Seminar III: R/Bioconductor

Misc. Stats and affy Quality Control.

José Víctor Moreno Mayar
jmoreno@lcg.unam.mx

LCG - UNAM

August - December, 2009

Misc. Stats and affy Control Quality.

1. Packages for this class.
2. lowess.
3. loess.
4. Affymetrix Chips.
5. AffyBatch
6. phenoData

Misc. Stats and affy Control Quality.

7 Affy quality Control.

Packages for this class.

For this class we will use two functions already included in R, and the affy package.

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("affy")
```

Some theory.

- It is a method for making smooth regressions of scatter plots.
- It uses the least squares method.
- Let us recall the method.

Least Squares.

- It consists, mainly, on fitting a curve to a given set of points.
- The total sum of the residuals has to be minimized. [1]
- The function depends on the degree of the polynomial that should be fitted.
- A system of linear equations is gotten so the coefficients of the regression curve are calculated.

[1] The square of the error is used, so it is called least squares

Locally Weighted Regression.

- The difference here is the weighting function W(x) used for each x_i.
- So each pair (x_i, y_i) has a different effect on the regression.
- Even more, W(x) is calculated based on the neighborhood of x_i.

The Weighting Function.

- An initial set of weights $w_k(x_i)$ is calculated for each x_i.
- An initial fitted value $\hat{y}_i$ is calculated for each x_i.
- This is made by means of weigthed least squares.
- The point is fitted to a *dth* degree polynomial function (usually, $d = 1$).

f and The Smoothness Assumption.

- The fact that the nieghbors of an x_i may be used in calculating its set of weights, makes an assumption of smoothness.
- So, you can select how wide will the neighborhood be.
- This is made through the paremeter f.
- Different values of f may be taken from the interval (0,1].
- The larger the f, the wider the neighborhood and the smoother the points will be.

Robust Lowess.

- The way to make this regression robust is by calculating a better $\hat{y}_i$.
- The regression is guarded against deviant points, which might distort the smoothed point by reiterating the procedure.
- This time, the computing of the $\hat{y}_i$ will be made with a different set of weights δ_i.
- δ_i is calculated based on the size of the residual $y_i - \hat{y}_i$.
  - Large residuals result in small weights and small residuals result in large weights.

Lowess in R.

- The function in R to generate this curves is lowess().
- The arguments to be passed to the function are the two related (or not) variables, f, which is the span of the smoothing process and the number of times the process should be iterated.
- Time for some practice.

An example.

Let us create a data frame with two variables x and y where:

- x are the first 151 consecutive natural numbers.
- $y_i = .2x_i + \epsilon_i$.
- ϵ_i is a random sample taken from a normal distribution with μ=0 and σ=1.
- Make an xyplot of it.
- Adjust 5 curves with different values for f f=.01, f=.2, f=.5, f=.8, f=.99, and plot them.
- Compare it to a curve made with lm().

An example.

Creating the data frame.

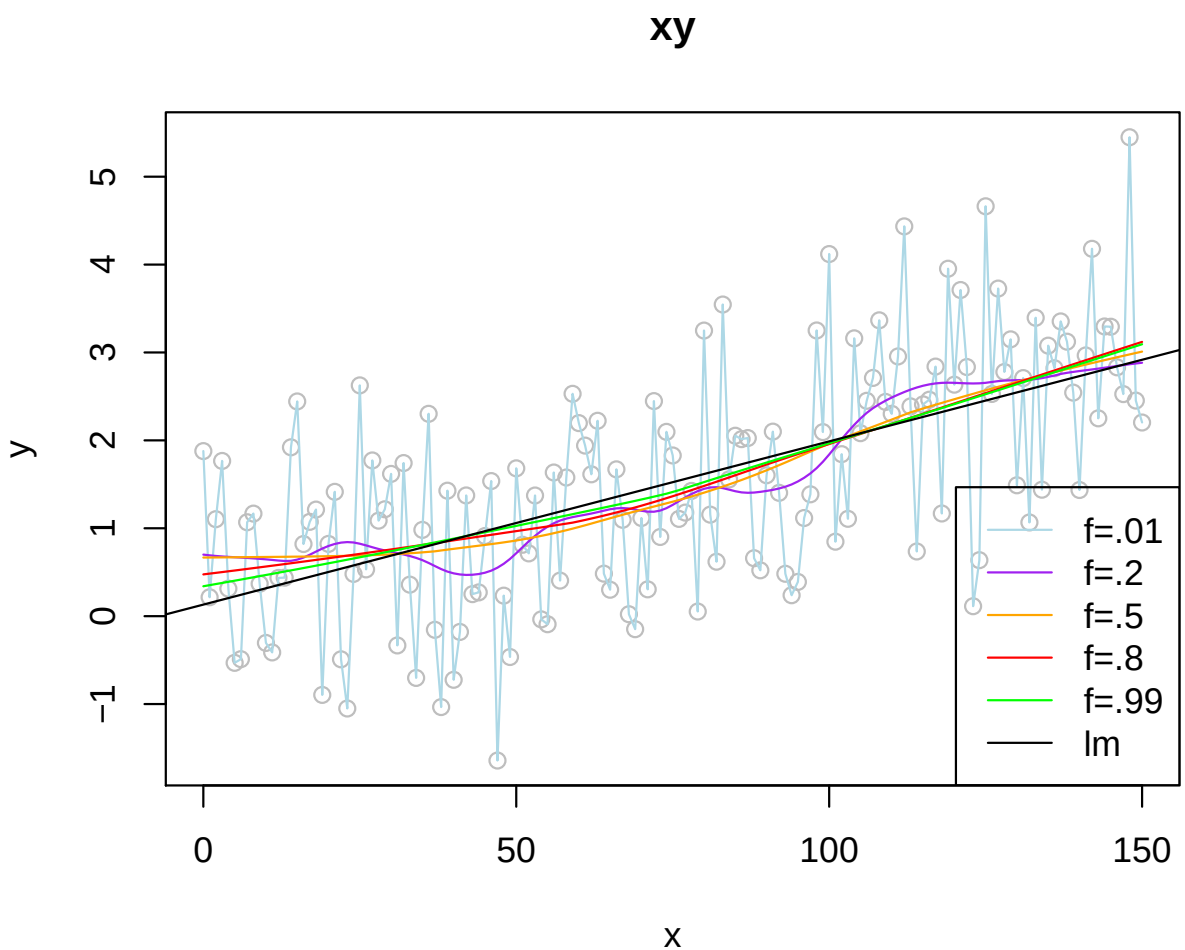
```
> x <- seq(from = 0, to = 150, by = 1)
> error <- rnorm(1000, 0, 1)
> error151 <- sample(error, 151,
+     replace = T)
> y <- 0.02 * x + error151
> xy <- data.frame(x, y)
```

An example.

Making the regressions.

```
> lo.xy.01 <- lowess(xy$x, xy$y,
+     f = 0.01)
> lo.xy.2 <- lowess(xy$x, xy$y, f = 0.2)
> lo.xy.5 <- lowess(xy$x, xy$y, f = 0.5)
> lo.xy.8 <- lowess(xy$x, xy$y, f = 0.8)
> lo.xy.99 <- lowess(xy$x, xy$y,
+     f = 0.99)
> lr <- lm(y ~ x, data = xy)
```

An example. (xyplot and regressions.)



Pros and Cons.

- What does the function lowess() return?[2]
- What are its cons when compared to lm()?[3]
- Which one do you consider to be the best fit?
- Do you notice any convergence?

[2] Use class().

[3] Which functions could be used on an object of class lm?

Choosing f.

- f should be chosen in order for it to maximize the smoothing.
- It should not make the pattern in the data diffuse or distorted.
- Most of the times, f should be chosen in the interval (.2,.8).
- When you do not know which is the best value for f, .5 should make the job, for an initial exploratory analysis.

loess.

- loess is a newer version of lowess.
- It implements the same local fitting method as lowess.
- The iterative method is optional.
- loess uses the formula notation.
- Here, you can decide the degree of the polynomial to which the data should be fitted.
- If you do not rely on the least squares approximation method, you can select some other one.
- An important argument of the function is loess.control.

loess.control.

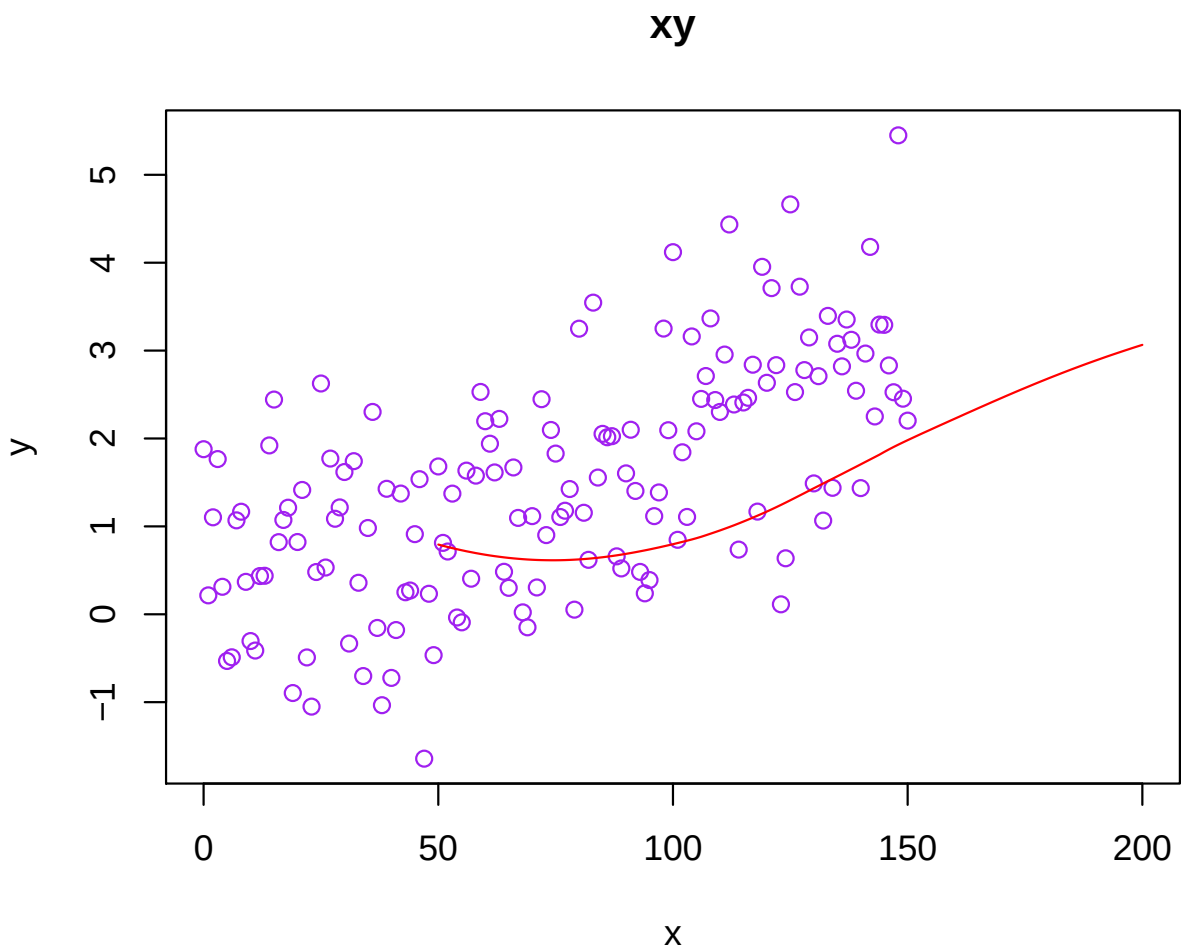
- loess.control helps you tune different parameters of the lowess computation.
- surface -> this regression may be used to extrapolate values based on the data.
- statistics, trace.hat -> to set these parameters to approximate would be useful if there are several points.
- Here, you can select the number of iterations to be made, as well.

Extrapolating with loess.

- Let us do an exptrapolation with the surface argument of the loess.control set to "direct".

```
> xy.lo2 <- loess(y ~ x, xy, control = loess.control(
> pred2 <- predict(xy.lo2, data.frame(xs = seq(50,
+     200, 1)), se = TRUE)
```

Plotting the Extrapolation.



About the chips.

- Let us study some important aspects of the Affimetrix oligonucleotide arrays tech.
- The goal is to probe an RNA sample (target) with different oligonucleotide probes.[4]
- Each feature is called a probe pair.

[4] 25 bp long

About the chips.

- Each probe pair contains a perfect match (PM) and a mismatch (MM) probe.
  - MM is used to measure the amount of non-specific binding.[5]
- A probe pair set is made up of all the PMs and MMs related to a common affyID.[6]

[5] It is created by changing the $13th$ base of the PM.

[6] An affyID is related to a gene or a gene fraction represented on the array.

About the chips.

- You can access to the PM, MM and probe names data with the pm(), mm(), probeNames() methods, for an AffyBatch object. [7]
- Another important function is geneNames(), which extracts unique affyIDs from an AffyBatch object.

[7] We will learn more about it, later.

Reading .CEL files.

- Affy chips are read into .CEL files which are read into R with the ReadAffy() function.
- First of all, you should have all your .CEL files in your working directory. You can check or set it with the getwd() and setwd() functions.
- All you have to do, is to keep your data into an R object.

```
> library(affy)
> data <- ReadAffy()
> data
```

Reading .CEL files.


```
AffyBatch object
size of arrays=732x732 features (18 kb)
cdf=HG-U133A_2 (22277 affyids)
number of samples=6
number of genes=22277
annotation=hgu133a2
notes=
```

AffyBatch Objects.

- What is the class of our object *data*?
- You can explore it by means of the slotNames() function.[8]
- If you want to access a particular slot, you can do it with the following syntax.

```
> slotNames(data)
 [1] "cdfName"
 [2] "nrow"
 [3] "ncol"
 [4] "assayData"
 [5] "phenoData"
 [6] "featureData"
 [7] "experimentData"
 [8] "annotation"
```

AffyBatch Objects.


```
 [9] "protocolData"
[10] ".__classVersion__"
```

```
> slot(data, "nrow")
```

```
[1] 732
```

[8] S3 and S4 objects can be accessed this way

phenoData.

- Microarray expression data is useless without metadata[9], so there must be a description of what is being assessed in each chip.
- You can find the description in the phenoData slot of an AffyBatch object.
- You can even assign a phenoData file to your data manually.
- This will be useful when testing for differentially expressed genes.

[9] Dr. Salt made a point on it last Monday.

Assigning a phenoData file

- Download the file pdata1422.txt and move it to your working directory.
- Then, read it with the read.AnnotatedDataFrame().
- Finally, put it into your AffyBatch object.

Assigning a phenoData file.


```
> pData(slot(data, "phenoData"))

          sample
AF14.CEL       1
AF15.CEL       2
AF16.CEL       3
AF6.CEL        4
AF7.CEL        5
AF8.CEL        6

> pd <- read.AnnotatedDataFrame(filename = "pdata1422
+      header = T)
> pData(pd)
```

Assigning a phenoData file.


```
                              Source.Name
AF16.CEL PROX1_siRNA-2_replicate2
AF7.CEL  PROX1_siRNA-1_replicate1
AF14.CEL     GFP_siRNA_replicate2
AF8.CEL  PROX1_siRNA-2_replicate1
AF15.CEL PROX1_siRNA-1_replicate2
AF6.CEL      GFP_siRNA_replicate1
```

```
> slot(data, "phenoData") <- pd
> pData(slot(data, "phenoData"))
```

Assigning a phenoData file.


```
                          Source.Name
AF16.CEL PROX1_siRNA-2_replicate2
AF7.CEL  PROX1_siRNA-1_replicate1
AF14.CEL     GFP_siRNA_replicate2
AF8.CEL  PROX1_siRNA-2_replicate1
AF15.CEL PROX1_siRNA-1_replicate2
AF6.CEL      GFP_siRNA_replicate1
```

Getting Microarray Data.

- As you know, you can use the ArrayExpress package to get expression data.
- I got my .CEL files as follows:

```
> library(ArrayExpress)
> Data = ArrayExpress("E-MEXP-1422",
+      save = T)
```

- This AffyBatch object already has its phenoData complete, so, if you do not know much about the sample, ArrayExpress is a must.

image.

- Sometimes, the chip will be damaged or some dust will fall on it, so some expression values obtained from it, will not be reliable.
- You can see the chips, as they were scanned, to check for this.

image.

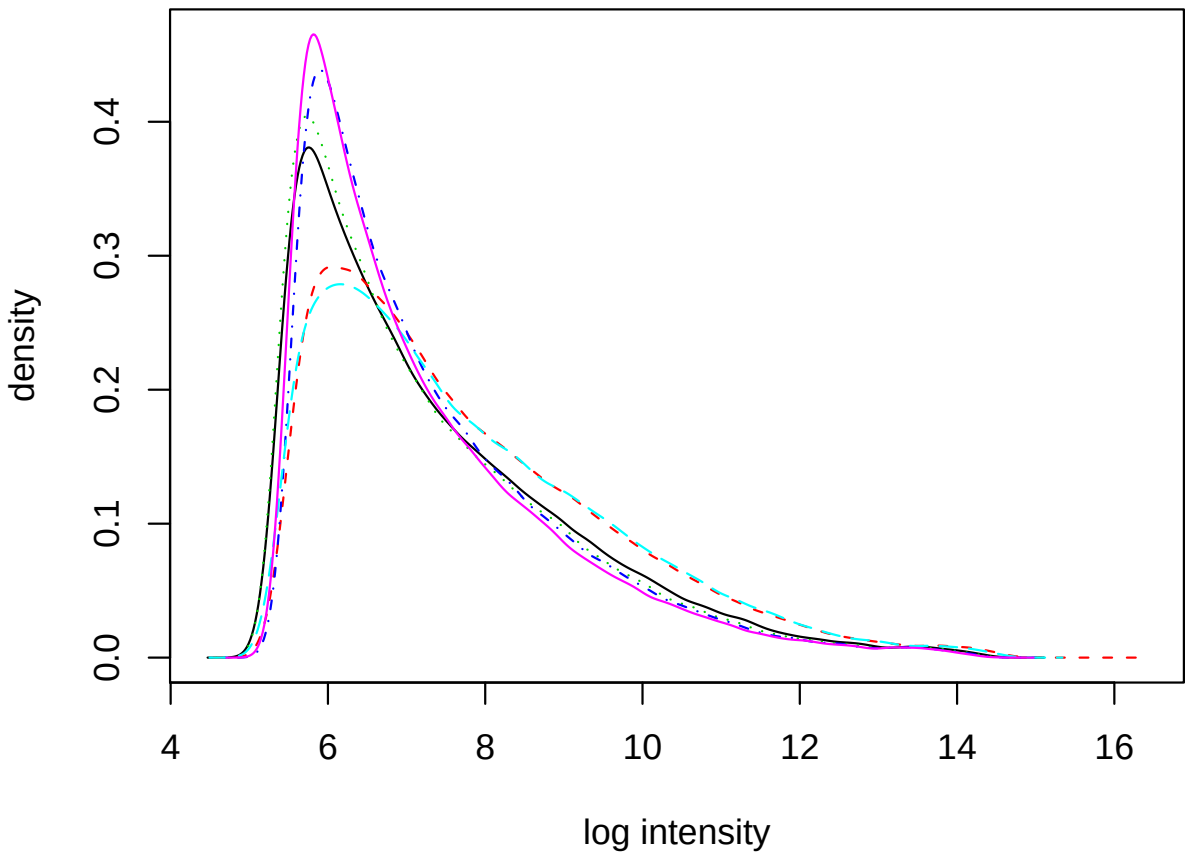

```
> par(mfrow = c(2, 3))
> image(Data)
> par(mfrow = c(1, 1))
```

hist.

- The hist function is well suited to deal with AffyBatch objects, as well.
- Make a histogram in order check if the intensities are distributed similarly among the arrays.
- You can infer the need for normalization between arrays with this plot.[10]

[10] We will discuss this in the next class.

hist.

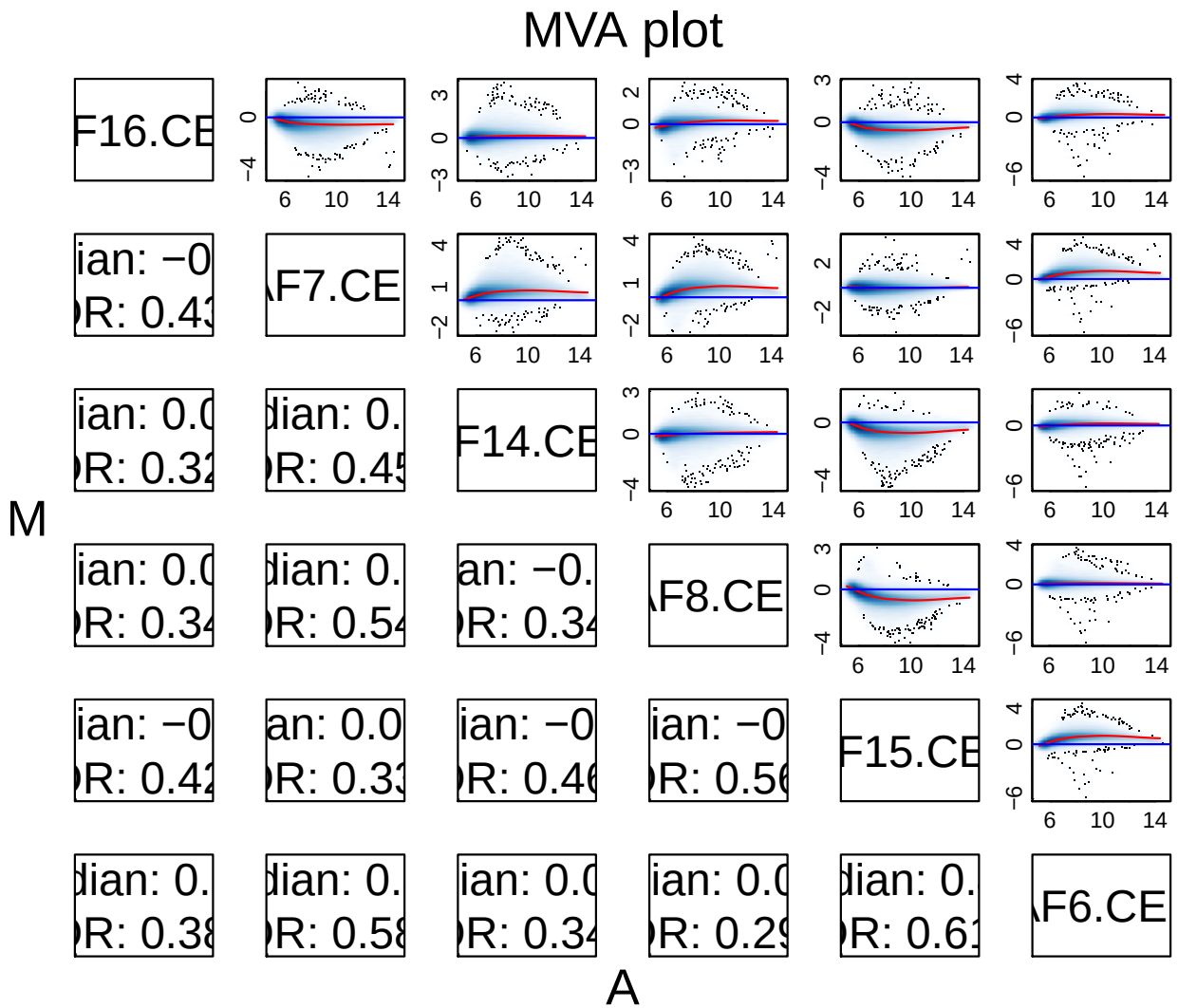


M vs A plot.

- The log ratio of the intensities is plotted on the vertical axis.
- The average of the log intensities is plotted on the horizontal axis.
- This plots offer a way of making pairwise graphical comparison of intensity data.
- Problems in replicate sets of arrays may be assessed with these graphs.
- We will use our Data object as it is ordered according to the replicates.[11]

[11] See the phenoData.

M vs A plot.



M vs A plot.

- The argument pairs sets the function to make pairwise comparisons.
- What would happen with pairs=F?[12]
- What can you notice about the red regression and the deviant points?[13]

[12] The chip is compared to a reference chip.

[13] We need to normalize, wait for the next class.

RNA Degradation.

- As you already know, different fragments of the same RNA are represented in different features of the array.
- An artefact of the microarray techinque is that features representing sites closer to the 5' end of the RNA are less intense.
- This is because RNA degradation in the assay starts at the 5' end of the molecule.
- There is a way to check for the RNA degradation rate in the assay.

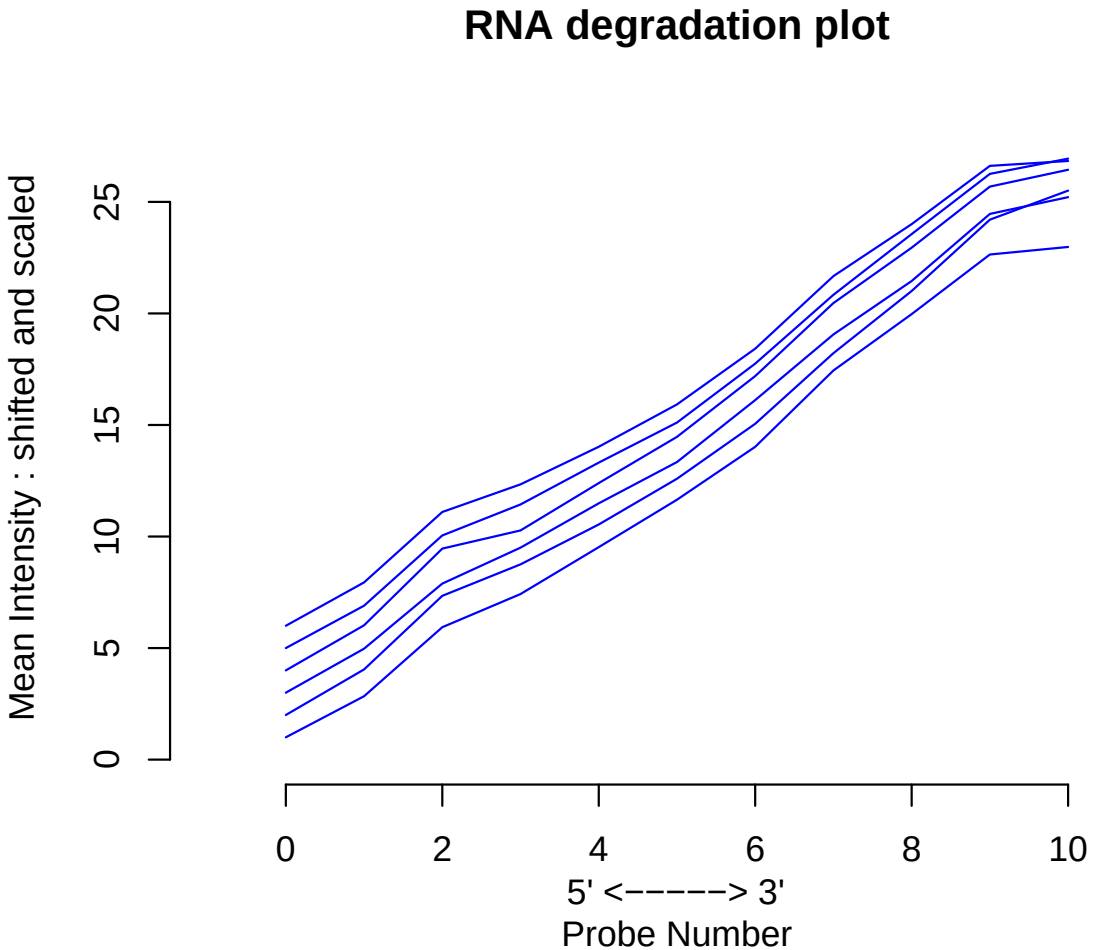
RNA Degradation.


```
> degradation <- AffyRNAdeg(Data)
> names(degradation)
[1] "N"                "sample.names"
[3] "means.by.number" "ses"
[5] "slope"            "pvalue"
```

- There is a way to make a summary of this object.[14]
- Now, we will plot degradation curves for this data.

[14] Use the summarizeAffyRNAdeg() function.

RNA Degradation Plots.



Pending Points.

Next class, we will learn about microarray normalization, finding DEGs and the SpeCond package.