Exploratory Data Analysis

Ian Donaldson
MVB-INF 4410/9410
September, 2010

This talk is a minor modification of the one given by Raphael Gottardo as part of the Canadian Bioinformatics Workshops course on ”Essential Statistics: Getting the numbers right”. The original material is available from http://bioinformatics.ca/workshops/2009/course-content

bioinformatics.ca

Essential Statistics in Biology: Getting the Numbers Right

Raphael Gottardo
Clinical Research Institute of Montreal (IRCM)
raphael.gottardo@ircm.qc.ca
http://www.rglab.org

Outline

- Exploratory Data Analysis
- 1-2 sample *t*-tests, multiple testing
- Clustering
- SVD/PCA
- Frequentists vs. Bayesians

The above lectures (as videos and powerpoints) are available from http://bioinformatics.ca/workshops/2009/course-content

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA)

- What is EDA?
- Basics of EDA: Boxplots, Histograms, Scatter plots, Transformations, QQ-plot
- Applications to microarray data

This talk is not (really) about statistics or distributions or R or microarray data.

It's about…..

Looking at your data

What is EDA?

- Statistical practice concerned with (among other things): uncover underlying structure, extract important variables, detect outliers and anomalies, test underlying assumptions, develop models
- Named by John Tukey
- Extremely Important

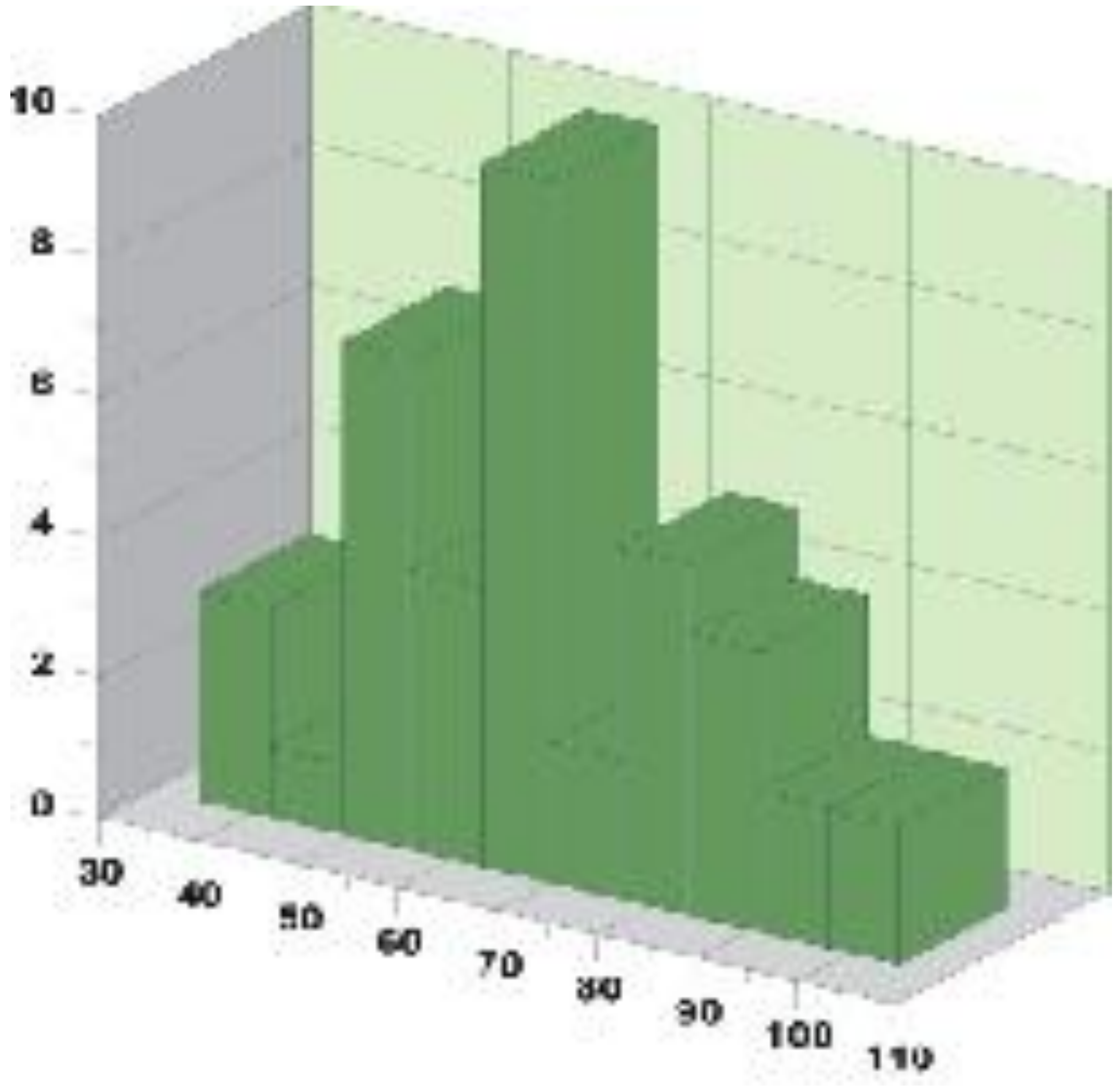
EDA techniques

- Mostly graphical
- Plotting the raw data (histograms, scatterplots, etc.)
- Also, plotting simple statistics such as means, standard deviations, medians, box plots, etc
- Positioning such plots so as to maximize our natural pattern-recognition abilities
- A **clear** picture is worth a thousand words!

A few tips

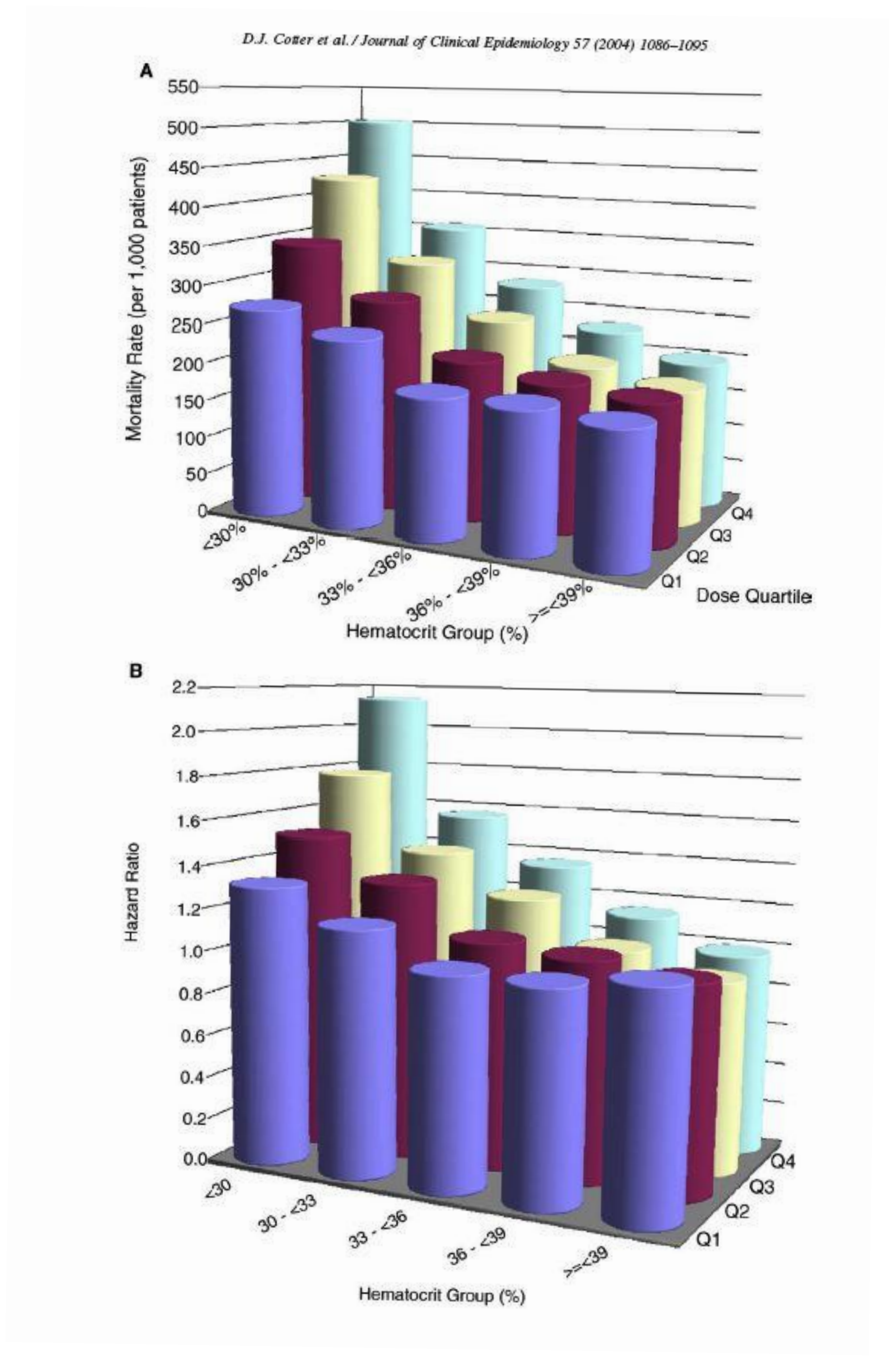
- Avoid 3-D graphics
- Don’t show too much information on the same graph (color, patterns, etc)
- Stay away from Excel, Excel is not a statistics package!
- R provides a great environment for EDA with good graphics capabilities

A few bad plots



Unecessary third dimension

A few bad plots



A 2D plot with four lines would be clearer

A few bad plots

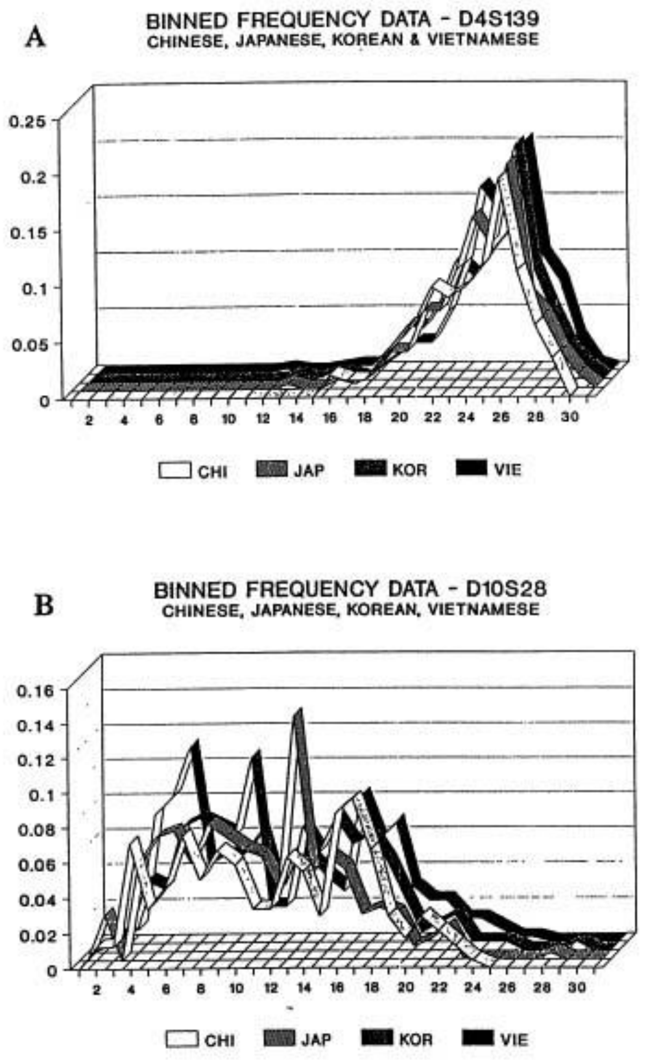


FIG. 4. *Fixed bin distribution (histogram) for two loci and four Asian subpopulations (used with permission from John Hartmann): the boundaries of the 30 bins (vertical axis) are determined by the FBI; these bins are not of equal length. Sample sizes (numbers of individuals) for Chinese, Japanese, Korean and Vietnamese are* 103, 125, 93 *and* 215 *for* D4S139 *and* 120, 137, 100 *and* 193 *for* D10S28. *The horizontal axis is the bin number; bins are not of equal length.*

A few bad plots

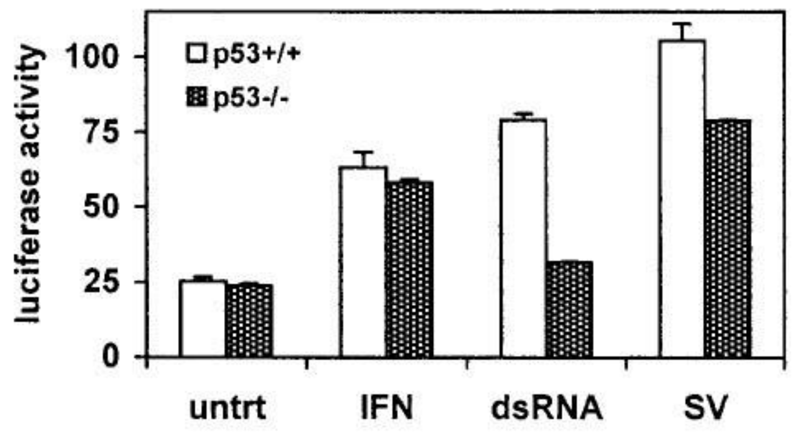


FIG. 4. ISG15 promoter activity mimics endogenous ISG15 mRNA regulation by p53, dsRNA, and virus. Cells (6×10^5 HCT 116) were seeded in 32-mm plates and allowed to attach overnight. Cells were transfected with 500 ng of pGL3/ISG15-Luc, 50 ng of pRL null (Promega), and 450 ng of pcDNA3 for carrier DNA by using Lipofectamine Plus (Life Technologies) following the manufacturer's instructions. Twenty-four hours posttransfection, the medium was aspirated and replaced with medium containing either 1,000 U of IFN-α/ml, 50 μg of dsRNA/ml, or Sendai virus (multiplicity of infection, 10). Cells were incubated for 12 h and then lysed, and luciferase assays were performed. Luciferase activity was assessed on 20 μl of each lysate as directed by the supplier (Dual Luciferase Kit, Promega) using a TD 20/20 luminometer (Turner Designs). Luciferase activity is presented as the ratio of firefly activity to renilla activity to control for differences in transfection efficiency. Each data point is the mean of triplicate samples ± the standard error; the data presented are representative of four independent experiments.

Only three replicates - just showing the numbers would be clearer and more accurate.

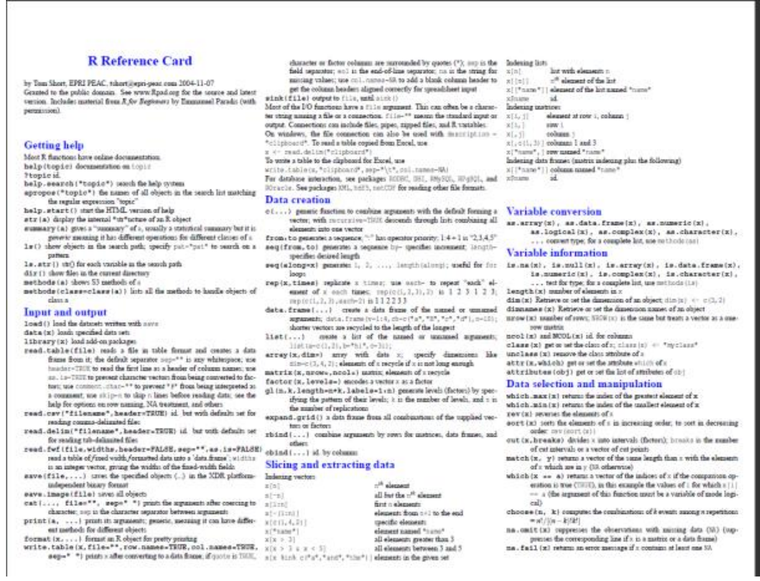
Show error bars above and below.

Want more?
Try http://www.biostat.wisc.edu/~kbroman/topten_worstgraphs/

What is R?

- R (http://www.r-project.org). R is a language and environment for statistical computing and graphics
- Provide many statistical techniques
- R provides a great environment for EDA with great graphics capabilities
- Open source
- Highly extensible (e.g. CRAN, Bioconductor)

R



R Reference Card

by Tom Short, EPRI PEAC, tshort@epri-peac.com 2004-11-07
Granted to the public domain. See www.Rpad.org for the source and latest version. Includes material from *R for Beginners* by Emmanuel Paradis (with permission).

Getting help

Most R functions have online documentation.
help(topic) documentation on topic
?topic id.
help.search("topic") search the help system
apropos("topic") the names of all objects in the search list matching the regular expression "topic"
help.start() start the HTML version of help
str(a) display the internal *str*ucture of an R object
summary(a) gives a "summary" of a, usually a statistical summary but it is *generic* meaning it has different operations for different classes of a
ls() show objects in the search path; specify pat="pat" to search on a pattern
ls.str() str() for each variable in the search path
dir() show files in the current directory
methods(a) shows S3 methods of a
methods(class=class(a)) lists all the methods to handle objects of class a

Input and output

load() load the datasets written with save
data(x) loads specified data sets
library(x) load add-on packages
read.table(file) reads a file in table format and creates a data frame from it; the default separator sep="" is any whitespace; use header=TRUE to read the first line as a header of column names; use as.is=TRUE to prevent character vectors from being converted to factors; use comment.char="" to prevent "#" from being interpreted as a comment; use skip=n to skip n lines before reading data; see the help for options on row naming, NA treatment, and others
read.csv("filename",header=TRUE) id. but with defaults set for reading comma-delimited files
read.delim("filename",header=TRUE) id. but with defaults set for reading tab-delimited files
read.fwf(file,widths,header=FALSE,sep="",as.is=FALSE) read a table of *f*ixed *w*idth *f*ormatted data into a 'data.frame'; widths is an integer vector, giving the widths of the fixed-width fields
save(file,...) saves the specified objects (...) in the XDR platform-independent binary format
save.image(file) saves all objects
cat(..., file="", sep=" ") prints the arguments after coercing to character; sep is the character separator between arguments
print(a, ...) prints its arguments; generic, meaning it can have different methods for different objects
format(x,...) format an R object for pretty printing
write.table(x,file="",row.names=TRUE,col.names=TRUE, sep=" ") prints x after converting to a data frame; if quote is TRUE, character or factor columns are surrounded by quotes ("); sep is the field separator; eol is the end-of-line separator; na is the string for missing values; use col.names=NA to add a blank column header to get the column headers aligned correctly for spreadsheet input
sink(file) output to file, until sink()
Most of the I/O functions have a file argument. This can often be a character string naming a file or a connection. file="" means the standard input or output. Connections can include files, pipes, zipped files, and R variables.
On windows, the file connection can also be used with description = "clipboard". To read a table copied from Excel, use
x <- read.delim("clipboard")
To write a table to the clipboard for Excel, use
write.table(x,"clipboard",sep="\t",col.names=NA)
For database interaction, see packages RODBC, DBI, RMySQL, RPgSQL, and ROracle. See packages XML, hdf5, netCDF for reading other file formats.

Data creation

c(...) generic function to combine arguments with the default forming a vector; with recursive=TRUE descends through lists combining all elements into one vector
from:to generates a sequence; ":" has operator priority; 1:4 + 1 is "2,3,4,5"
seq(from,to) generates a sequence by= specifies increment; length= specifies desired length
seq(along=x) generates 1, 2, ..., length(along); useful for for loops
rep(x,times) replicate x times; use each= to repeat "each" element of x each times; rep(c(1,2,3),2) is 1 2 3 1 2 3; rep(c(1,2,3),each=2) is 1 1 2 2 3 3
data.frame(...) create a data frame of the named or unnamed arguments; data.frame(v=1:4,ch=c("a","B","c","d"),n=10); shorter vectors are recycled to the length of the longest
list(...) create a list of the named or unnamed arguments; list(a=c(1,2),b="hi",c=3i);
array(x,dim=) array with data x; specify dimensions like dim=c(3,4,2); elements of x recycle if x is not long enough
matrix(x,nrow=,ncol=) matrix; elements of x recycle
factor(x,levels=) encodes a vector x as a factor
gl(n,k,length=n*k,labels=1:n) generate levels (factors) by specifying the pattern of their levels; k is the number of levels, and n is the number of replications
expand.grid() a data frame from all combinations of the supplied vectors or factors
rbind(...) combine arguments by rows for matrices, data frames, and others
cbind(...) id. by columns

Slicing and extracting data

Indexing vectors

x[n]	n^th element
x[-n]	all but the n^th element
x[1:n]	first n elements
x[-(1:n)]	elements from n+1 to the end
x[c(1,4,2)]	specific elements
x["name"]	element named "name"
x[x > 3]	all elements greater than 3
x[x > 3 & x < 5]	all elements between 3 and 5
x[x %in% c("a","and","the")]	elements in the given set

Indexing lists

x[n]	list with elements n
x[[n]]	n^th element of the list
x[["name"]]	element of the list named "name"
x$name	id.

Indexing matrices

x[i,j]	element at row i, column j
x[i,]	row i
x[,j]	column j
x[,c(1,3)]	columns 1 and 3
x["name",]	row named "name"

Indexing data frames (matrix indexing plus the following)

x[["name"]]	column named "name"
x$name	id.

Variable conversion

as.array(x), as.data.frame(x), as.numeric(x), as.logical(x), as.complex(x), as.character(x), ... convert type; for a complete list, use methods(as)

Variable information

is.na(x), is.null(x), is.array(x), is.data.frame(x), is.numeric(x), is.complex(x), is.character(x), ... test for type; for a complete list, use methods(is)
length(x) number of elements in x
dim(x) Retrieve or set the dimension of an object; dim(x) <- c(3,2)
dimnames(x) Retrieve or set the dimension names of an object
nrow(x) number of rows; NROW(x) is the same but treats a vector as a one-row matrix
ncol(x) and **NCOL(x)** id. for columns
class(x) get or set the class of x; class(x) <- "myclass"
unclass(x) remove the class attribute of x
attr(x,which) get or set the attribute which of x
attributes(obj) get or set the list of attributes of obj

Data selection and manipulation

which.max(x) returns the index of the greatest element of x
which.min(x) returns the index of the smallest element of x
rev(x) reverses the elements of x
sort(x) sorts the elements of x in increasing order; to sort in decreasing order: rev(sort(x))
cut(x,breaks) divides x into intervals (factors); breaks is the number of cut intervals or a vector of cut points
match(x, y) returns a vector of the same length than x with the elements of x which are in y (NA otherwise)
which(x == a) returns a vector of the indices of x if the comparison operation is true (TRUE), in this example the values of i for which x[i] == a (the argument of this function must be a variable of mode logical)
choose(n, k) computes the combinations of *k* events among *n* repetitions = $n!/[(n-k)!k!]$
na.omit(x) suppresses the observations with missing data (NA) (suppresses the corresponding line if x is a matrix or a data frame)
na.fail(x) returns an error message if x contains at least one NA

- R is for statistical analysis
- r-project.org
- Bioconductor is a biology specific R project
- bioconductor.org
- For serious, heavy-duty processing
- Steep learning curve with payoff’s
- *Tutorial tomorrow*

….but what about Excel

- We all do it
- Spread-sheet programme
- Limited statistical functions add-on
- Ability to create simple graphs
- Excellent for simpler work – does not scale well for larger processing
- Shallow learning curve (with lots more if you look)
- Use it for viewing, sorting and filtering data tables quickly.

Why you should not use MS Excel for statistics

- Read http://www.practicalstats.com/xlsstats/excelstats.html

- Limited statistical functions
- Misleading/wrong procedures
- Precision errors
- Graphing ”glitz”
- Excel is not evil – but know when not to use it and
- Dont box yourself into knowing only Excel

What to learn - summary

- Learn to use Excel well and appropriately
- Learn one other package
- R is optimal because you are likely to see it again
- There are a lot of other packages – consider using what people around you use.

Probability distributions

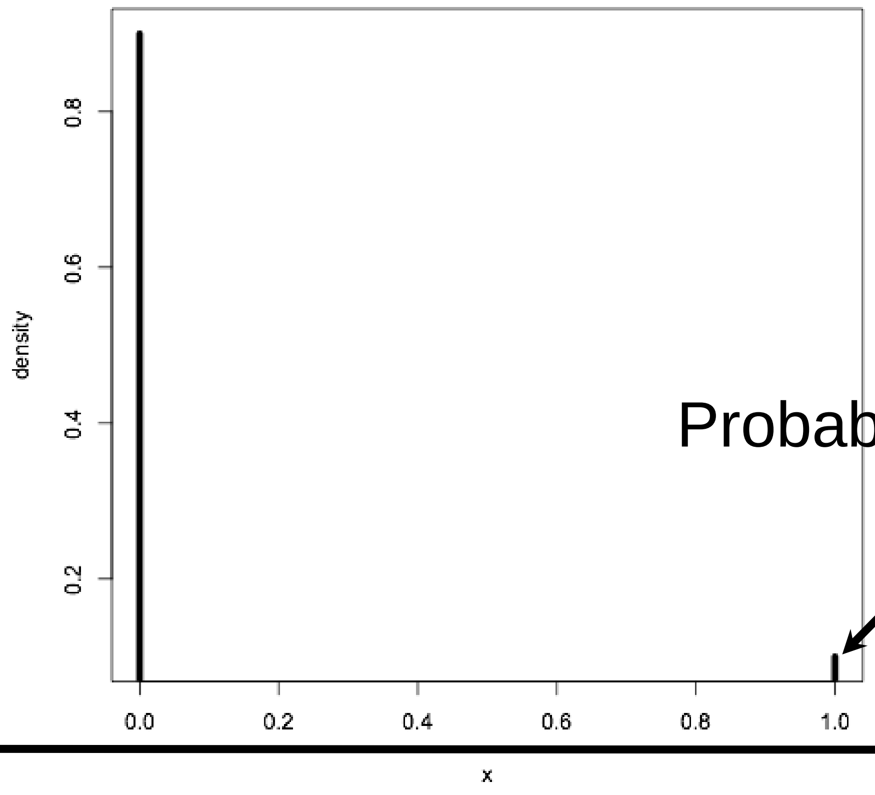
Can be either discrete or continuous (uniform, bernoulli, normal, etc)

Defined by a density function, *p*(x) or *f*(x)

Bernoulli distribution Be(p)

Flip a coin (T=0, H=1). Probability of H is .1.

```
x<-0:1
f<-dbinom(x, size=1, prob=.1)
plot(x,f,xlab="x",ylab="density",type="h",lwd=5)
```



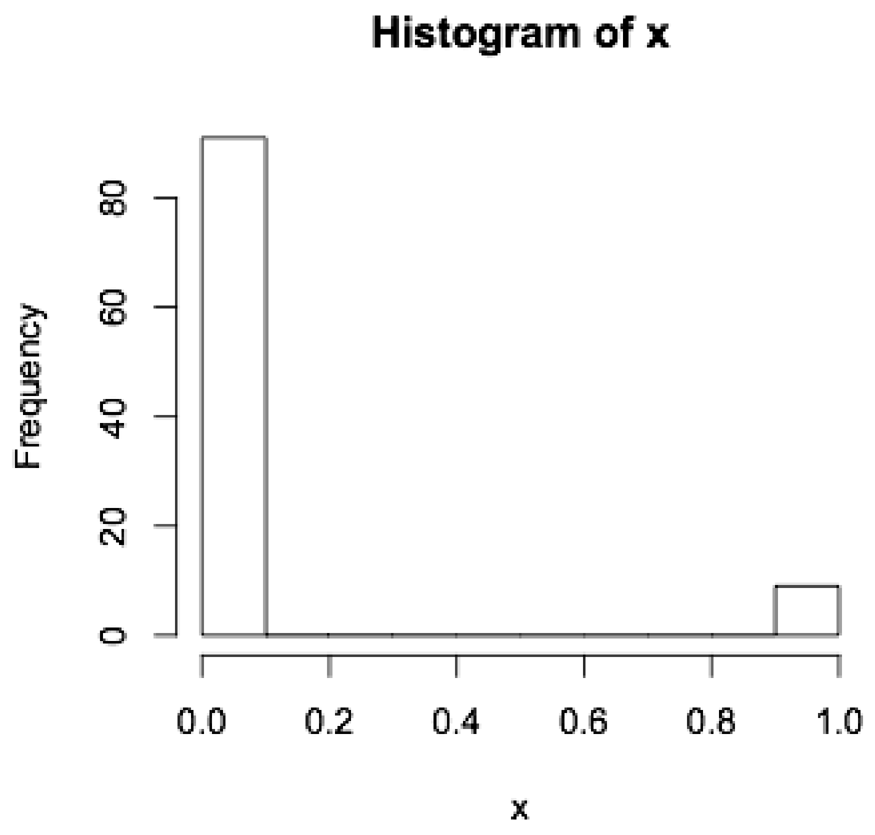
Probability of having a 1

Probability distributions

Random sampling

Generate 100 observations from a Be(.1)

```
set.seed(100)
x<-rbinom(100, size=1, prob=.1)hist(x)
hist(x)
```

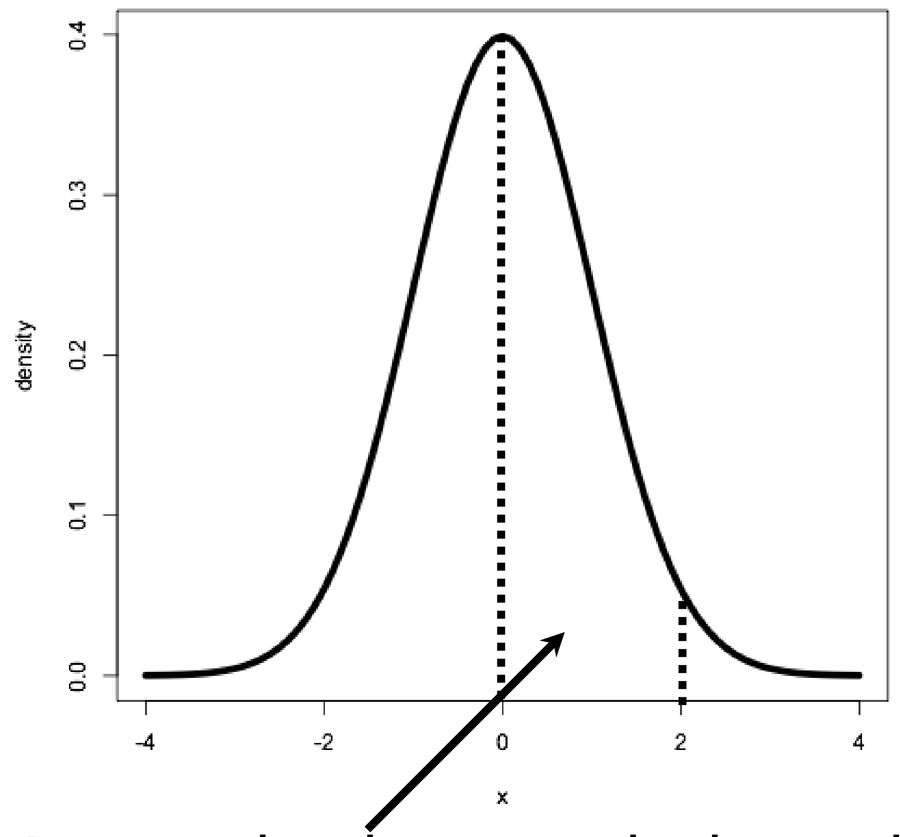


Probability distributions

Normal distribution $N(\mu,\sigma^2)$

μ is the mean and σ^2 is the variance

```
x<-seq(-4,4,.1)
f<-dnorm(x, mean=0, sd=1)
plot(x,f,xlab="x",ylab="density",lwd=5,type="l")
```



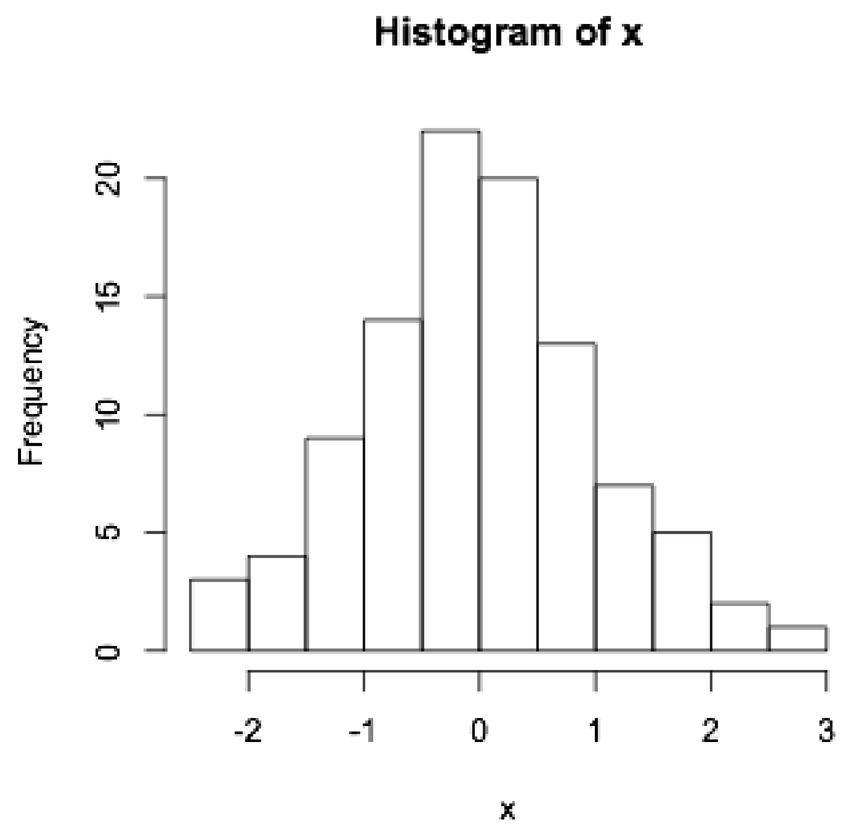
Area under the curve is the prob of having an observation beween 0 and 2.

Probability distributions

Random sampling

Generate 100 observations from a N(0,1)

```
set.seed(100)
x<-rnorm(100, mean=0, sd=1)
hist(x)
```

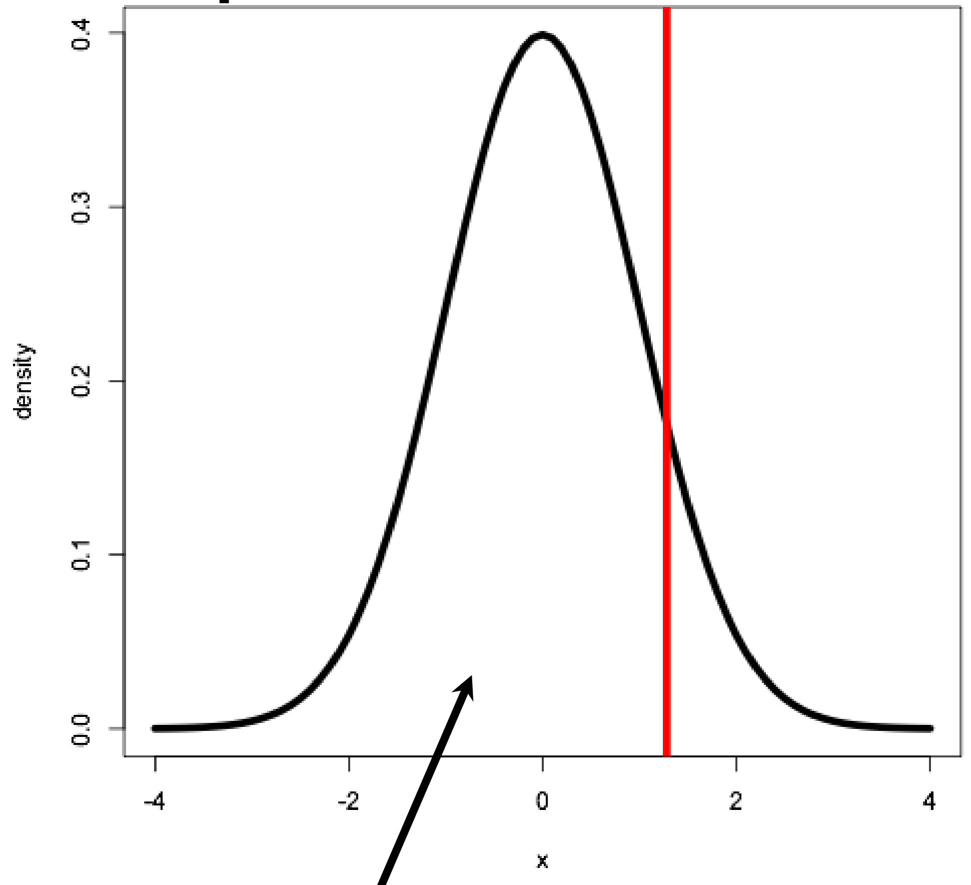


Histograms can be used to estimate densities!

Quantiles

(Theoritical) Quantiles: The *p*-quantile is the value with the property that there is a probability *p* of getting a value less than or equal to it.

```
q90<-qnorm(.90, mean = 0, sd = 1)
x<-seq(-4,4,.1)
f<-dnorm(x, mean=0, sd=1)
plot(x,f,xlab="x",ylab="density",type="l",lwd=5)
abline(v=q90,col=2,lwd=5)
```



The 50% quantile is called the median

90% of the prob. (area under the curve) is on the left of red vertical line.

Descriptive Statistics

Empirical Quantiles: The *p*-quantile is the value with the property that *p*% of the observations are less than or equal to it.

Empirical quantiles can easily be obtained in R.

```
set.seed(100)
x<-rnorm(100, mean=0, sd=1)
quantile(x)
```

```
        0%        25%        50%        75%       100% 
-2.2719255 -0.6088466 -0.0594199  0.6558911  2.5819589 
```

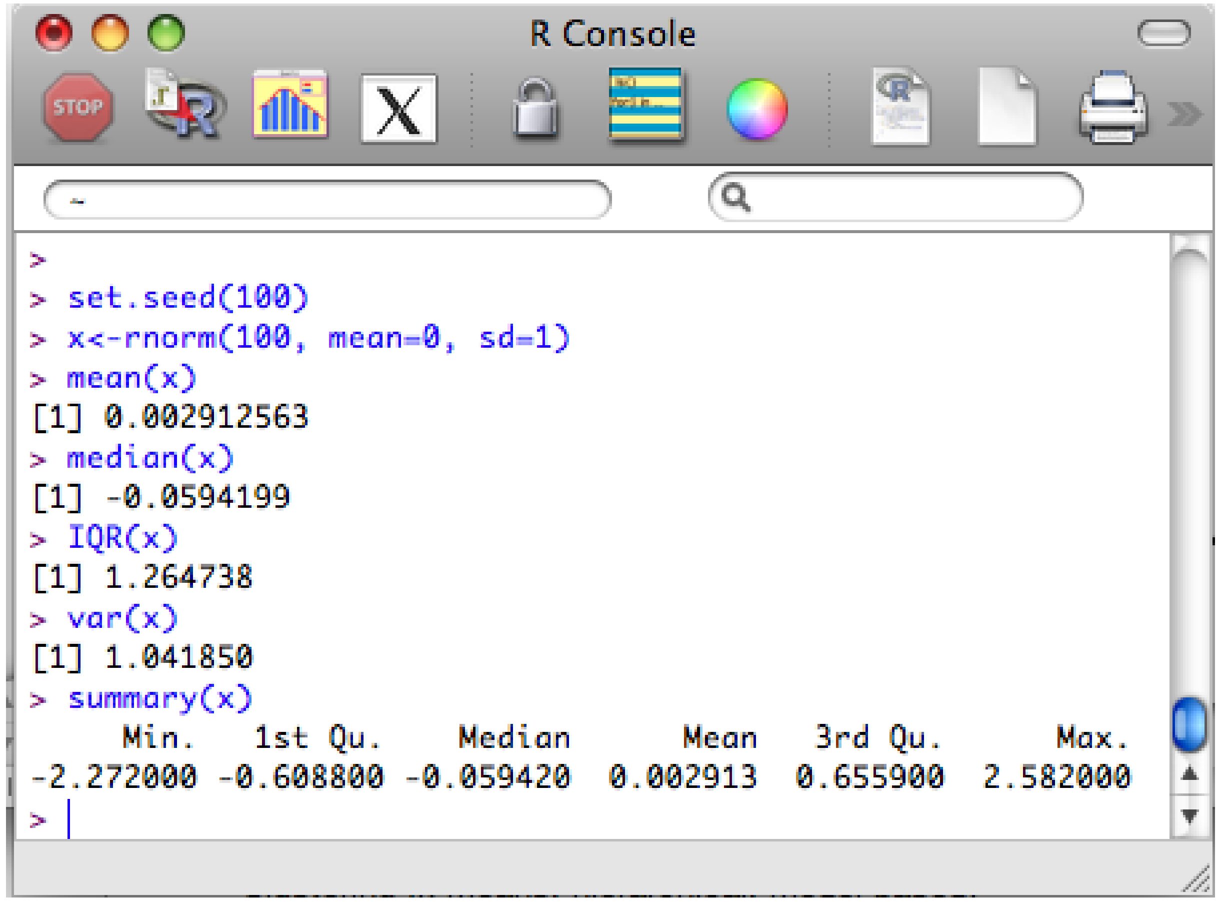
```
quantile(x,probs=c(.1,.2,.9))
```

```
       10%        20%        90% 
-1.1744996 -0.8267067  1.3834892 
```

Descriptive Statistics

We often need to quickly 'quantify' a data set, and this can be done using a set of summary statistics (mean, median, variance, standard deviation)

```
set.seed(100)
x<-rnorm(100, mean=0, sd=1)
mean(x)
median(x)
IQR(x)
var(x)
summary(x)
```

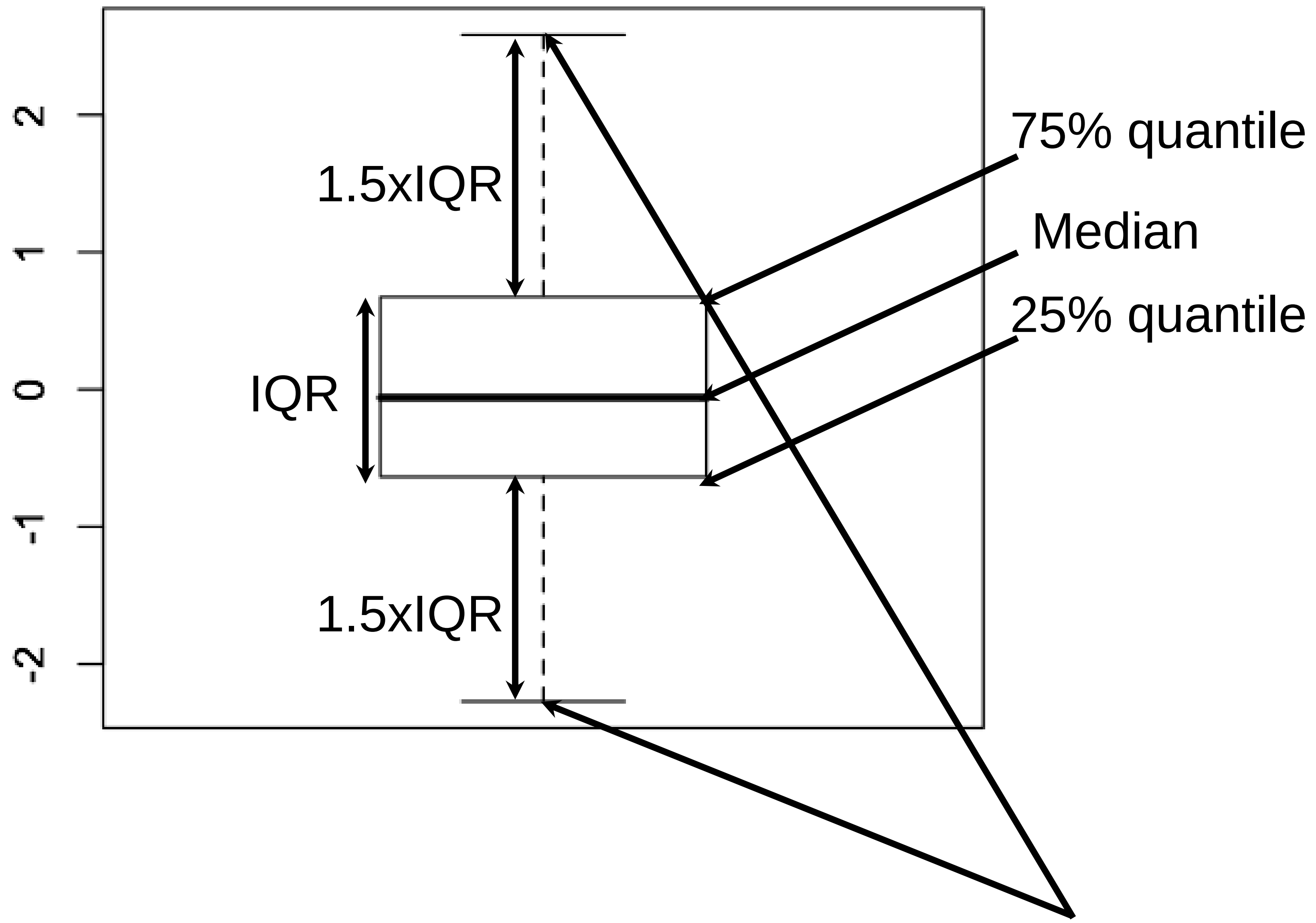


```
>
> set.seed(100)
> x<-rnorm(100, mean=0, sd=1)
> mean(x)
[1] 0.002912563
> median(x)
[1] -0.0594199
> IQR(x)
[1] 1.264738
> var(x)
[1] 1.041850
> summary(x)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-2.272000 -0.608800 -0.059420  0.002913  0.655900  2.582000
> 
```

'summary' can be used for almost any R object!
R is object oriented (methods/classes).

Descriptive Statistics - Box-plot

```
set.seed(100)
x<-rnorm(100, mean=0, sd=1)
boxplot(x)
```



IQR= 75% quantile -25% quantile= Inter Quantile Range

Everything above or below are considered outliers

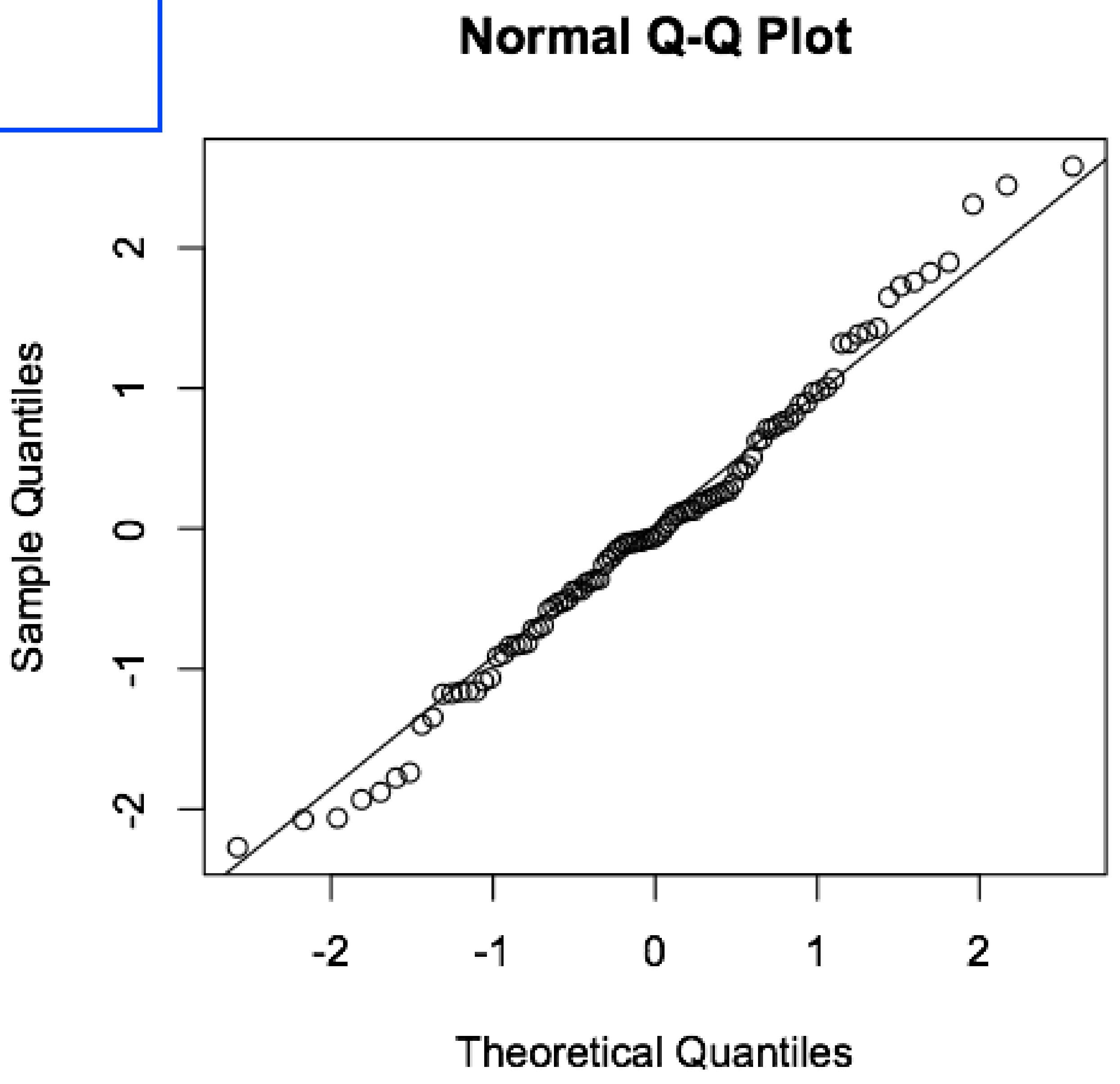
QQ-plot

- Many statistical methods make some assumption about the distribution of the data (e.g. Normal).
- The quantile-quantile plot provides a way to visually verify such assumptions.
- The QQ-plot shows the theoretical quantiles versus the empirical quantiles. If the distribution assumed (theoretical one) is indeed the correct one, we should observe a straight line.

QQ-plot

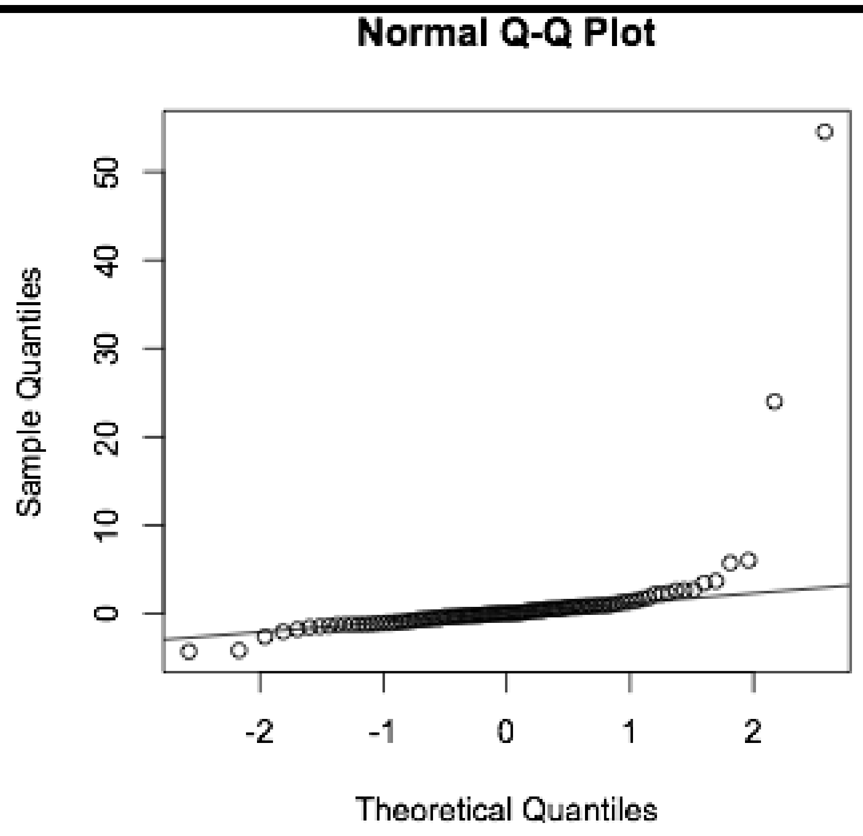
```
set.seed(100)
x<-rnorm(100, mean=0, sd=1)
qqnorm(x)
qqline(x)
```

Only valid for the normal distribution!



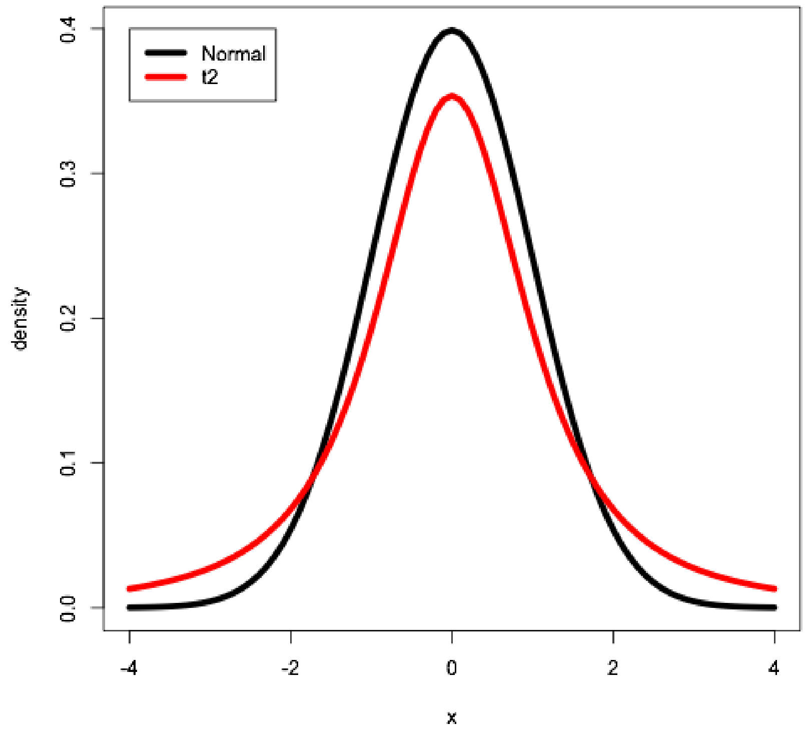
QQ-plot

```
set.seed(100)
x<-rt(100,df=2)
qqnorm(x)
qqline(x)
```



Clearly the *t* distribution with two degrees of freedom is different from the Normal!

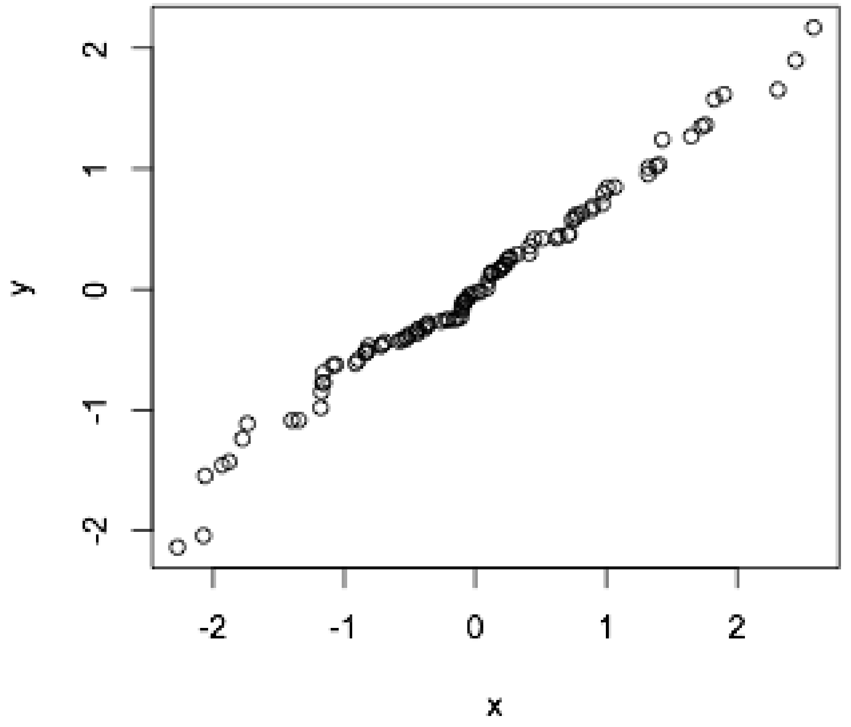
```
x<-seq(-4,4,.1)
f1<-dnorm(x, mean=0, sd=1)
f2<-dt(x, df=2)
plot(x,f1,xlab="x",ylab="density",lwd=5,type="l")
lines(x,f2,xlab="x",ylab="density",lwd=5,col=2)
```



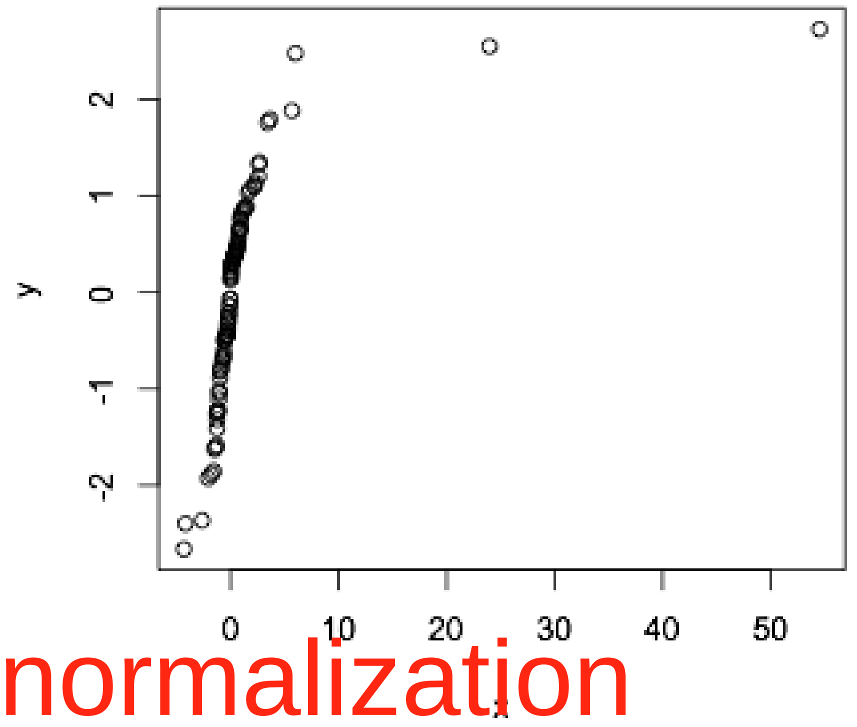
QQ-plot

Comparing two samples

```
set.seed(100)
x<-rnorm(100, mean=0, sd=1)
y<-rnorm(100, mean=0, sd=1)
qqplot(x,y)
```



```
set.seed(100)
x<-rt(100, df=2)
y<-rnorm(100, mean=0, sd=1)
qqplot(x,y)
```



Ex: Try with different values of df.

Main idea behind quantile normalization

Scatter plots

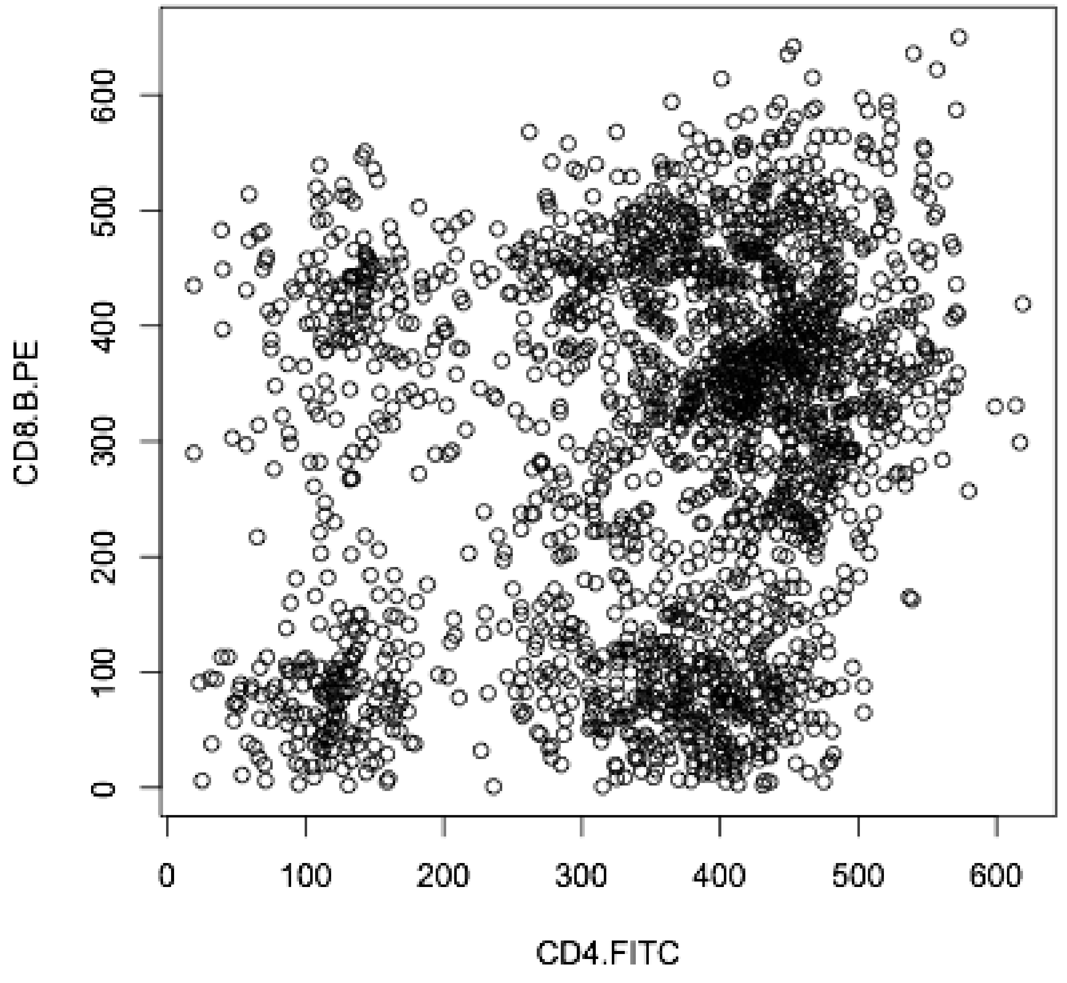
Biological data sets often contain several variables

So they are multivariate.

Scatter plots allow us to look at two variables at a time.

```
# GvHD flow cytometry data
gvhd<-read.table("GvHD+.txt", header=TRUE)
# Only extract the CD3 positive cells
gvhdCD3p<-as.data.frame(gvhd[gvhd[,5]>280,3:6])
cor(gvhdCD3p[,1],gvhdCD3p[,2])
```

This can be used to visually assess independence!



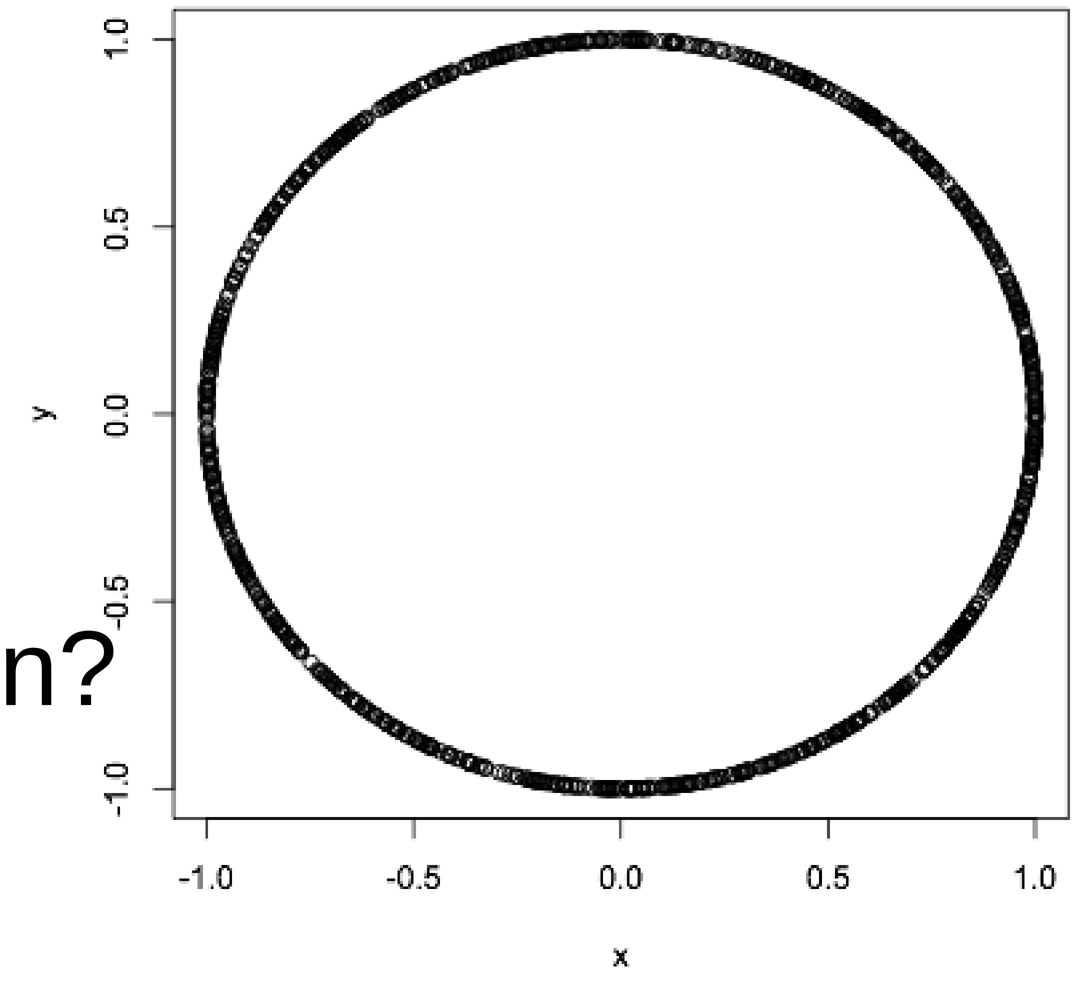
Scatter plots vs. correlations

Note that in this example, the correlation between CD8.B.PE and CD4.FITC is 0.23.

Correlation is only good for linear dependence.

```
# Quick comment on correlation
set.seed(100)
theta<-runif(1000,0,2*pi)
x<-cos(theta)
y<-sin(theta)
plot(x,y)
cor(x,y)
[1] -0.05328118
```

What is the correlation?



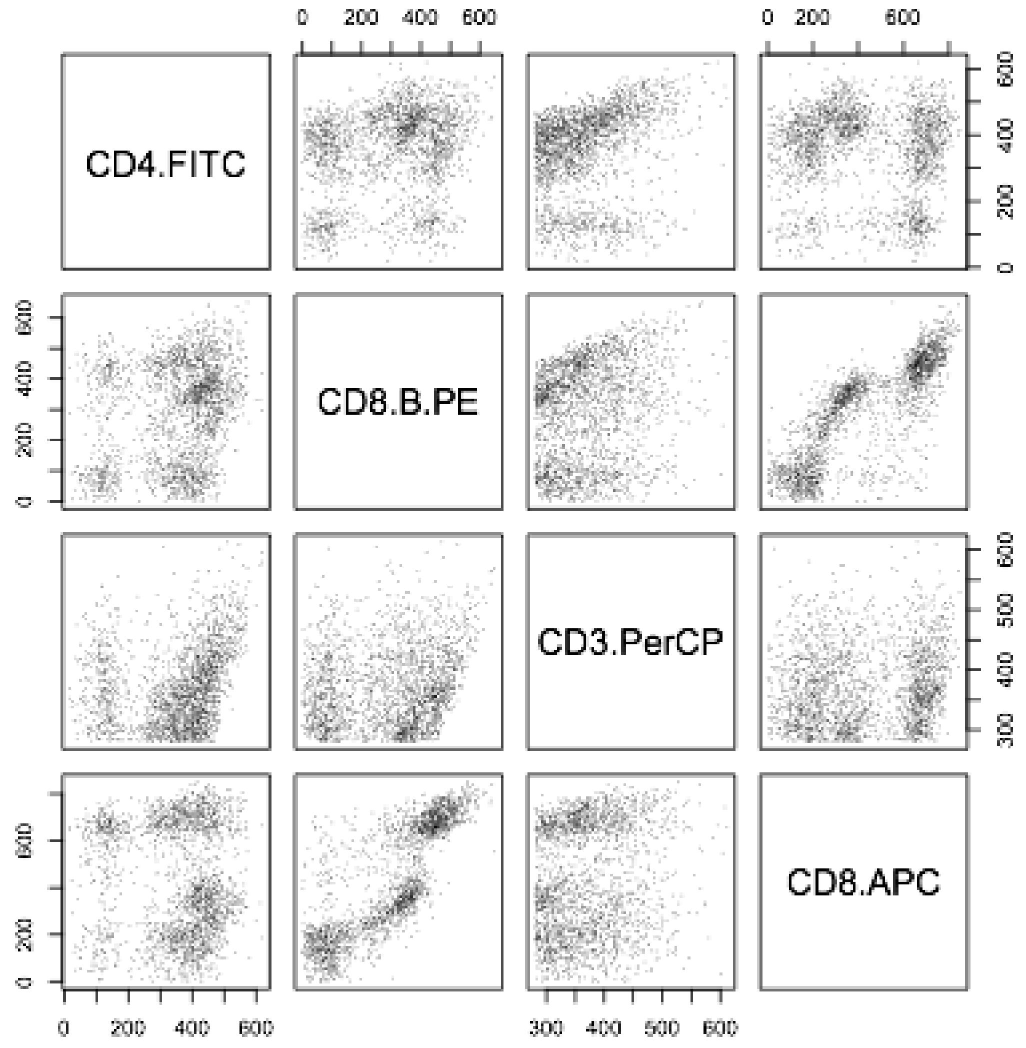
Trellis graphics

Trellis Graphics is a family of techniques for viewing complex, multi-variable data sets.

```
plot(gvhdCD3p, pch=".")
```

Note that I have changed the plotting symbol.

Many more possibilities in the ‘lattice’ package!

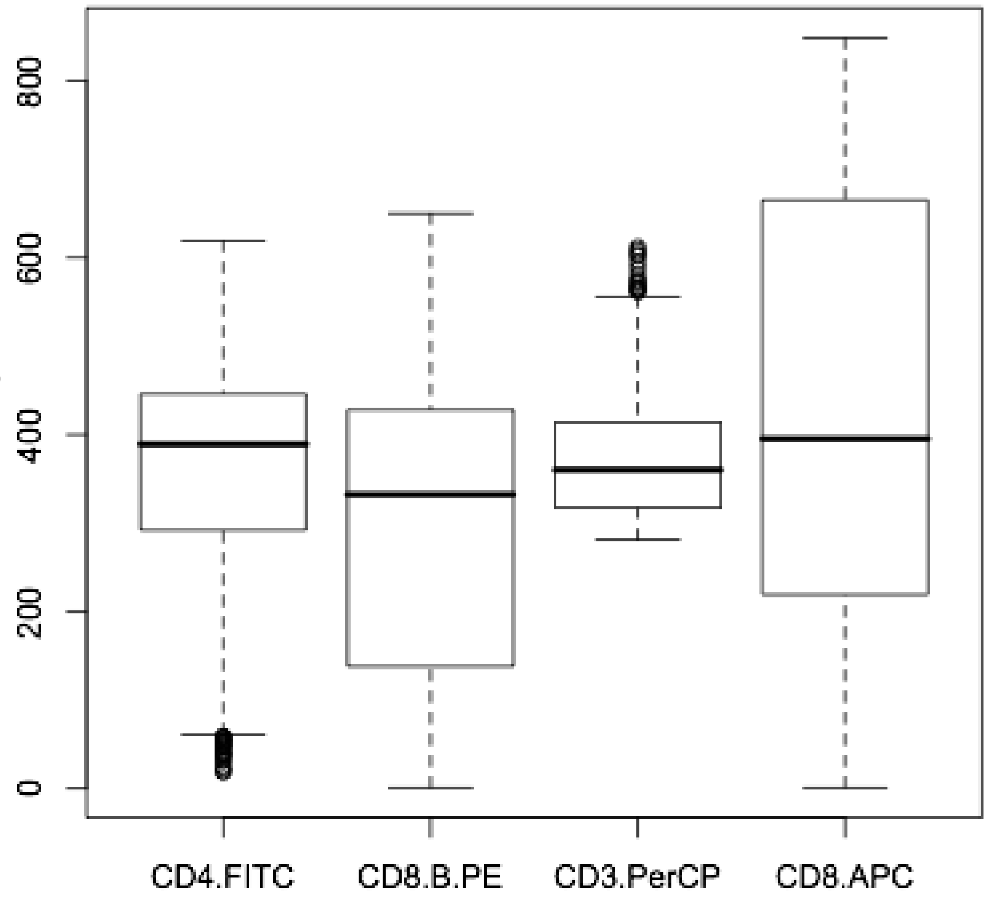


EDA of flow data

```
boxplot(gvhdCD3p)
```

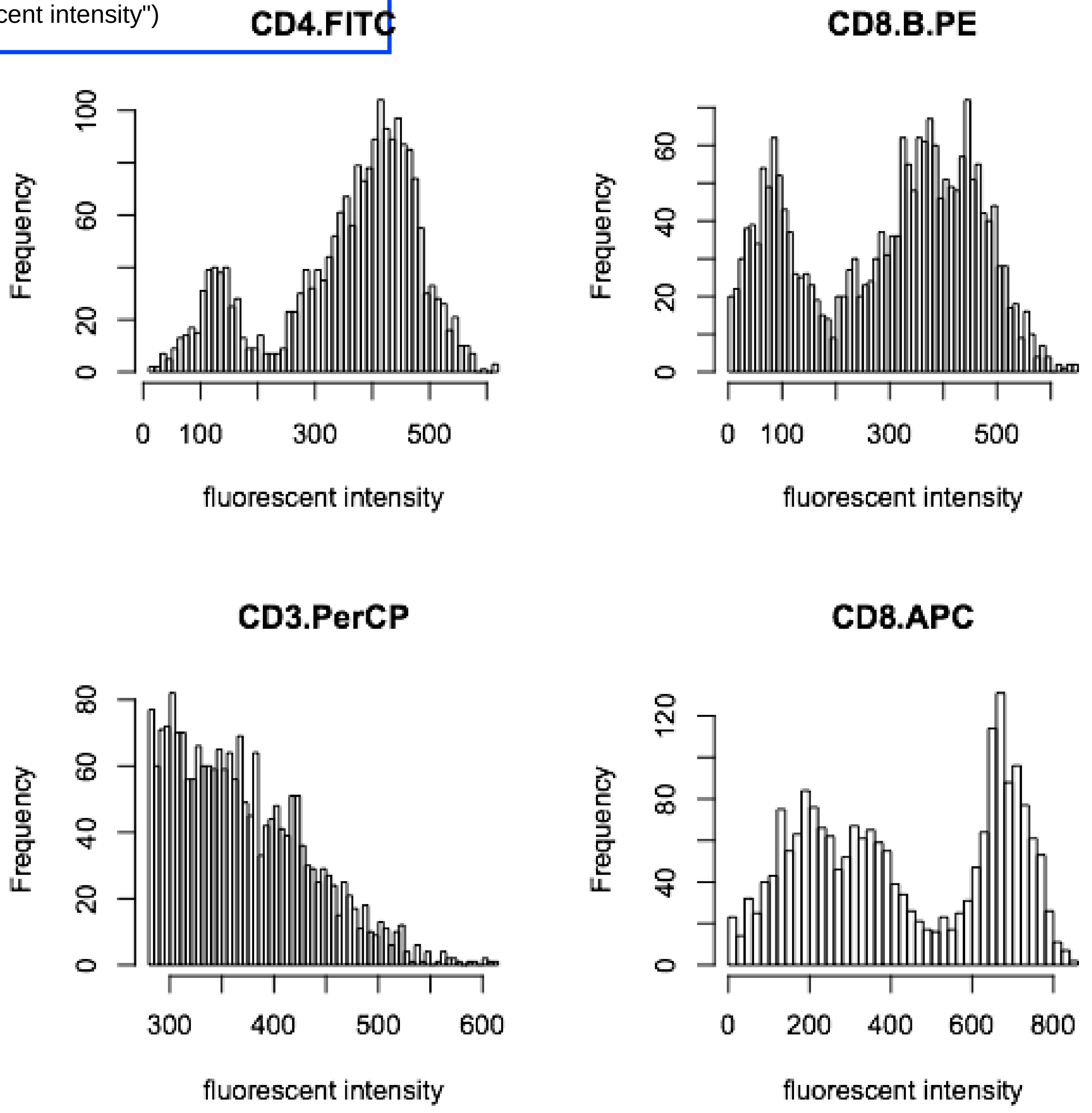
The boxplot function can be used to display several variables at a time!

What can you say here?



EDA of flow data

```
par(mfrow=c(2,2))hist(gvhdCD3p[,1],50,main=names(gvhdCD3p)[1],xlab="fluorescent intensity")
hist(gvhdCD3p[,2],50,main=names(gvhdCD3p)[2],xlab="fluorescent intensity")
hist(gvhdCD3p[,3],50,main=names(gvhdCD3p)[3],xlab="fluorescent intensity")
hist(gvhdCD3p[,4],50,main=names(gvhdCD3p)[4],xlab="fluorescent intensity")
```



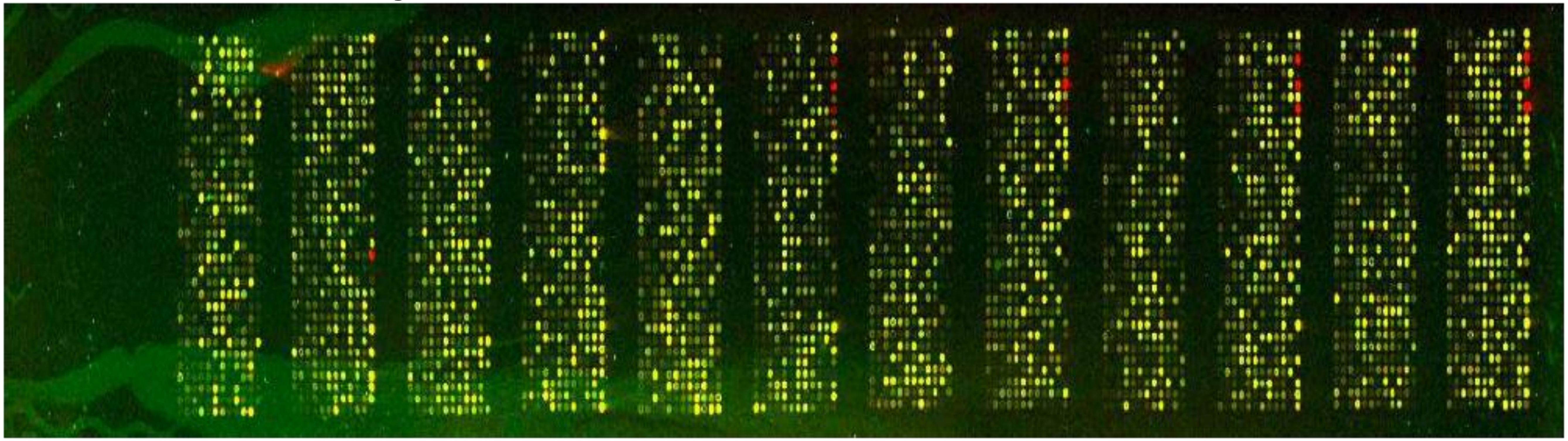
Mix of cell sub-populations!

EDA: HIV data

- HIV Data
- The expression levels of 7680 genes were measured in CD4-T-cell lines at time t = 24 hours after infection with HIV type 1 virus. 12 positive controls (HIV genes).
- 4 replicates (2 with a dye swap)

EDA: HIV data

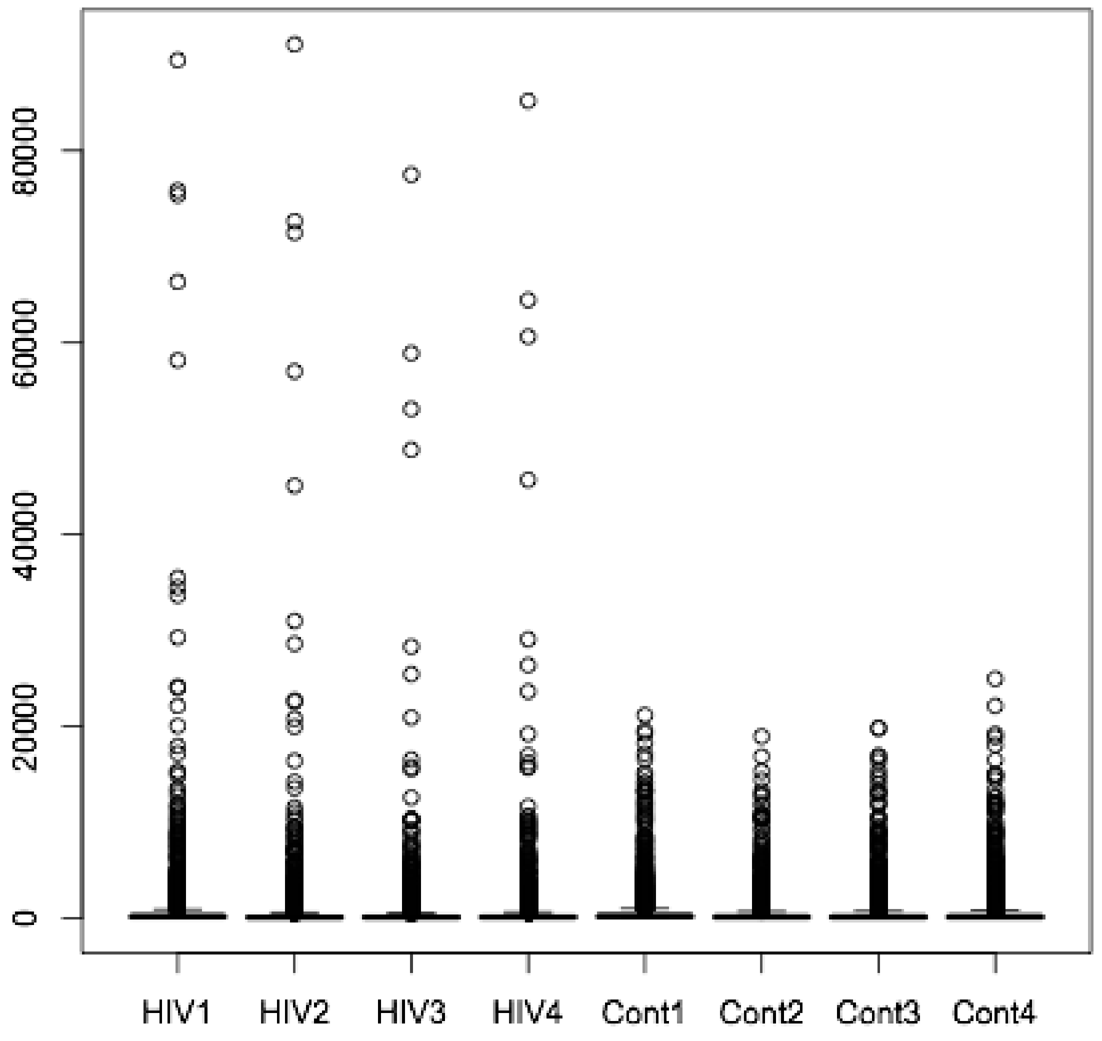
- One of the array



- Assume the image analysis is done
- For each gene (spot) we have an estimate of the intensity in both channels
- Data matrix of size 7680x8

EDA: HIV data – this is a box-plot!

```
data<-read.table(file="hiv.raw.data.24h.txt",sep="\t",header=TRUE)
summary(data)
boxplot(data)
#this really is a box plot!
```

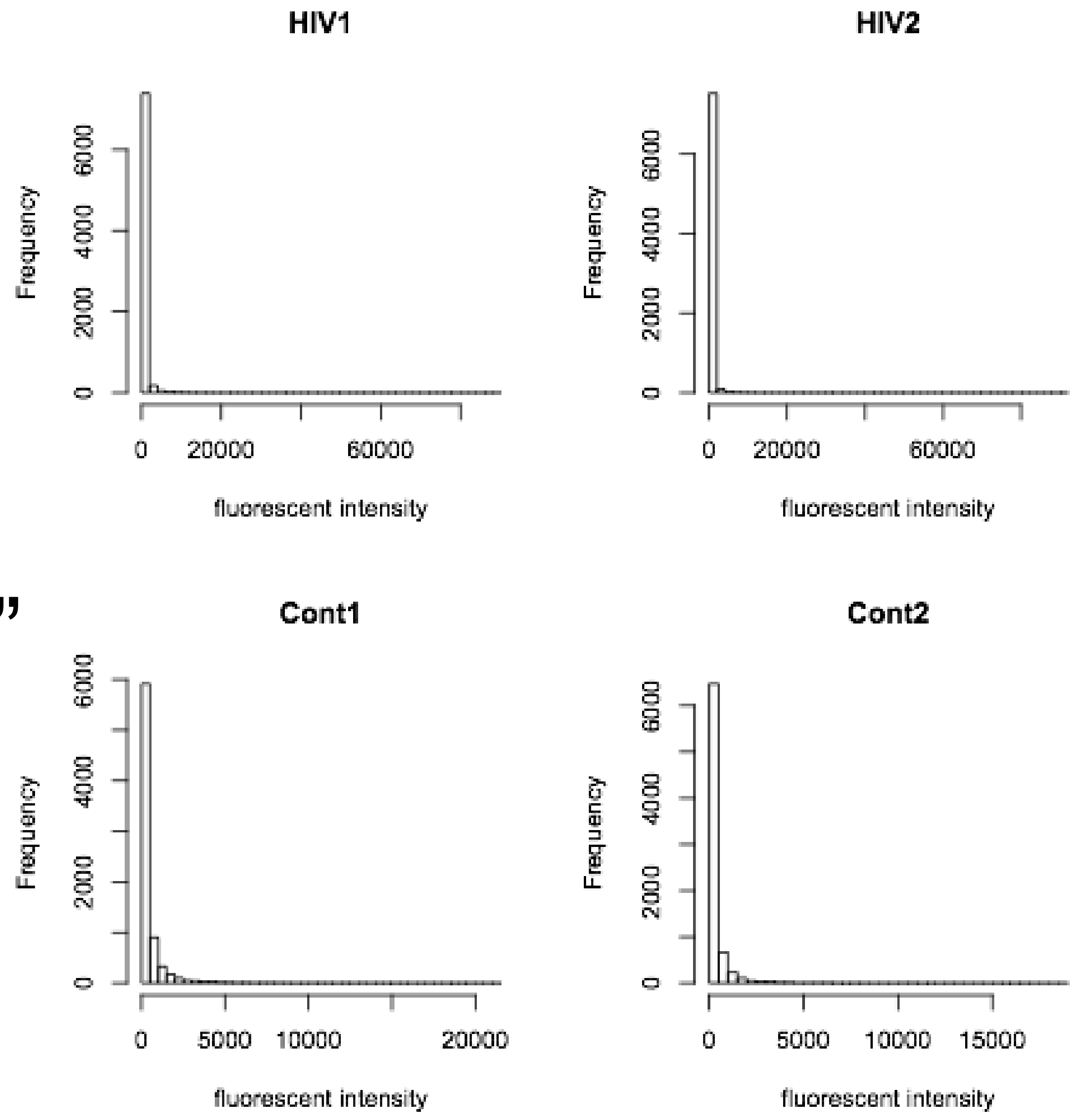


EDA: HIV data

```
par(mfrow=c(2,2))hist(data[,1],50,main=names(data)[1],xlab="fluorescent intensity")
hist(data[,2],50,main=names(data)[2],xlab="fluorescent intensity")hist(data[,5],50,main=names(data)[5],xlab="fluorescent intensity")
hist(data[,6],50,main=names(data)[6],xlab="fluorescent intensity")
```

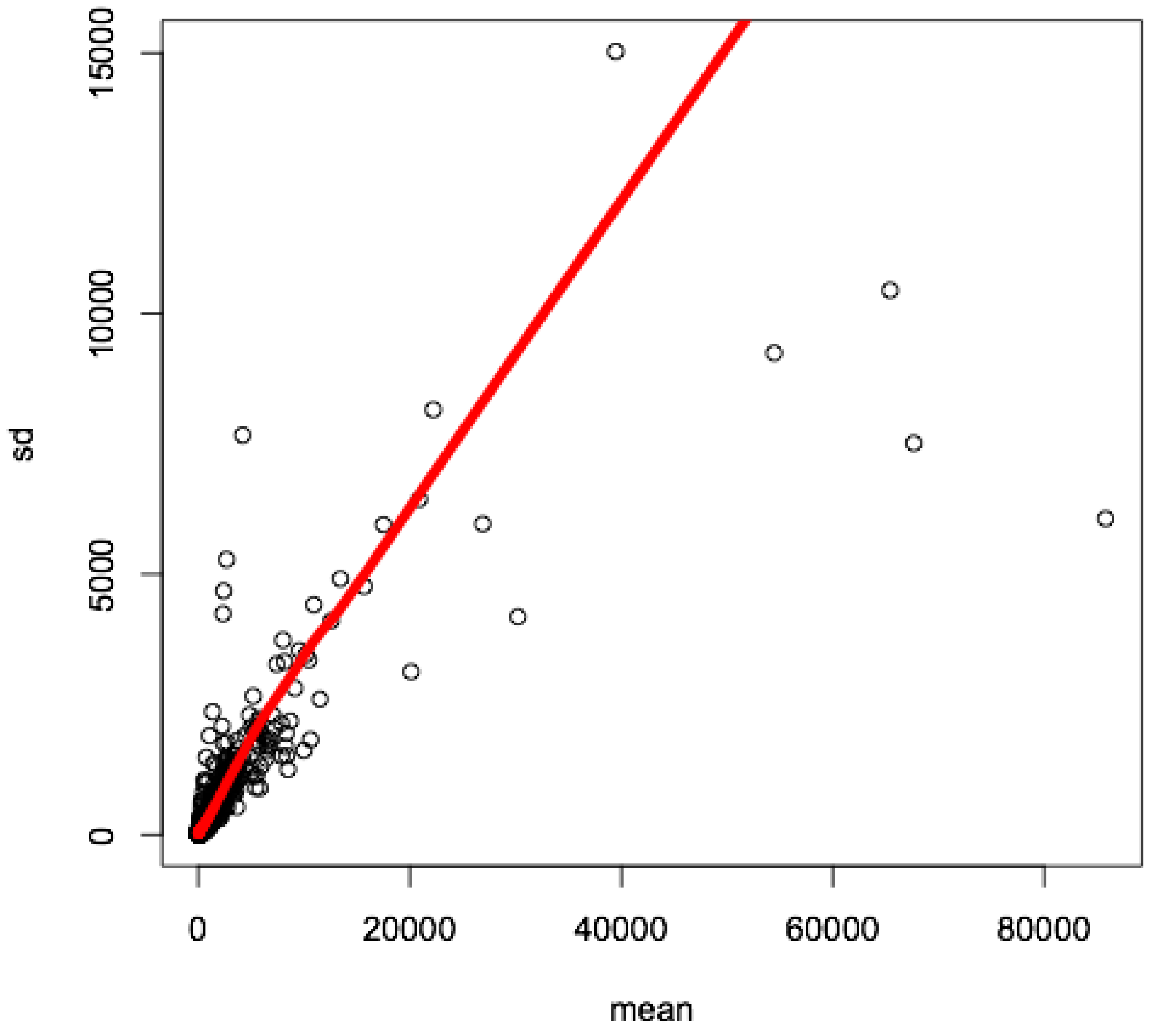
Does this look Normal to you?

The box-plot ”hides” this skewness.



EDA: HIV data

The standard deviation is not constant as it increases with the mean.

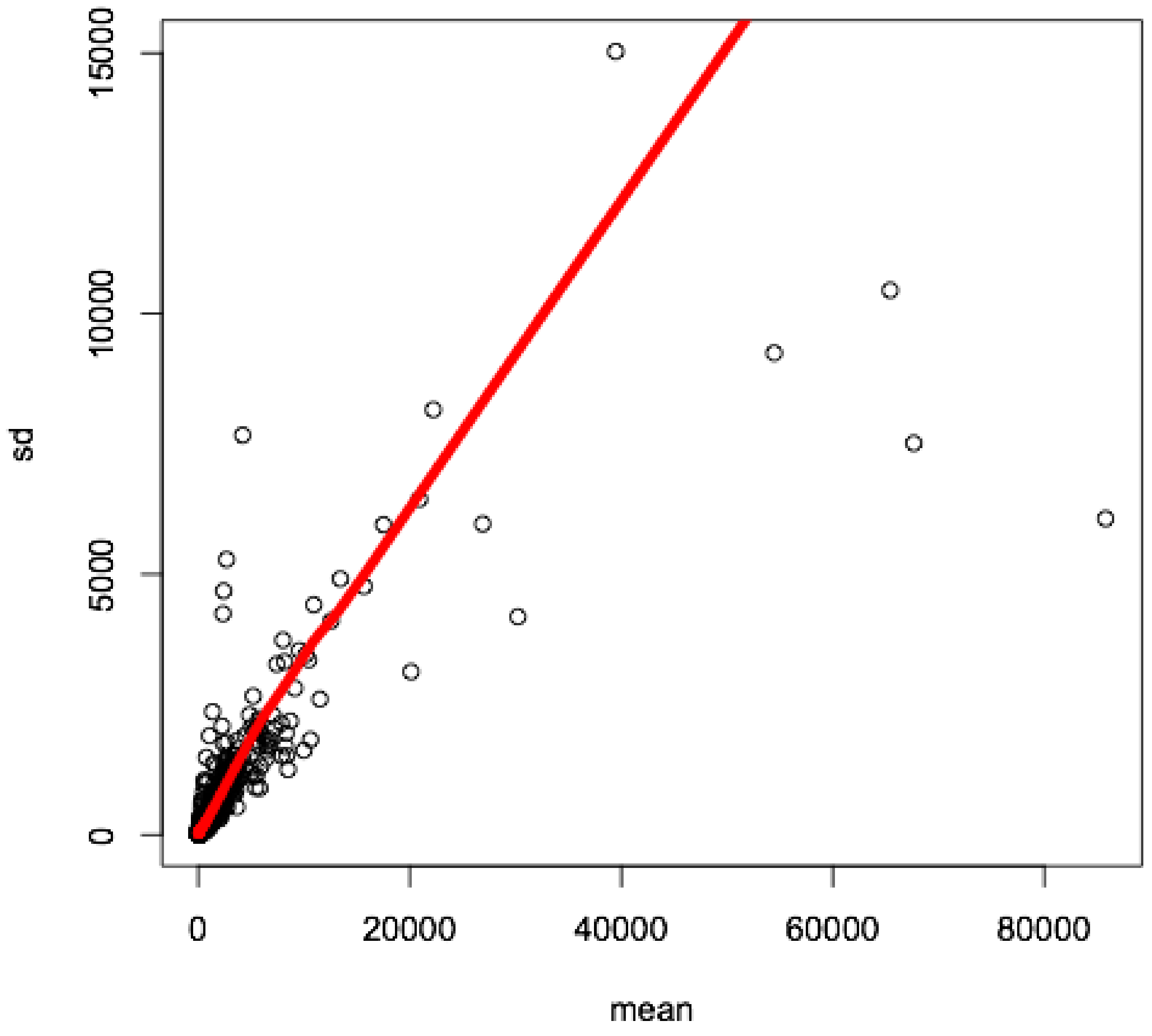


EDA: HIV data

```
# 'apply' will apply the function to all rows of the data matrix
mean<-apply(data[,1:4],1,"mean")
sd<-apply(data[,1:4],1,"sd")
plot(mean,sd)
trend<-lowess(mean,sd)lines(trend,col=2,lwd=5)
```

—— lowess fit

LOcally WEighted Scatter plot Smoother used to estimate the trend in a scatter plot
Non parametric!



EDA: Transformations

Observations:

The data are highly skewed.

The standard deviation is not constant as it increases with the mean.

Solution:

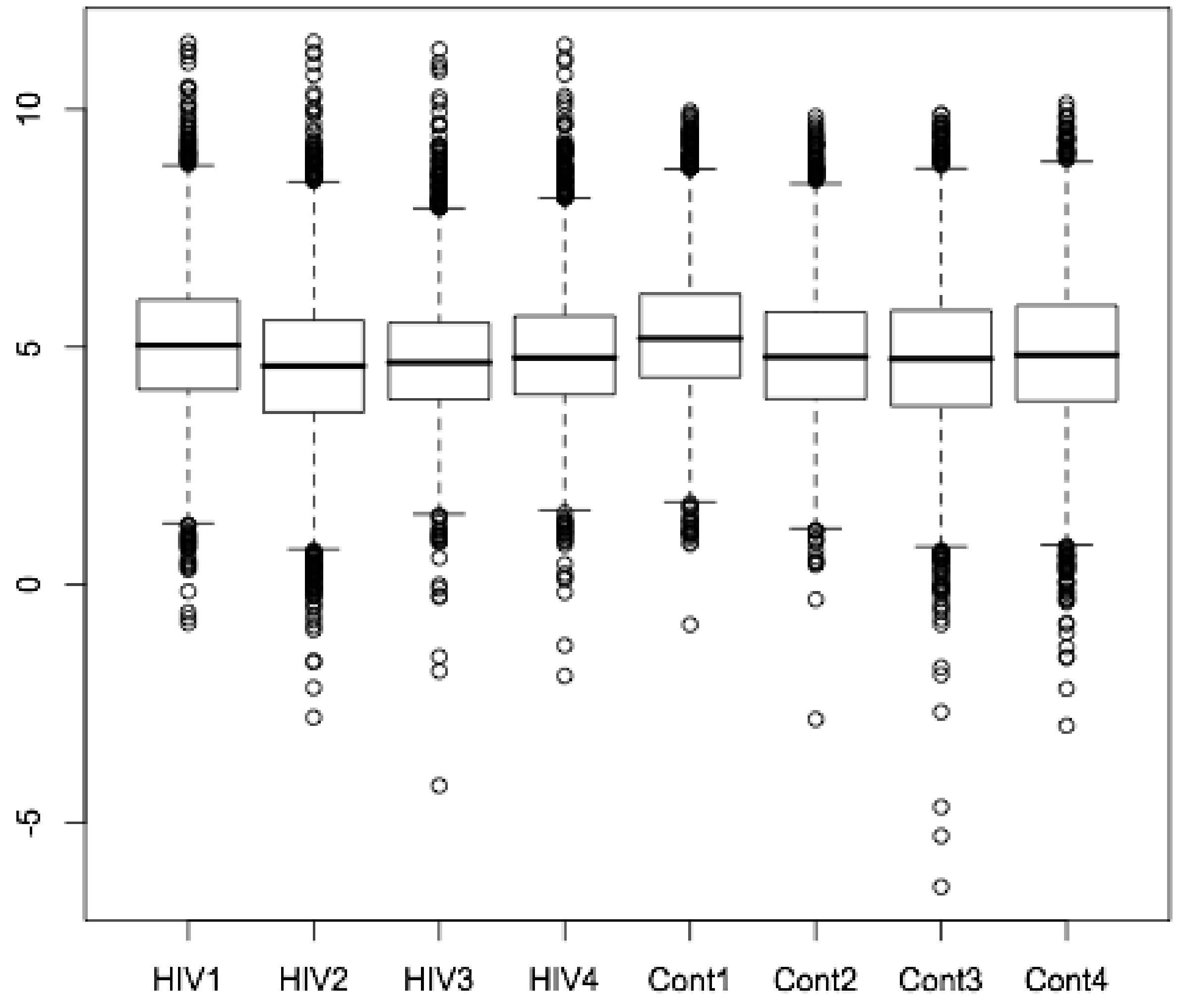
Look for a transformation that will make the data more symmetric and the variance more constant.

With positive data the log transformation is often appropriate.

EDA: Transformations

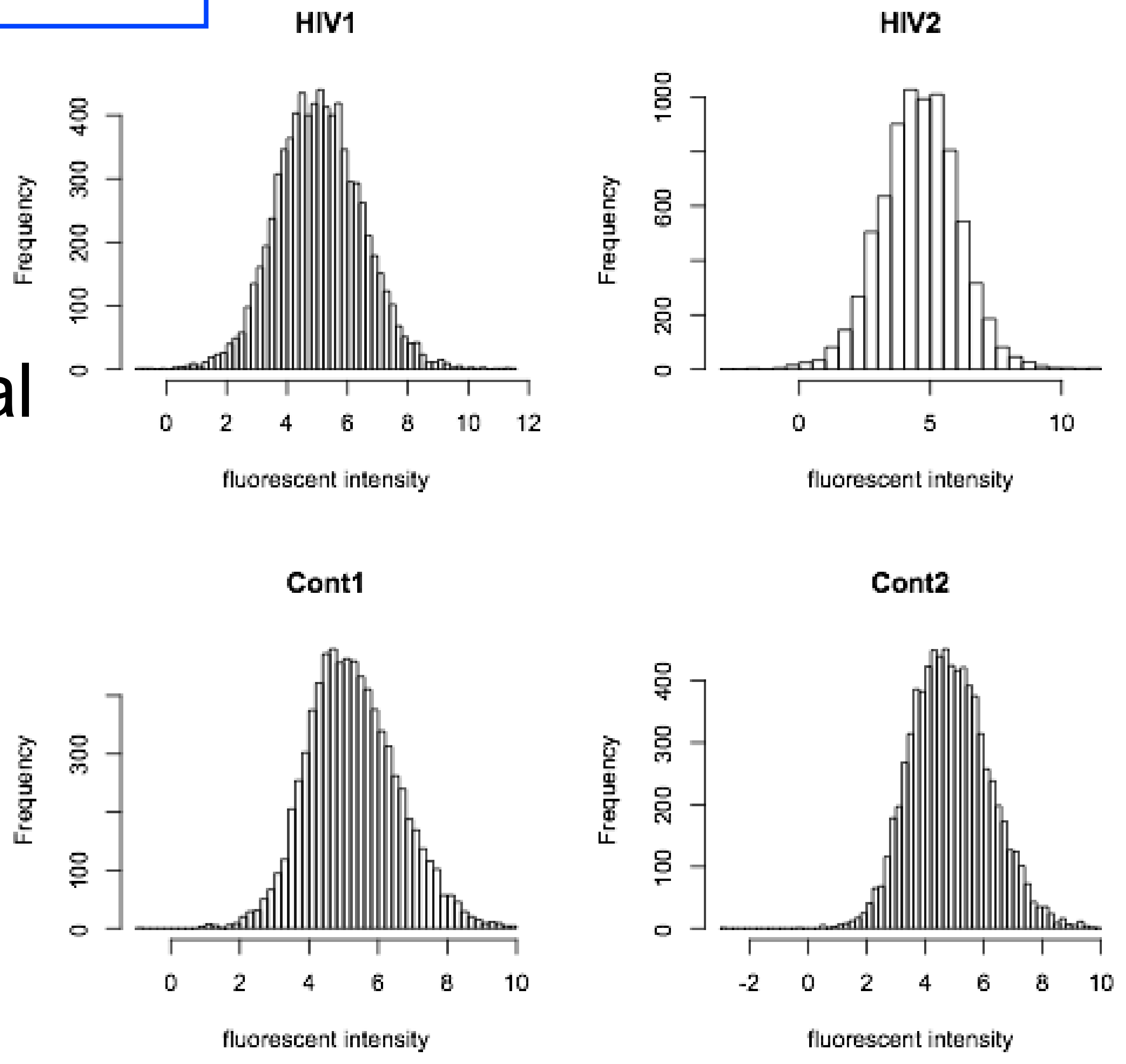
```
data<-log(read.table(file="hiv.raw.data.24h.txt",sep="\t",header=TRUE))
summary(data)
boxplot(data)
```

The data is now less skewed.



EDA: Transformations

```
par(mfrow=c(2,2))
hist(data[,1],50,main=names(data)[1],xlab="fluorescent intensity")
hist(data[,2],50,main=names(data)[2],xlab="fluorescent intensity")
hist(data[,5],50,main=names(data)[5],xlab="fluorescent intensity")
hist(data[,6],50,main=names(data)[6],xlab="fluorescent intensity")
```



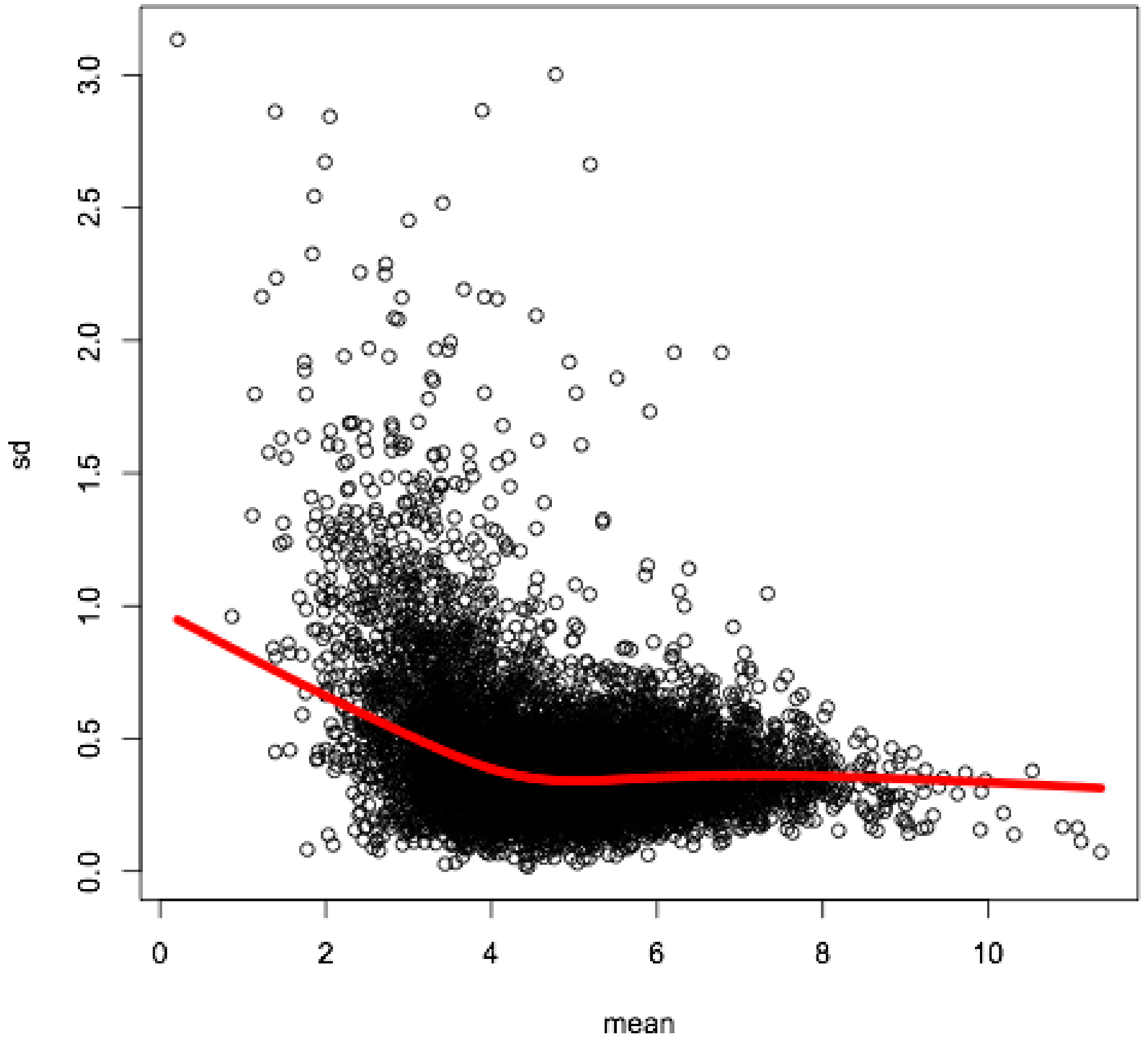
…and follows a more normal distribution.

All the data is more easily visualized.

EDA: Transformations

```
mean<-apply(data[,1:4],1,"mean")
sd<-apply(data[,1:4],1,"sd")
plot(mean,sd)
trend<-lowess(mean,sd)
lines(trend,col=2,lwd=5)
```

The sd is almost independent of the mean now!



EDA and microarray: Always log

- Makes the data more symmetric, large observations are not as influential
- The variance is (more) constant
- Turns multiplication into addition (log(ab)=log(a)+log(b))

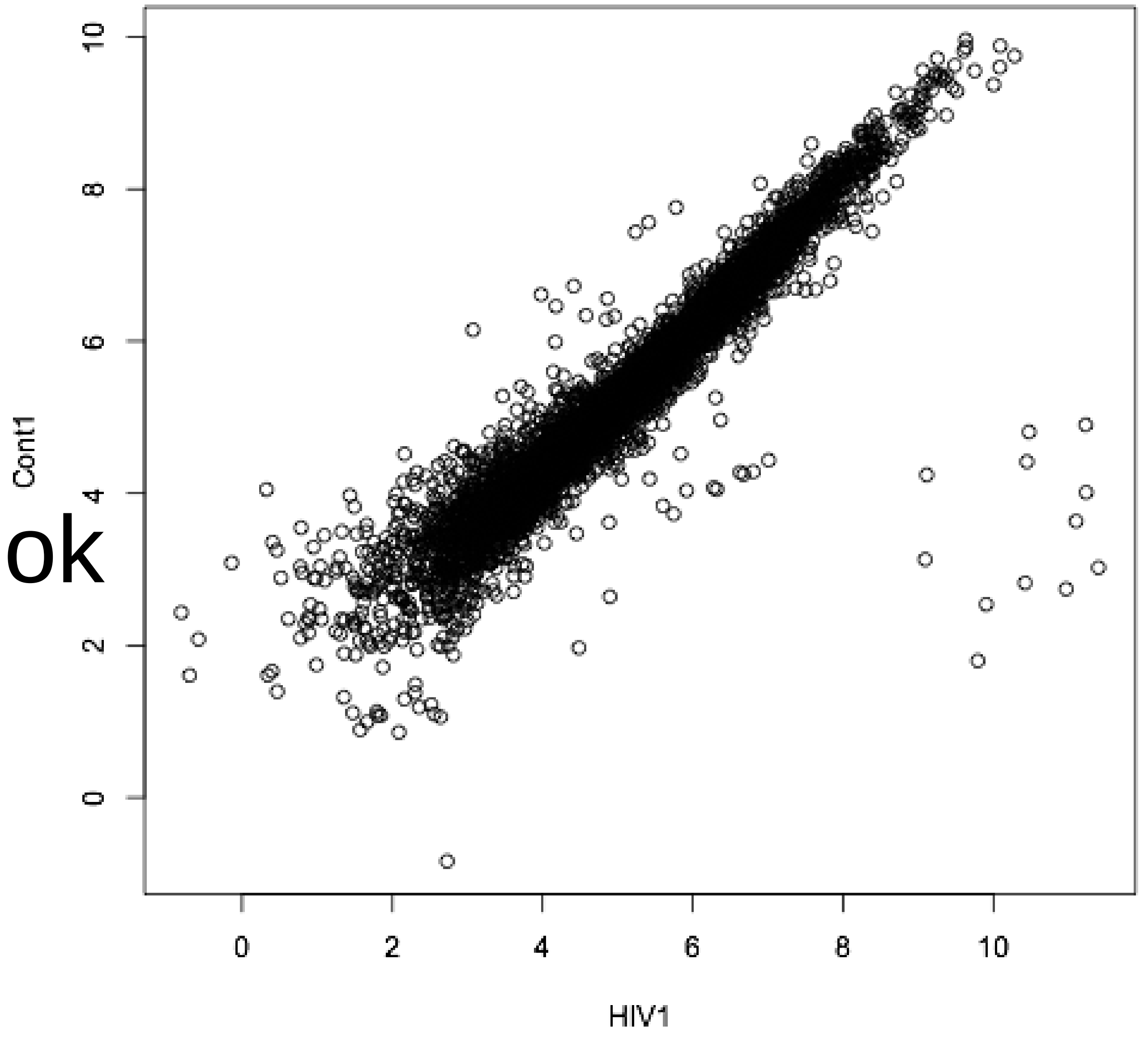
- So fold change in expression of two genes is just the difference between the logs of the original signal.
- In practice use log base 2, log2(x)=log(x)/log(2)

EDA for gene expression

```
# scatter plot
plot(data[,1],data[,5],xlab=names(data)[1],ylab=names(data)[5])
```

What can you say?

Is this the best way to look at the data?

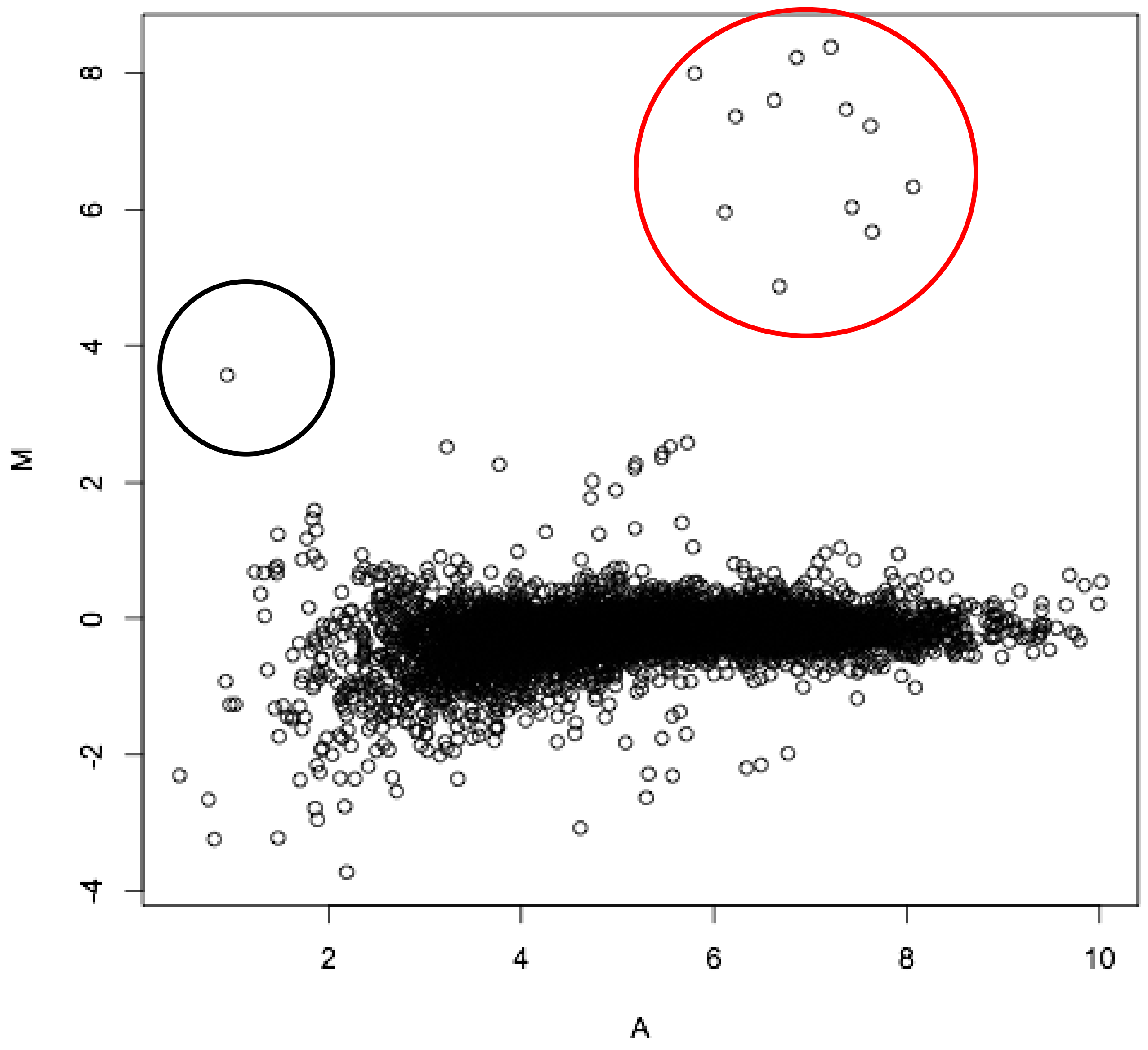


EDA for gene expression : MA plots

```
# MA plots per replicate
A<-(data[,1]+data[,5])/2
M<-(data[,1]-data[,5])
plot(A,M,xlab="A",ylab="M")
```

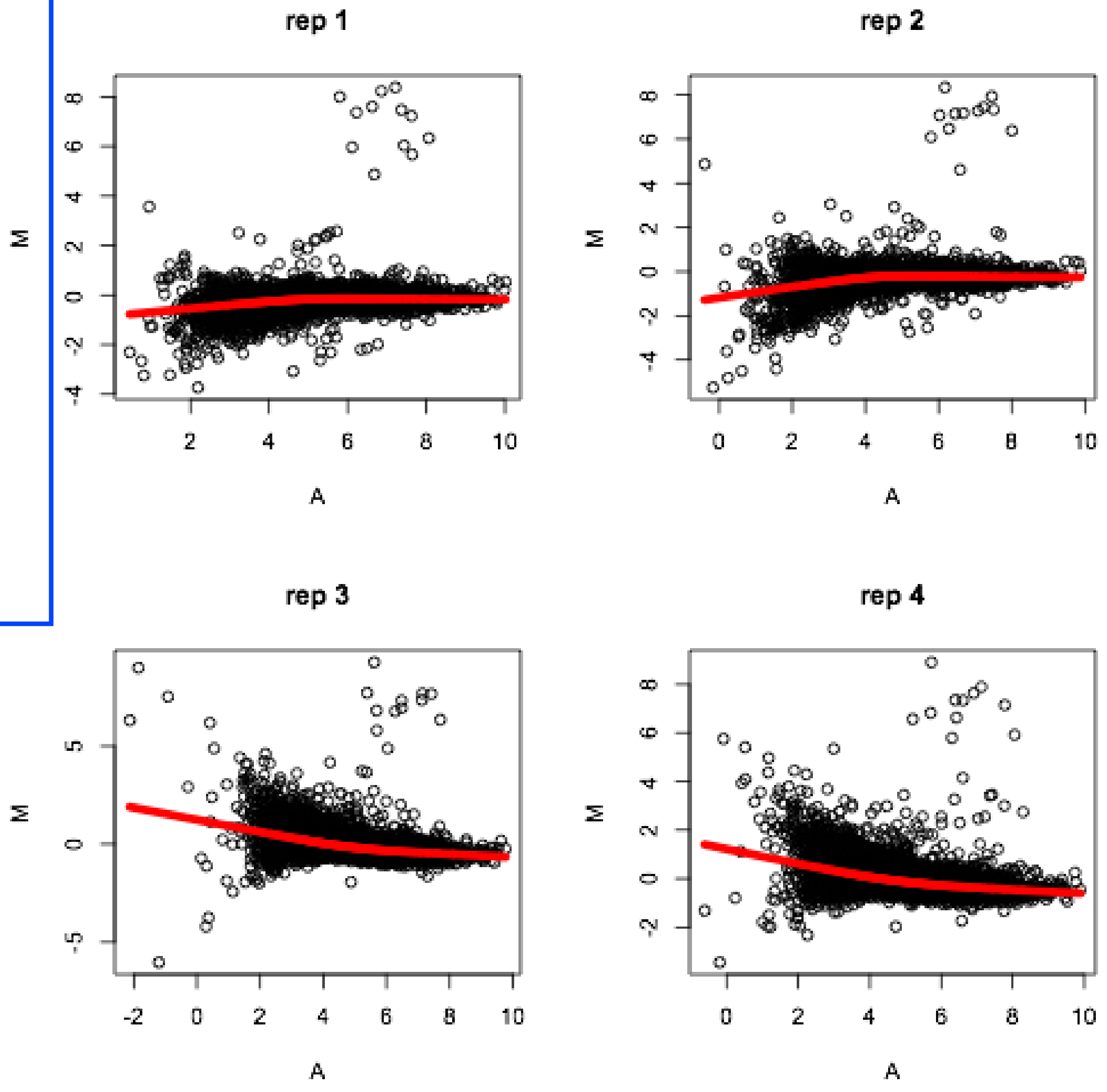
M (minus) is the log ratio
A (average) is overall intensity.

So here a MA plot is superior to a straight scatter-plot because we can differentiate between differences that we might trust more because they are based on higher signal.



EDA for gene expression : MA plots

```
# MA plots per replicate
par(mfrow=c(2,2))
A<-(data[,1]+data[,5])/2
M<-(data[,1]-data[,5])
plot(A,M,xlab="A",ylab="M",main="rep 1")
trend<-lowess(A,M)
lines(trend,col=2,lwd=5)
A<-(data[,2]+data[,6])/2
M<-(data[,2]-data[,6])
plot(A,M,xlab="A",ylab="M",main="rep 2")
trend<-lowess(A,M)lines(trend,col=2,lwd=5)
A<-(data[,3]+data[,7])/2
M<-(data[,3]-data[,7])
plot(A,M,xlab="A",ylab="M",main="rep 3")
trend<-lowess(A,M)
lines(trend,col=2,lwd=5)A<-(data[,4]+data[,8])/2
M<-(data[,4]-data[,8])
plot(A,M,xlab="A",ylab="M",main="rep 4")
trend<-lowess(A,M)
lines(trend,col=2,lwd=5)
```

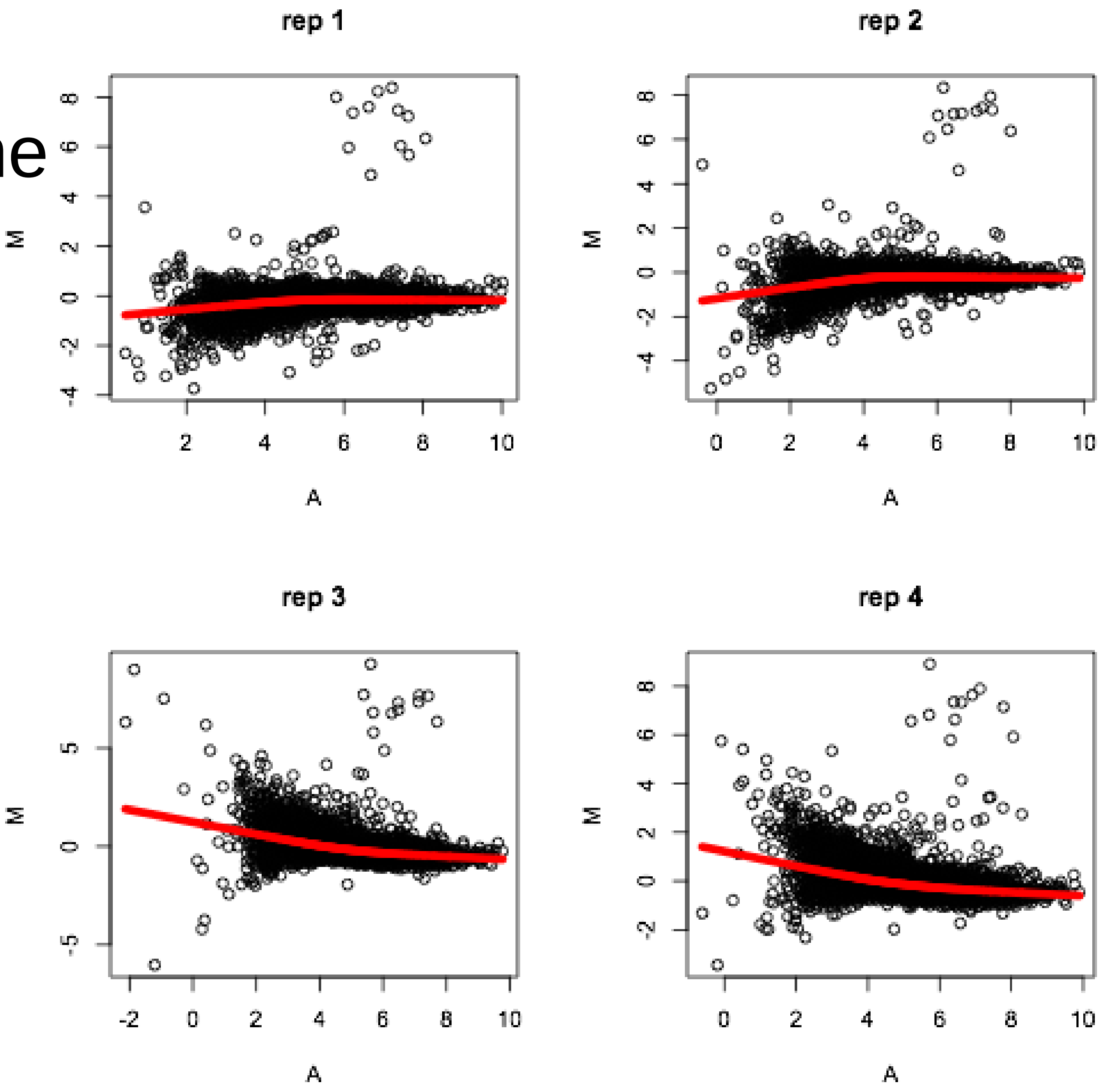


EDA for gene expression : MA plots

Here, the analysis hows that one of the dyes is skewed towards slightly higher values. (Replicates 3 and 4 – the red line should be around zero)

It becomes apparent that this should be corrected for.

This becomes important when selecting diff expressed genes.



Summary

- EDA should be the first step in any statistical analysis!
- Extremely Important
- Good modeling starts and ends with EDA
- R provides a great framework for EDA

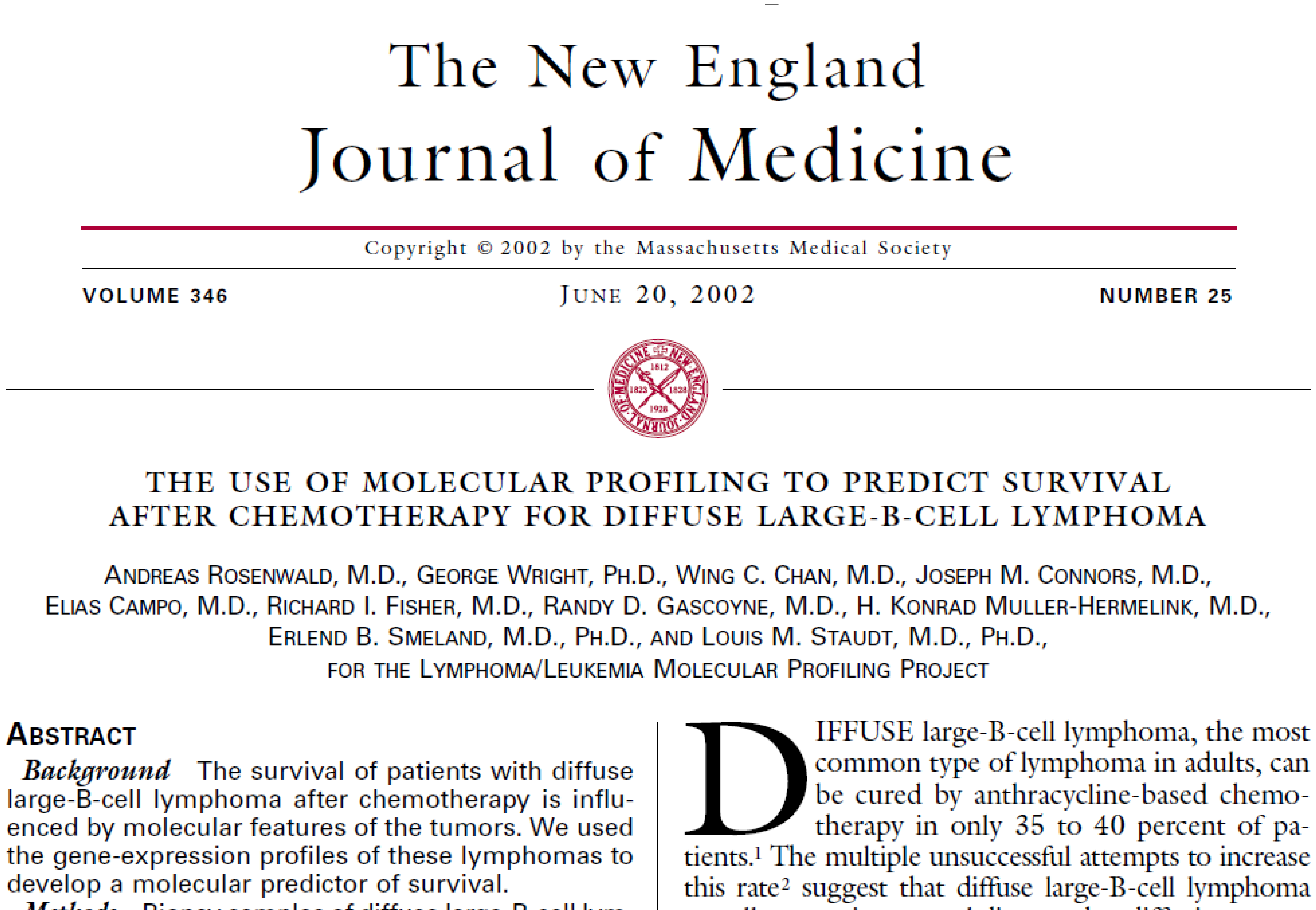
Explore the data

Anja Bråthen Kristoffersen

Biomedical Research Group

Slides and R commands to accompany extra material that may differ from the above talk

http://llmpp.nih.gov/D



The New England Journal of Medicine

Copyright © 2002 by the Massachusetts Medical Society

VOLUME 346 JUNE 20, 2002 NUMBER 25

THE USE OF MOLECULAR PROFILING TO PREDICT SURVIVAL AFTER CHEMOTHERAPY FOR DIFFUSE LARGE-B-CELL LYMPHOMA

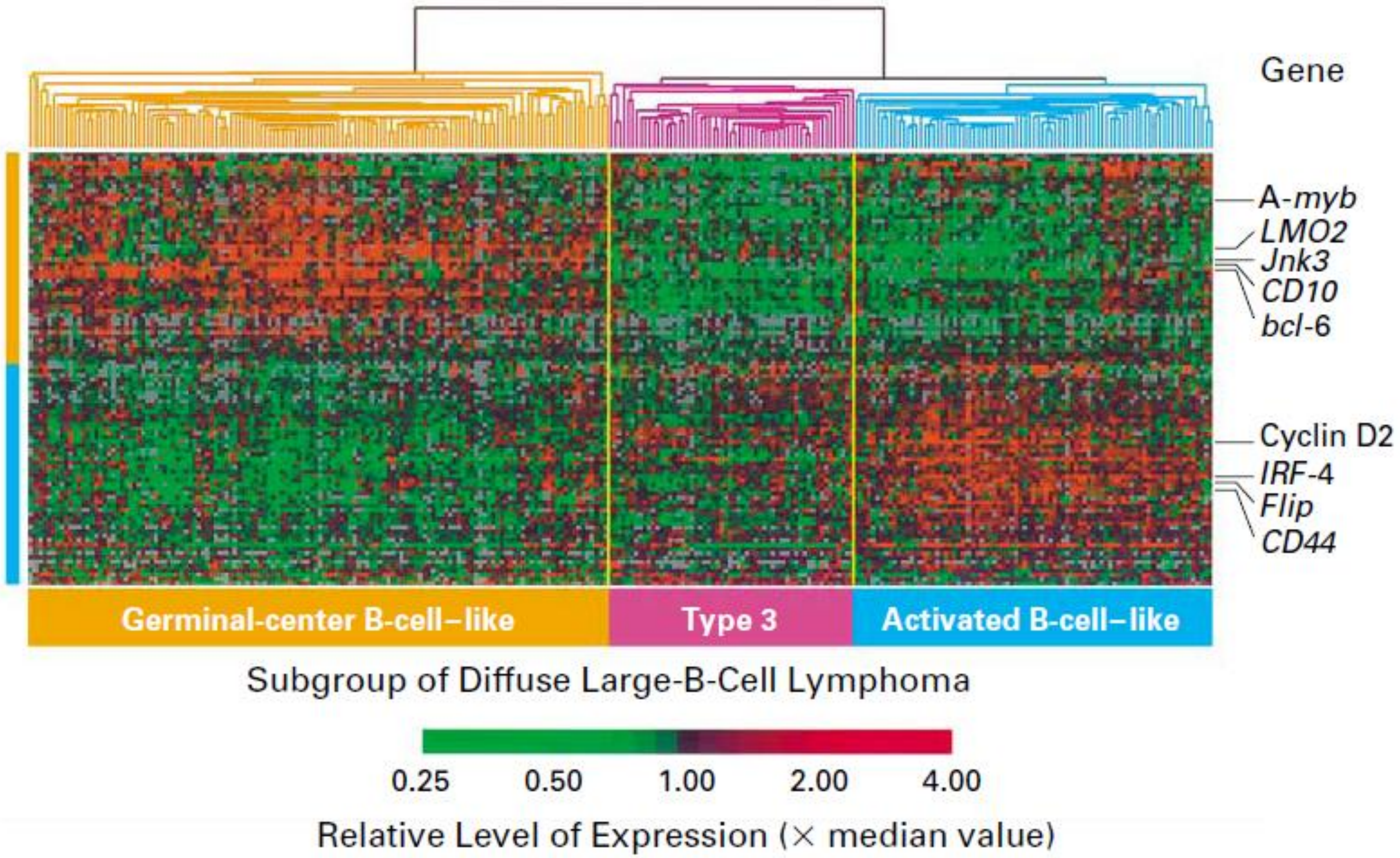
ANDREAS ROSENWALD, M.D., GEORGE WRIGHT, PH.D., WING C. CHAN, M.D., JOSEPH M. CONNORS, M.D., ELIAS CAMPO, M.D., RICHARD I. FISHER, M.D., RANDY D. GASCOYNE, M.D., H. KONRAD MULLER-HERMELINK, M.D., ERLEND B. SMELAND, M.D., PH.D., AND LOUIS M. STAUDT, M.D., PH.D., FOR THE LYMPHOMA/LEUKEMIA MOLECULAR PROFILING PROJECT

ABSTRACT

Background The survival of patients with diffuse large-B-cell lymphoma after chemotherapy is influenced by molecular features of the tumors. We used the gene-expression profiles of these lymphomas to develop a molecular predictor of survival.

DIFFUSE large-B-cell lymphoma, the most common type of lymphoma in adults, can be cured by anthracycline-based chemotherapy in only 35 to 40 percent of patients.[1] The multiple unsuccessful attempts to increase this rate[2] suggest that diffuse large-B-cell lymphoma

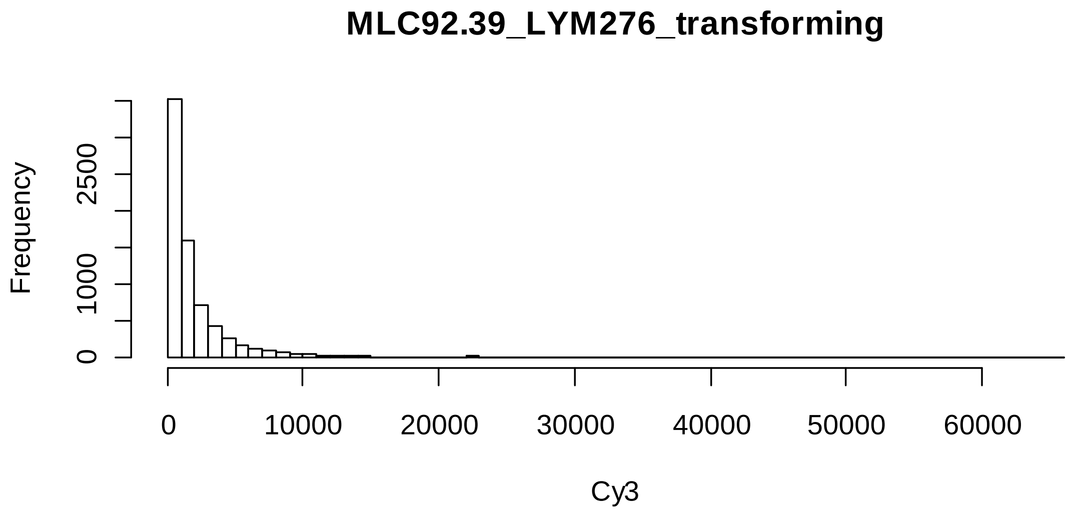
Microarray data

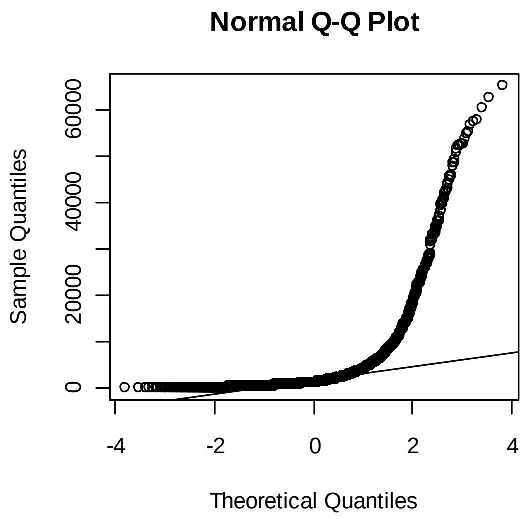


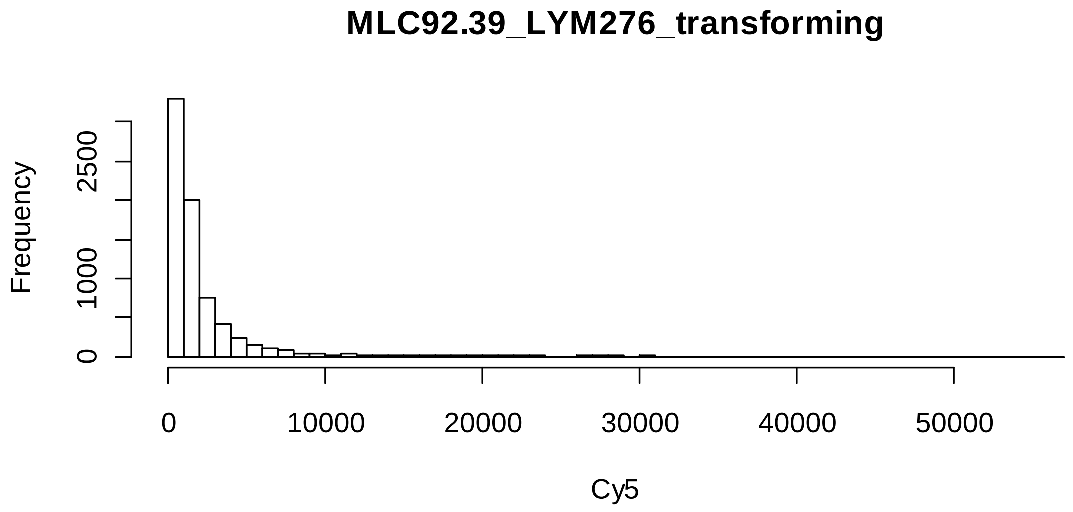
Microarray raw data

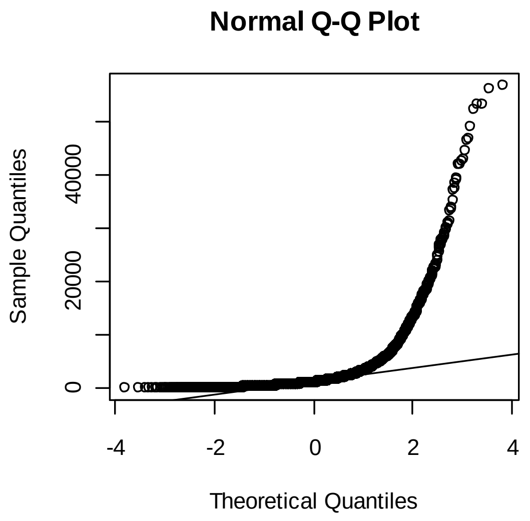
```
Cy3 <- read.table(file="NEJM_Web_Fig1data_CY3.txt", header=TRUE, sep="\t", dec=",")
Cy5 <- read.table(file="NEJM_Web_Fig1data_CY5.txt", header=TRUE, sep="\t", dec=",")

par(mfrow=c(2,1))
hist(Cy3[,55], 50,main=names(Cy3)[55], xlab="Cy3")
hist(Cy5[,55], 50, main=names(Cy3)[55], xlab="Cy5")
```

```
par(mfrow=c(2,1))
qqnorm(Cy3[,55])
qqline(Cy3[,55]
qqnorm(Cy5[,55])
qqline(Cy5[,55])
```

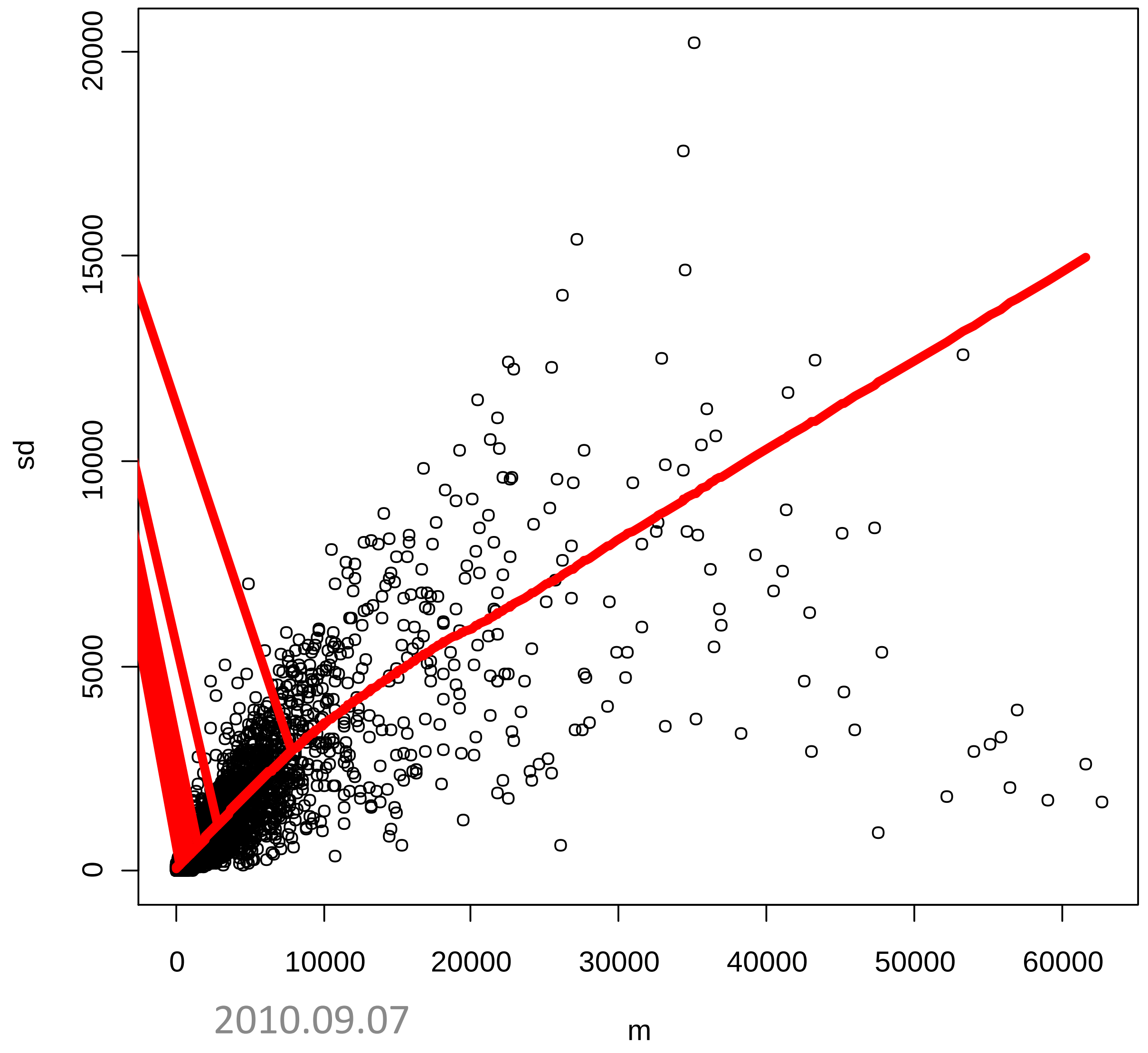








Standard deviation depends on signal



```
# 'apply' will apply the function to all rows of the data matrix
m <- apply(Cy3[,55:58],1,mean,na.rm=TRUE)
sd <- apply(Cy3[,55:58],1,sd,na.rm=TRUE)
plot(m,sd)
trend<-lowess(m,sd)
lines(trend,col=2,lwd=5)
```

🟥 lowess fit

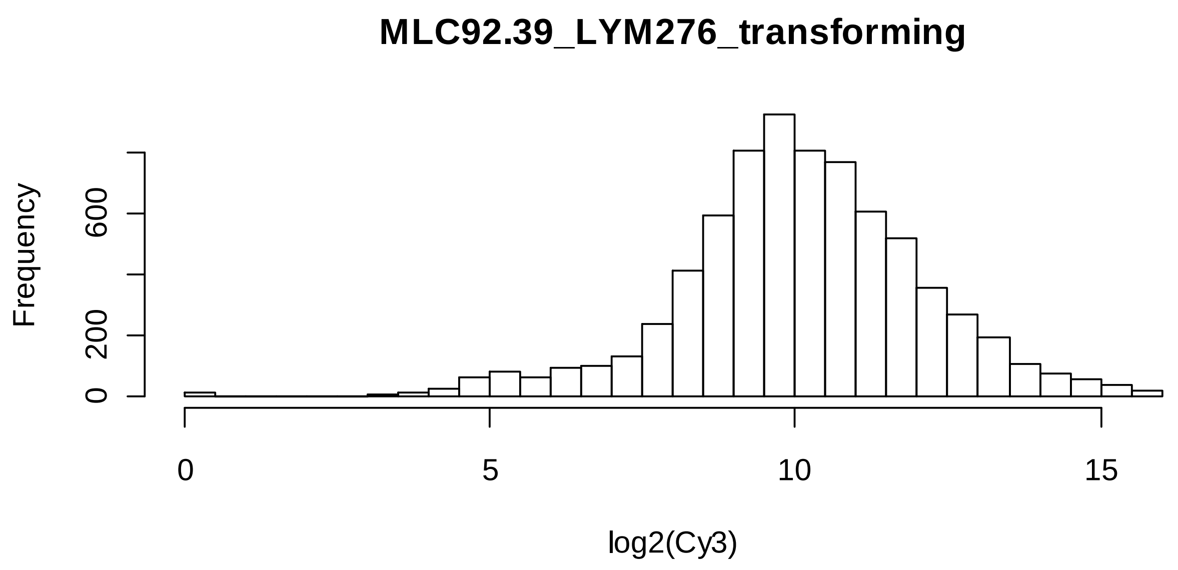
LOcally WEighted Scatter plot Smoother used to estimate the trend in a scatter plot

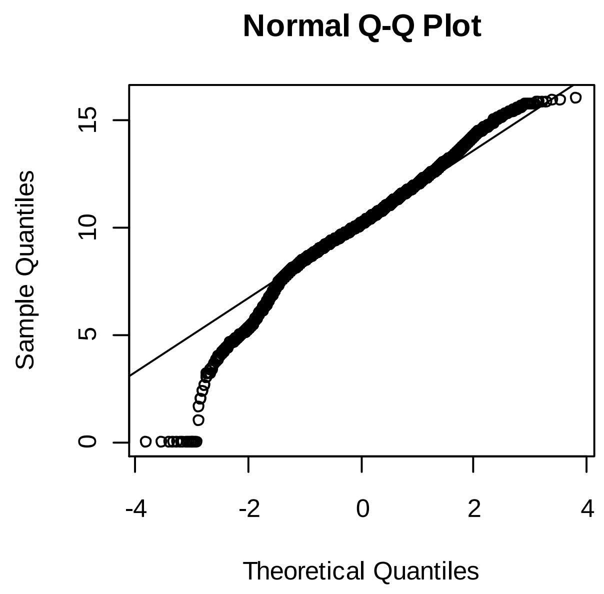
Non parametric!

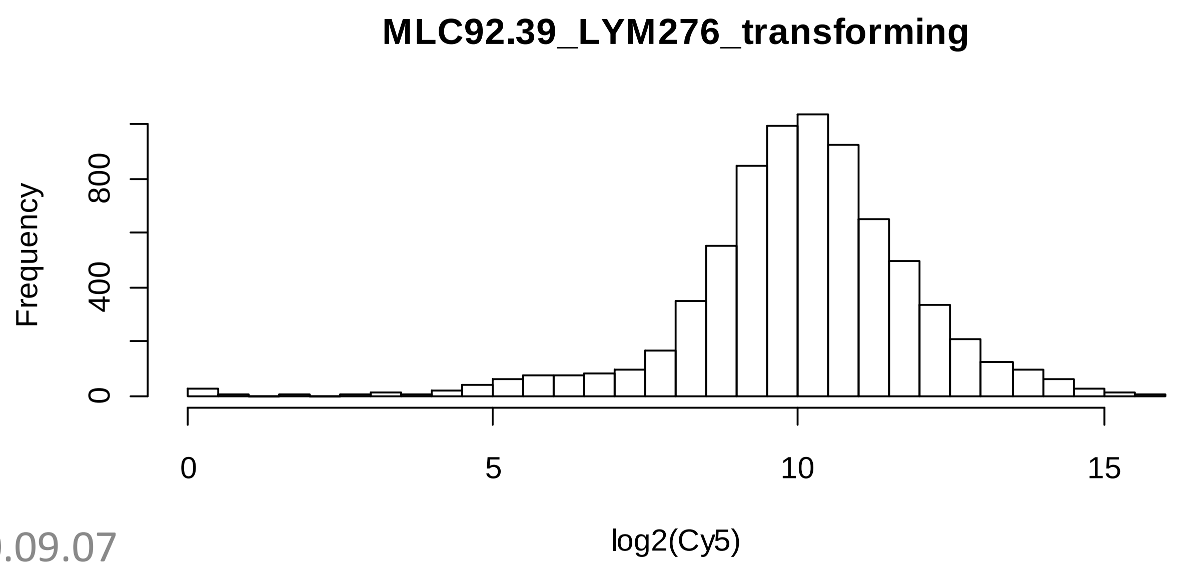
Transformation

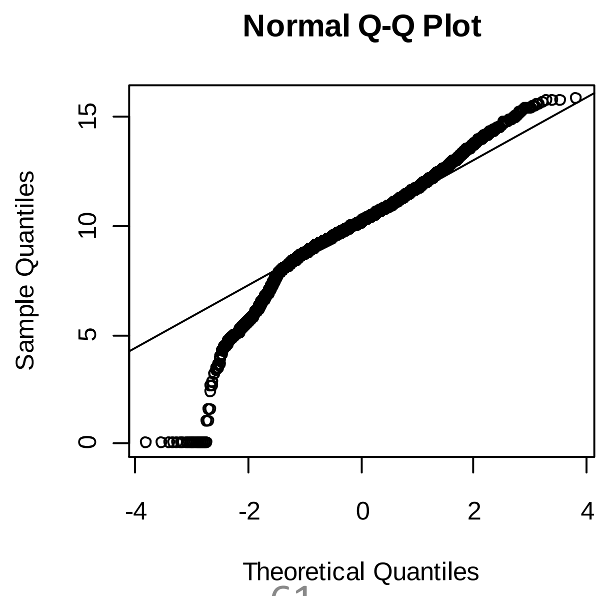
```
par(mfrow=c(2,1))
qqnorm(log2(Cy3[,55]))
qqline(log2(Cy3[,55]))
qqnorm(log2(Cy5[,55]))
qqline(log2Cy5[,55]))
```

```
hist(log2(Cy3[,55]), 50,main=names(Cy3)[55], xlab="log2(Cy3)")
hist(log2(Cy5[,55]), 50, main=names(Cy3)[55], xlab="log2(Cy5)")
```



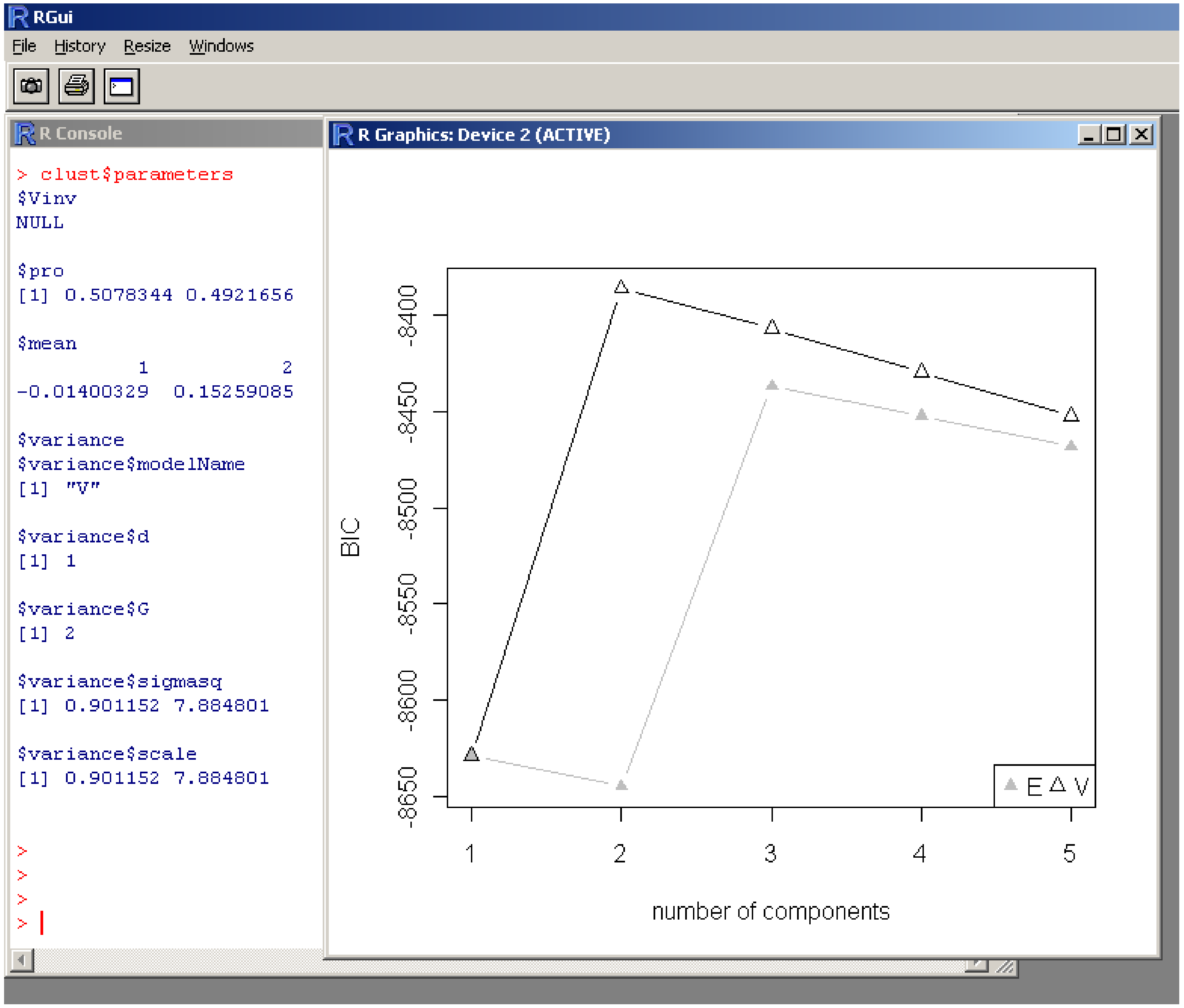




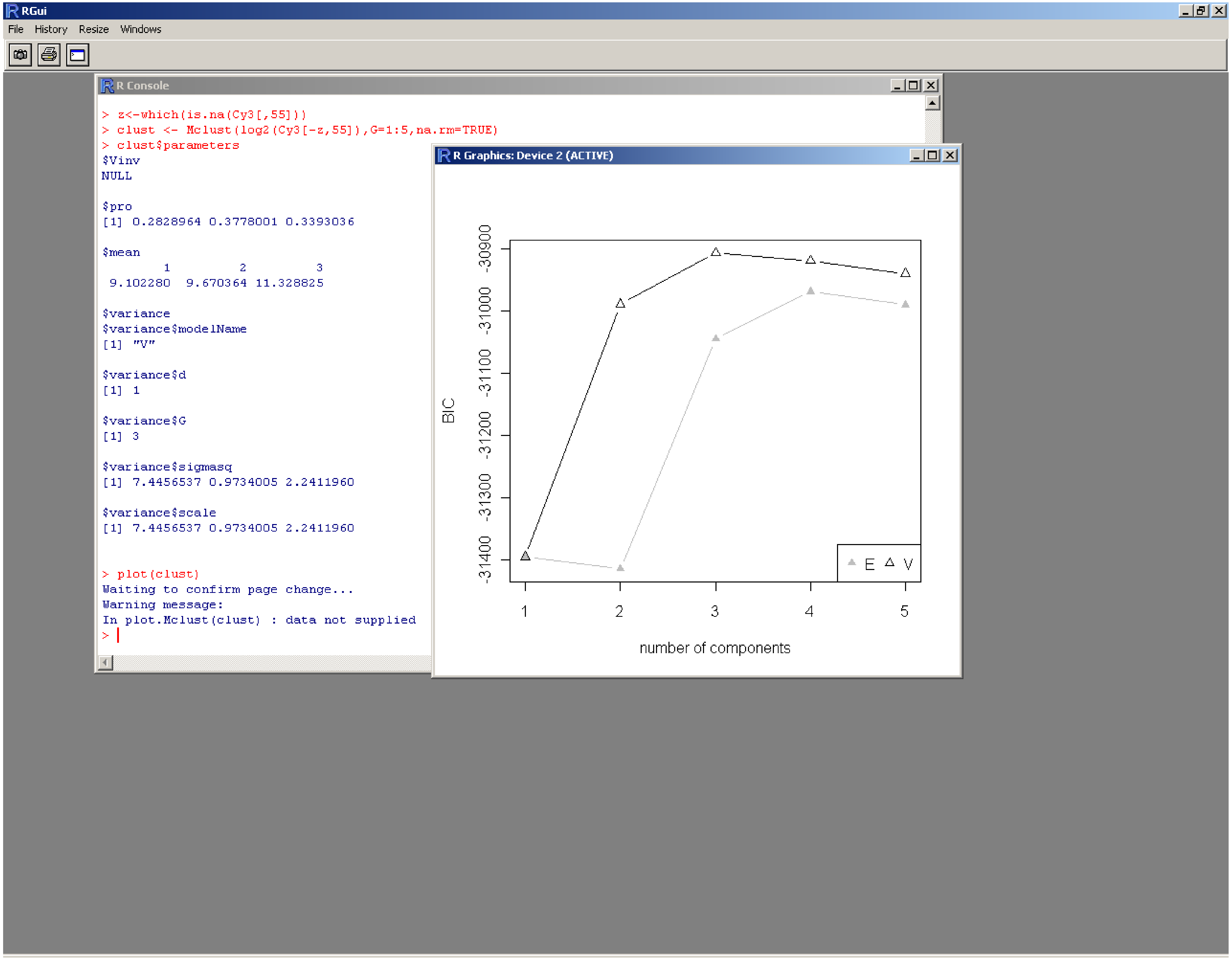


One Gaussian distribution?

```
library(mclust)
y<-rnorm(1000,0,1)
x<-rnorm(1000,0,3)
clust <- Mclust(c(x,y), G=1:5)
plot(clust)
clust$parameters
```



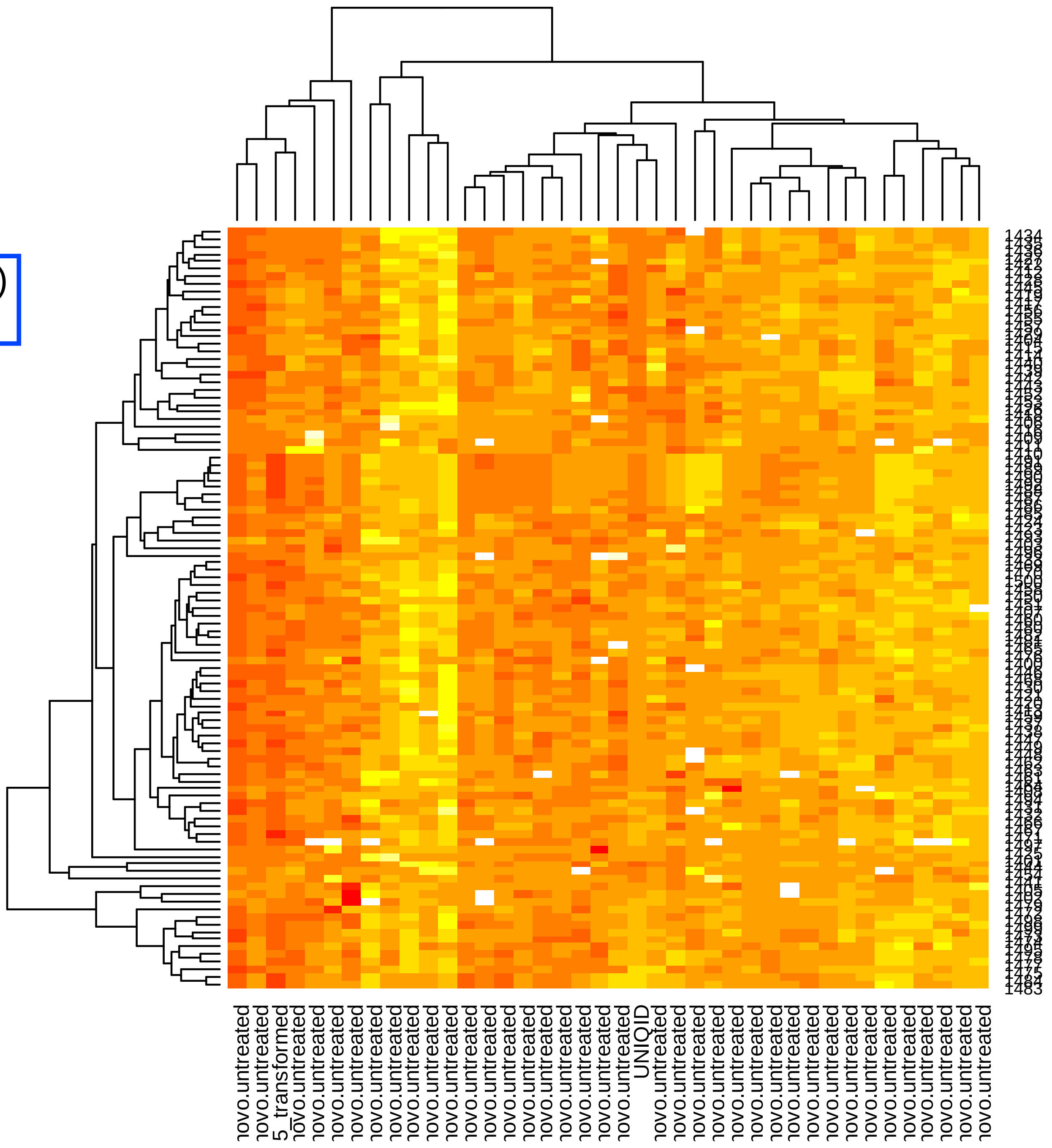
```
> clust$parameters
$Vinv
NULL

$pro
[1] 0.5078344 0.4921656

$mean
          1          2
-0.01400329  0.15259085

$variance
$variance$modelName
[1] "V"

$variance$d
[1] 1

$variance$G
[1] 2

$variance$sigmasq
[1] 0.901152 7.884801

$variance$scale
[1] 0.901152 7.884801


>
>
>
>
```



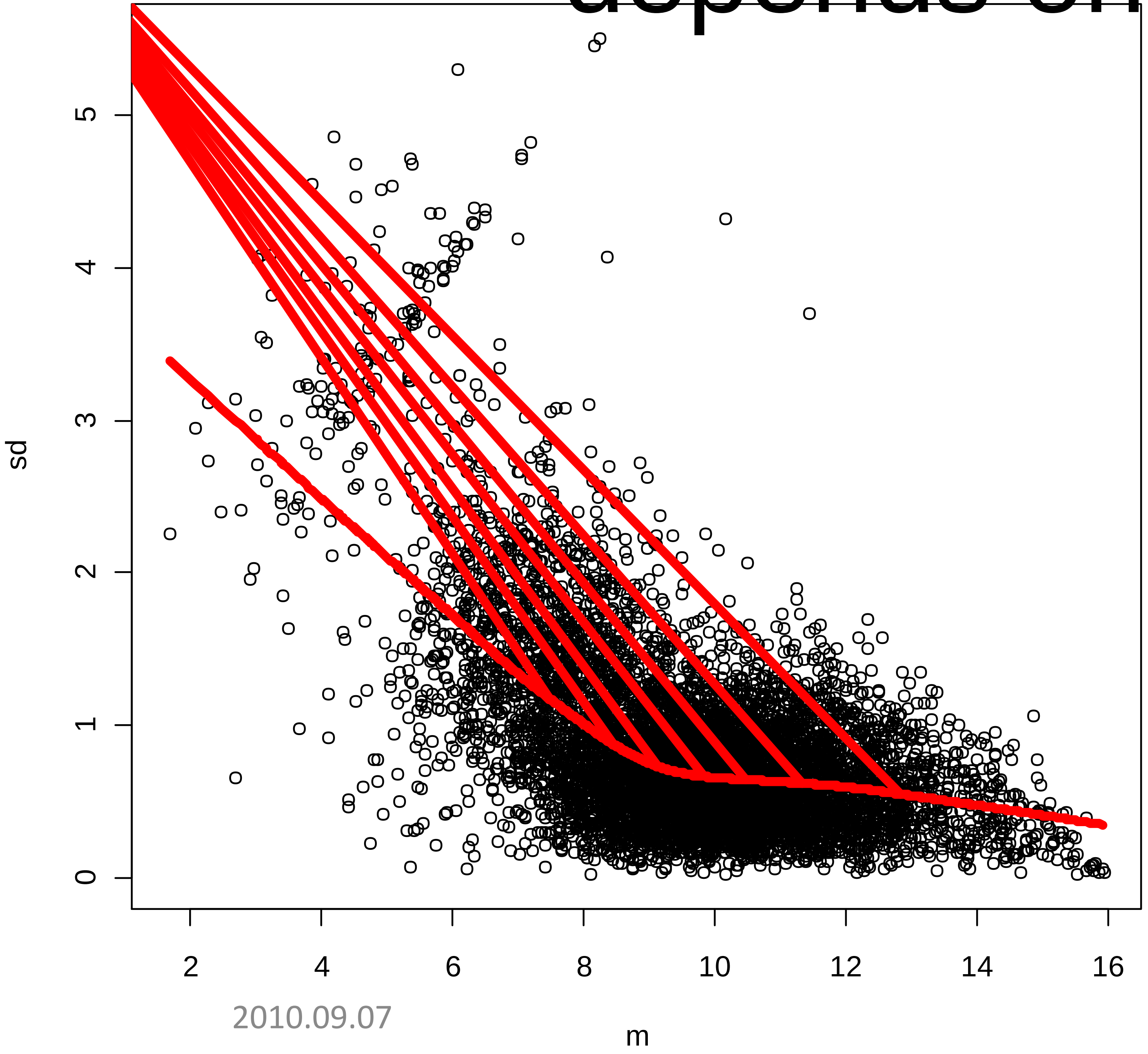
```
> z<-which(is.na(Cy3[,55]))
> clust <- Mclust(log2(Cy3[-z,55]),G=1:5,na.rm=TRUE)
> clust$parameters
$Vinv
NULL

$pro
[1] 0.2828964 0.3778001 0.3393036

$mean
        1         2         3
 9.102280  9.670364 11.328825

$variance
$variance$modelName
[1] "V"

$variance$d
[1] 1

$variance$G
[1] 3

$variance$sigmasq
[1] 7.4456537 0.9734005 2.2411960

$variance$scale
[1] 7.4456537 0.9734005 2.2411960


> plot(clust)
Waiting to confirm page change...
Warning message:
In plot.Mclust(clust) : data not supplied
>
```

Heatmap

```
z<-as.matrix(log2(Cy3[1400:1500,1:40])-log2(Cy5[1400:1500,1:40]))
heatmap(z)
```



Standard deviation depends on signal



```
# 'apply' will apply the function to all rows of the data matrix
m <- apply(log2(Cy3[,55:58]),1,mean,na.rm=TRUE)
sd <- apply(log2(Cy3[,55:58]),1,sd,na.rm=TRUE)
plot(m,sd)
trend<-lowess(m,sd)
lines(trend,col=2,lwd=5)
```

But the dependency is weaker

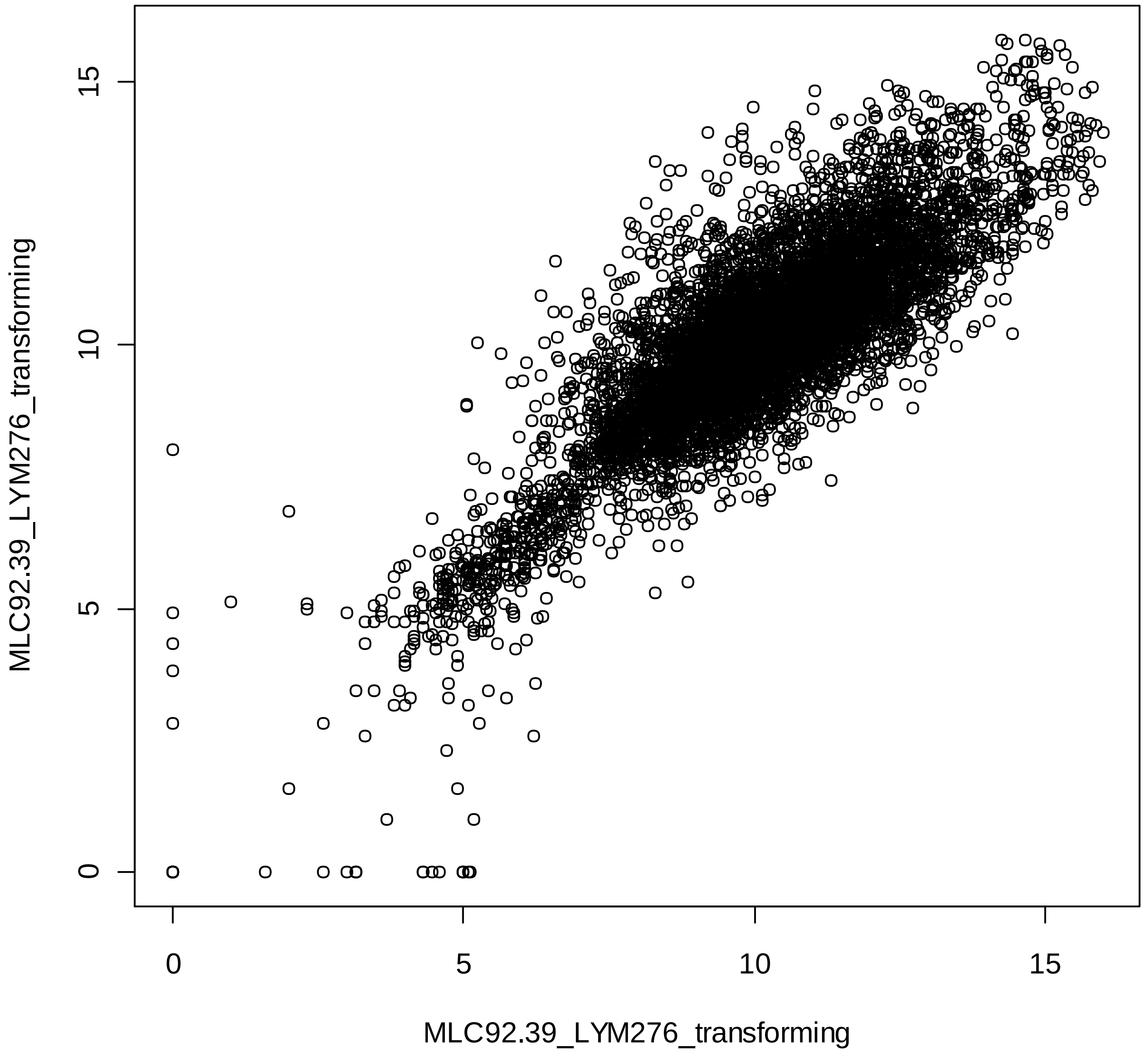
Especially where most of the data are located.

gene expression

```
plot(Cy3[,55],Cy5[,55], xlab=names(Cy3)[55], ylab=names(Cy5)[55])
```

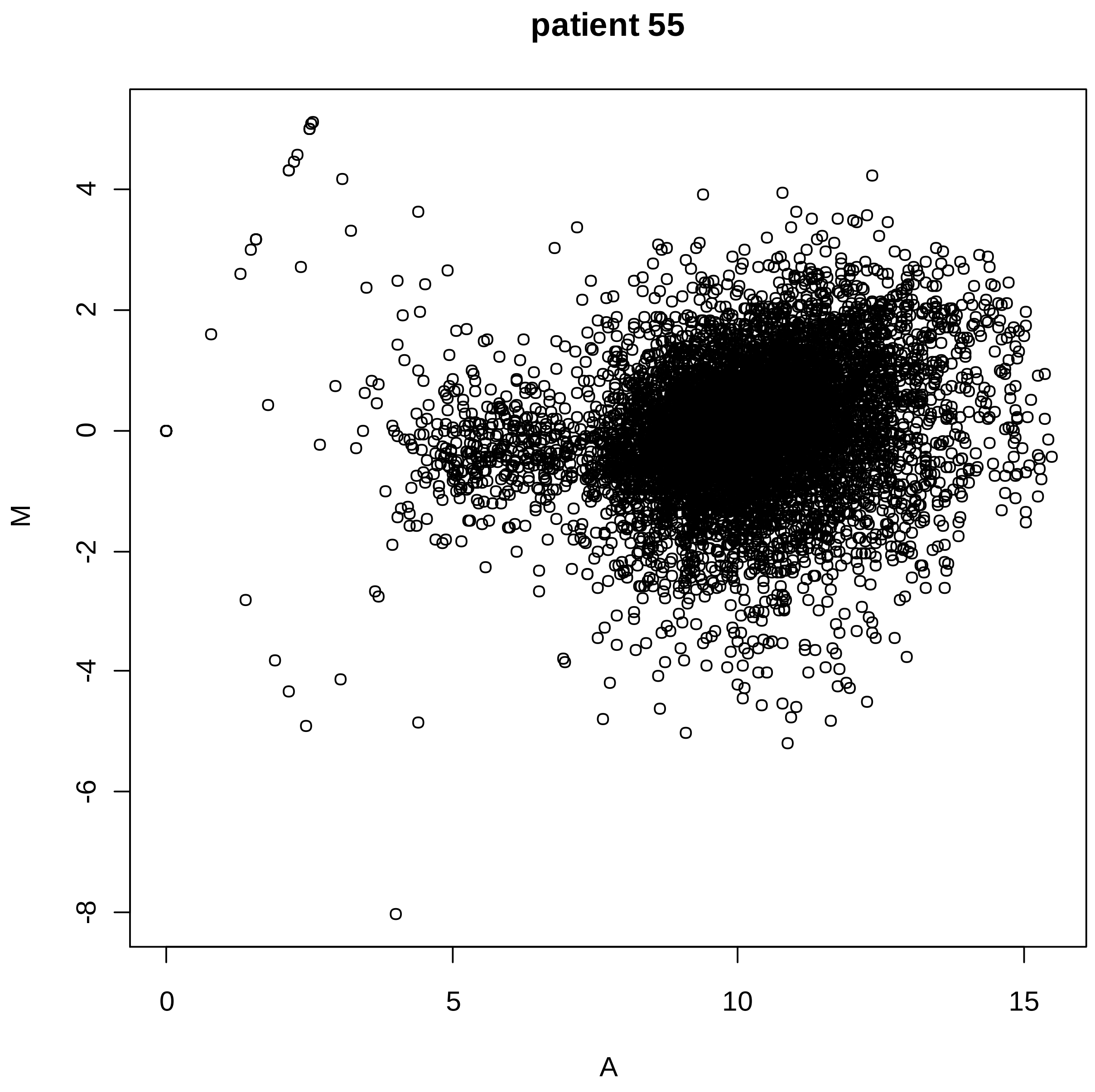
What can you say?

Is this the best way to look at the data?



MA plots

```
# MA plots per replicate
A<-(log2(Cy3[,55])+log2(Cy5[,55]))/2
M<-(log2(Cy3[,55])-log2(Cy5[,55]))
plot(A,M,xlab="A",ylab="M",main="patient 55")
```

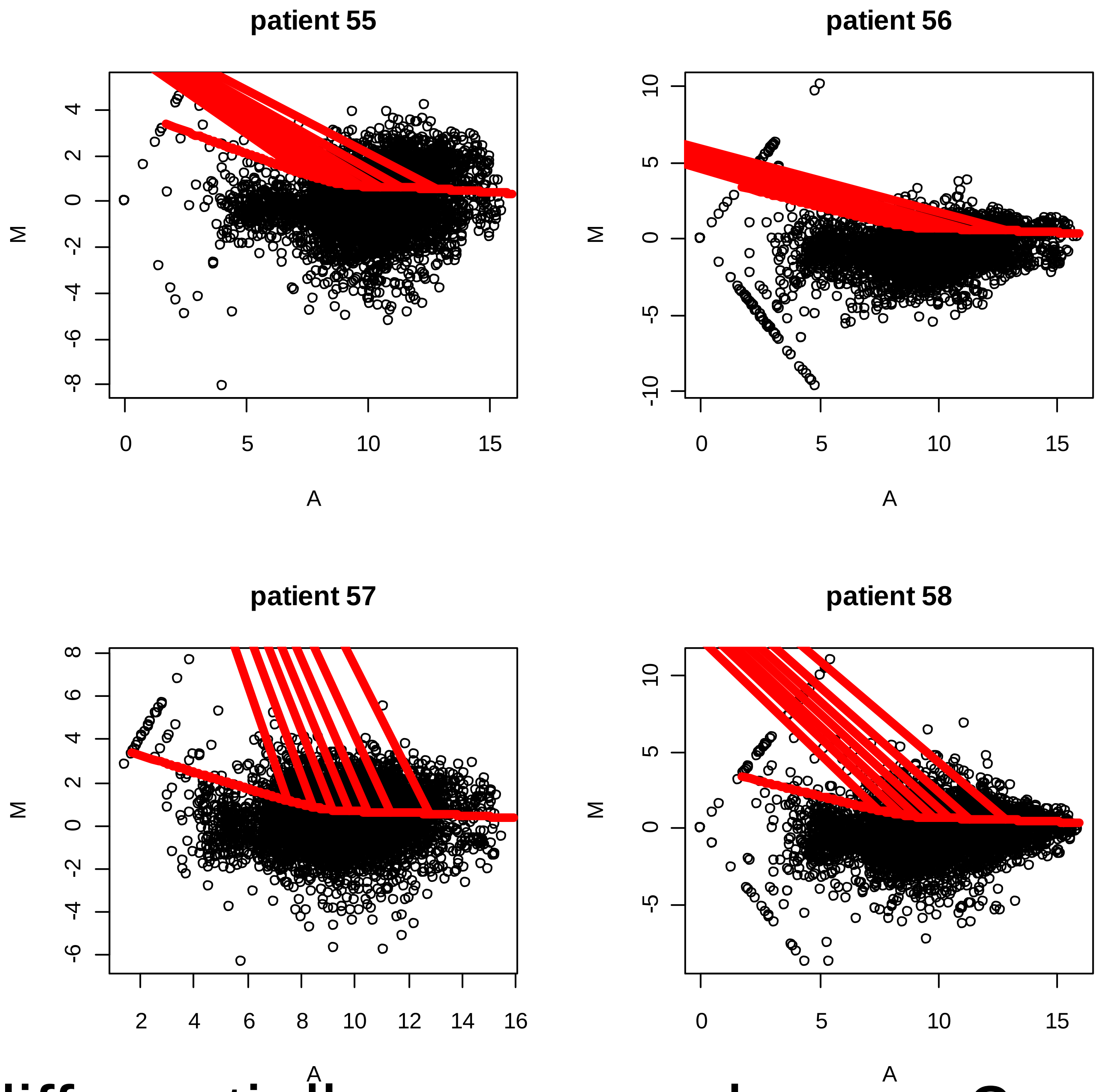


M (minus) is the log ratio

A (average) is overall intensity

MA plots

```
par(mfrow=c(2,2))
A<-(log2(Cy3[,55])+log2(Cy5[,55]))/2
M<-(log2(Cy3[,55])-log2(Cy5[,55]))
plot(A,M,xlab="A",ylab="M",main="patient 55")
trend<-lowess(A,M)
lines(trend,col=2,lwd=5)

A<-(log2(Cy3[,56])+log2(Cy5[,56]))/2
M<-(log2(Cy3[,56])-log2(Cy5[,56]))
plot(A,M,xlab="A",ylab="M",main="patient 56")
trend<-lowess(A,M)
lines(trend,col=2,lwd=5)

A<-(log2(Cy3[,57])+log2(Cy5[,57]))/2
M<-(log2(Cy3[,57])-log2(Cy5[,57]))
plot(A,M,xlab="A",ylab="M",main="patient 57")
trend<-lowess(A,M)
lines(trend,col=2,lwd=5)

A<-(log2(Cy3[,58])+log2(Cy5[,58]))/2
M<-(log2(Cy3[,58])-log2(Cy5[,58]))
plot(A,M,xlab="A",ylab="M",main="patient 58")
trend<-lowess(A,M)
lines(trend,col=2,lwd=5)
```

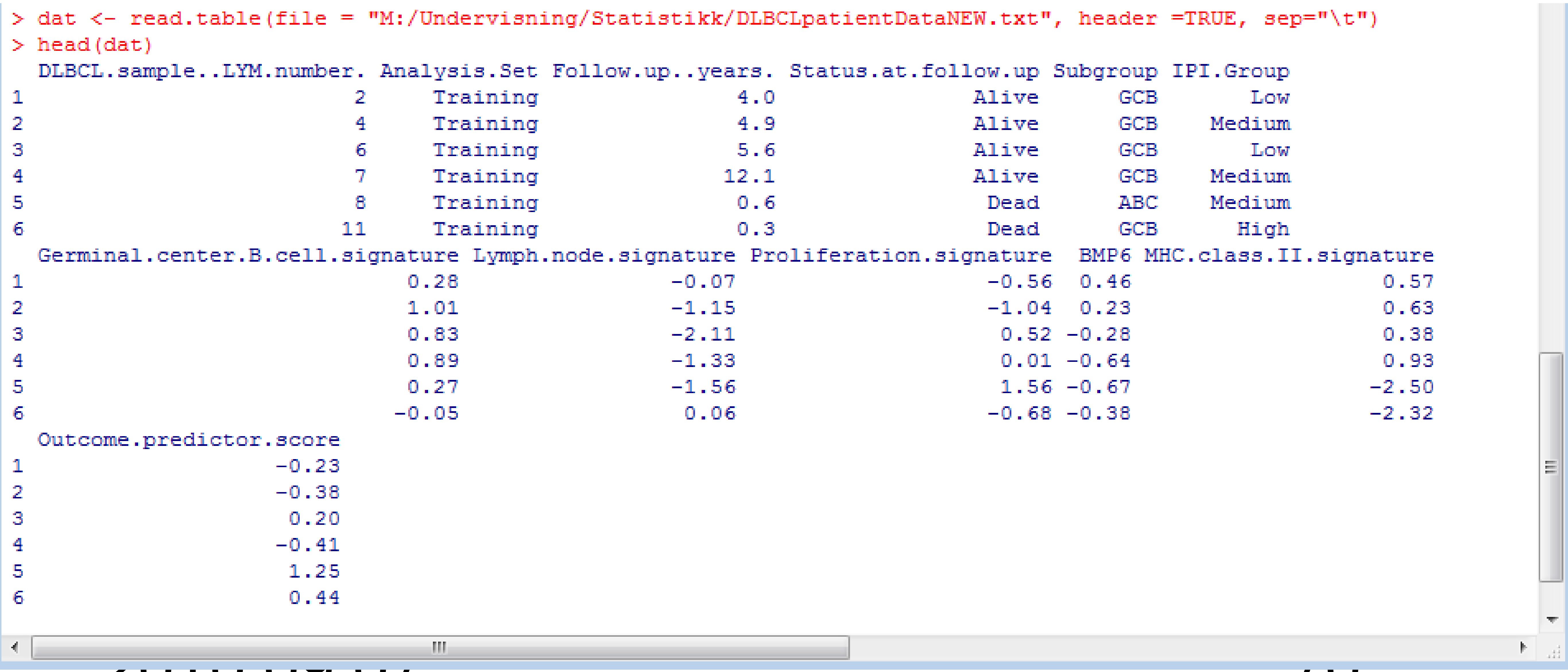


How do we find differentially expressed genes?

Combining micro-array and survival data

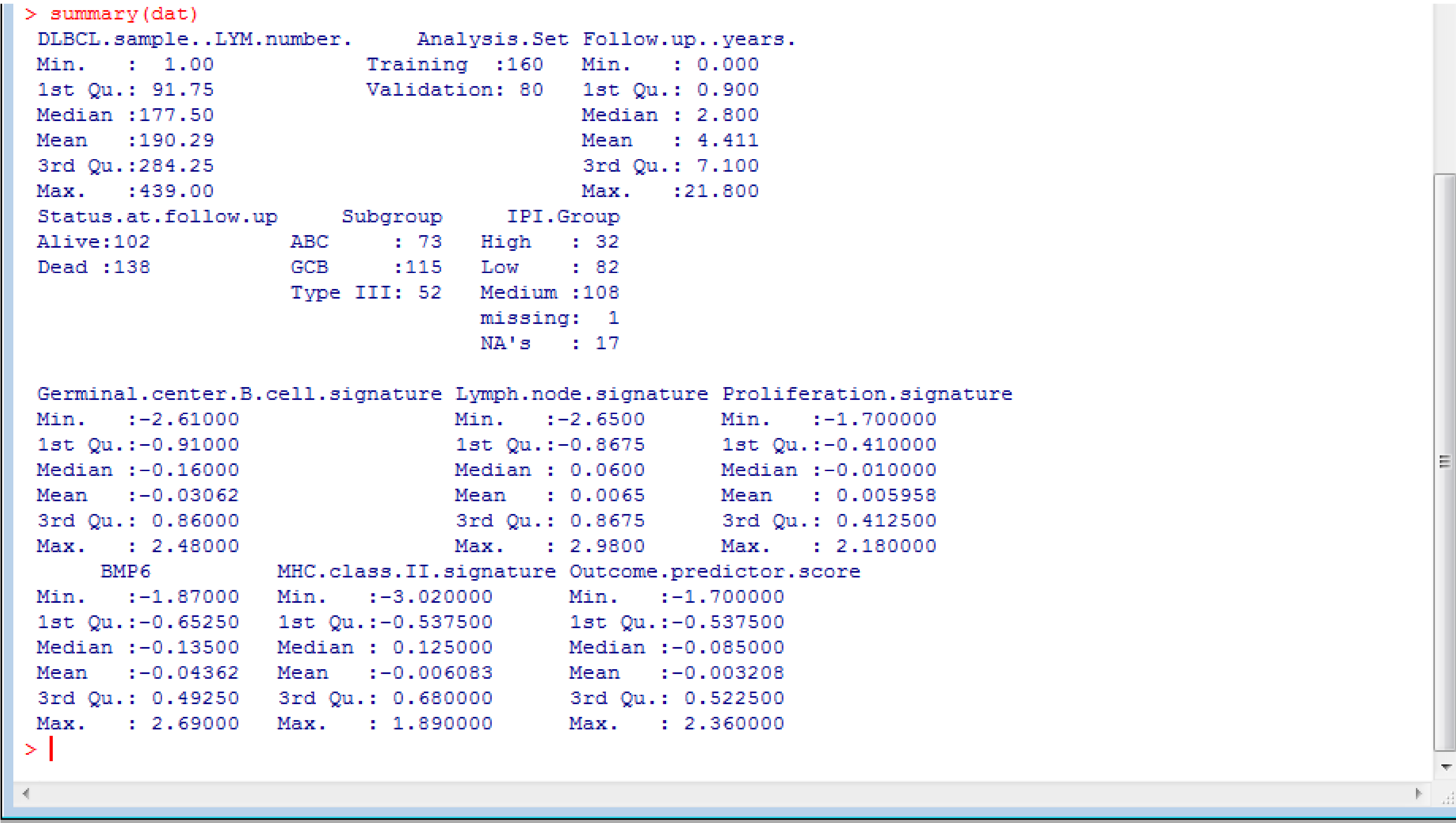
- For each patient five signature are calculated from the micro-array as the mean of the signal from each of the group of genes:
  - Germinal.center.B.cell.signature
  - Lymph.node.signature
  - Proliferation.signature
  - BMP6
  - MHC.class.II.signature

head(dat)



```
> dat <- read.table(file = "M:/Undervisning/Statistikk/DLBCLpatientDataNEW.txt", header =TRUE, sep="\t")
> head(dat)
  DLBCL.sample..LYM.number. Analysis.Set Follow.up..years. Status.at.follow.up Subgroup IPI.Group
1                         2     Training               4.0               Alive      GCB       Low
2                         4     Training               4.9               Alive      GCB    Medium
3                         6     Training               5.6               Alive      GCB       Low
4                         7     Training              12.1               Alive      GCB    Medium
5                         8     Training               0.6                Dead      ABC    Medium
6                        11     Training               0.3                Dead      GCB      High
  Germinal.center.B.cell.signature Lymph.node.signature Proliferation.signature  BMP6 MHC.class.II.signature
1                             0.28                -0.07                   -0.56  0.46                   0.57
2                             1.01                -1.15                   -1.04  0.23                   0.63
3                             0.83                -2.11                    0.52 -0.28                   0.38
4                             0.89                -1.33                    0.01 -0.64                   0.93
5                             0.27                -1.56                    1.56 -0.67                  -2.50
6                            -0.05                 0.06                   -0.68 -0.38                  -2.32
  Outcome.predictor.score
1                   -0.23
2                   -0.38
3                    0.20
4                   -0.41
5                    1.25
6                    0.44
```

summary(dat)



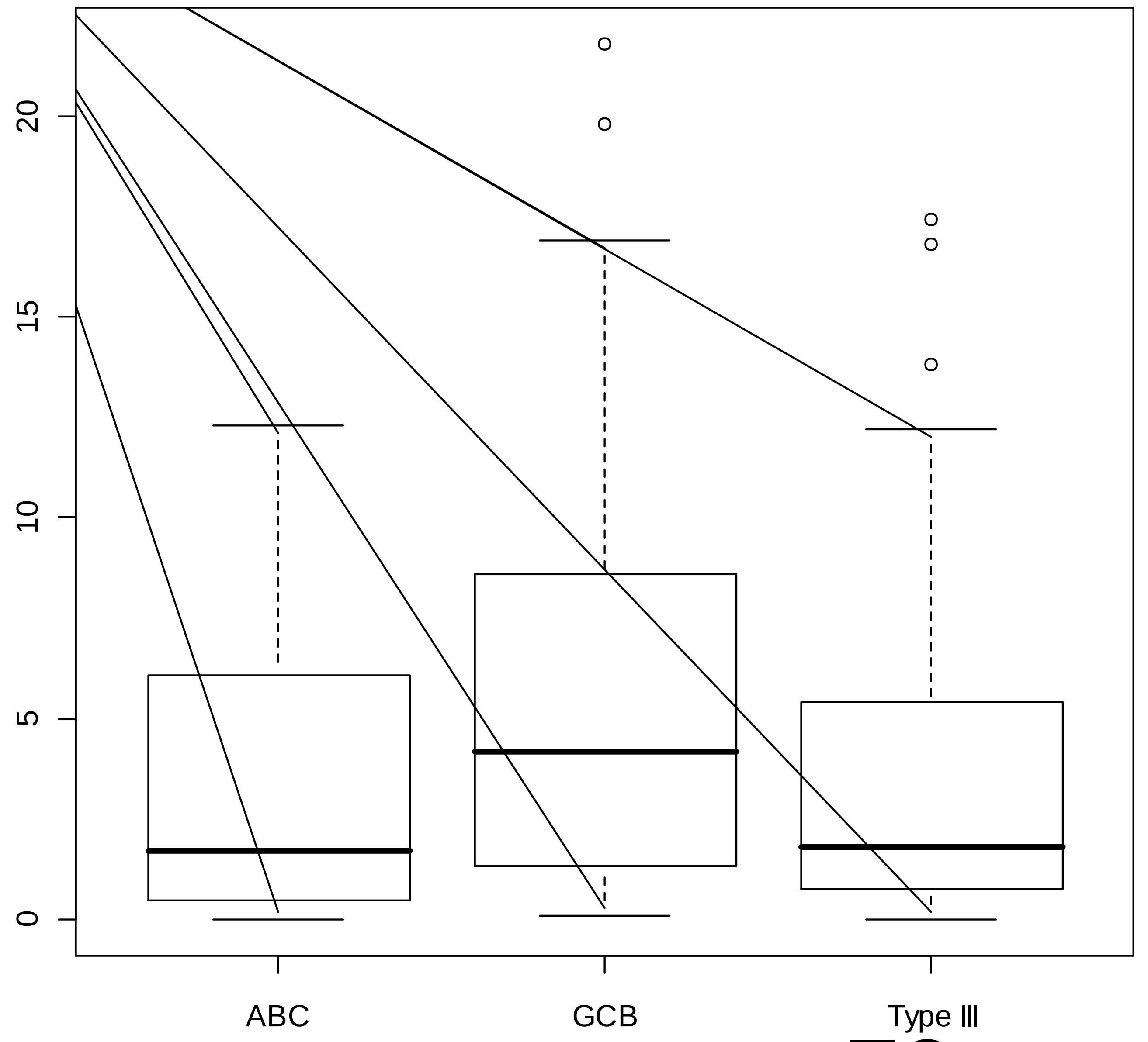
```
> summary(dat)
 DLBCL.sample..LYM.number.     Analysis.Set Follow.up..years.
 Min.   :  1.00           Training  :160   Min.   : 0.000
 1st Qu.: 91.75           Validation: 80   1st Qu.: 0.900
 Median :177.50                            Median : 2.800
 Mean   :190.29                            Mean   : 4.411
 3rd Qu.:284.25                            3rd Qu.: 7.100
 Max.   :439.00                            Max.   :21.800
 Status.at.follow.up     Subgroup      IPI.Group
 Alive:102           ABC      : 73   High    : 32
 Dead :138           GCB      :115   Low     : 82
                     Type III: 52   Medium  :108
                                     missing:  1
                                     NA's   : 17

 Germinal.center.B.cell.signature Lymph.node.signature Proliferation.signature
 Min.   :-2.61000                 Min.   :-2.6500      Min.   :-1.700000
 1st Qu.:-0.91000                 1st Qu.:-0.8675      1st Qu.:-0.410000
 Median :-0.16000                 Median : 0.0600      Median :-0.010000
 Mean   :-0.03062                 Mean   : 0.0065      Mean   : 0.005958
 3rd Qu.: 0.86000                 3rd Qu.: 0.8675      3rd Qu.: 0.412500
 Max.   : 2.48000                 Max.   : 2.9800      Max.   : 2.180000
      BMP6            MHC.class.II.signature Outcome.predictor.score
 Min.   :-1.87000   Min.   :-3.020000      Min.   :-1.700000
 1st Qu.:-0.65250   1st Qu.:-0.537500      1st Qu.:-0.537500
 Median :-0.13500   Median : 0.125000      Median :-0.085000
 Mean   :-0.04362   Mean   :-0.006083      Mean   :-0.003208
 3rd Qu.: 0.49250   3rd Qu.: 0.680000      3rd Qu.: 0.522500
 Max.   : 2.69000   Max.   : 1.890000      Max.   : 2.360000
> 
```

Boxplot: follow up time for each subgroup

```
boxplot(Follow.up..years.~Subgroup, data = dat)
```

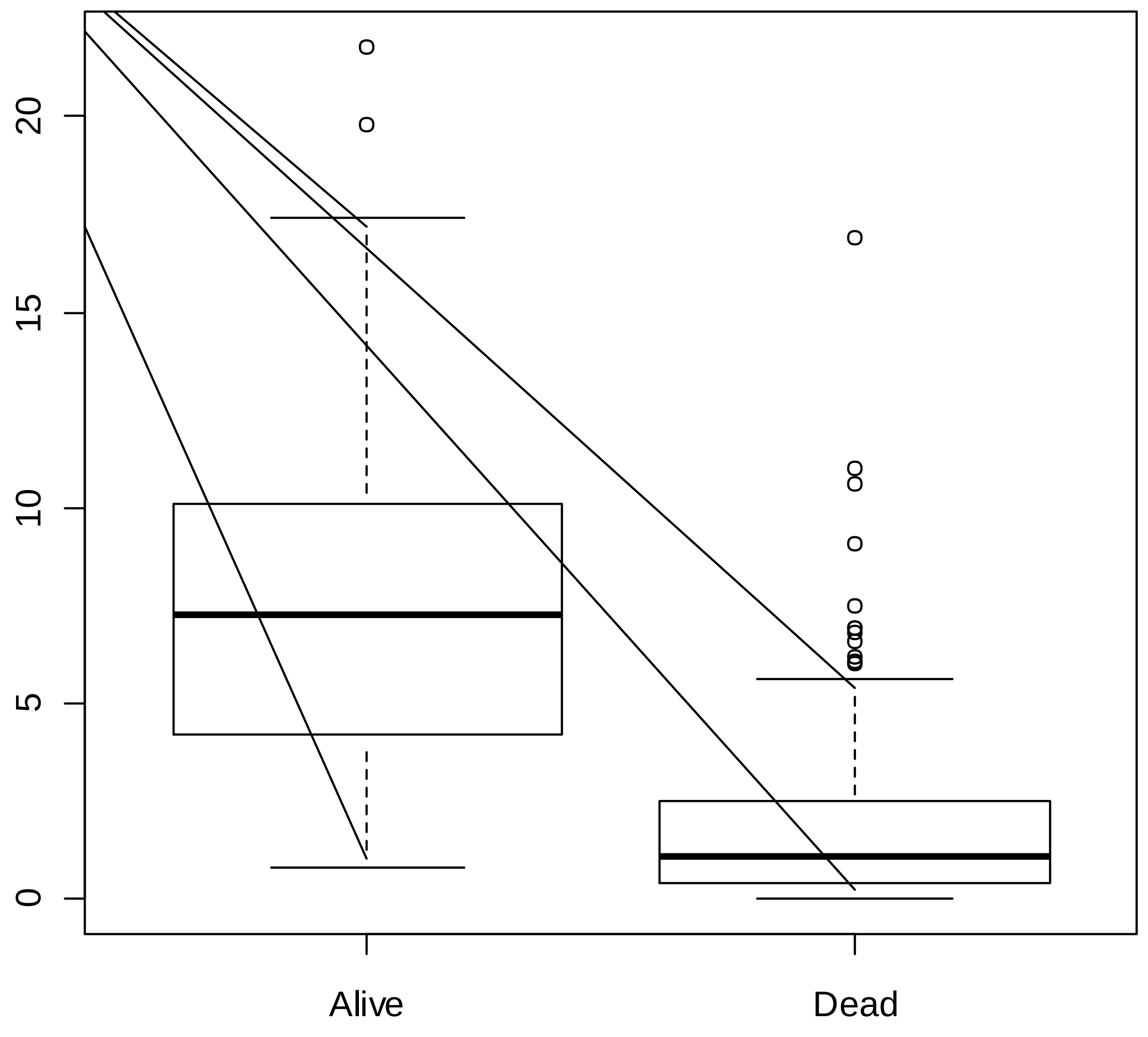
The boxplot function can be used to display several variables at a time!

What can you say here?



Boxplot: follow up time for each subgroup

```
boxplot(Follow.up..years.~Status.at.follow.up, data = dat)
```

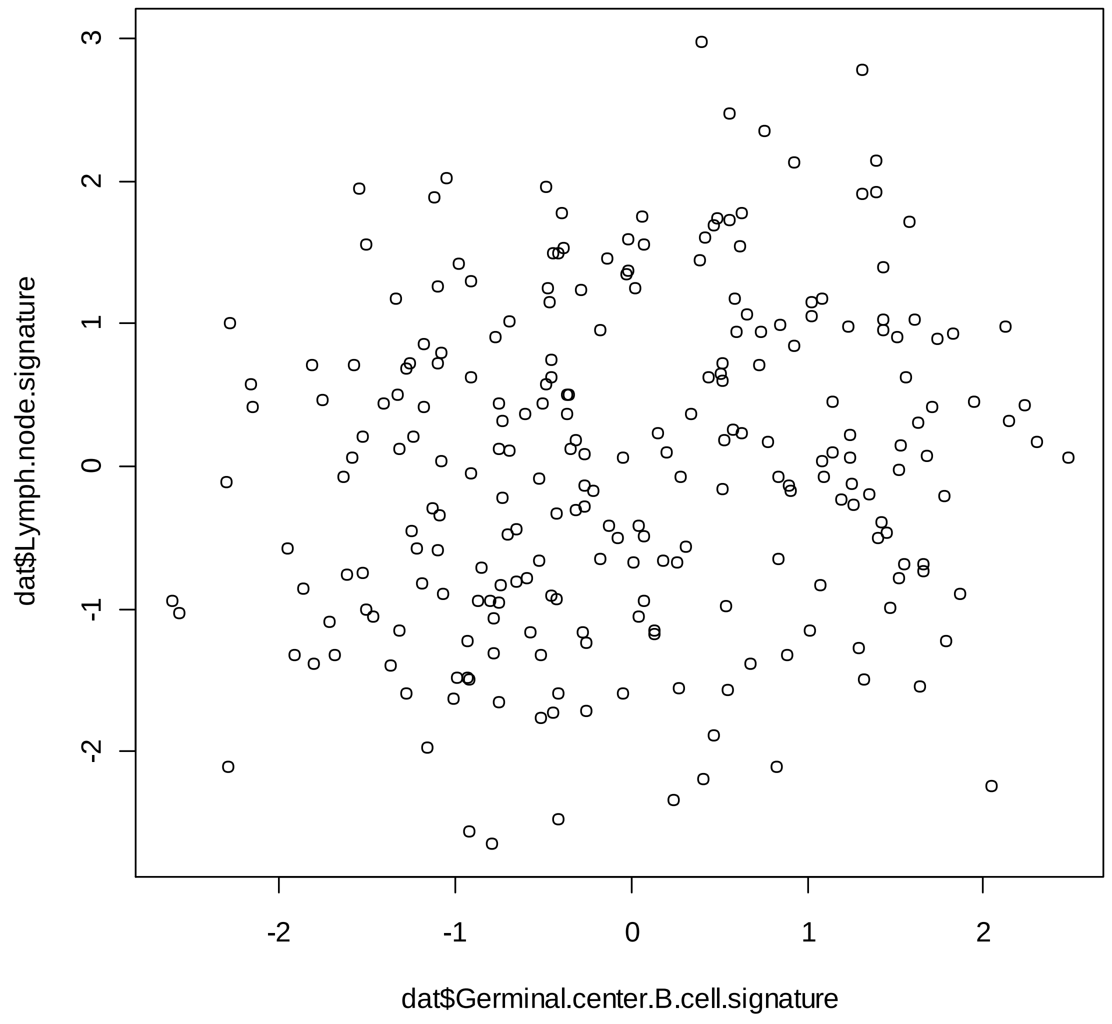


Scatter plots

Biological data sets often contain several variables
So they are multivariate.

Scatter plots allow us to look at two variables at a time

```
plot(dat$Germinal.center.B.cell.signature,dat$Lymph.node.signature)
cor(dat$Germinal.center.B.cell.signature,dat$Lymph.node.signature)
#[1] 0.1633608
```



This can be used
to assess independence!

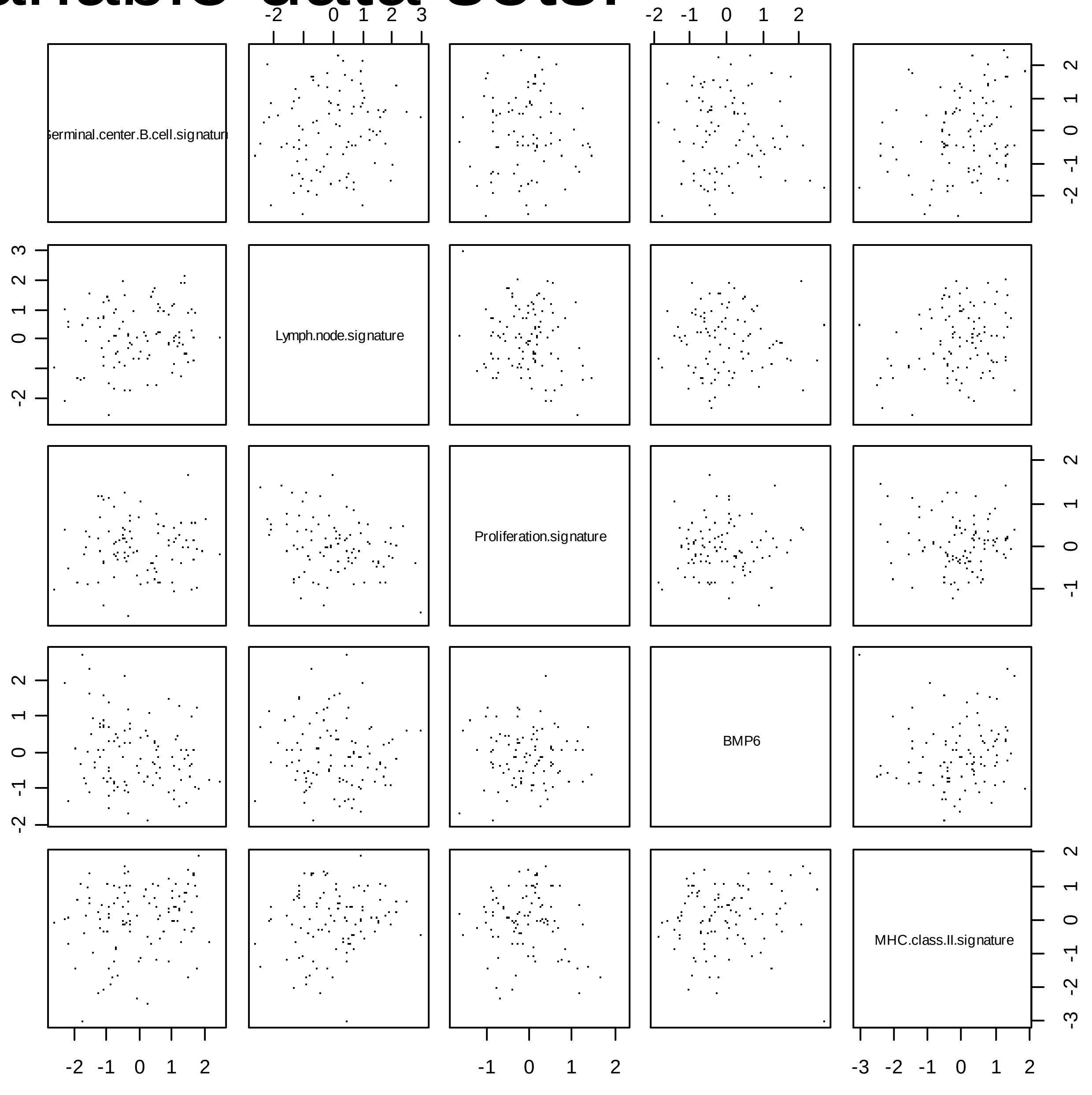
Trellis graphics

Trellis Graphics is a family of techniques for viewing complex, multi-variable data sets.

```
plot(dat[,7:11] pch=".")
```

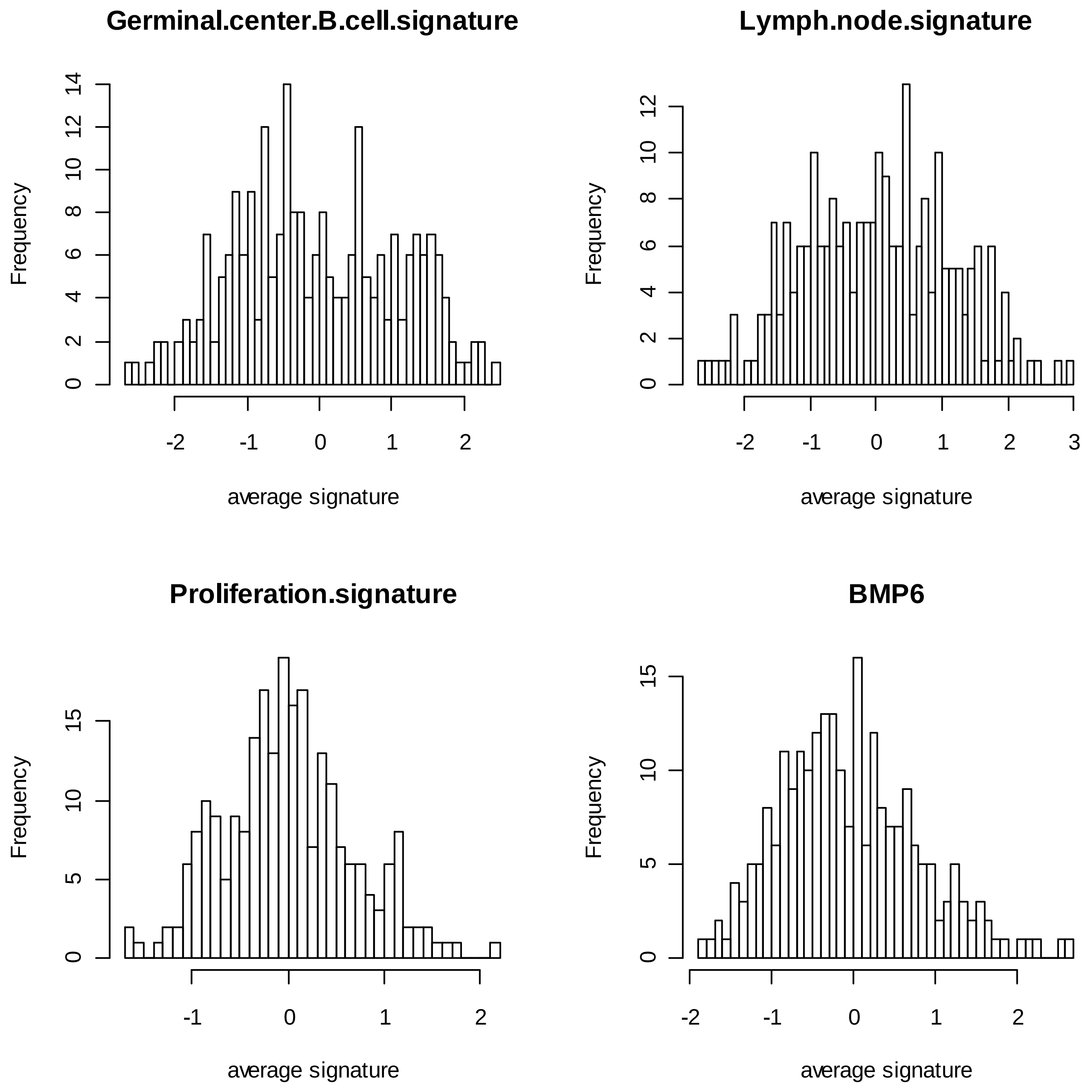
Note that the plotting symbol is changed.

Many more possibilities in the 'lattice' package!



Histogram

```
par(mfrow=c(2,2))
hist(dat[,7], 50, main = names(dat)[7], xlab="average signature")
hist(dat[,8], 50, main = names(dat)[8], xlab="average signature")
hist(dat[,9], 50, main = names(dat)[9], xlab="average signature")
hist(dat[,10], 50, main = names(dat)[10], xlab="average signature")
```


Germinal.center.B.cell.signature
Lymph.node.signature
Proliferation.signature
BMP6
Frequency
average signature