# Introduction to gene expression microarray data analysis

# Outline

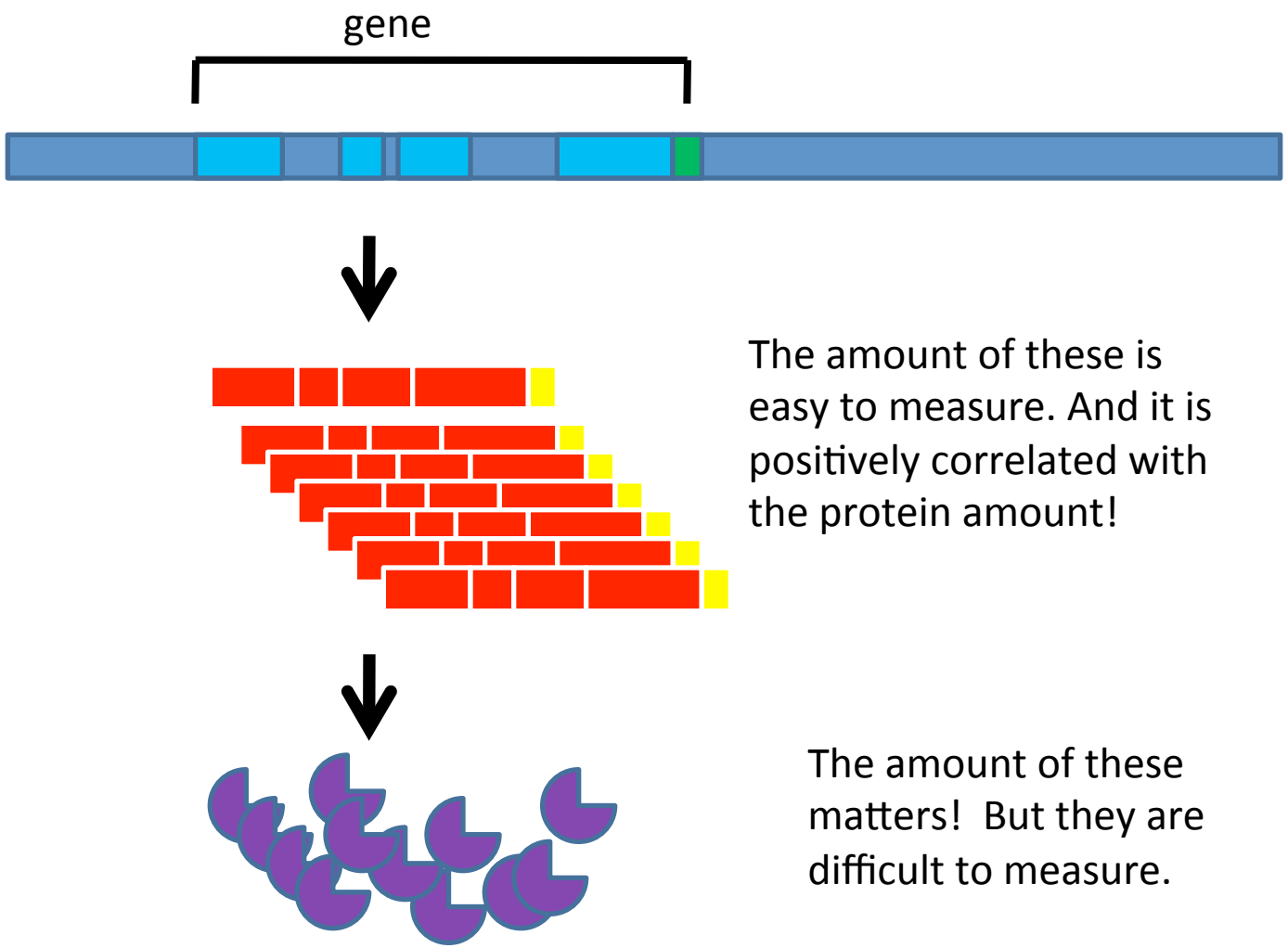
- Brief introduction:
  - Technology and data.
  - Statistical challenges in data analysis.
- Preprocessing
- Differential expression
- Useful Bioconductor packages

Still microarray?

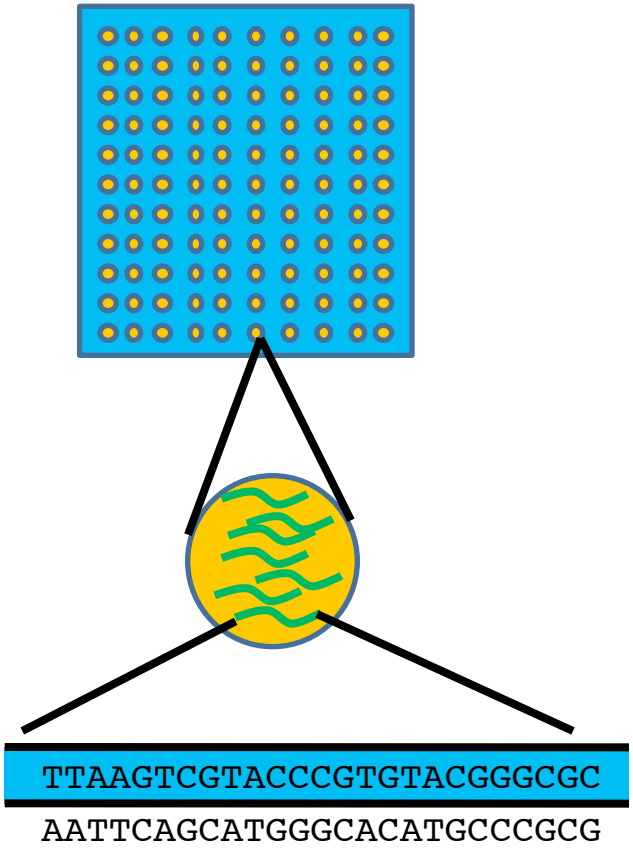
- Microarray is still widely used because of lower costs, easier experiment and more comprehensive analysis methods.
- Similar problems are presented in newer technologies such as RNA-seq, and similar statistical techniques can be borrowed.

# Introduction to technology and array designs

# Goal: measure mRNA abundance

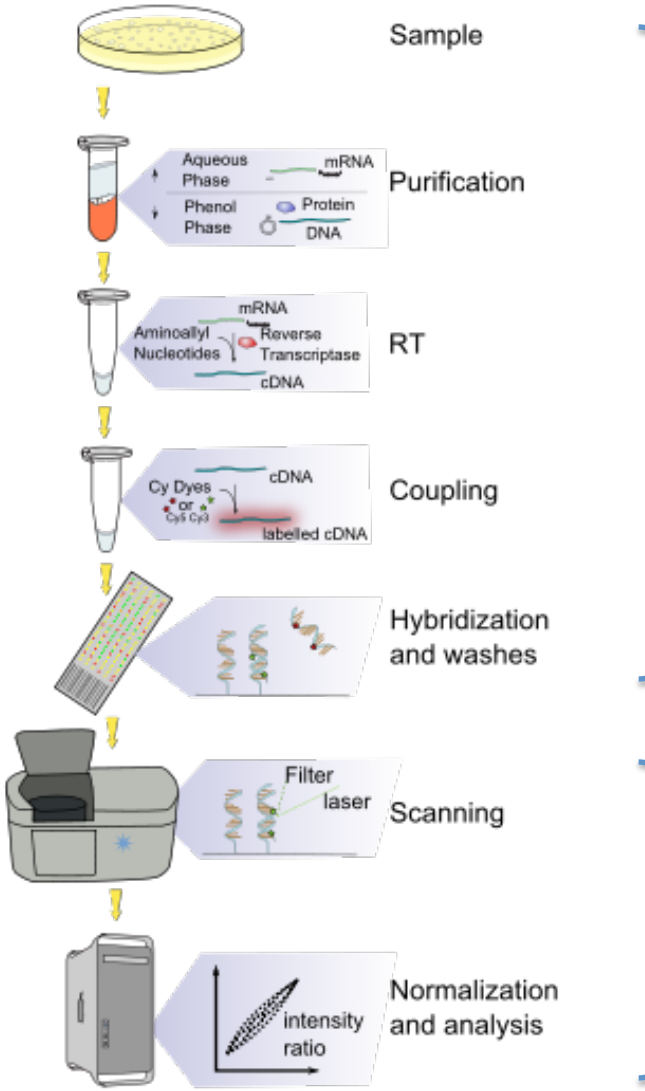


# Gene expression microarray design




- A collection of DNA spot on a solid surface.
- Each spot contains many copies of the same DNA sequence (called “probes”).
  - Probe sequences are designed to target specific genes.
- Genes with part of sequence complementary to a probe will hybridized on (stick to) that probe.
- The amount of hybridization on each probe measures the amount of mRNA for its target gene.

# Experimental procedure




wet lab: perform experiment

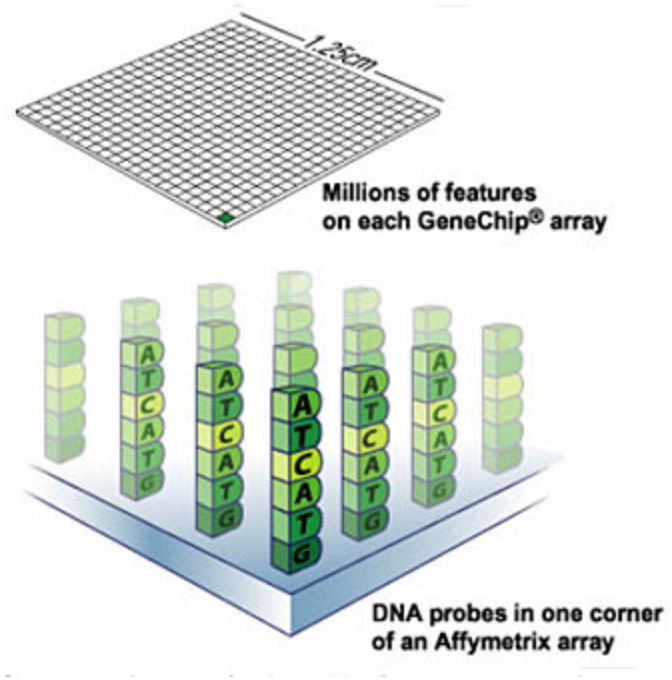
dry lab: data analysis

# Available platforms

- Affymetrix
- Agilent
- Nimblegene
- Illumina
- ABI
- Spotted cDNA

# Affymetrix Gene expression arrays

The Affymetrix platform is one of the most widely used.

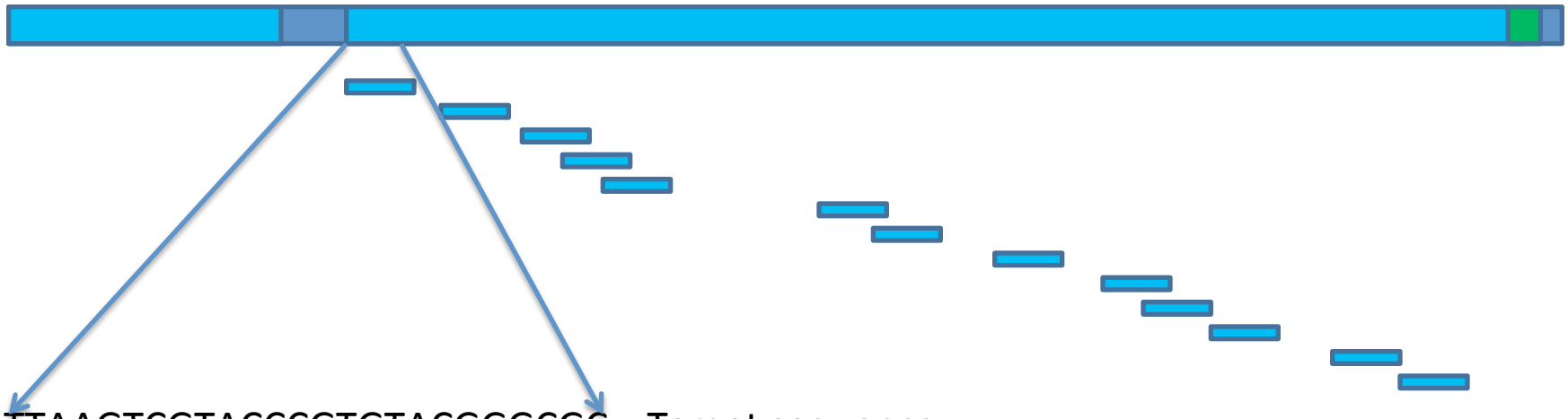




http://www.affymetrix.com/

# Affymetrix GeneChip array design

Use U133 system for illustration:

- Around 20 probes per gene;
- Not necessarily evenly spaced: sequence property matters;
- The probes are located at random locations on the chip;

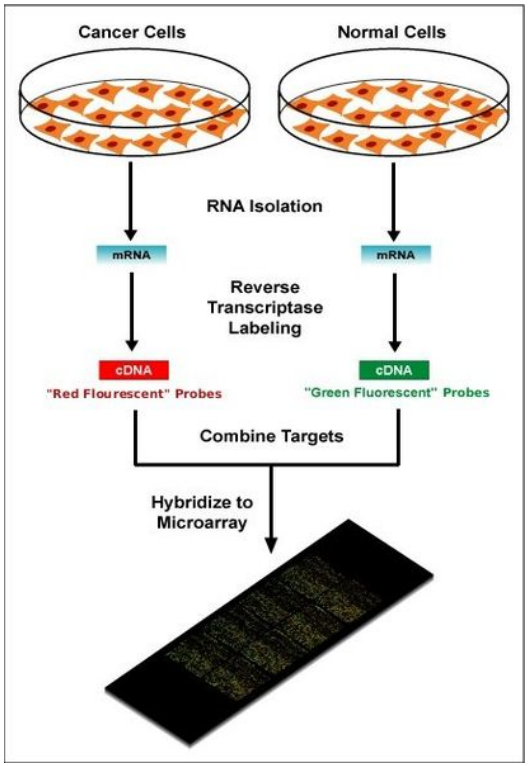


TTAAGTCGTACCCGTGTACGGGCGC Target sequence

AATTCAGCATGGGCACATGCCCGCG Perfect match (PM) probe: measure signals

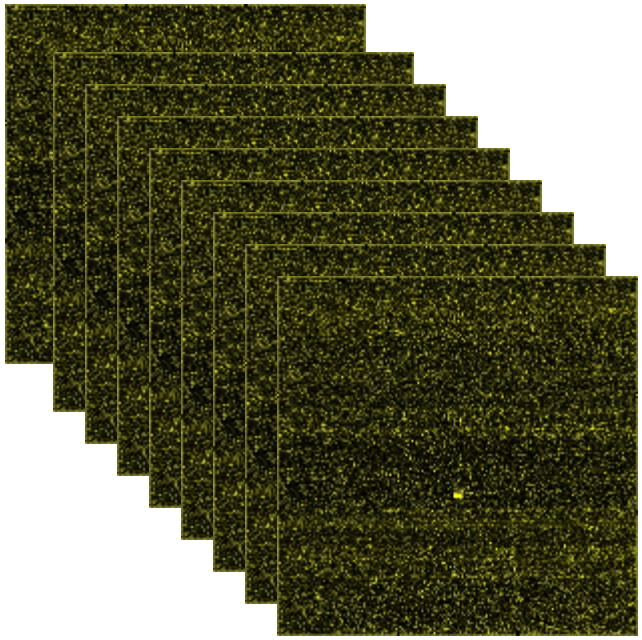
AATTCAGCATGGACACATGCCCGCG Mis-match (MM) probe: measure background

# One-color vs. two-color arrays




- Two-color (two-channel) arrays hybridize two samples on the same array with different colors (red and green).
  - Each spot produce two numbers.
  - Agilent, Nimblegen
- One-color (single-channel) arrays hybridize one sample per array.
  - Easier when comparing multiple groups.
  - Have to use twice as many arrays.
  - Affymetrix, Illumina.

# Data from microarray



- Data are fluorescent intensities:
  - extracted from the images with artifacts (e.g., cross-talk) removed, which Involves many statistical methods.
  - Final data are stored in a matrix: row for probes, column for samples.
  - For each sample, each probe has one number from one-color arrays and two numbers for tw-color arrays.

| | sample1 | sample2 | sample3 | sample4 |
|---|---|---|---|---|
| 1007_s_at | 8.575758 | 8.915618 | 9.150667 | 8.967870 |
| 1053_at | 6.959002 | 7.039825 | 6.898245 | 7.136316 |
| 117_at | 7.738714 | 7.618013 | 7.499127 | 7.610726 |
| 121_at | 10.114529 | 10.018231 | 10.003332 | 9.809068 |
| 1255_g_at | 5.056204 | 4.759066 | 4.629297 | 4.673458 |
| 1294_at | 8.009337 | 7.980694 | 8.343183 | 8.025335 |
| 1316_at | 6.899290 | 7.045843 | 6.976185 | 7.063050 |
| 1320_at | 7.218898 | 7.600437 | 7.433031 | 7.201984 |
| 1405_i_at | 6.861933 | 6.042179 | 6.165090 | 6.200671 |
| 1431_at | 5.073265 | 5.114023 | 5.159933 | 5.063821 |
| ... | | | | |

# Statistical challenges

- Data normalization: remove systematic technical artifacts.
  - Within array: variations of probe intensities are caused by:
    - cross-hybridization: probes capture the "wrong" target.
    - probe sequence: some probes are "sticker".
    - others: spot sizes, smoothness of array surface, etc.
  - Between array: intensity-concentration response curve can be different from different arrays, caused by variations in sample processing, image reader, etc.
- Summarization of gene expressions:
  - summarize values for multiple probes belonging to the same gene into one number.
- Differential expression detection:
  - Find genes that are expressed differently between different experimental conditions, e.g., cases and controls.

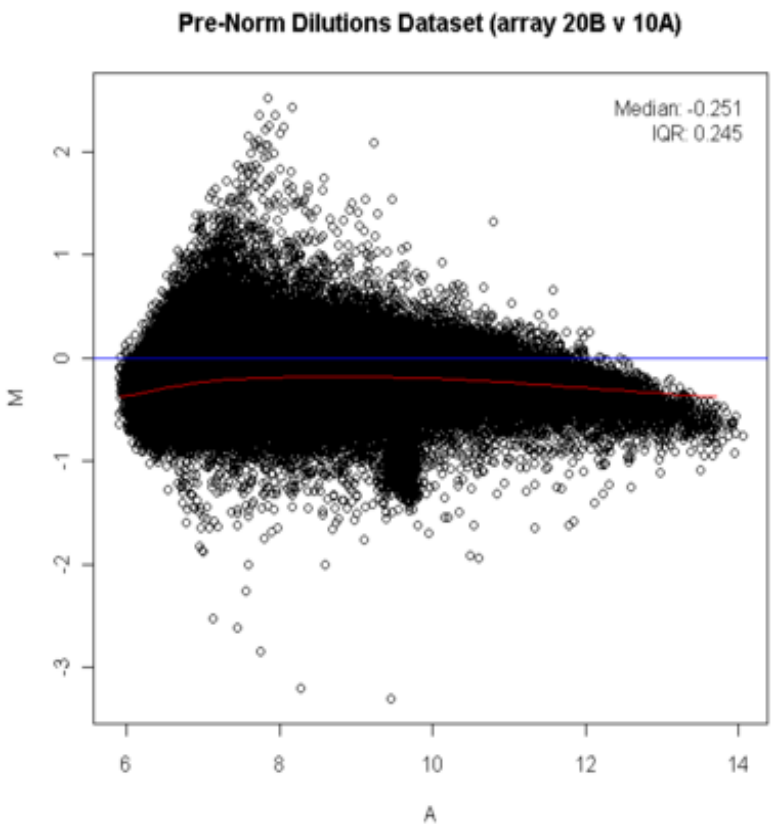
# Gene expression microarray data preprocessing

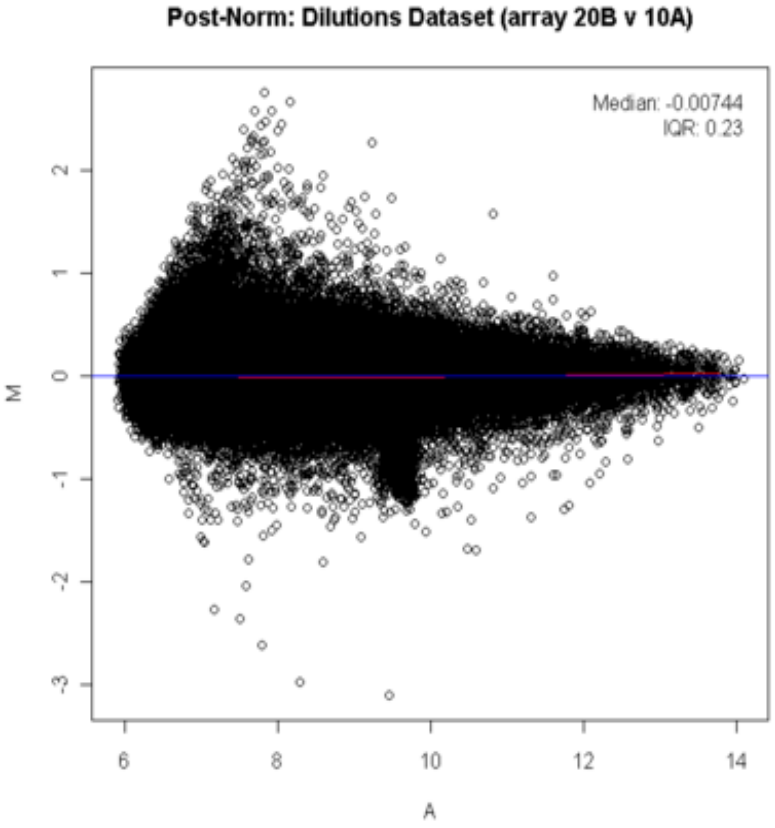
# Within array normalization: two-color arrays

Normalization based on M vs. A plot, or MA plot: (Yang et al. 2002, *Nucleic Acids Research*).

- For spot *i*, let $R_i$ and $G_i$ be the intensities, define:
  - $M_i=log_2R_i-log_2G_i$, A=$(log_2R_i+log_2G_i)/2$.
  - M measures relative expression, A measures total expression.
- MA plots is used to visualize relative vs. total expression dependence.
- The normalization is based on the assumption that M and A are independent.
- Normalization procedure:
  - Fit a smooth curve of M vs. A using loess, e.g., *M=c(A)+epsilon*, c(.) is smooth.
  - $M_{norm}=M-c(A)$

# Loess normalization: before and after



# Within array normalization: one-color arrays

- RMA (Robust Multi-array Average) background model (Irizarry et al. 2003, *Biostatistics*)
- For each array, assume:

  $PM = S + B$

  Signal: $S \sim Exp(\lambda)$

  Background: $B \sim N(\mu, \sigma^2)$ left-truncated at zero
- Observed: PM; of interest: S.
- Predict S from PM using $E[S \mid PM]$
- Full derivation at

  http://www.biochem.ucl.ac.uk/~harry/MAD/rma_bg.pdf

# An extension: GCRMA

$$
\begin{aligned}
Y_{gij} &= O_{gij} + N_{gij} + S_{gij} \\
&= O_{gij} + \exp(\mu_{gij} + \varepsilon_{gij}) + \exp(s_g + \delta_g X_i + a_{gij} + b_i + \xi_{gij}).
\end{aligned}
$$

Here $Y_{gij}$ is the *PM* intensity for the probe $j$ in probeset $g$ on array $i$, $\varepsilon_{gij}$ is a normally distributed error that account for NSB for the same probe behaving differently in different arrays, $s_g$ represents the baseline log expression level for probeset $g$, $a_{gij}$ represents the signal detecting ability of probe $j$ in gene $g$ on array $i$, $b_i$ is a term used to describe the need for normalization, $\xi_{gij}$ is a normally distributed term that accounts for the multiplicative error, and $\delta_g$ is the expected differential expression for every unit difference in covariate $X$. Notice $\delta_g$ is the parameter of interest. As described by Naef and Magnasco (2003) $a_{gj}$ is a function of $\alpha$.

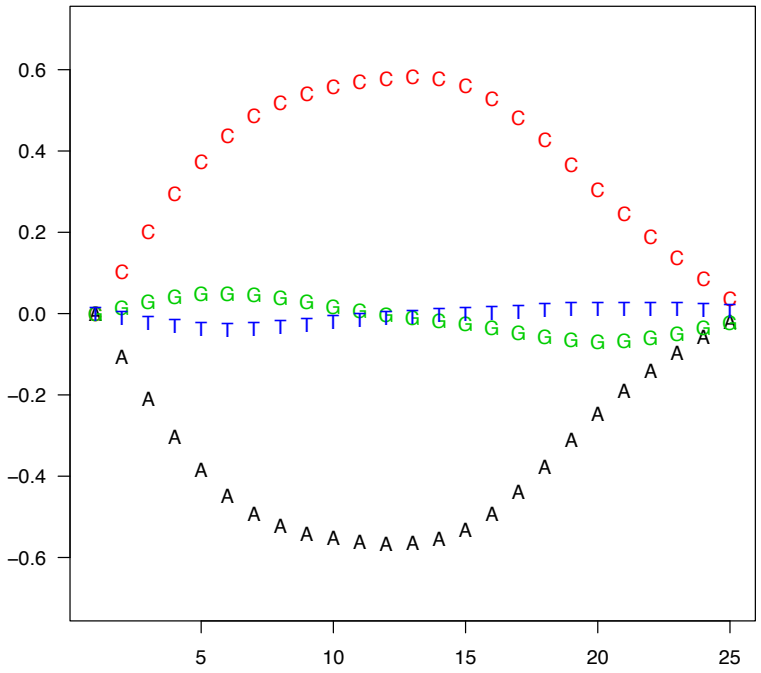
Wu et al. (2005) *JASA*

# Probe sequence effects

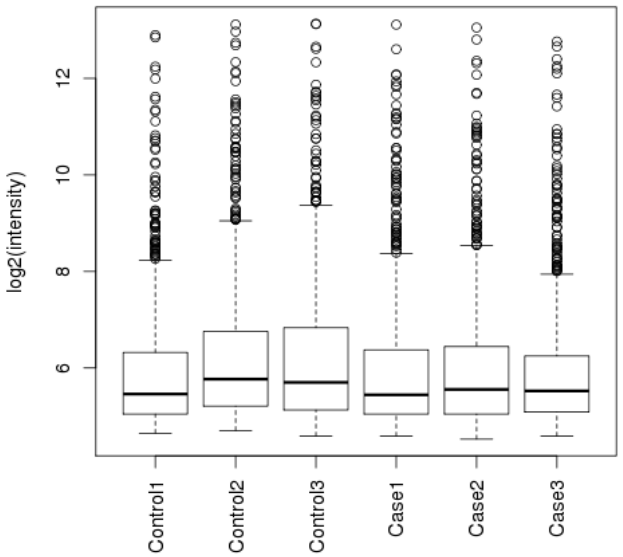
- Probe affinity is modeled as:

$$\alpha = \sum_{k=1}^{25} \sum_{j \in \{A,T,G,C\}} \mu_{j,k} 1_{b_k = j} \text{ with } \mu_{j,k} = \sum_{l=0}^{3} \beta_{j,l} k^l,$$

- This kind of modeling is widely used in other microarrays and sequencing data!



# Between array normalization




- Remember data from arrays (intensity values) estimate mRNA quantities, but the intensity-concentration response can be different from different arrays. So 5.5 on arrays 1 doesn't mean the same on array 2.
- This could be caused by:
  - Total amount of mRNA used
  - Properties of the agents used.
  - Array properties
  - Settings of laser scanners
  - etc.
- Goal: normalize so that data from different arrays are comparable!

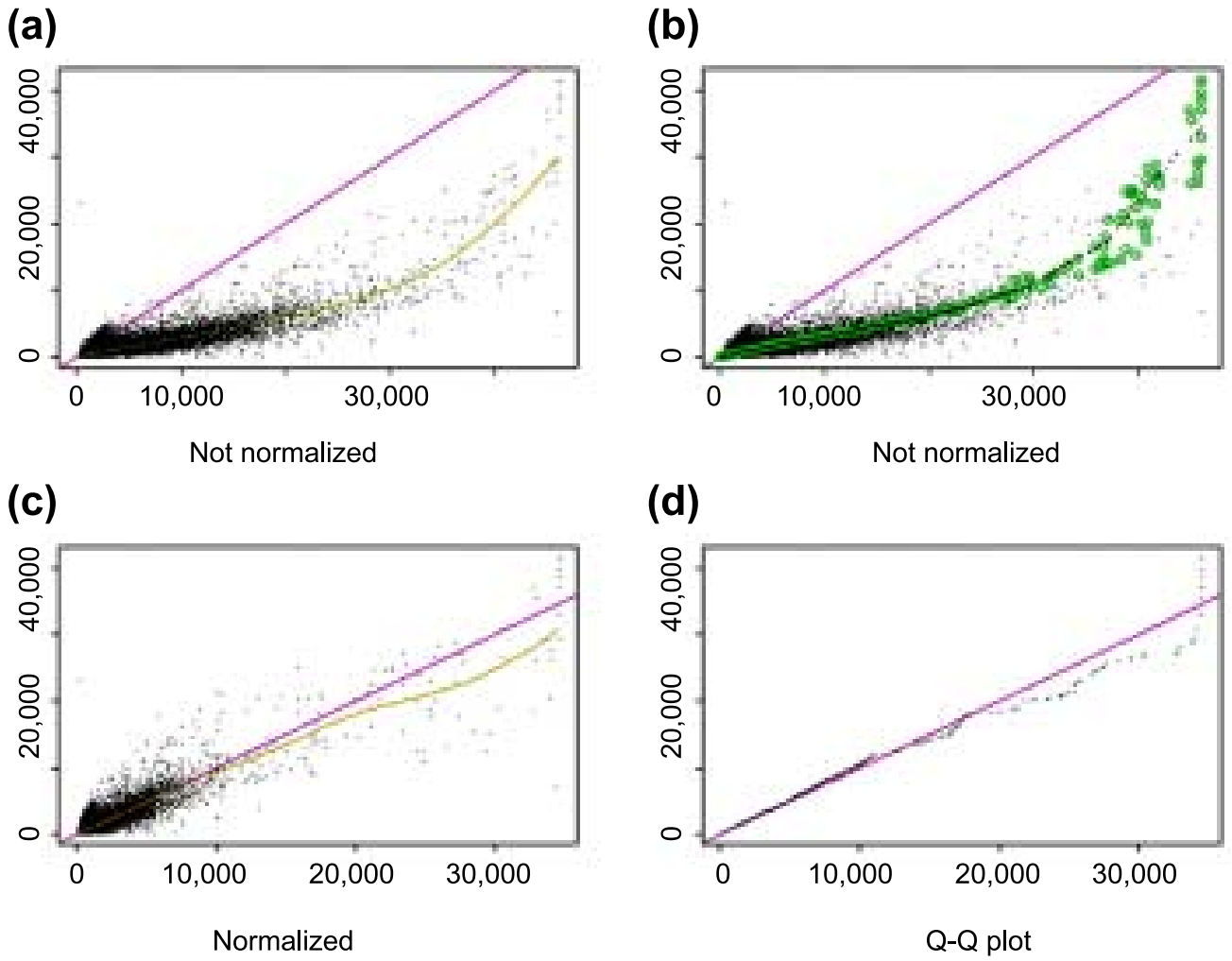
# Linear scaling method

- Used in Affymetrix software MAS:
  - Use a number of “housekeeping” genes and assume their expressions are identical across all arrays.
  - Shift and rescale all data so the average expression of these genes are the same across all arrays.

# Non-linear smoothing based

- Implemented in dChip (Li and Wong 2001, *Genome Bio*.)
  - Find a set of genes invariant across arrays.
  - Find a “baseline” array
  - For every other arrays fit a smooth curve on expressions of invariant genes
  - Normalize based on the fitted curve.

# dChip normalization



# Quantile normalization

Proposed in Bolstad *et al.* 2003, *Bioinformatics*:

- Force the distribution of all data from all arrays to be the same, but keep the ranks of the genes.
- Procedures:
  1. Create a target distribution, usually use the average from all arrays.
  2. For each array, match its quantiles to that of the target. To be specific: $x_{norm} = F_2^{-1}(F_1(x))$:
     - *x:* value in the chip to be normalized
     - $F_1$*:* distribution function in the array to be normalized
     - $F_2$*:* target distribution function

# A simple example for quantile normalization

| Gene | sample1 | Sample2 | Sample3 | Sample4 |
|---|---|---|---|---|
| 1 | 8 | 15 | 9 | 13 |
| 2 | 7 | 2 | 7 | 15 |
| 3 | 3 | 6 | 5 | 8 |
| 4 | 1 | 5 | 2 | 9 |
| 5 | 9 | 13 | 6 | 11 |

1. Find the Smallest Value for each sample

| Gene | sample1 | Sample2 | Sample3 | Sample4 |
|---|---|---|---|---|
| 1 | 8 | 15 | 9 | 13 |
| 2 | 7 | **2** | 7 | 15 |
| 3 | 3 | 6 | 5 | **8** |
| 4 | **1** | 5 | **2** | 9 |
| 5 | 9 | 13 | 6 | 11 |

2. Average them

(1+2+2+8)/4=3.25

# Replace Each Value by the Average

| Gene | sample1 | Sample2 | Sample3 | Sample4 |
|---|---|---|---|---|
| 1 | 8 | 15 | 9 | 13 |
| 2 | 7 | 3.25 | 7 | 15 |
| 3 | 3 | 6 | 5 | 3.25 |
| 4 | 3.25 | 5 | 3.25 | 9 |
| 5 | 9 | 13 | 6 | 11 |

# Find the Next Smallest Values, then average

| Gene | sample1 | Sample2 | Sample3 | Sample4 |
|---|---|---|---|---|
| 1 | 8 | 15 | 9 | 13 |
| 2 | 7 | 3.25 | 7 | 15 |
| 3 | 3 | 6 | 5 | 3.25 |
| 4 | 3.25 | 5 | 3.25 | 9 |
| 5 | 9 | 13 | 6 | 11 |

(3+5+5+9)/4=5.5

# Replace Each Value by the Average

| Gene | sample1 | sample2 | sample3 | sample4 |
|---|---|---|---|---|
| 1 | 8 | 15 | 9 | 13 |
| 2 | 7 | 3.25 | 7 | 15 |
| 3 | 5.50 | 6 | 5.50 | 3.25 |
| 4 | 3.25 | 5.50 | 3.25 | 5.50 |
| 5 | 9 | 13 | 6 | 11 |

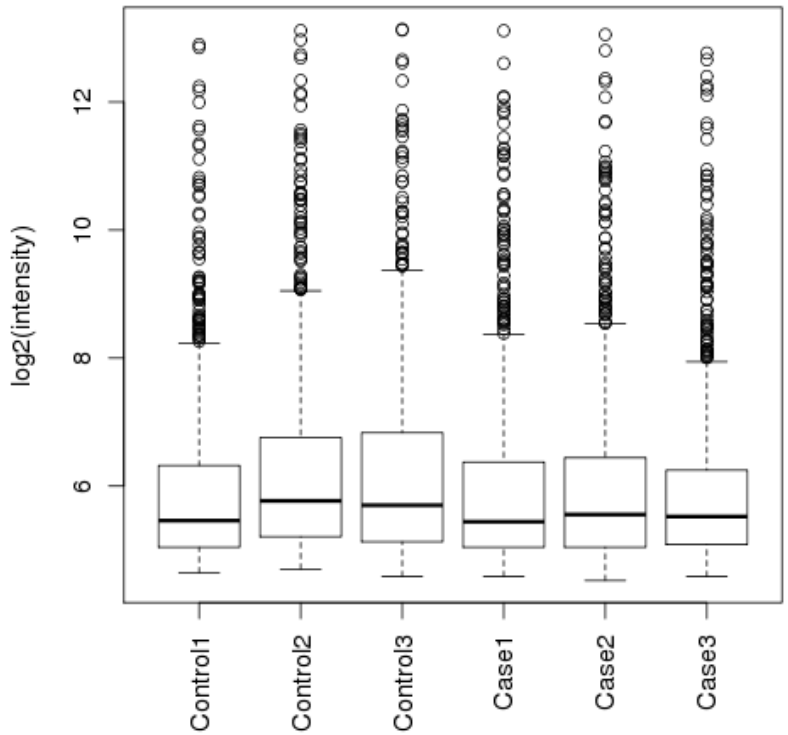
Continue the process, we get the following matrix after finishing:

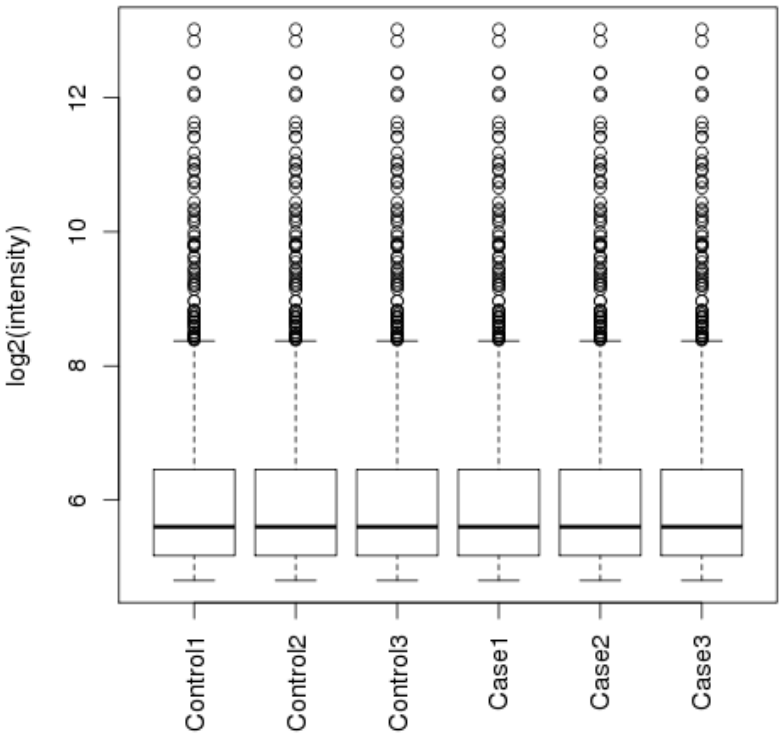
| Gene | sample1 | sample2 | sample3 | sample4 |
|---|---|---|---|---|
| 1 | 10.25 | 12.00 | 12.00 | 10.25 |
| 2 | 7.50 | 3.25 | 10.25 | 12.00 |
| 3 | 5.50 | 7.50 | 5.50 | 3.25 |
| 4 | 3.25 | 5.50 | 3.25 | 5.50 |
| 5 | 12.00 | 10.25 | 7.50 | 7.50 |

The result matrix has following properties:

- The values taken in each column are exactly the same.
- The ranks of genes in each column are the same as before normalization.

# Before/after QN boxplot



# Microarray data summarization

- There are usually multiple probes corresponding to a gene. The task is to summarize the readings from these probes into one number to represent the gene expression.
- Naïve methods: mean, median.
- From MAS 5.0: use one-step Tukey Biweight (TBW) to obtain a robust weighted mean that is resistant to outliers.
  - Probes with intensities far away from median will have smaller weights in the average.

# RMA summarization

$$Y_{ijn} = \mu_{in} + \alpha_{jn} + \varepsilon_{ijn}, i = 1, \ldots, I, j = 1, \ldots, J, n = 1, \ldots, n$$

log transformed $PM$ intensities, denoted with $Y$

$\mu_i$ representing the log scale expression level for array $i$,

$\alpha_j$ a probe affinity effect,

each probe set $n$

- Borrow information from multiple samples to estimate probe effects.
- Model-fitting: Median Polish (robust against outliers)
  - Iteratively removing the row and column medians until convergence
  - The remainder is the residual;
  - After subtracting the residual, the row medians are the estimates of the expression, and column medians are probe effects.

Irizarry et al. (2003) *Biostatistics*.

# Detect differentially expressed genes

# Test for differential expression (DE)

- To compare two groups, the easiest is to perform two group t-test gene by gene, with multiple-testing corrected. But,
  - T-statistics from two-group t-test is: $(\overline{X} - \overline{Y}) / \sqrt{S_X^2 / N_X + S_Y^2 / N_Y}$
  - By chance some genes have very small variance, which result in large t-statistics and tiny p-values even when the difference is small.
  - This motivates many different versions of modified t-test, empirical Bayes and variance shrinkage methods.

# SAM t-test

- Add a small constant to the denominator in calculating t statistics:

$$d(i) = \frac{\bar{x}_{\mathrm{I}}(i) - \bar{x}_{\mathrm{U}}(i)}{s(i) + s_0}$$

- Coefficient of variation of *d(i)* is computed as a function of *s(i)* in moving window across the data.
- $S_0$ is chosen to minimize that coefficient of variation.
- Bioconductor package `siggenes`.

Tusher *et al.* (2001) *PNAS*

# Empirical Bayes method from `limma`

- Use a Bayesian hierarchical model in multiple regression setting.
- Borrow information from all genes to estimate gene specific variances.
  - As a result, variance estimates will be "shrunk" toward the mean of all variances. So very small variance scenarios will be alleviated.
- Implemented in Bioconductor package "limma".

Smyth *et al.* (2004) *Statistical Applications in Genetics and Molecular Biology*

Let $\beta_{gj}$ be coefficient for gene g, factor j, assume

$$\hat{\beta}_{gj} \mid \beta_{gj}, \sigma_g^2 \sim N(\beta_{gj}, v_{gj}\sigma_g^2) \qquad s_g^2 \mid \sigma_g^2 \sim \frac{\sigma_g^2}{d_g}\chi^2_{d_g}$$ with priors:

$$P(\beta_{gj} \neq 0) = p_j. \quad \beta_{gj} \mid \sigma_g^2, \beta_{gj} \neq 0 \sim N(0, v_{0j}\sigma_g^2). \quad \frac{1}{\sigma_g^2} \sim \frac{1}{d_0 s_0^2}\chi^2_{d_0}.$$

Posterior variance estimator: $\tilde{s}_g^2 = \dfrac{d_0 s_0^2 + d_g s_g^2}{d_0 + d_g}.$

Moderated t-statistics for testing $\beta_{gj}$=0 : $\tilde{t}_{gj} = \dfrac{\hat{\beta}_{gj}}{\tilde{s}_g \sqrt{v_{gj}}}.$

# Correct for multiple testing

- Multiple testing problem is severe in high throughput data analysis because a large number of tests were performed.
  - Under type I error $\alpha$=0.05, 1000 out of 20000 genes will be falsely declared DE (false positive)by chance.
  - If there are a total of 2000 genes declared DE, the false discovery rate (FDR) is 0.5!
- Multiple testing correction
  - Bonferroni correction: use $\alpha$=0.05/20000 (too conservative).
  - FDR control (Benjamini and Hochberg, 1995 JRSS-B)

# After DE detection

- Identifying DE genes is not the end of the game. Possible downstream analysis:
  - Functional analysis of DE genes: GO (Gene Ontology) analysis.
  - Clustering.

# Bioconductor packages for microarray analysis

# Bioconductor for microarray data

- There're a rich collection of bioc packages for microarrays. In fact, Bioconductor started for microarray analysis.
- There are currently 228 packages for microarray.
- Important ones include:
  - affy: one of the earliest bioc packages. Designed for analyzing data from Affymetrix arrays.
  - limma and siggenes: DE detection using limma and SAM-t model.
  - oligo: preprocessing tools for many types of oligonecleotide arrays. This is designed to replace affy package.
  - Many annotation data package to link probe names to genes.

# My suggestion

- Use `oligo` to reading in data, normalization and summarization.
- Use `siggenes` or `limma` for detecting DE genes.

# An exmple of Analyzing a set of Affymetrix data

- Data generated by MAQC (MicroArray Quality Control) project.
- Five brain samples and five reference samples on human exon arrays.
- Raw data are CEL files (binary file generated by factory).
- Each CEL file is around 65Mb.
- The platform design package (pd.huex.1.0.st.v2) needs to be installed.

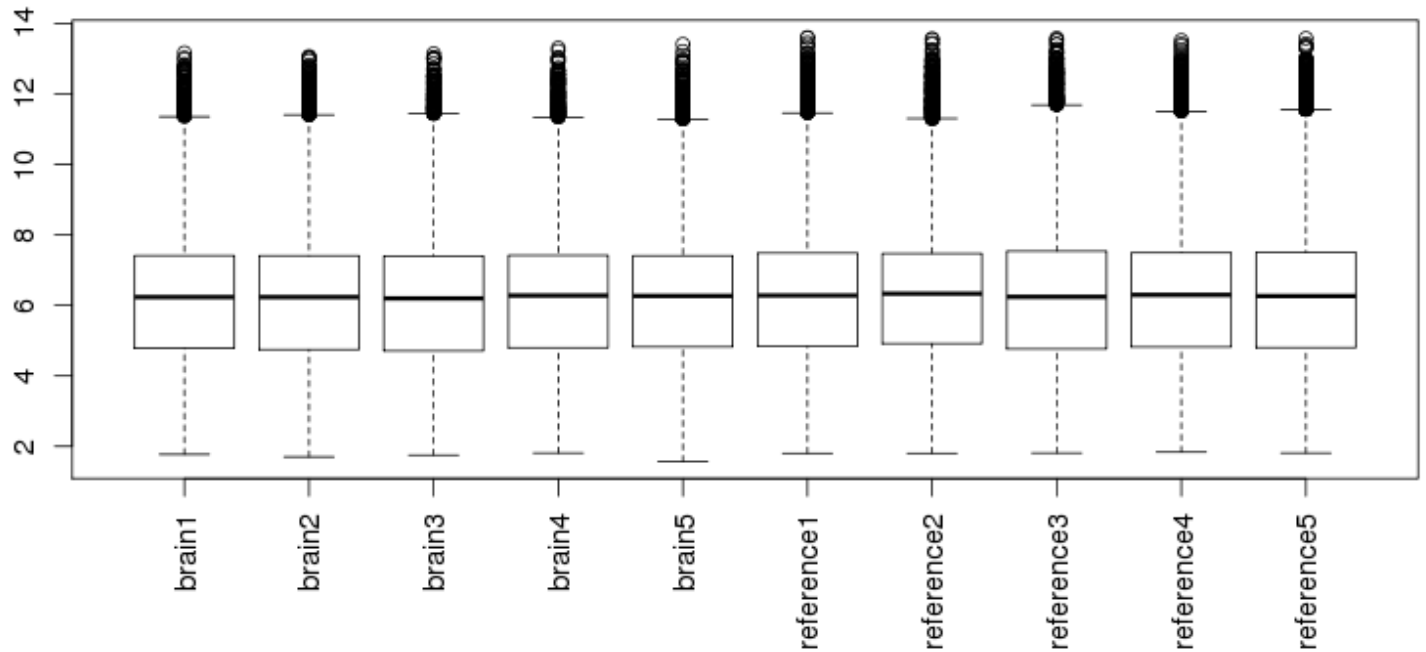
# Read in data

```
## load in necessary libraries
> library(oligo)
> library(limma)
## get a list of CEL files
> CELfiles=dir(pattern="CEL")
## read in all raw data
> rawdata=read.celfiles(CELfiles)
> rawdata
ExonFeatureSet (storageMode: lockedEnvironment)
assayData: 6553600 features, 10 samples
  element names: exprs
protocolData
  rowNames: ambion_A1.CEL, ambion_A2.CEL, ..., stratagene_K2.CEL
(10 total)
...
Annotation: pd.huex.1.0.st.v2
```

# Normalization and summarization

```
## using RMA
> normdata=rma(rawdata, target = "core")
> normdata
ExpressionSet (storageMode: lockedEnvironment)
assayData: 22011 features, 10 samples
  element names: exprs
...
## extract expression values using expr function
> data=exprs(normdata)
> head(data)
           sample 1  sample 2  sample 3  sample 4
1007_s_at 10.160224 10.214496 10.090697 11.020649
1053_at    9.501826  9.500412  9.574311  7.361141
117_at     5.669447  5.478072  5.648788  6.048142
121_at     8.061479  8.154549  8.156215  7.902597
1255_g_at  4.307739  4.017903  3.992333  4.668972
1294_at    7.108730  7.185586  7.122404  6.597161
```

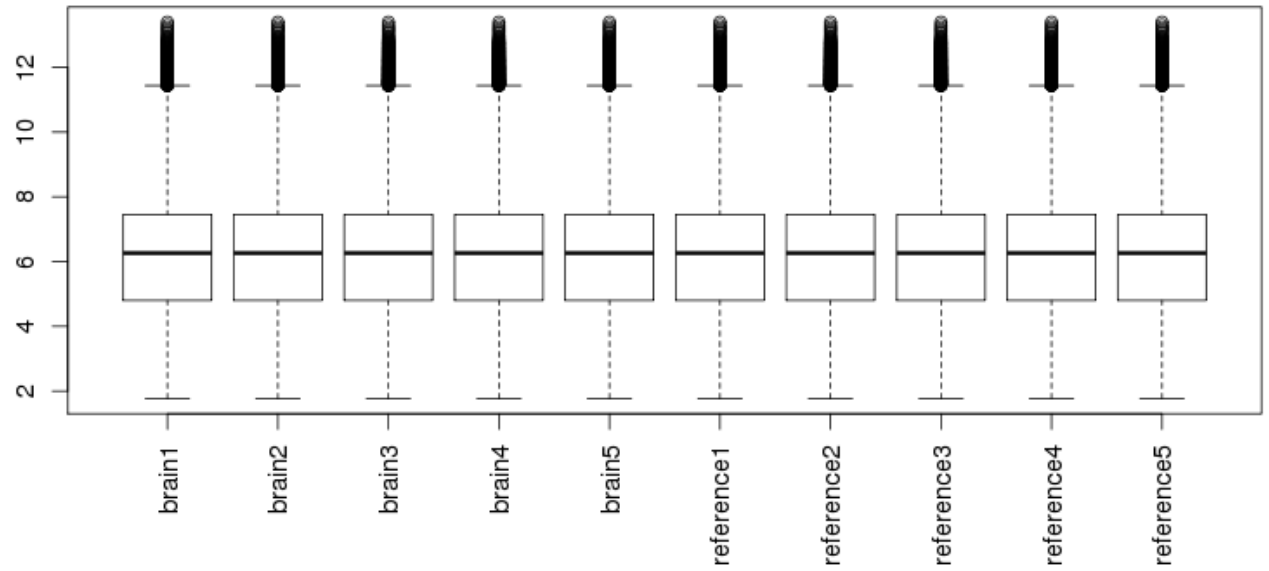
```
## check data distribution after RMA
> boxplot(data)
```



The boxplot looks really good after RMA, so between array normalization is unnecessary. But in case you need it, use `normalizeQuantiles` function from `limma` for quantile normalization :

```
> data2=normalizeQuantiles(data)
```

Now the new boxplot after quantile normalization:



# DE detection using SAM t-test

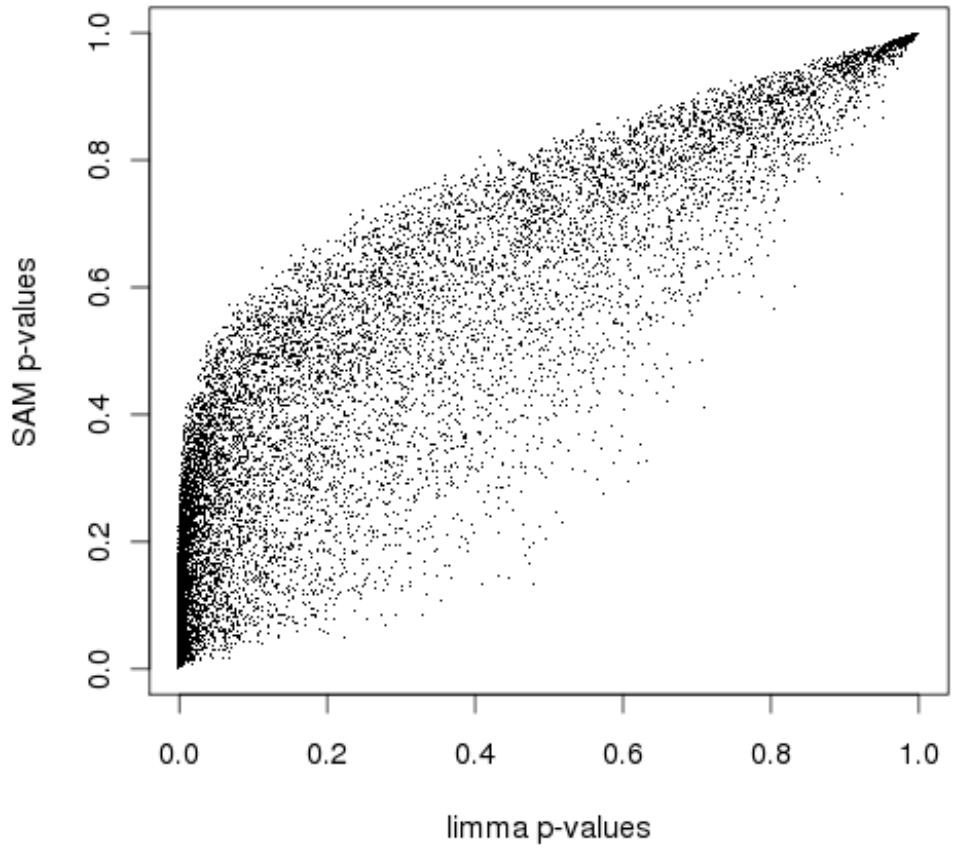
```
> library(siggenes)
## create a vector for design.
> design <- c(rep(0,5),rep(1,5))
> sam.result=sam(data2, cl=design)
> sam.result
SAM Analysis for the Two-Class Unpaired
Case Assuming Unequal Variances
```

# DE detection using limma

```
## create design matrix. Intercept must be included
> design=cbind(mu=1,beta=c(rep(0,5),rep(1,5)))
## fit linear model and compute estimates
> limma.result=lmFit(data2, design=design)
## Empirical Bayes method to get p-values
> limma.result=eBayes(limma.result)
## get p-values for the comparison
> pval=limma.result$p.value[,"beta"]
```

# Compare results from limma and SAM

- Agreement is good, 0.95 Spearman rank correlation.
- Limma seems to be more liberal.



# Obtain gene annotations

- Now you get p-values for all genes, but you also need gene names for generating report. This could be very troublesome!
- Easiest: use `getNetAffx` from `oligo` (but it's not always working):

```
> anno=getNetAffx(normdata, "transcript")
> genes=pData(anno)$geneassignment
```

- Alternatively, there are many annotation packages available for different array platforms. For example, hgu133a.db is for HGU133A arrays.
- These packages contain comprehensive information for all probes, including their sequences, chromosome, position, corresponding gene IDs, GO terms, etc.
- A typical way to convert probeset names to accession number or gene alias is:

```
> library(hgu133a.db)
## convert to accession numbers:
> geneAcc=as.character(hgu133aACCNUM[rownames(data)])
## convert to gene names
> geneNames=as.character(hgu133aSYMBOL[rownames(data)])
```

# Finally generate a report table

```
> ix=sam.result@q.value<0.1
> result=data.frame(gene=geneNames[ix],
    pvalue=sam.result@p.value[ix],
    fold=sam.result@fold[ix])
## sort by fold change
> ix2=sort(result$fold, decreasing=TRUE, index.return=TRUE)$ix
> result=result[ix2,]
> head(result)
                  gene pvalue     fold
2731192    NM_000477      0 185.5720
3457336    NM_006928      0 155.7143
2772566    NM_144646      0 152.8232
2731230    NM_001134      0 132.8515
> write.table(result, file="report.txt", sep="\t")
```

# Review

- We have covered microarray analysis, including:
  - Data preprocessing: within and between array normalization.
  - Summarization.
  - DE detection.

# To do list

- Review the slides.
- Read the review article on *Nature Reviews Genetics* (link on the class webpage).