Lab 3B: An Introduction to the `affy` package

June 4, 2003

In this lab, we demonstrate the main functions in the *affy* package for pre-processing Affymetrix microarray data. We first load the packages that we will be using.

```
> library(affy)

Welcome to Bioconductor
         Vignettes contain introductory material.  To view,
         simply type: openVignette()
         For details on reading vignettes, see
         the openVignette help page.
Creating a new generic function for "summary" in package
reposTools

Synching your local package management information ...

> library(affydata)
```

For a more detailed introduction, consult the package vignettes which can be listed by the command `openVignette("affy")`. A demo can also be accessed by `demo(affy)`. A number of sample datasets are available in the package and in the *affydata* add-on; to list these, type `data(package="affy")` and `data(package="affydata")`. To interact with this and other vignettes you can use `vExplorer`.

The function `ReadAffy` is available for reading CEL files. However, in this lab we will work mainly with the `Dilution` dataset, which is included in the package. For a description of `Dilution`, type `?Dilution`. To load this dataset

```
> data(Dilution)
```

One of the main classes in *affy* is the `AffyBatch` class. For details on this class consult the help file, `help("AffyBatch-class")`; methods for manipulating instances of this class are also described in the help file. Other classes include `ProbeSet` (PM and MM intensities for individual probe sets), `Cdf` (information contained in a CDF file), and `Cel` (single array cel intensity data). The object `Dilution` is an instance of the class `AffyBatch`. Try the following commands to obtain information of this object

```
> class(Dilution)

[1] "AffyBatch"

> slotNames(Dilution)

[1] "cdfName"     "nrow"        "ncol"        "exprs"       "se.exprs"
[6] "phenoData"   "description" "annotation"  "notes"

> Dilution

AffyBatch object
size of arrays=640x640 features (12805 kb)
cdf=HG_U95Av2 (12625 affyids)
number of samples=4
number of genes=12625
annotation=hgu95av2

> annotation(Dilution)

[1] "hgu95av2"
```

For a description of the target samples hybridized to the arrays

```
> phenoData(Dilution)

         phenoData object with 3 variables and 4 cases
         varLabels
                liver: amount of liver RNA hybridized to array in micrograms
                sn19: amount of central nervous system RNA hybridized to array in micro
                scanner: ID number of scanner used

> pData(Dilution)

    liver sn19 scanner
20A    20    0       1
20B    20    0       2
10A    10    0       1
10B    10    0       2
```

The exprs slot contains a matrix with columns corresponding to arrays and rows to individual probes on the array. To obtain the matrix of intensities for all four arrays

```
> e <- exprs(Dilution)
> nrow(Dilution) * ncol(Dilution)
```

```
[1] 409600
```

```
> dim(e)
```

```
[1] 409600      4
```

The values in this array are the raw values for the mean probe expression in your `CEL` files.

You can access probe-level PM and MM intensities using

```
> PM <- pm(Dilution)
> dim(PM)
```

```
[1] 201800      4
```

```
> PM[1:5, ]
```

```
           20A   20B   10A   10B
1000_at1 468.8 282.3 433.0 198.0
1000_at2 430.0 265.0 308.5 192.8
1000_at3 182.3 115.0 138.0  86.3
1000_at4 930.0 588.0 752.8 392.5
1000_at5 171.0 128.0 152.3  97.8
```

The array `PM` contains only the perfect match probes.

To get the probe set names (Affy IDs)

```
> gnames <- geneNames(Dilution)
> length(gnames)
```

```
[1] 12625
```

```
> gnames[1:5]
```

```
[1] "1000_at"   "1001_at"   "1002_f_at" "1003_s_at" "1004_at"
```

```
> nrow(e)/length(gnames)
```

```
[1] 32.44356
```

The length of `gnames`, 12625, indicates that there are that many probesets on the chip.

As with other microarray objects in Bioconductor packages, you can use standard subsetting commands on `AffyBatch` objects. The

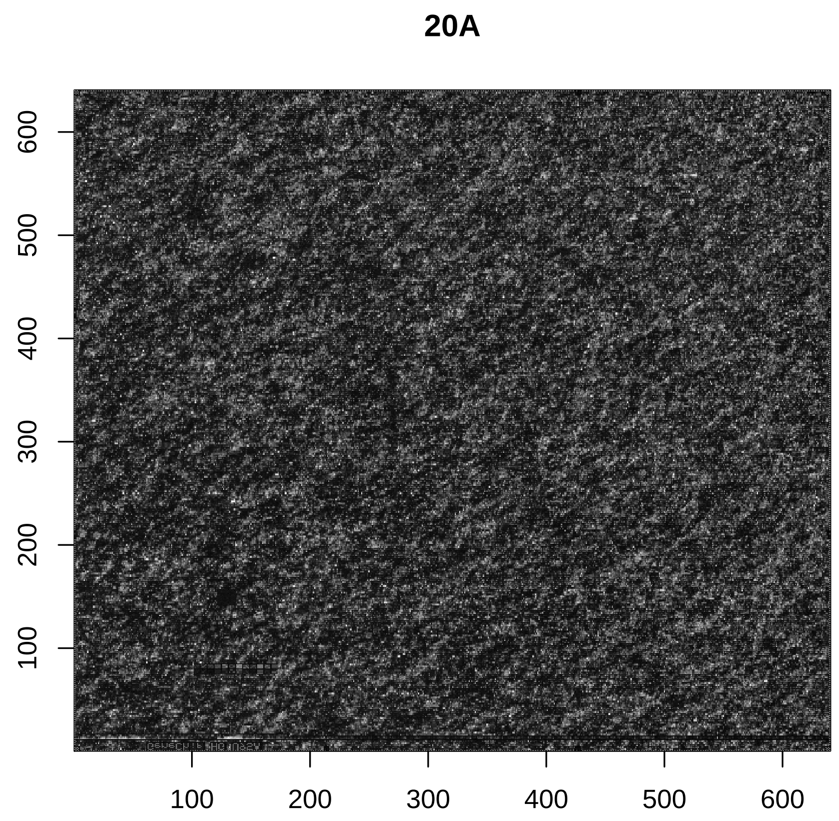
```
> dil1 <- Dilution[1]
> class(dil1)
```

```
[1] "AffyBatch"

> dil1

AffyBatch object
size of arrays=640x640 features (3204 kb)
cdf=HG_U95Av2 (12625 affyids)
number of samples=1
number of genes=12625
annotation=hgu95av2

> cel1 <- Dilution[[1]]
> class(cel1)

[1] "Cel"

> cel1

Cel object
name=20A
cdfName=HG_U95Av2
intensity=640 x 640 (3200 kb)
masked= 0 %
outliers= 0 %
history=
```

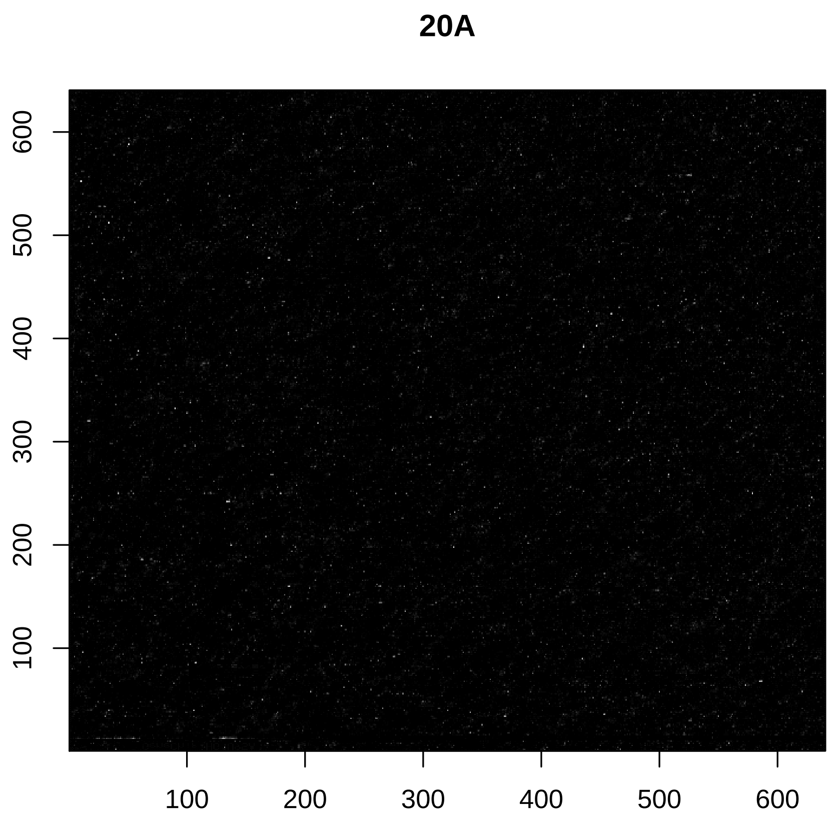
One of the main functions for reading in Affymetrix data is `ReadAffy`. It reads in data from `CEL` and `CDF` files and creates objects of class `AffyBatch`. Using `ReadAffy(widget=TRUE)` uses a widget for interactive data input.

To produce a spatial image of probe log intensities and probe raw intensities

```
> image(Dilution[1])
```

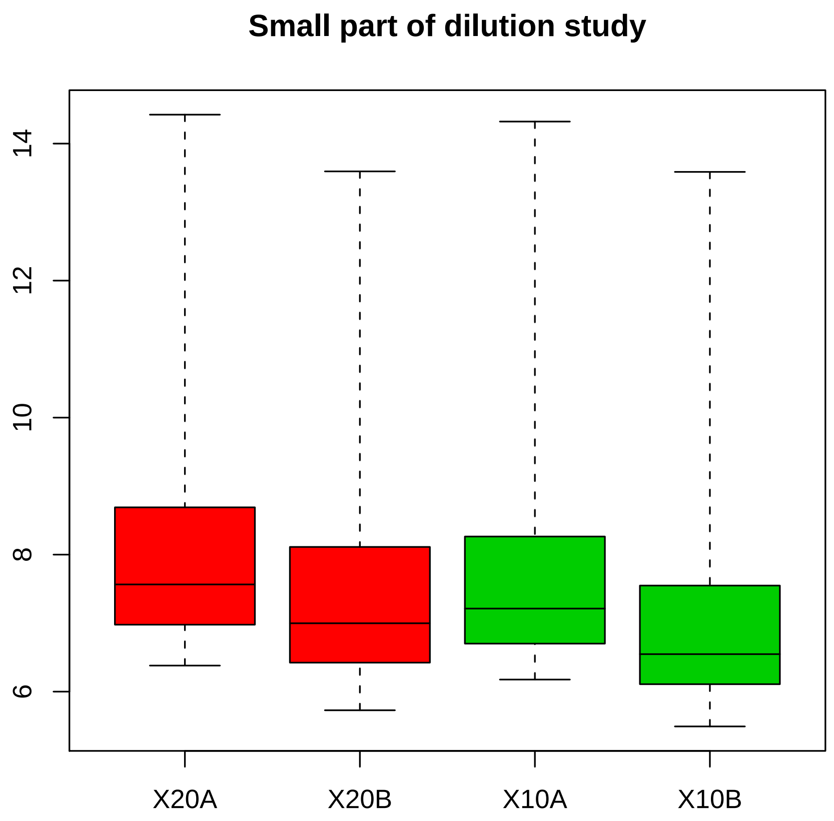


```
> image(cel1)
```



To produce boxplots of probe log intensities

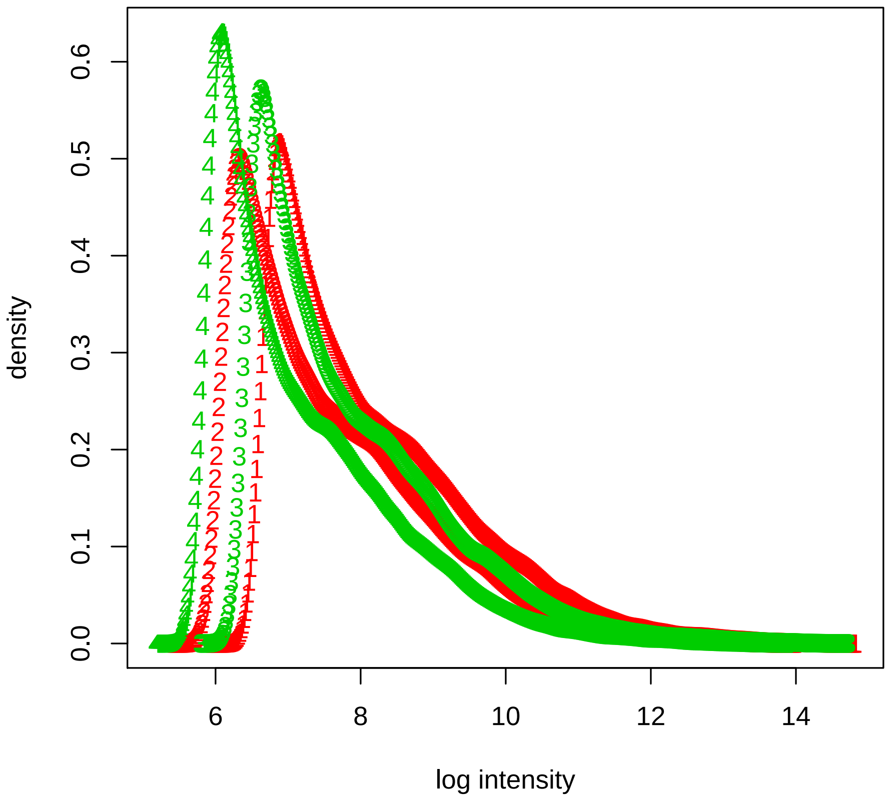
```
> boxplot(Dilution, col = c(2, 2, 3, 3))
```



Note that scanner effects seem stronger than concentration effect (the first and third boxplots are from the same scanner).

To produce density plots of probe log intensities

```
> hist(Dilution, type = "l", col = c(2, 2, 3, 3), lty = rep(1:2,
+     2), lwd = 3)
```

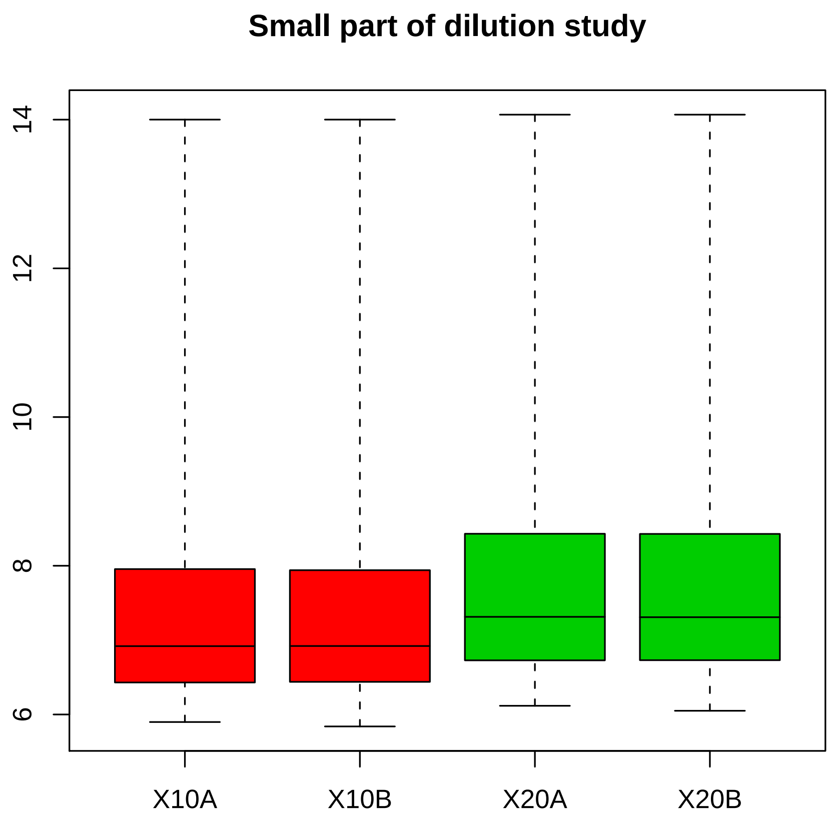


These boxplots and histograms show that the Dilution data needs normalizaton. Arrays that should be the same are different. Arrays that should be different are similar. Because of these arrays have increasing concentrations they have to be normalized in concnetration groups.

```
> Dil10 <- normalize(Dilution[1:2])
> Dil20 <- normalize(Dilution[3:4])
> normDil <- merge(Dil20, Dil10)
```

Notice how the boxplot now looks better:

```
> boxplot(normDil, col = c(2, 2, 3, 3))
```



The `affy` package provides implementations for a number of methods for background correction, probe-level normalization (e.g., quantile, curve-fitting (Bolstad et al., 2002)), and computation of expression measures (e.g., MAS 4.0, MAS 5.0, MBEI (Li & Wong, 2001), RMA (Irizarry et al., 2003)). To list available methods for `AffyBatch` objects

```
> bgcorrect.methods
```

```
[1] "mas"  "none" "rma"  "rma2"
```

```
> normalize.AffyBatch.methods
```

```
[1] "constant"          "contrasts"         "invariantset"      "loess"
[5] "qspline"           "quantiles"         "quantiles.robust"
```

```
> pmcorrect.methods
```

```
[1] "mas"        "pmonly"     "subtractmm"
```

```
> express.summary.stat.methods
```

```
[1] "avgdiff"      "liwong"       "mas"          "medianpolish" "playerout"
```

The main probe level pre-processing function is `expresso`. You can select pre-processing methods interactively using widgets by typing `expresso(Dilution, widget=TRUE)`. The function operates on objects of class `AffyBatch` and returns objects of class `exprSet`.

The function `rma` provides a more efficient implementation of Robust Multi-array Average (RMA)

We don't normalize because we already did above.

Data packages for CDF information can be download from `www.bioconductor.org`|. These packages contain environment objects which provide mappings between Affymetrix identifiers and matrices of probe locations, with rows corresponding to probe-pairs and columns to PM and MM cels. CDF environments for HGU95Av2 and HGU133A chips are already in the package. For information on the environment object `? hgu95av2cdf`

```
> annotation(Dilution)

[1] "hgu95av2"

> data(hgu95av2cdf)
> pnames <- ls(env = hgu95av2cdf)
> length(gnames)

[1] 12625

> gnames[1:5]

[1] "1000_at"   "1001_at"   "1002_f_at" "1003_s_at" "1004_at"

> get(gnames[1], env = hgu95av2cdf)

          pm     mm
 [1,] 358160 358800
 [2,] 118945 119585
 [3,] 323731 324371
 [4,] 223978 224618
 [5,] 313420 314060
 [6,] 349209 349849
 [7,] 199525 200165
 [8,] 213669 214309
 [9,] 236739 237379
[10,] 298099 298739
[11,] 282744 283384
[12,] 281443 282083
```

```
[13,] 349198 349838
[14,] 297953 298593
[15,] 317054 317694
[16,] 404069 404709
```

You can also use the `indexProbe`, `pmindex`, and `mmindex` to get information on probe location

```
> plocs <- indexProbes(Dilution, which = "both")
> plocs[[1]]
```

```
 [1] 358160 118945 323731 223978 313420 349209 199525 213669 236739 298099
[11] 282744 281443 349198 297953 317054 404069 358800 119585 324371 224618
[21] 314060 349849 200165 214309 237379 298739 283384 282083 349838 298593
[31] 317694 404709
```

```
> pmindex(Dilution, genenames = gnames[1], xy = TRUE)
```

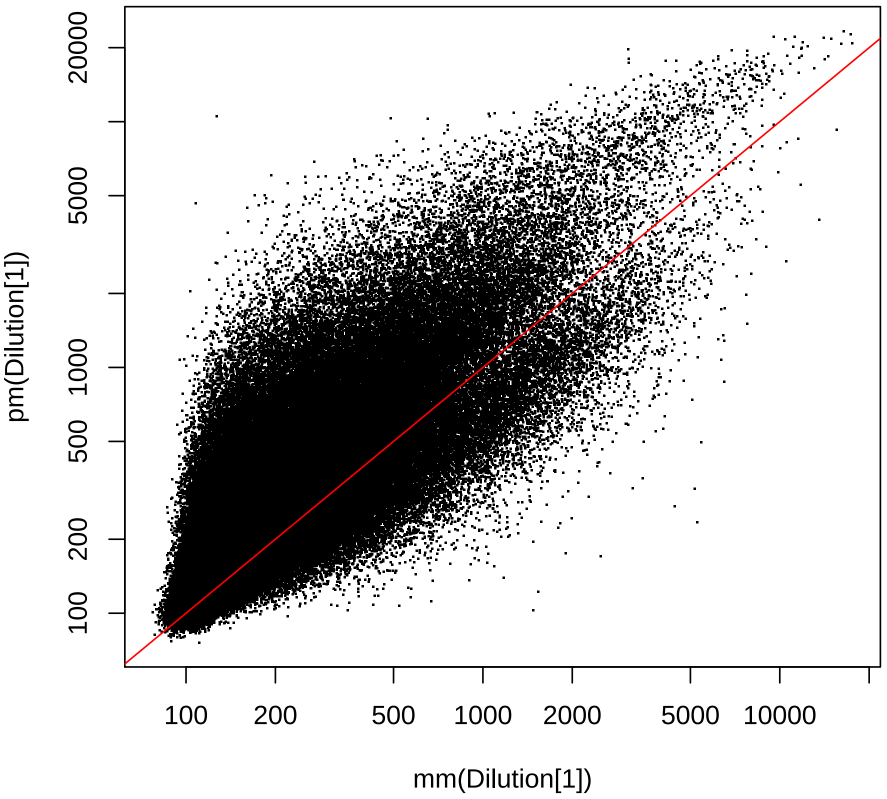
```
$"1000_at"
        x   y
 [1,] 400 560
 [2,] 545 186
 [3,] 531 506
 [4,] 618 350
 [5,] 460 490
 [6,] 409 546
 [7,] 485 312
 [8,] 549 334
 [9,] 579 370
[10,] 499 466
[11,] 504 442
[12,] 483 440
[13,] 398 546
[14,] 353 466
[15,] 254 496
[16,] 229 632
```

```
> pmindex(Dilution, genenames = gnames[1])
```

```
$"1000_at"
 [1] 358160 118945 323731 223978 313420 349209 199525 213669 236739 298099
[11] 282744 281443 349198 297953 317054 404069
```

Having access to PM and MM data can be useful. Let's look at a plot of PM vs. MM

```
> plot(mm(Dilution[1]), pm(Dilution[1]), pch = ".", log = "xy")
> abline(0, 1, col = "red")
```



An example of a `ProbeSet` is included in the package. The object represents the PM and MM intensities of a particular control probeset, `AFFX-BioB-5\_at`, from 12 arrays from the spike-in data set. The transcripts related to this probeset where spiked-in at known concentrations in the hybridization mixture. The concentrations were varied from chip to chip. The different concentration values are used as the sample names:

```
> data(SpikeIn)
> sampleNames(SpikeIn)
```

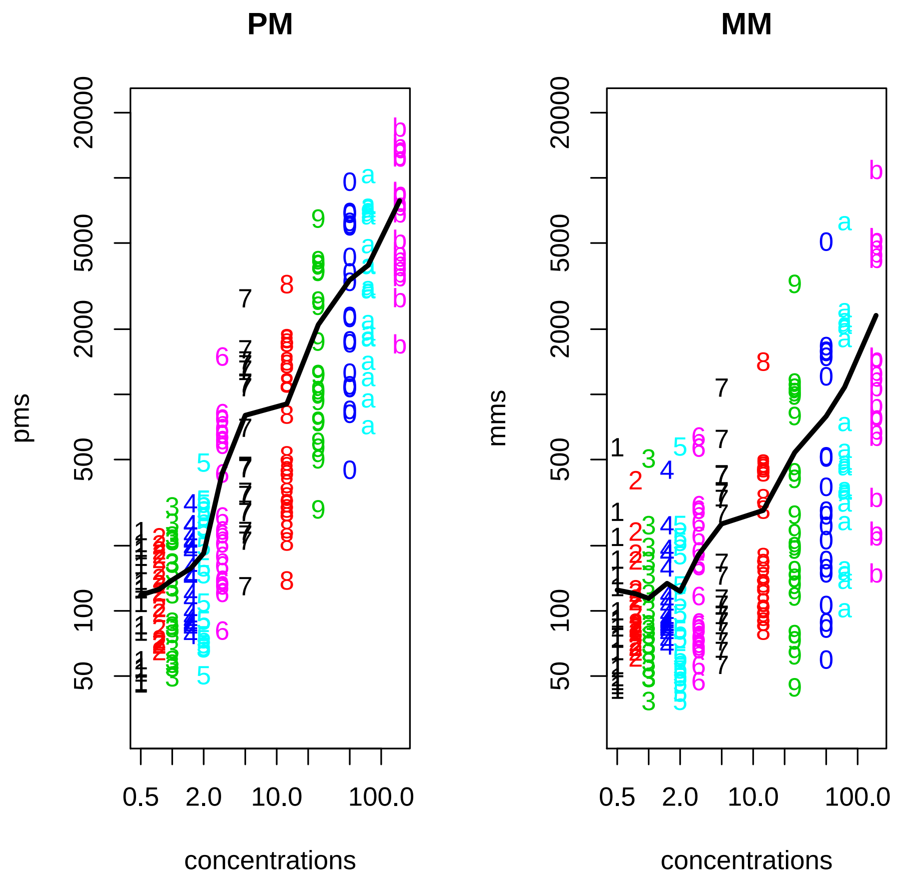
```
 [1] "0.50"   "0.75"   "1.00"   "1.50"   "2.00"   "3.00"   "5.00"   "12.50"
 [9] "25.00"  "50.00"  "75.00"  "150.00"
```

Notice that the concentrations are growing exponentially.

The help file describes the data set in more detail. The The next figure uses this data set to demonstrate that the MM also detect some transcript signal.

In the next plot we see that both the MM values and the PM values increase as the concentration does. This means that the MM probes are detecting signal and not just background or non-specific hybridization.

```
> pms <- pm(SpikeIn)
> mms <- mm(SpikeIn)
> par(mfrow = c(1, 2))
> concentrations <- matrix(as.numeric(sampleNames(SpikeIn)), 20,
+     12, byrow = TRUE)
> matplot(concentrations, pms, log = "xy", main = "PM", ylim = c(30,
+     20000))
> lines(concentrations[1, ], apply(pms, 2, mean), lwd = 3)
> matplot(concentrations, mms, log = "xy", main = "MM", ylim = c(30,
+     20000))
> lines(concentrations[1, ], apply(mms, 2, mean), lwd = 3)
```



These PM and MM intensities were obtained using the `probeset` method on an `AffyBatch` (not included or shown) representing a large spike-in experiment.

1 Using sequence information

Sequence information can be used to improve on pre-processing. Two examples are the use of GC content to adjust for non-specific hybridization and the use of probe location to take into account RNA degradation. We demonstrate this with some examples

1.1 GC content

To explore issues of GC content we need to load a special library that contains information on the GC content of the probes. For any chip the R package *makeMatchAffy* can be used, together with the appropriate metadata package to construct the necessary package. The package named *mAffyu95A* was built using the HGU95Av2 chip. This package contains the probe sequences (for most probes) and their location on the RNA transcript represented by their probeset. We start by loading this information

```
> if (.Platform$OS.type == "unix") {
+     library(mAffyu95A)
+     data(affyprobeids)
+     data(affylocs)
+ }
```

Now we are ready to put together each probe with their sequence and their GC-content.

```
> if (.Platform$OS.type == "unix") {
+     Index <- affylocs[, 1] + affylocs[, 2] * ncol(Dilution) +
+         1
+     probenames <- probeNames(Dilution)
+     myindex <- unlist(pmindex(Dilution))
+     probenames <- probenames[match(Index, myindex)]
+     seq <- getprobes(seq(along = Index))
+     tmp <- sapply(seq, cgcontent)
+     tmp <- matrix(unlist(tmp), nrow = 4)
+     atc <- tmp[1, ] + tmp[2, ]
+     gcc <- tmp[3, ] + tmp[4, ]
+ }
```

To see that the GC has an effect on hybridization notice how as GC content grows so does the measured intensity. So probes with higher GC content have higher intensities. This appears to be independent of the amount of transcript available.

```
> if (.Platform$OS.type == "unix") {
+     y <- intensity(Dilution)[Index, 1]
+     boxplot(split(y, gcc), xlab = "GC content", ylab = "Iintensity",
+         log = "y", las = 1, range = 0, col = "grey")
+ }
```

Now does this matter at the expression level? To see that the average GC across probes within probeset can be variable notice the distribution shown in the next figure. The average GC content in probesets is quite variable.

One still might ask whether this matters. If all we are doing is comparing samples for specific genes then it shouldn't matter too much, but there may be some cases where it does. For example if a probeset has a GC rich probe to which some mRNA other than the target hybridizes then we may perceive differences in the target genes expression, between the samples (if the samples differ in abundance of the other mRNA). We are still unsure as to how large that effect might be. These tools will help you to explore the data more extensively if desired.

```
> if (.Platform$OS.type == "unix") {
+     avggcc <- tapply(gcc, probenames, mean)
+     barplot(table(round(avggcc)), xlab = "Average GC content in probeset",
+         las = 1)
+ }
```

Subtracting the MM does a relatively good job at removing this effect. In the next figure the boxplots are centered around zero.

```
> if (.Platform$OS.type == "unix") {
+     pmi <- unlist(pmindex(Dilution))
+     mmi <- unlist(mmindex(Dilution))
+     pmIndex <- match(pmi, Index)
+     y <- intensity(Dilution)[pmi, 1] - intensity(Dilution)[mmi,
+         1]
+     x <- gcc[pmIndex]
+     boxplot(split(y, x), xlab = "GC content", ylab = "PM-MM",
+         las = 1, range = 0, col = "grey", ylim = c(-300, 500))
+ }
```

If instead of using RMA we use MASS 5.0 (at least our interpretation of it) the effect due to GC content is much smaller. The reason for this is that MASS 5.0 does subtract the MM values and hence removes the GC content effect.

```
> if (.Platform$OS.type == "unix") {
+     mas5Dil <- mas5(Dilution, normalize = FALSE)
+     boxplot(split(exprs(mas5Dil)[names(avggcc), 1], round(avggcc)),
+         log = "y", range = 0, las = 1, yaxt = "n", xlab = "Average GC content in pr
+         ylab = "MAS 5.0 expression", col = "grey")
+     axis(2, c(0.1, 10, 1000), c("0.01", "10", "1000"), las = 1)
+ }
```

```
background correction: mas
PM/MM correction : mas
expression values: mas
background correcting...done.
12625 ids to be processed
.........
```

However, as pointed out by Irizarry et al. (2003) subtracting MM results in expression measures that are very noisy at low expression values. RMA is not as noisy, but the next plot shows it does not do as well in removing the dependence on GC content.

```
> if (.Platform$OS.type == "unix") {
+     RMA <- rma(Dilution)
+ }

Background correcting
Normalizing
Calculating Expression
```

We could write a new version of RMA that does the background adjustment within GC content strata (and indeed that work is under way).

1.2 Probe Location

The package also includes methods useful for assessing RNA quality. Individual probes in a probe set are ordered by location relative to the $5'$ end of the targeted RNA molecule. Since RNA degradation typically starts from the $5'$ end of the molecule, we would expect probe intensities to be systematically lowered at that end of a probe set when compared to the $3'$ end. Affymetrix software includes some $5'$ and $3'$ probe sets that are used to assess this. The *affy* package includes some simple methods that complement the Affymetrix assessment.

When using the probe location functions described above the PM and MM are, in general, returned in location order. This makes it easy to perform the following assessment: On each chip, probe intensities are averaged by location in probe set, with the average taken over probe sets. The package provides a method that produces side-by-side plots of these means, making it easy to notice any $5'$ to $3'$ trend. The function `AffyRNAdeg` does all the computation. The object returned can be summarized and plotted using `summaryAffyRNAdeg` and `plotAffyRNAdeg` respectively.

The summary shows standardized slope estimates of the regression of intensity versus probe number (not location per se). The plots shows the average across all probes versus probe number. This plot can be used to compare arrays. An array with degradation should stand out becuase it has a bigger slope. In this example we dont see any evidence of this.

```
> if (.Platform$OS.type == "unix") {
+     rnadeg <- AffyRNAdeg(Dilution)
+     summaryAffyRNAdeg(rnadeg)
+     plotAffyRNAdeg(rnadeg)
+ }
```

```
X20A X20B X10A X10B
slope -0.0239 0.0363 0.0273 0.0849
pvalue 0.892 0.84 0.875 0.616
```

If one wants to delve deaper one can use the MatchAffy package, which returns the exact location (as opposed to order location). The following code gives an example of how one can obtain a similar plot to the above:

The next plot uses the actual location instead of probe number, however, we re-scale because otherwise one does not see much (there is too much noise).

```
> if (.Platform$OS.type == "unix") {
+     scale01 <- function(x) {
+         r <- range(x)
+         x - r[1]/(r[2] - r[1])
+     }
+     locs <- tapply(as.double(affylocs[, 3]), probenames, scale01)
+     locs <- unlist(locs)
+     plot(locs, locs, type = "n", xlab = "5' <-----> 3'\nRelative position",
+         ylab = "Relative log intesity", yaxt = "n", xaxt = "n",
+         ylim = c(-0.02, 0.35))
+     legend(0.1, 0.3, c("normal RNA", "degraded RNA"), col = c("blue",
+         "red"), lty = c(2, 1))
+     i <- 1
+     o <- sample(length(gcc), 5000)
+     pms <- log2(intensity(Dilution)[Index, i])
+     res <- tapply(pms, probenames, function(x) x - median(x))
+     tmp <- loess(unlist(res) ~ gcc, subset = o, span = 4/5, degree = 1)
+     oo <- seq(1, length(tmp$x), len = 100)
+     lines(sort(tmp$x)[oo], tmp$fitted[order(tmp$x)][oo], col = "red")
+ }
```