

---

# Gene Expression – an Overview of Problems & Solutions: 1&2

---

Utah State University

Bioinformatics: Problems and Solutions

Summer 2006

---

# Review

- DNA → mRNA → Proteins → action!
  - mRNA transcript abundance ~ expression level
  - microarrays
    - represent thousands of gene segments on chip
    - mRNA transcripts hybridize to matching spots
    - look at spot intensities
    - oligo & cDNA arrays exist
  - Often interested in identifying genes that change expression levels between disease states
  - Bioconductor: a [free!] tool
  - Reference: Bioinformatics and Computational Biology Solutions Using R and Bioconductor, edited by Gentleman et al.
-

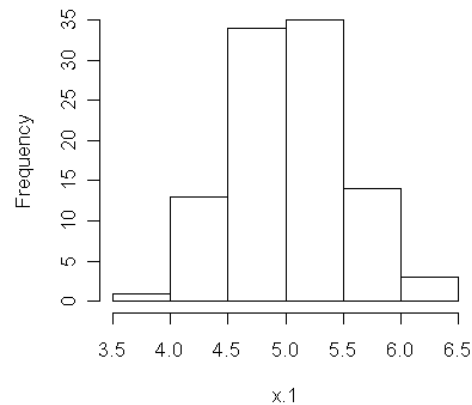
---

# Problem 1: GIGO

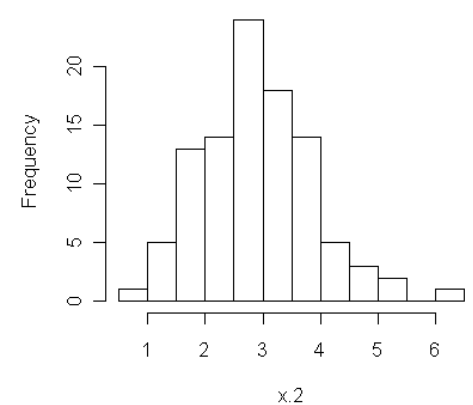
- How to assess data quality?
  - Focus on graphical checks:
    - Histograms
    - Boxplots
    - Images
    - Residual Images
-

```
### Dummy example of histogram
### & boxplot
set.seed(123)
x.1 <- rnorm(100,mean=5,sd=0.5)
x.2 <- rnorm(100,mean=3,sd=1)
x.12 <- c(x.1,x.2)
par(mfrow=c(2,2))
hist(x.1)
hist(x.2)
hist(x.12)
boxplot(x.12,
        main='Boxplot of x.12')
```

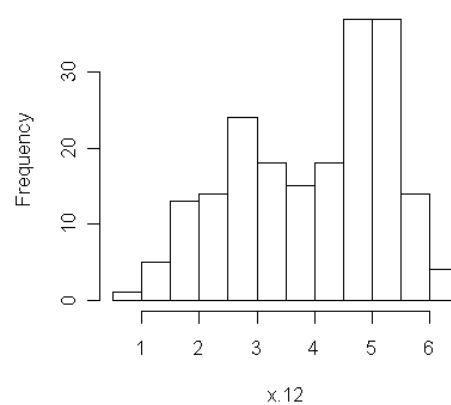
Histogram of x.1



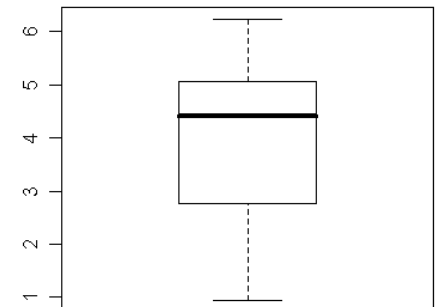
Histogram of x.2



Histogram of x.12



Boxplot of x.12

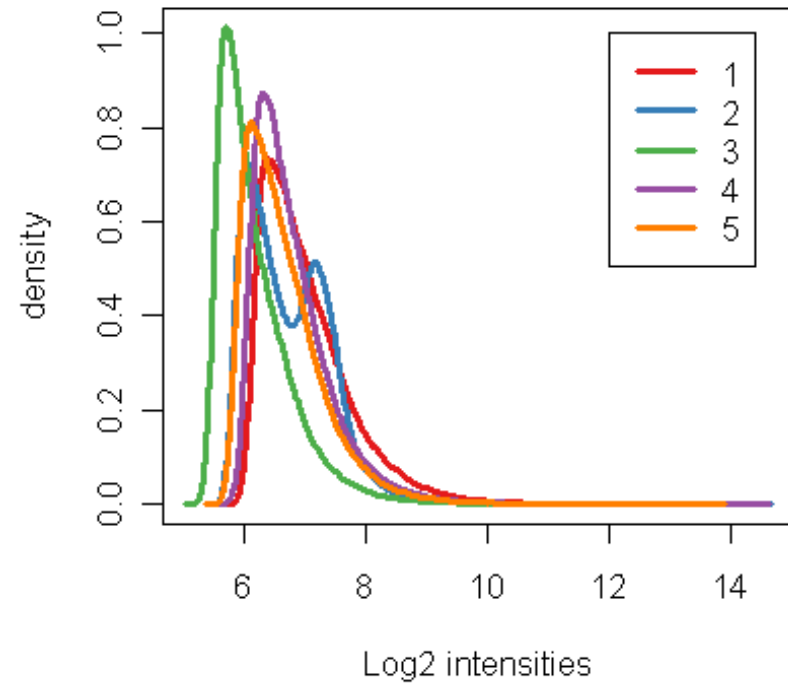
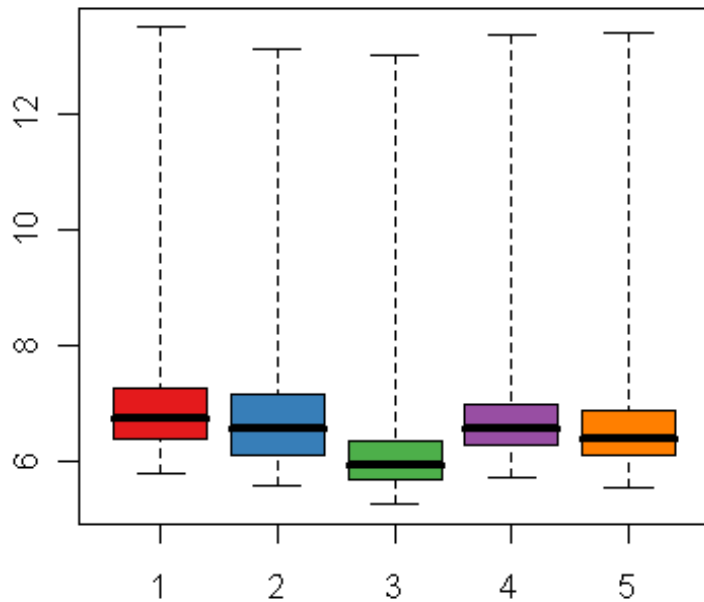


```
# Startup
memory.limit(size=4000)
Sys.putenv("http_proxy"="http://proxy.usu.edu:80/")
Sys.getenv("http_proxy")
source("http://www.stat.usu.edu/~jrstevens/get.packages.R")
library(affy); windows(record=T)

# Load data
library(ALLMLL); data(MLL.B)
Data <- MLL.B[,c(1:5)] # an AffyBatch object w/ 5 arrays

####
#### Problem 1: Quality Checks
####

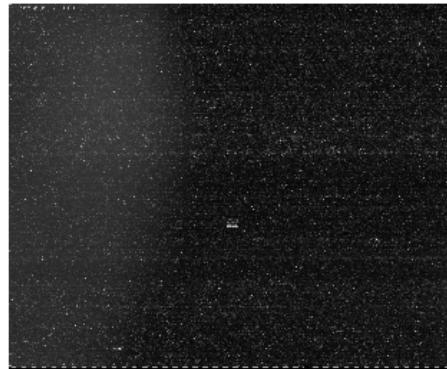
# Look at boxplots and histograms
library(RColorBrewer)
par(mfrow=c(2,2))
cols <- brewer.pal(5, "Set1")
boxplot(Data,col=cols,names=1:5)
# Possible difference in array 3
hist(Data,col=cols,lty=1,xlab="Log2 intensities",lwd=2)
legend(12,1,1:5,lty=1,col=cols,lwd=2)
# Notice difference in shape of array 2
```



Get a boxplot and a smoothed histogram of  
log2 intensities on each array  
– look for differences in shape and location

```
# Look at images
par(mfrow=c(2,2))
image(Data[,2:3])
# Fit probe-level model
# and look at images of
# residuals
library(affyPLM)
Pset <- rmaPLM(Data)
image(Pset,type="resids",
      which=c(2,3))
```

/tmp/MLL/JD-ALD051-v5-U133B.CEL



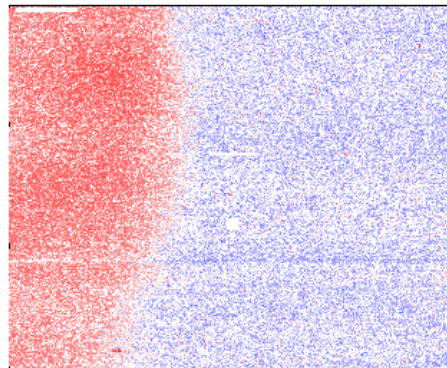
/tmp/MLL/JD-ALD051-v5-U133B.CEL



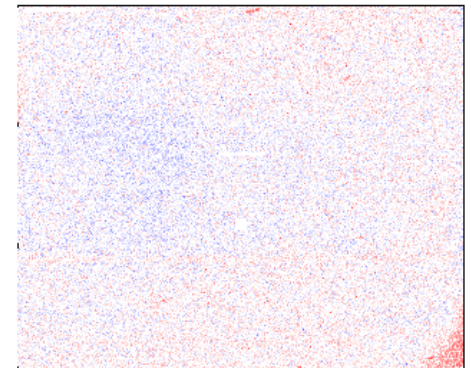
Look for:

- Substantial artifacts
- Systematic patterns

JD-ALD051-v5-U133B.CEL



JD-ALD052-v5-U133B.CEL



# MA plot – compare 2 arrays (X & Y)

- MA plot:  $M=Y-X$  vs.  $A=0.5(Y+X)$   
rotate and scale Y vs. X scatterplot  
(X & Y on log-scale  $\rightarrow$  M is log fold-change)
- What would this mean?  
(look at whether degree of change depends on average expression level; it shouldn't)
- Loess curve (to look at overall trend):  
locally weighted polynomial regression

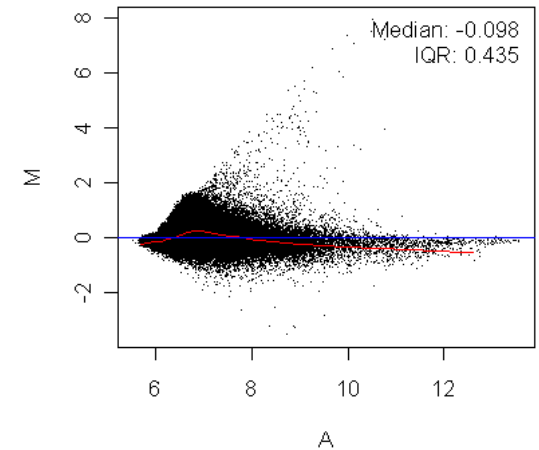
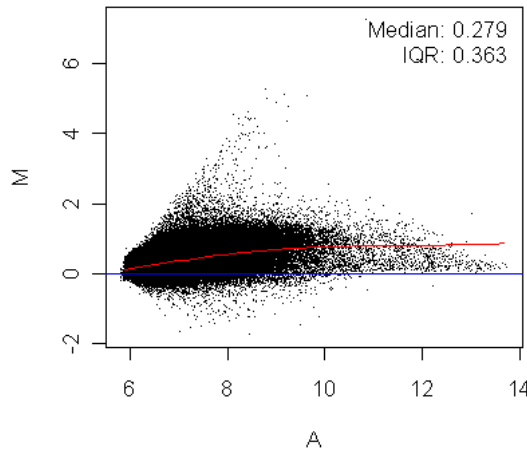
# MA plot

```
# Look at MA-plots  
par(mfrow=c(2,2))  
MAplot(Data[,1:4],  
       cex=1)
```

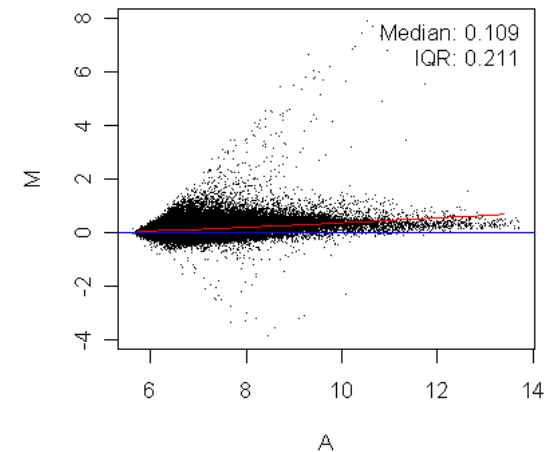
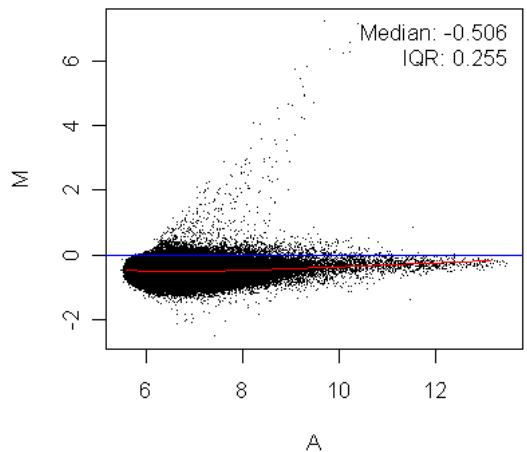
Quality problems  
most apparent when:

- Loess line oscillates much
- M-variability is much greater than other arrays

JD-ALD009-v5-U133B.CEL vs pseudo-median JD-ALD051-v5-U133B.CEL vs pseudo-median

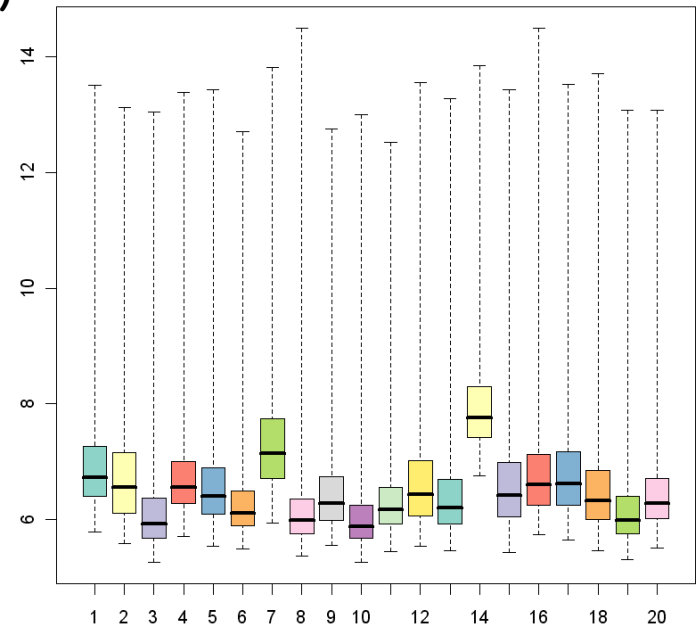


JD-ALD052-v5-U133B.CEL vs pseudo-median JD-ALD057-v5-U133B.CEL vs pseudo-median



# Problem 2: Preprocessing

```
####  
#### Problem 2: Preprocessing  
####  
  
Data <- MLL.B #AffyBatch object, 20 arrays  
# 'Fix' a few slots on this object  
sample <- 1:20  
Data@reporterInfo<-data.frame(sample=sample)  
sampleNames(Data) <- as.character(sample)  
sample.frame <- data.frame(sample=sample)  
rownames(sample.frame) <- sample  
pData(Data) <- sample.frame  
# Look at boxplots  
cols <- brewer.pal(20, "Set3")  
par(mfrow=c(1,1))  
boxplot(Data,col=cols,names=1:20)
```



# Preprocessing: motivation & main idea

- Shouldn't have dramatic array effect, especially for the same "treatment" levels
  - Want to make measurements from different arrays (replicates) directly comparable
  - Three components:
    - Background correction
      - to remove small local artifacts
    - Normalization
      - to remove array effects
    - Summarization
      - to combine probe intensities into one measure
-

# Example: Robust Multi-Array Average

- RMA Background Correction: Convolution
    - Separate out probe-level signal from background signal (optical noise & non-specific binding)
    - Fits a probe-level model using PM only
  - RMA Normalization: Quantile
    - Make all arrays have same quantiles
    - Essentially force arrays to have same histogram
  - RMA Summarization: Median Polish
    - A robust measure of “center”, using probe-level background-corrected and quantile-normalized intensities
    - A little ad-hoc, but works well here
  - Final RMA expression level on the log<sub>2</sub> scale
-

# Going back a bit: PLM Residual Image

- Probe Level Model
- RMA model:

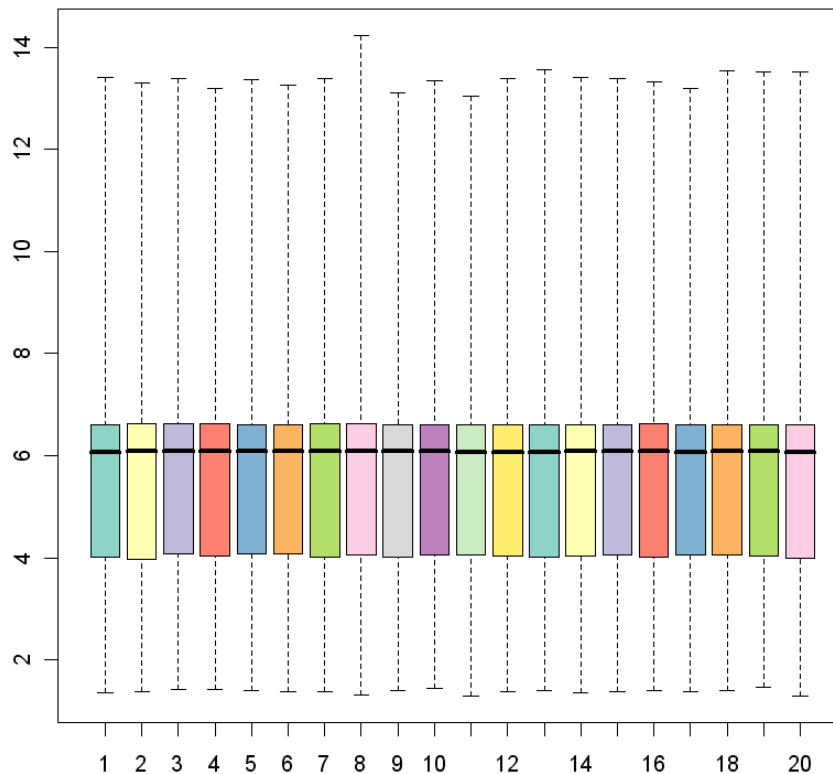
$$Y_{ijk} = \underbrace{\mu_{ik}} + \underbrace{\alpha_{jk}} + \varepsilon_{ijk}$$

Probe affinity effect

↑  
Log-scale expression level for gene k on array i

- Use robust measure to estimate model parameters
- To identify quality problems, look at residuals

```
# Obtain RMA expression estimates
eset <- rma(Data)
# Look at what this eset has for us
eset@exprs[1:2,]
# Look at what this did to the intensities
rma.abatch <- bg.correct(Data,method="rma")
quant.abatch <- normalize(rma.abatch, method="quantiles")
boxplot(quant.abatch,col=cols,names=1:20)
```



---

# Other preprocessing methods

– each has strengths and weaknesses

- MAS5

- Affymetrix's algorithm (use PM-MM)

- dChip

- Li & Wong's model (PM or PM&MM, in probe-level model)

- vsn

- “Variance Stabilization”

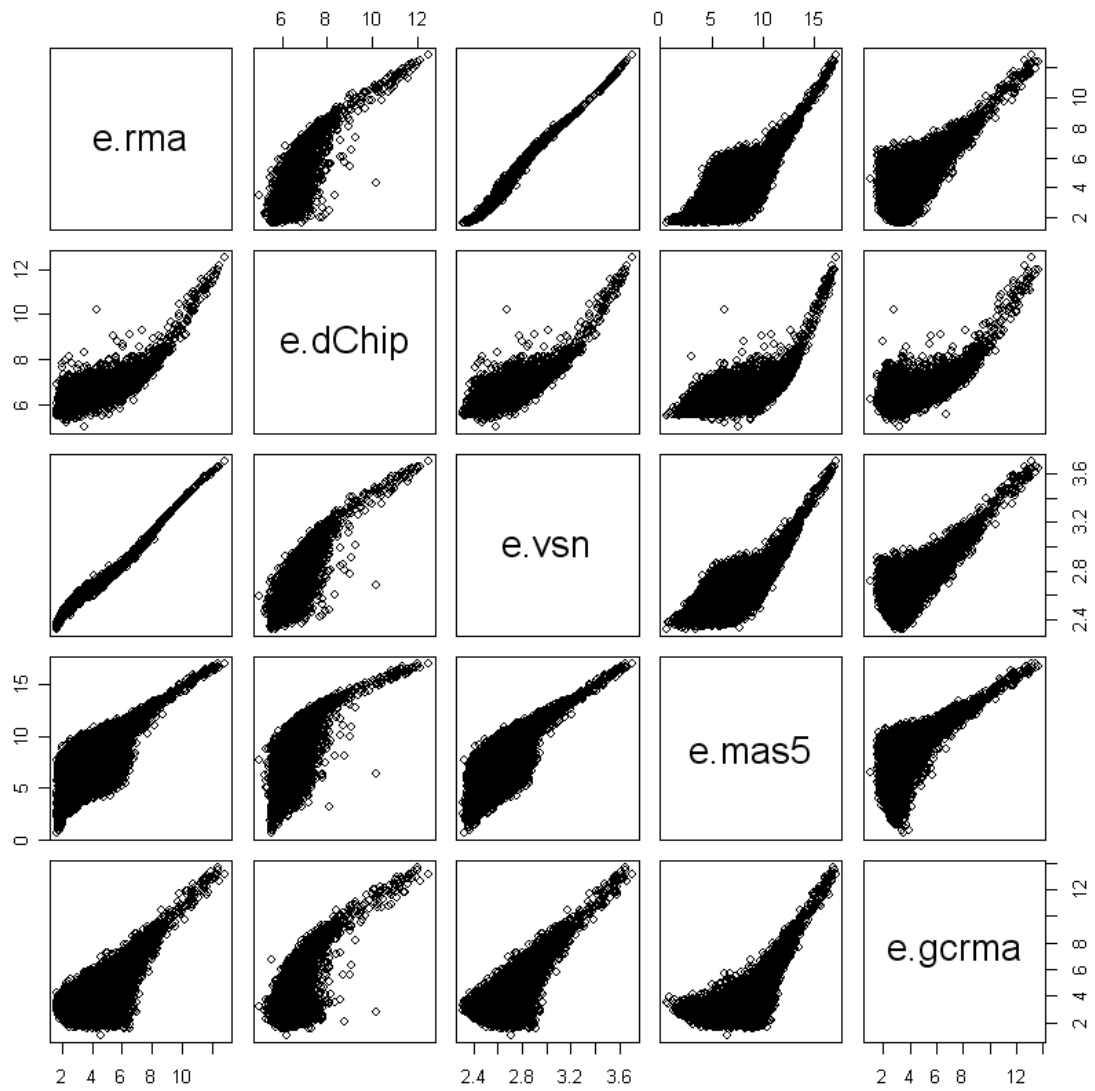
- GCRMA

- RMA-based, with additional adjustment for position of G & C nucleotides in probe sequence

- (more proposed all the time – active area of research)

---

```
# dChip with PM only:
eset.dChip.pm <- suppressWarnings(expresso(Data,
  normalize.method="invariantset", bg.correct=FALSE,
  pmcorrect.method="pmonly", summary.method = "liwong"))
# vsn:
library(vsn)
eset.vsn <- suppressWarnings(expresso(Data,
  normalize.method="vsn", bg.correct=FALSE,
  pmcorrect.method="pmonly", summary.method="medianpolish"))
# MAS 5.0:
eset.mas5 <- mas5(Data)
# GCRMA:
eset.gcrma <- gcrma(Data)
# Compare these: (written to file to save time later)
e.rma <- eset@exprs[,1]
e.dChip <- log2(eset.dChip.pm@exprs[,1])
e.vsn <- log2(eset.vsn@exprs[,1])
e.mas5 <- log2(eset.mas5@exprs[,1])
e.gcrma <- eset.gcrma@exprs[,1]
e.mat <- as.data.frame(cbind(e.rma,e.dChip,e.vsn,e.mas5,e.gcrma))
write.csv(e.mat,"C:\\folder\\e.mat.csv",quote=F,row.names=F)
e.mat <- read.csv("C:\\folder\\e.mat.csv",header=T)
plot(e.mat)
```



Each point represents a single gene's expression level on one array after preprocessing all the arrays together

Notice similarities and differences

Notice "overplotting"  
→ lost information

```
# More efficient scatterplots
```

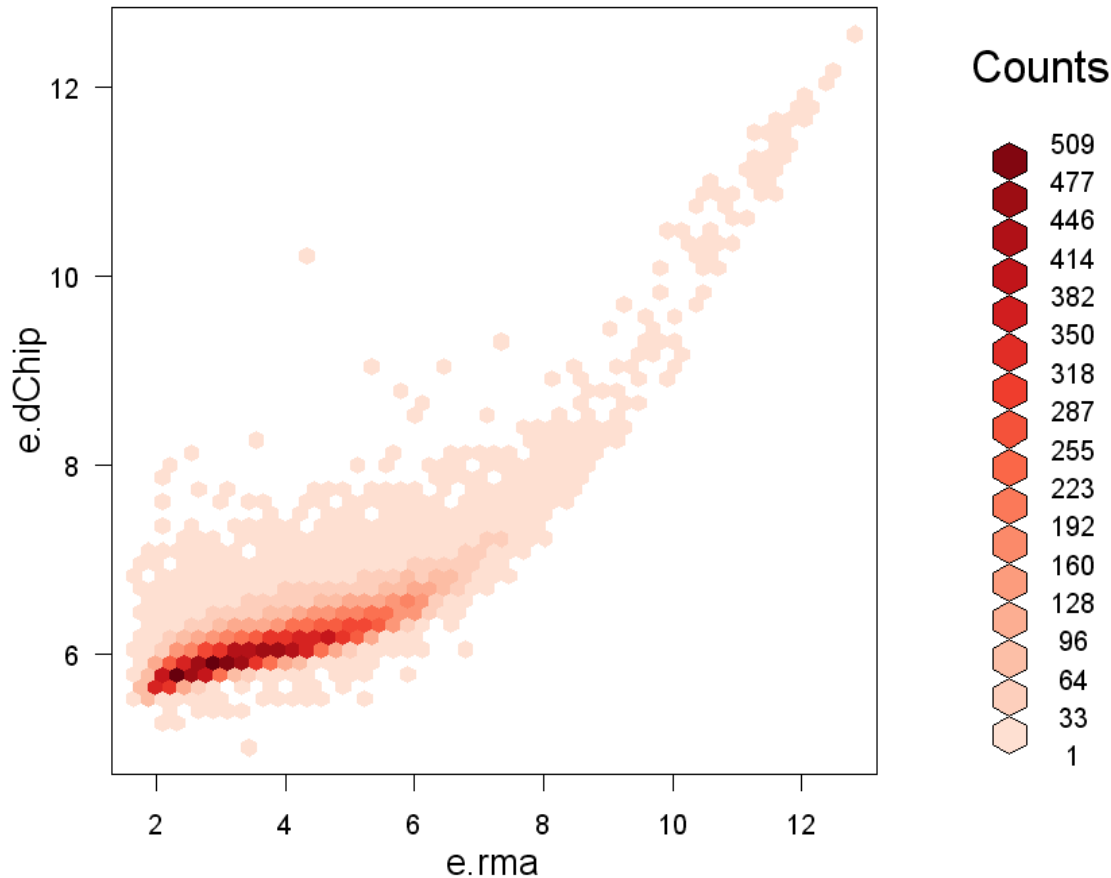
```
library(RColorBrewer); library(hexbin)
```

```
red.ramp <- colorRampPalette(brewer.pal(9, "Reds")[-1])
```

```
mat <- e.mat[,c(1,2)]
```

```
hb <- hexbin(mat,xbins=50)
```

```
plot(hb,colramp=red.ramp)
```



Here, “density” is represented by color

Coming up: more on effective visualization