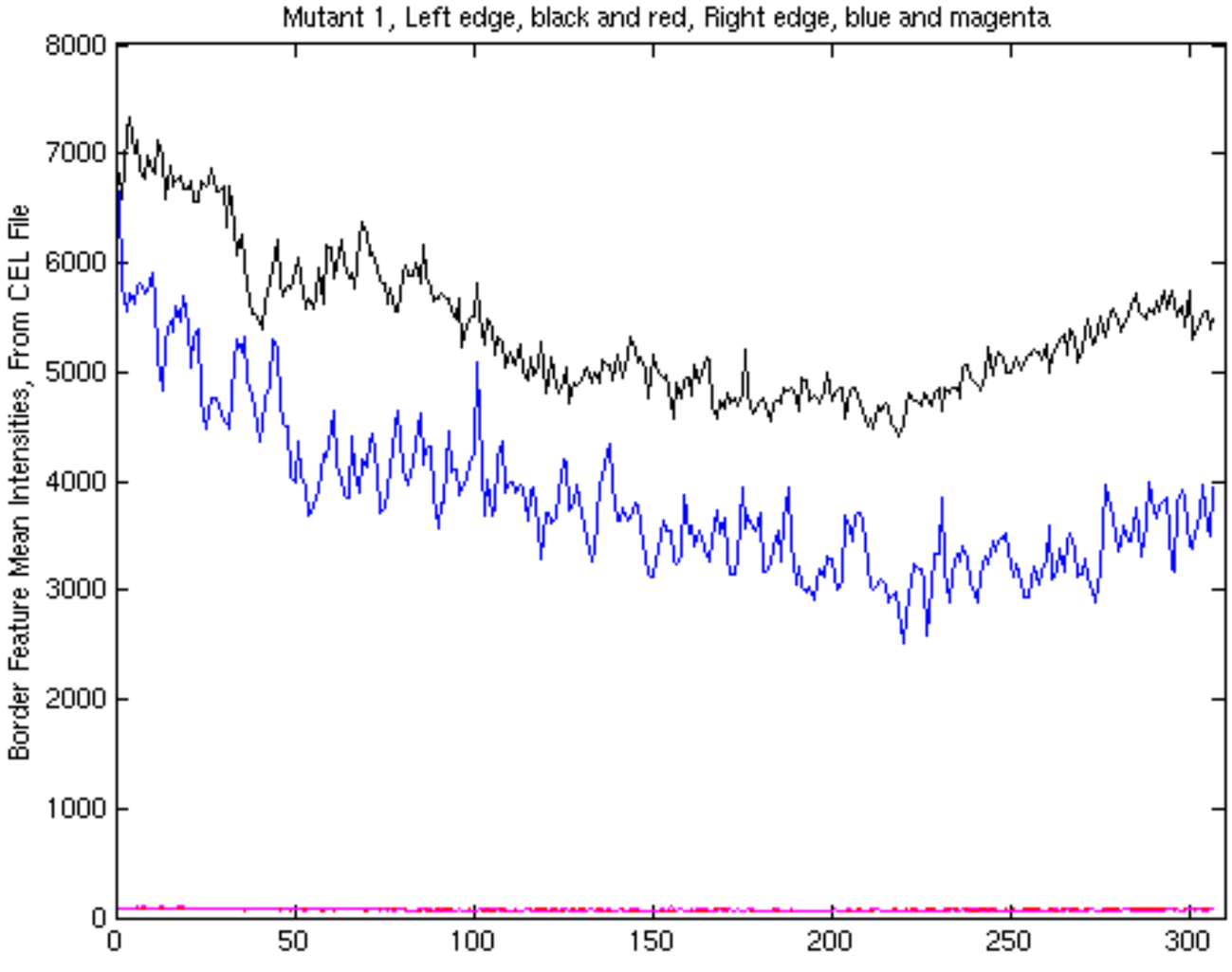
# Chips 1 and 2



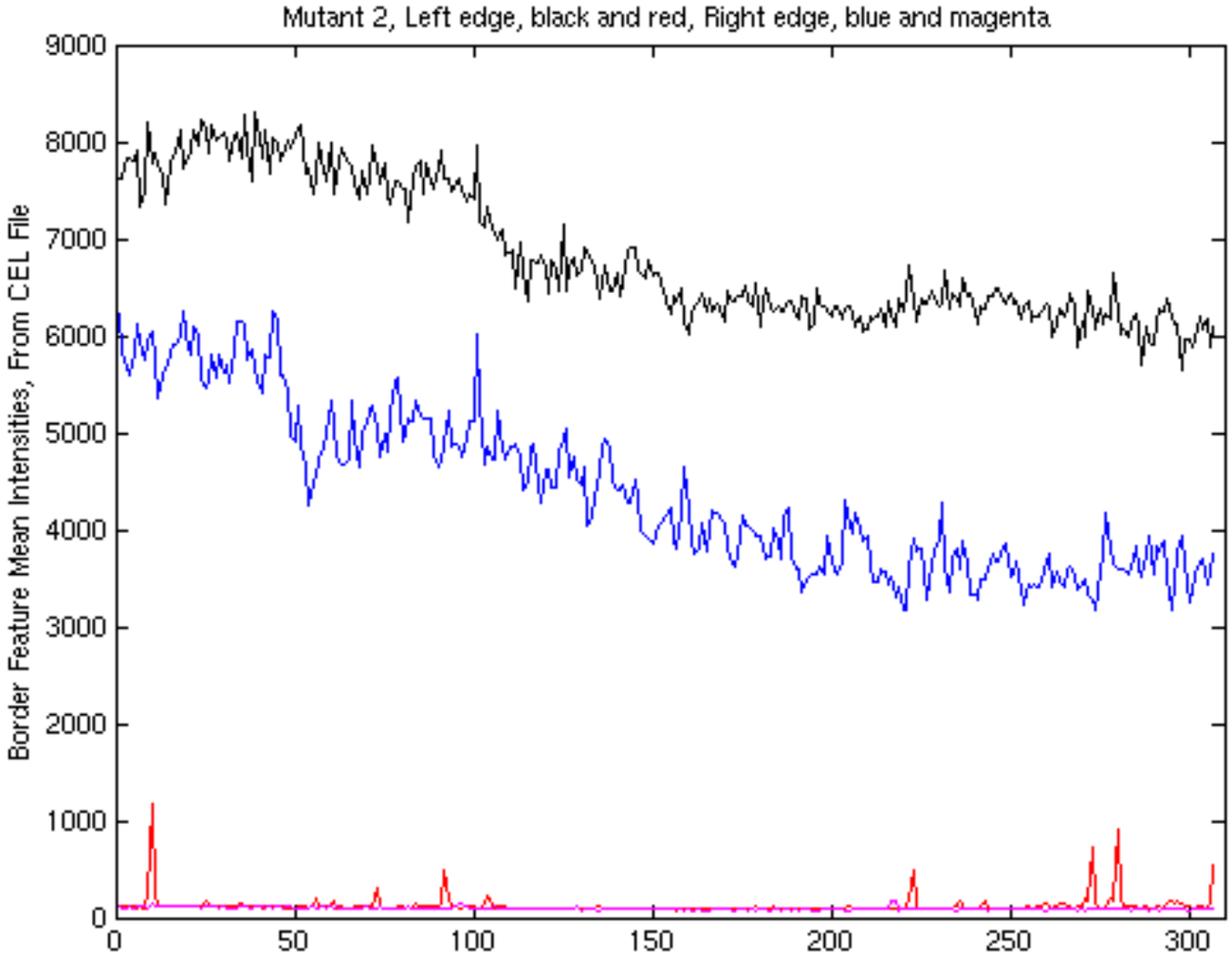
# and Within the Chip...



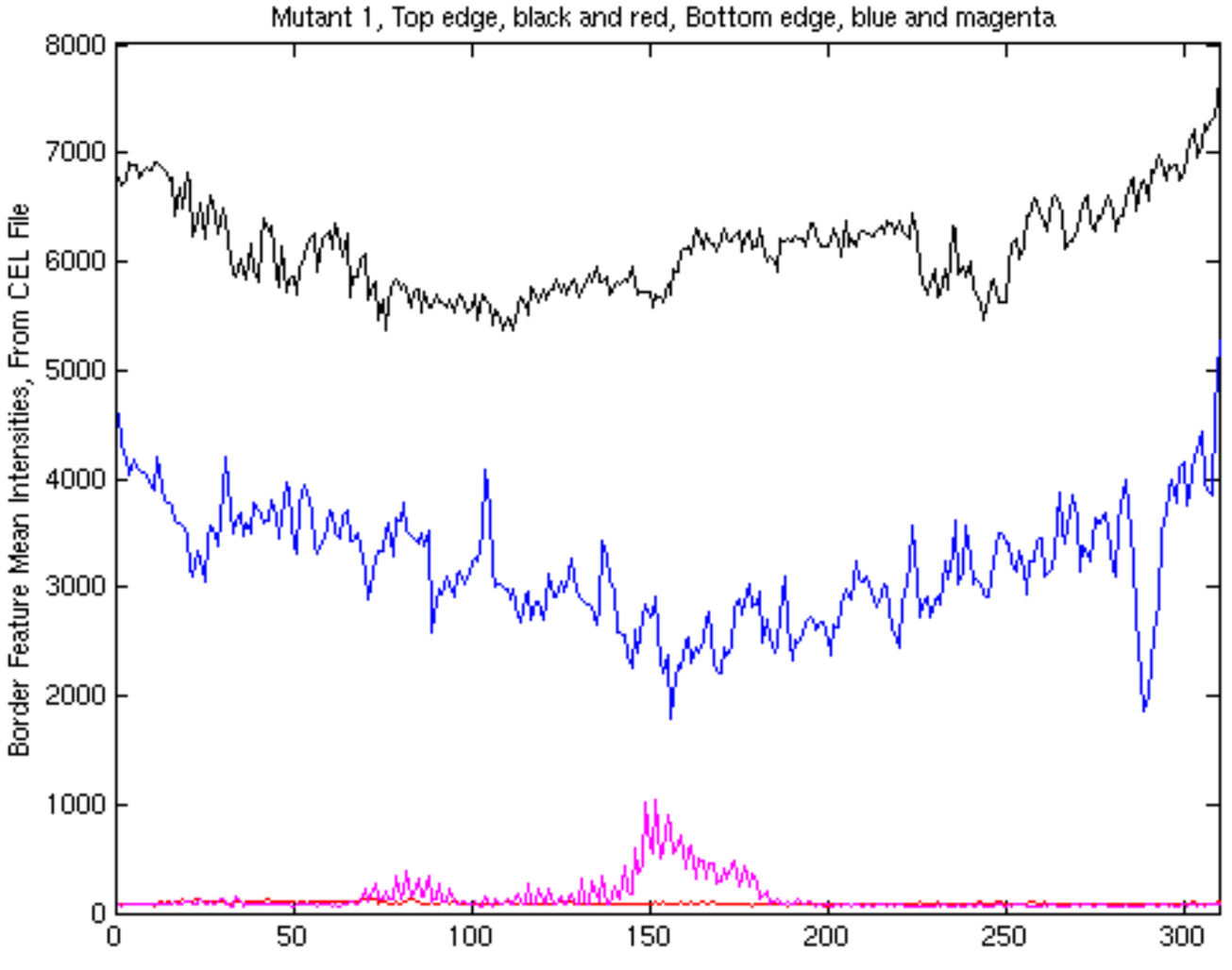
# and Within the Chip...



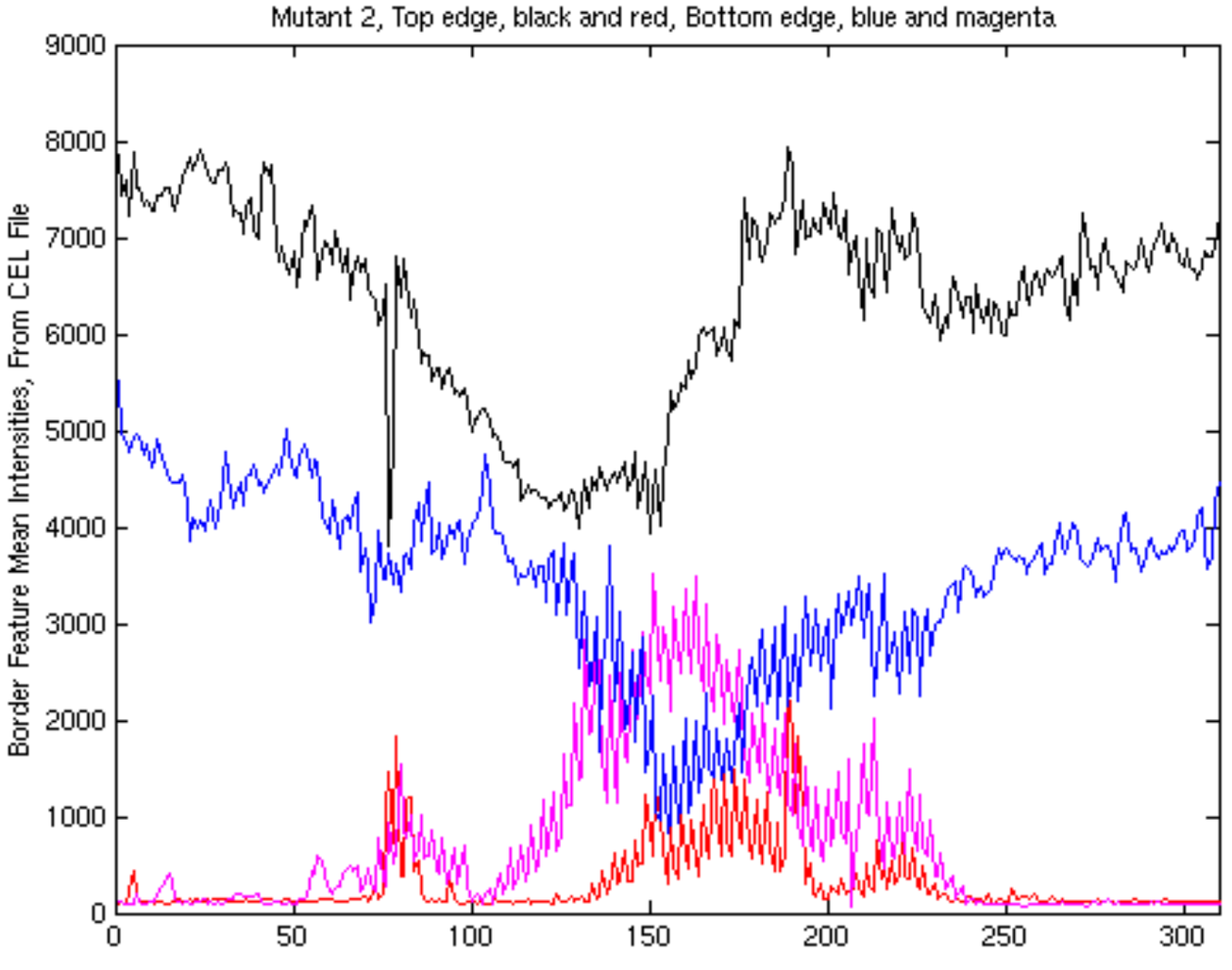
# and Within the Chip...



# and Within the Chip...

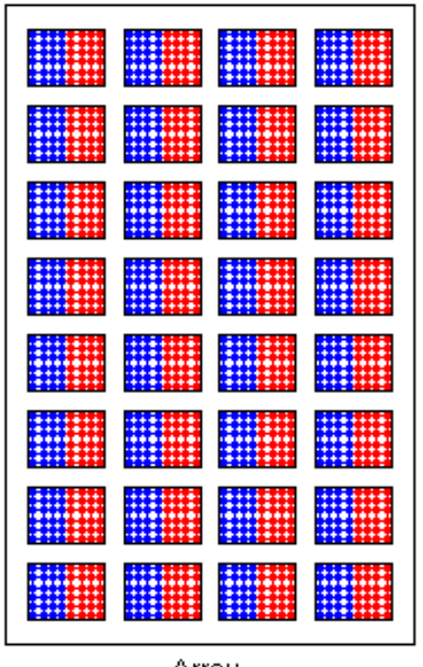


# and Within the Chip...

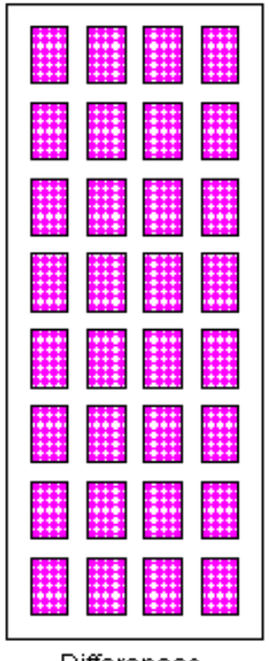


# But What About 2-Color Arrays?

In many cases, replicate spots (if they are used) will be printed on the array in a regular fashion.

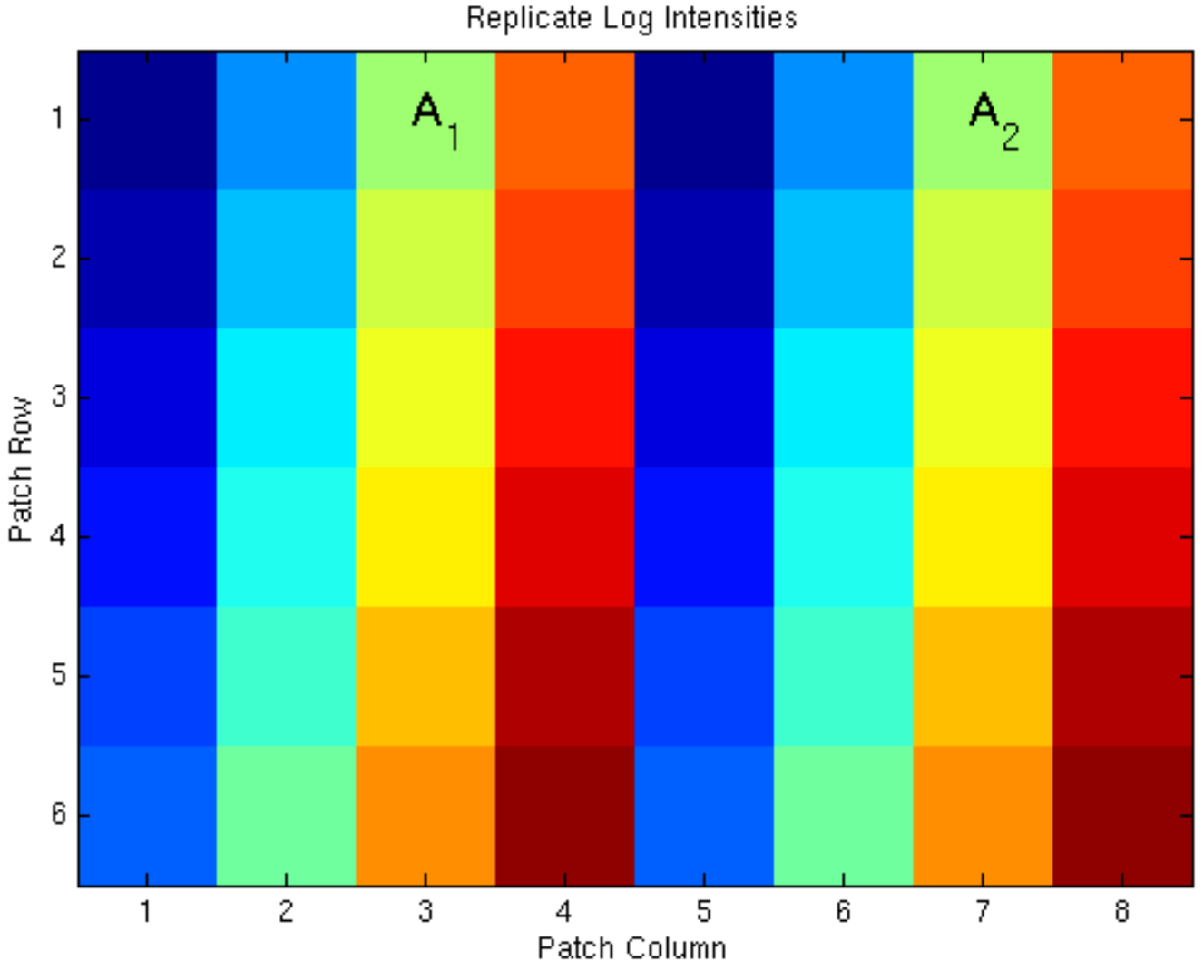


Array

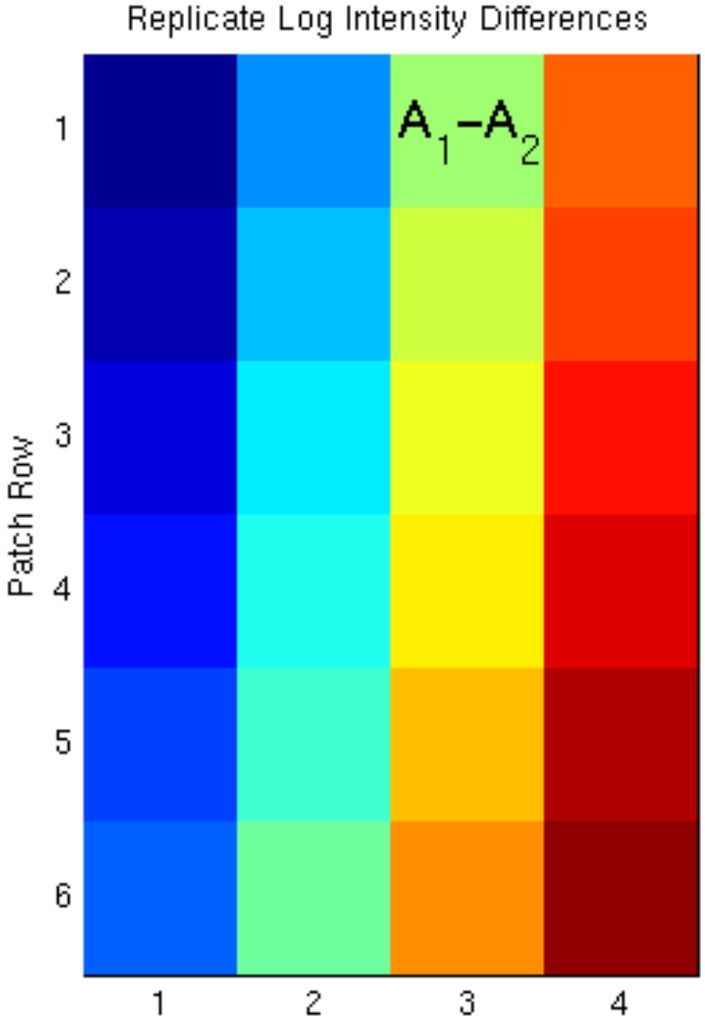


Differences

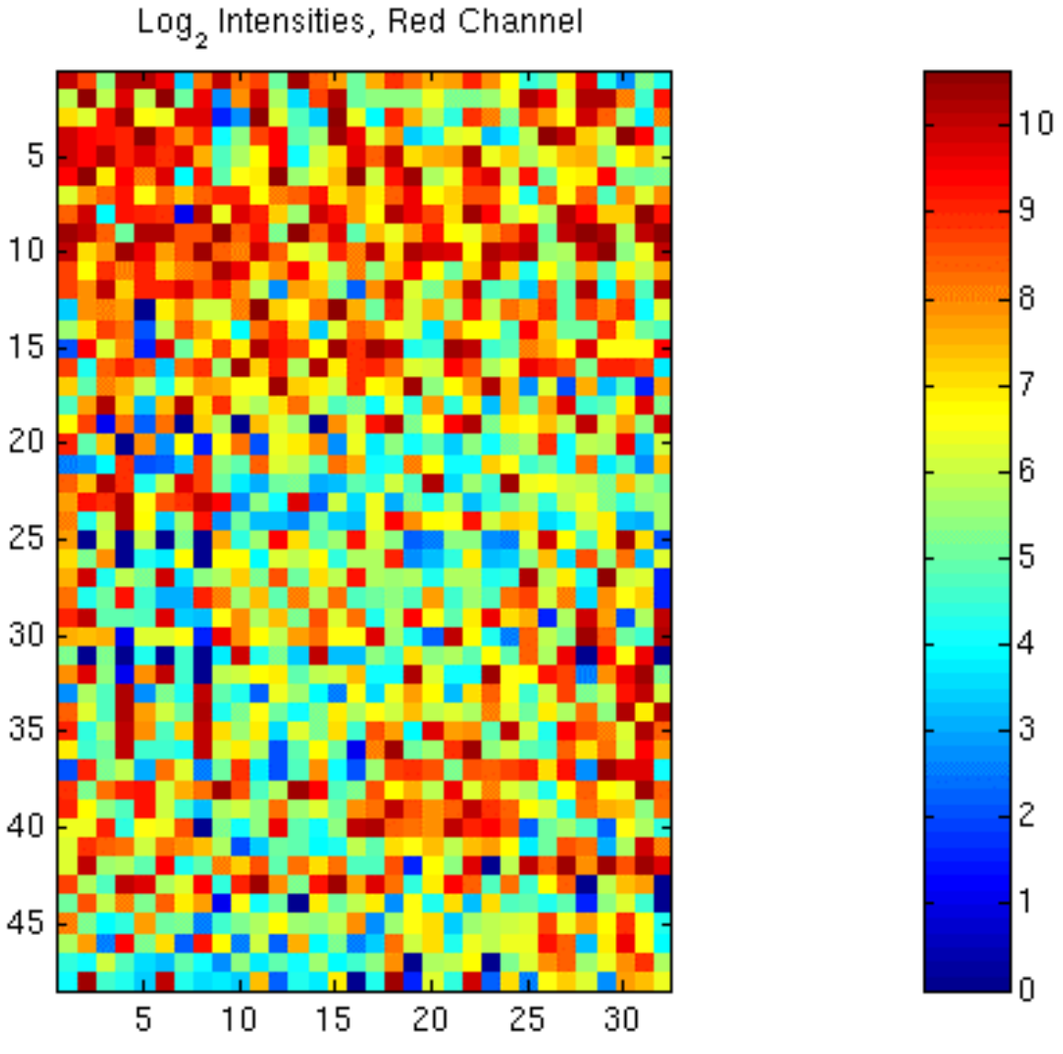
# Focus on a Grid



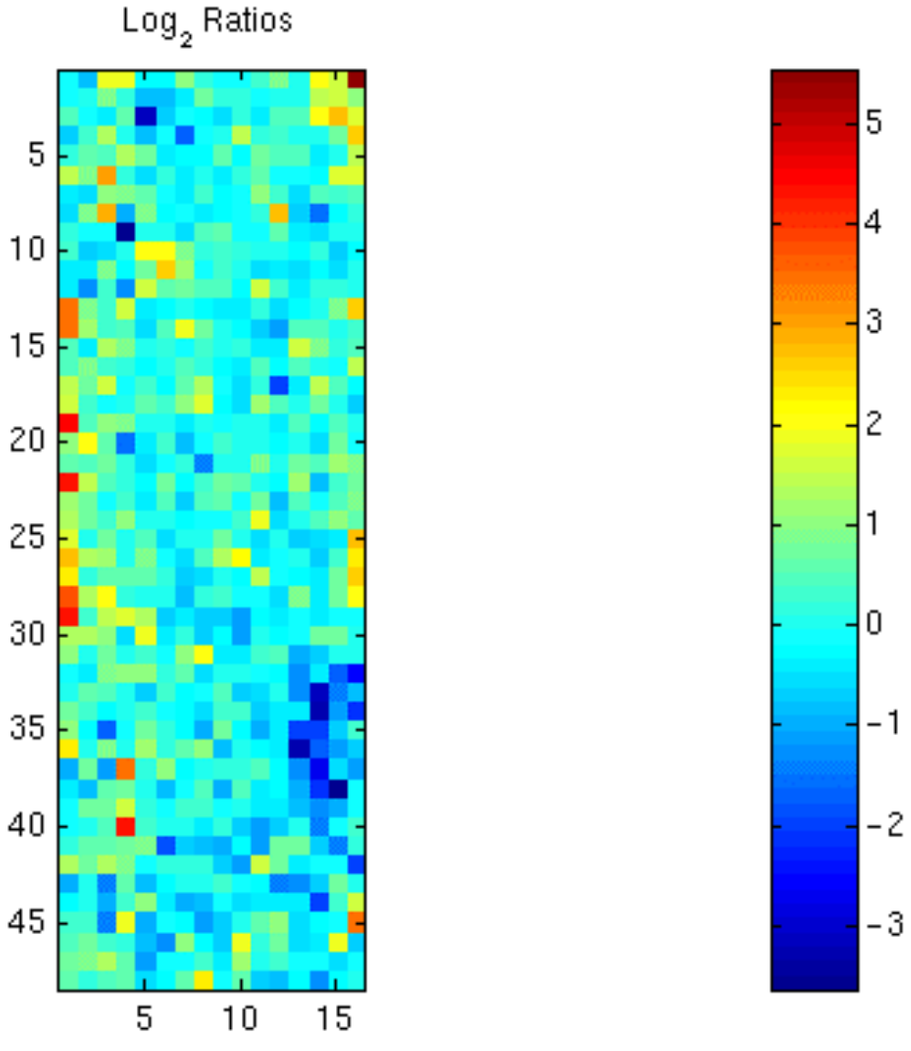
# Take Differences



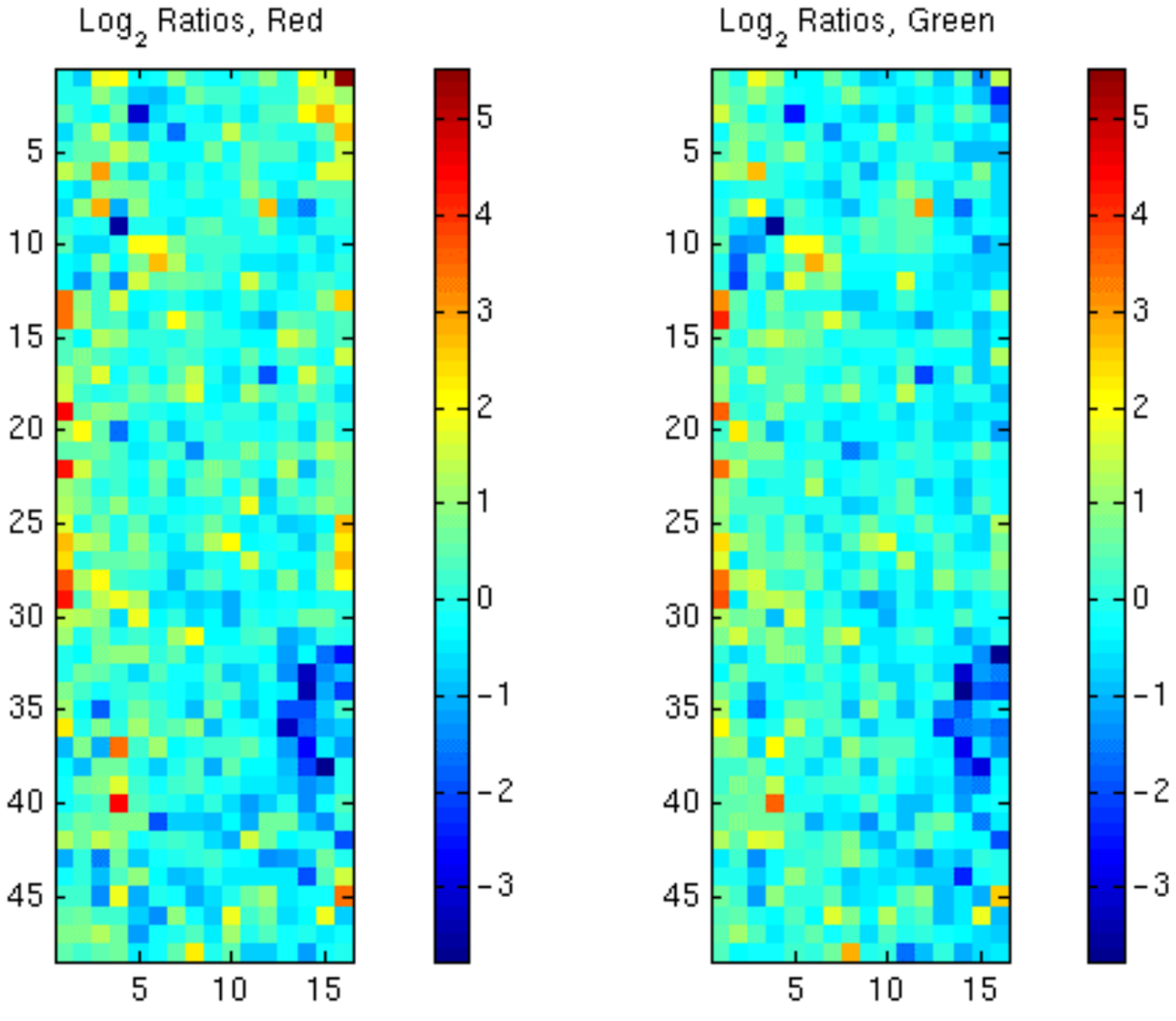
# Now Focus on One Channel



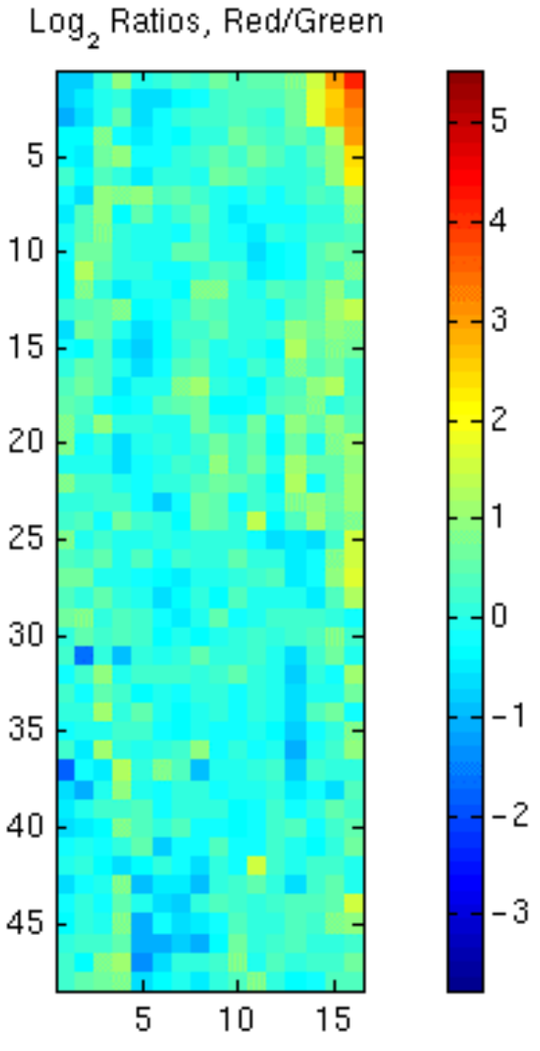
# Take Differences



# Check the Other Channel



# Check the Ratio



# What did Replicates Suggest?

There was a trend in the hybridization pattern across the chip. This is what we use the log ratios to correct for. However, if we are only working with a single channel, then we could use the replicates to estimate the trend and flatten it out.

But it's not quite what we want.

# Why Are Differences Suboptimal?

The use of differences does not generalize nicely to other patterns of replicate spotting.

There is no 1-to-1 correspondence with other points on the array.

Since it used fold differences, the visual effect can be dominated by spots with greater natural variation.

Colors can be misleading.

# Surface Estimates are Better

Using the differences and estimates of their standard errors, we'd prefer to construct an estimate of the "hybridization surface". This is similar to fitting a regression surface in more standard statistics, but there are differences.

(1) We don't have direct measurements of the surface that we're trying to estimate; just differences.

(2) Variation is local

(3) Continuity is imposed by a lattice.

# Mapping to Math

$$\sum_{\text{allwedges}} e_w^2, \qquad e_w = \hat{f}(i,j) - \hat{f}(i+1,j)$$

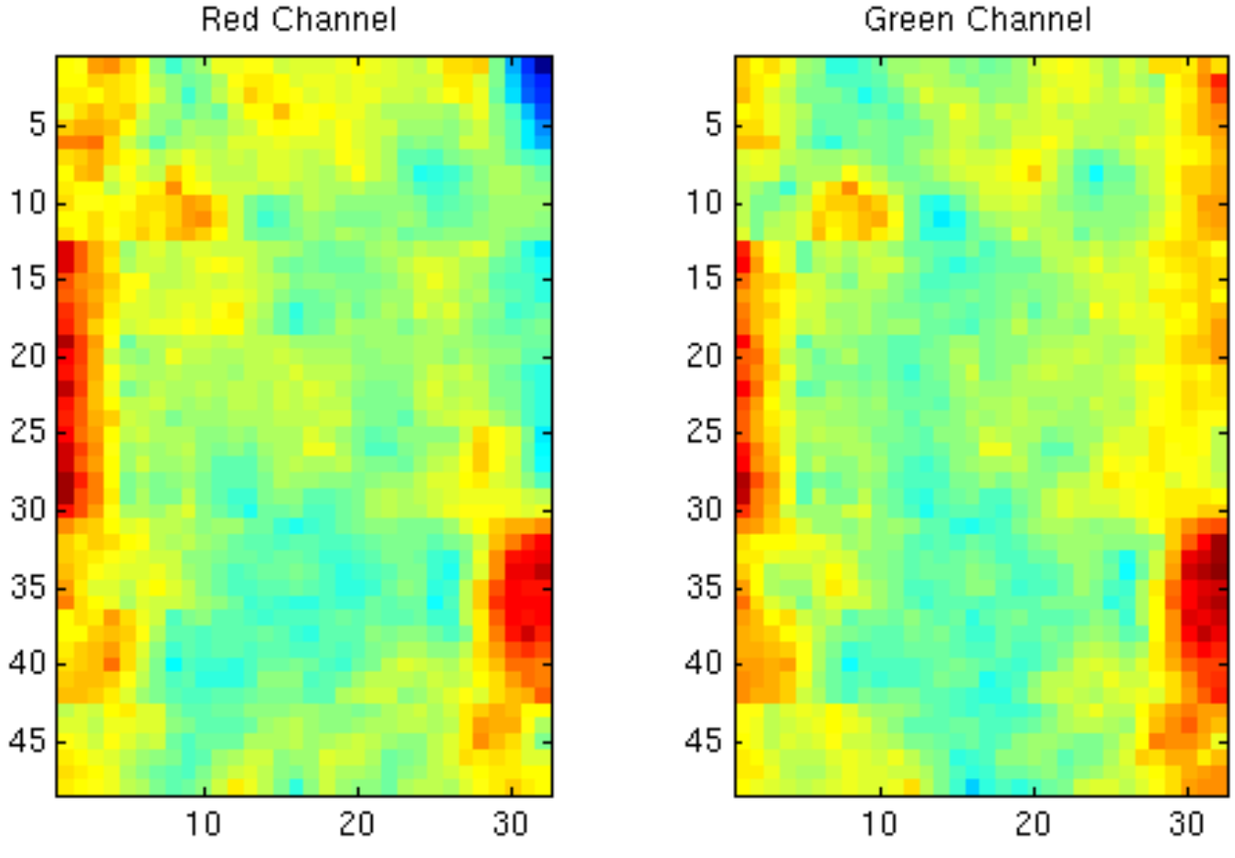
$$\sum_{\text{allbedges}} e_b^2, \qquad e_b = \hat{f}(i,j) - \hat{f}(i+1,j)$$

$$\sum_{\text{allreppairs}} w_r d_r^2, \qquad d_r = \left(\hat{f}(i_1,j_1) - \hat{f}(i_2,j_2)\right) - diff_r$$

At the end of the day, we want to minimize

$$\sum e_w^2 + k \sum e_b^2 + \lambda \sum w_r d_r$$

# Fitted Surfaces



# Things We Glossed Over, 1

How did we normalize the data?

We made a call to maNorm, but what does that do?

By default, this deals with print-tip loess, but there are two things to consider here. First, print-tip loess makes stronger assumptions that loess. Ratios from each print tip are assumed to have the same distributions, and this is a dangerous assumption if the allocation of clones to plates is nonrandom. Spots grouped by function may be brighter.

# Things We Glossed Over, 2

The second aspect of normalization that's a bit more difficult in the initial analysis is the question of how to deal with multiple array layouts.

I would tend to use some type of print-tip or spatial loess within each array to correct for overall trends, and then use quantile normalization to line up the ratios for genes spotted on both platforms.