

Analysis of NimbleGen Expression Data with the oligo Package

Benilton Carvalho

August 20, 2009

1 Introduction

This document presents a non-trivial use of the `oligo` Package for the analysis of NimbleGen Expression data. This vignette follows the structure of the chapter **From CEL files to a list of interesting genes** by R. A. Irizarry *in Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, which shows a case study for Affymetrix Expression arrays.

In order to analyze microarray data using `oligo`, the user is expected to have installed on the system a package with the annotation for the particular array design on which the experiment was performed. For the example in question here, the design is `hg18.60mer.expr` and the annotation package associated to it is `pd.hg18.60mer.expr`, which is built by using the `pdInfoBuilder` package.

2 Initialization of the environment

We start by loading the packages that are going to be used in this session. The `maqExpression4plex` package provides a set of six samples on the MAQC Study; the set is comprised of samples on two groups: universal reference and brain. The remaining packages offer additional functionality, like tools for filtering, plotting and visualization.

```
R> library(oligo)
R> library(maqExpression4plex)
R> library(genefilter)
R> library(limma)
R> library(RColorBrewer)
R> palette(brewer.pal(8, "Dark2"))
```

Once the package is loaded, we can easily get the location of the `XYs` files that contain the intensities by calling `list.xysfiles`, which takes the same arguments as `list.files`. To minimize the chance of problems, we strongly recommend the use of `full.names=TRUE`.

```
R> extdata <- system.file("extdata",
  package = "maqExpression4plex")
R> xys.files <- list.xysfiles(extdata,
  full.names = TRUE)
R> basename(xys.files)

[1] "9868701_532.xys" "9868901_532.xys"
[3] "9869001_532.xys" "9870301_532.xys"
[5] "9870401_532.xys" "9870601_532.xys"
```

To read the XYS files, we provide the `read.xysfiles` function, which also takes `phenoData`, `experimentData` and `featureData` objects and returns an appropriate subclass of *FeatureSet*.

```
R> pd <- dir(extdata, pattern = "phenoData",
  full.names = TRUE)
R> pd <- read.AnnotatedDataFrame(pd)
R> maqc <- read.xysfiles(xys.files, phenoData = pd)
R> class(maqc)

[1] "ExpressionFeatureSet"
attr(,"package")
[1] "oligoClasses"
```

3 Exploring the feature-level data

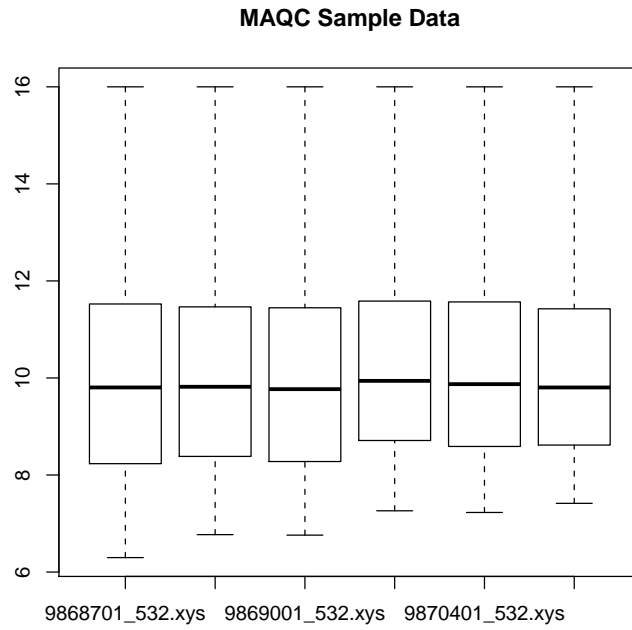
The `read.xysfiles` function returns, in this case, an instance of *ExpressionFeatureSet* and the intensities of these files are stored in its `exprs` slot, which can be accessed with a method with the same name.

```
R> exprs(maqc)[10001:10010, 1:2]

      9868701_532.xys 9868901_532.xys
10001             735             742
10002             4786            4435
10003            25600            26155
10004             1079            1093
10005             3056            3128
10006              310             385
10007              NA             NA
10008              NA             NA
10009              599             713
10010            28712            29795
```

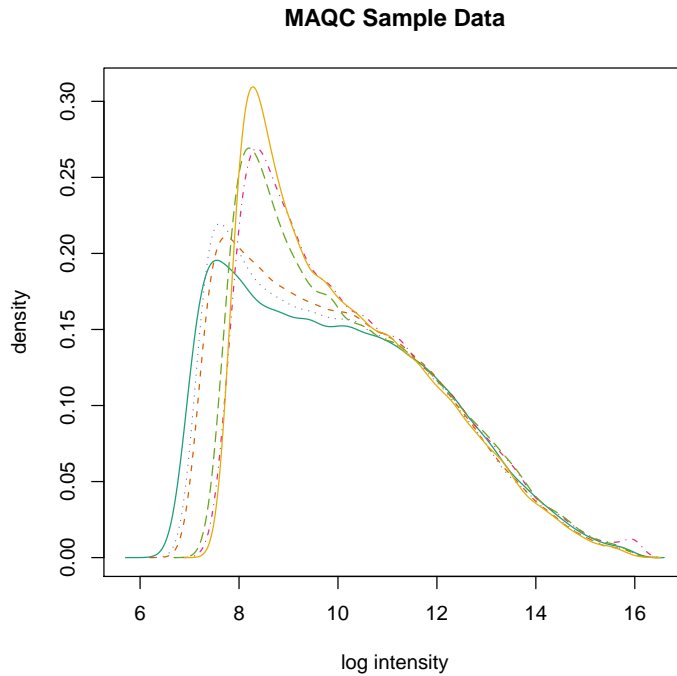
The `boxplot` method can be used to produce boxplots for the feature-level data.

```
R> boxplot(maqc, main = "MAQC Sample Data")
```



Similarly, a smoothed histogram for the feature-level data can be obtained with the `hist` method.

```
R> hist(maqc, main = "MAQC Sample Data")
```



4 RMA algorithm

The RMA algorithm can be applied to the raw data of expression arrays. It is available via the `rma` method. The algorithm will perform background subtraction, quantile normalization and summarization via median polish. The result of `rma` is an instance of `ExpressionSet` class, which also contains an `exprs` slot and method.

```
R> eset <- rma(maqc)
```

```
Background correcting
Normalizing
Calculating Expression
```

```
R> class(eset)
```

```
[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"
```

```
R> show(eset)
```

```

ExpressionSet (storageMode: lockedEnvironment)
assayData: 24000 features, 6 samples
  element names: exprs
phenoData
  sampleNames: 9868701_532.xys, 9868901_532.x
ys, ..., 9870601_532.xys (6 total)
  varLabels and varMetadata description:
    Key:
    additional varMetadata: channel
featureData
  featureNames: NM_000014, NM_000015, ..., XM
_928211 (24000 total)
  fvarLabels and fvarMetadata description: none
experimentData: use 'experimentData(object)'
Annotation: pd.hg18.60mer.expr

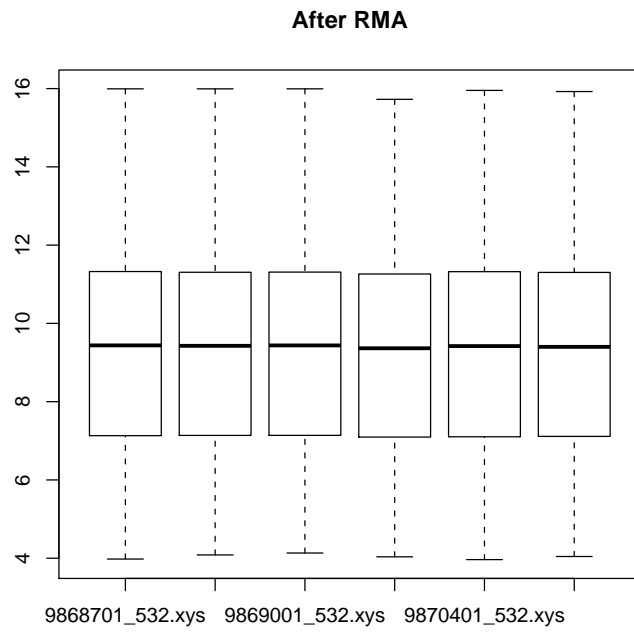
```

```
R> exprs(eset)[1:10, 1:2]
```

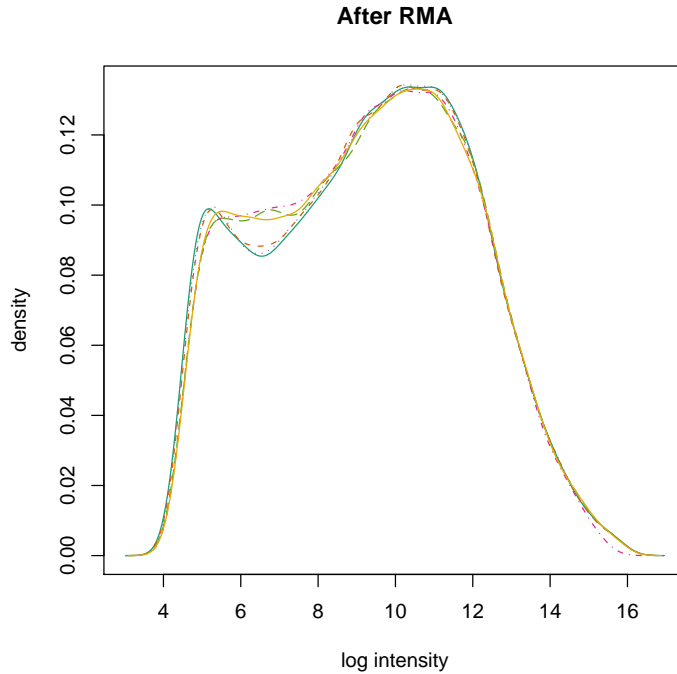
	9868701_532.xys	9868901_532.xys
NM_000014	12.3	12.3
NM_000015	4.5	4.6
NM_000016	12.4	12.2
NM_000017	8.5	8.5
NM_000018	12.6	12.4
NM_000019	11.7	11.6
NM_000020	8.9	9.2
NM_000021	11.8	11.8
NM_000022	8.9	8.4
NM_000023	8.9	9.1

The `boxplot` and `hist` methods are also implemented for *ExpressionSet* objects. Note that `rma`'s output is in the \log_2 scale, so we call such methods using the argument `transfo=identity`, so the data are not transformed in any way.

```
R> boxplot(eset, transfo = identity,
  main = "After RMA")
```



```
R> hist(eset, transfo = identity, main = "After RMA")
```



5 Assessing differential expression

One simple approach to assess differential expression is to flag units with log-ratios greater (in absolute value) than 1, i.e. a change greater than 2-fold when comparing brain vs. universal reference.

```
R> e <- exprs(eset)
R> index <- which(eset[["Key"]] == "brain")
R> d <- rowMeans(e[, index]) - rowMeans(e[,
  -index])
R> a <- rowMeans(e)
R> sum(abs(d) > 1)

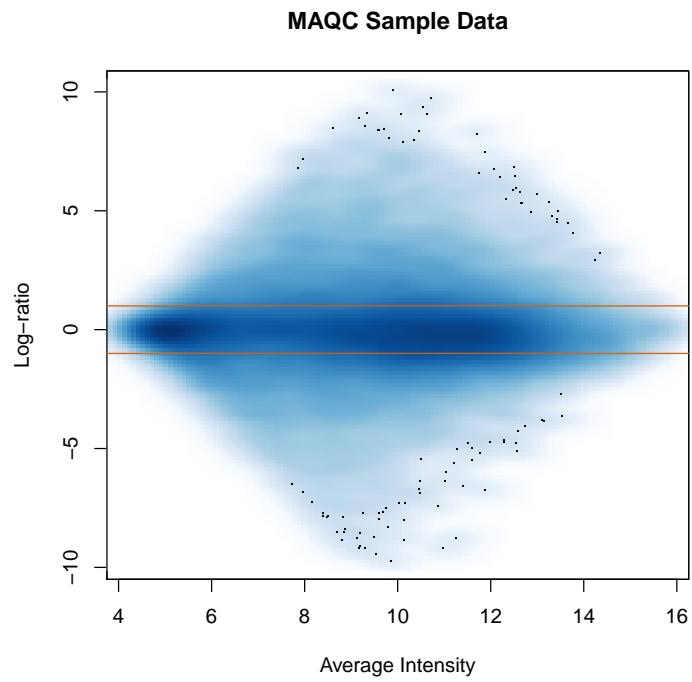
[1] 10043
```

Another approach is to use *t*-tests to infer whether or not there is differential expression.

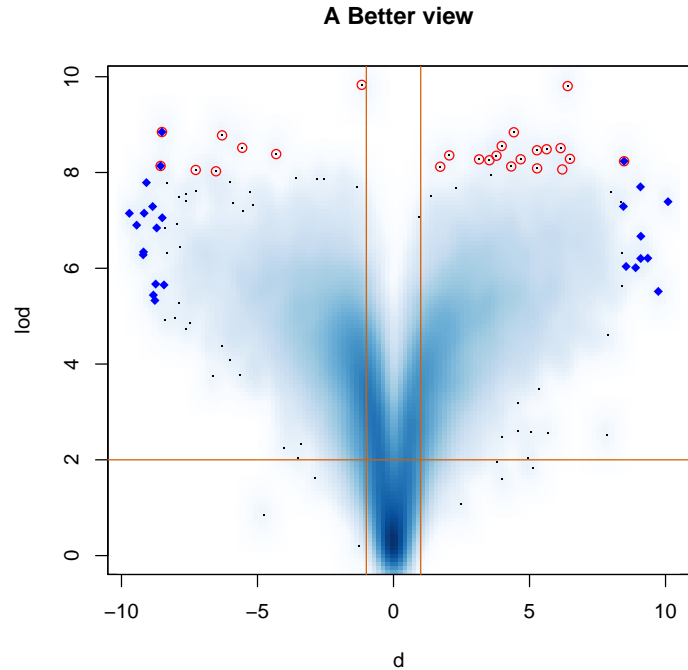
```
R> tt <- rowttests(e, factor(eset[["Key"]]))
R> lod <- -log10(tt[["p.value"]])
```

The MA plot can be used to visualize the behavior of the log-ratio as a function of average log-intensity. Features with log-ratios greater (in absolute value) than 1 are candidates for being classified as differentially expressed.

```
R> smoothScatter(a, d, xlab = "Average Intensity",
  ylab = "Log-ratio", main = "MAQC Sample Data")
R> abline(h = c(-1, 1), col = 2)
```



The use of t -tests allows us to use the volcano plot to visualize candidates for differential expression. Below, we highlight, in blue, the top 25 in log-ratio and,



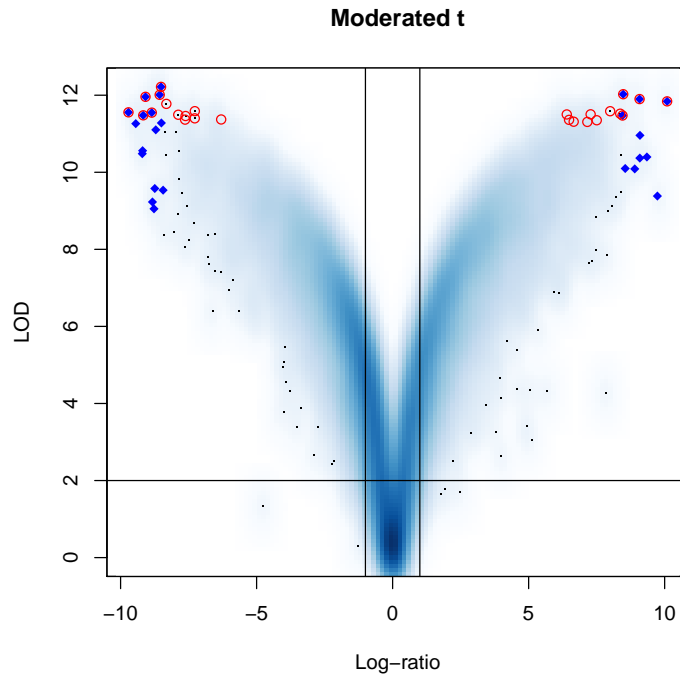
in red, the top 25 in effect size.

The *limma* Package can also be used to assess difference in expression between the two groups.

```
R> design <- model.matrix(~factor(eset[["Key"]]))
R> fit <- lmFit(eset, design)
R> eBayes <- eBayes(fit)
R> lod <- -log10(eBayes[["p.value"]][,
  2])
R> mtstat <- eBayes[["t"]][, 2]
```

The Empirical Bayes approach implemented in *limma* provides moderated *t*-statistic, shown to have a better performance when compared to the standard *t*-statistic. Below, we reconstruct the volcano plot, but using the moderated *t*-statistic.

```
R> o1 <- order(abs(d), decreasing = TRUE)[1:25]
R> o2 <- order(abs(mtstat), decreasing = TRUE)[1:25]
R> o <- union(o1, o2)
R> smoothScatter(d, lod, main = "Moderated t",
  xlab = "Log-ratio", ylab = "LOD")
R> points(d[o1], lod[o1], pch = 18, col = "blue")
R> points(d[o2], lod[o2], pch = 1, col = "red")
R> abline(h = 2, v = c(-1, 1))
```



The `topTable` command provides us a way of ranking genes for further evaluation. In the case below, we adjust for multiple testing by FDR and look at the Top-10 genes.

```
R> tab <- topTable(ebayes, coef = 2,
  adjust = "fdr", n = 10)
R> tab
```

	ID	logFC	AveExpr	t	P.Value
13761	NM_021871	8.5	8.7	118	6.1e-13
746	NM_000806	-8.5	8.6	-111	9.4e-13
169	NM_000184	8.6	9.2	111	9.8e-13
13760	NM_021870	9.1	9.2	109	1.1e-12
10465	NM_014841	-9.1	10.1	-107	1.3e-12
7467	NM_005277	-10.1	9.9	-105	1.4e-12
3286	NM_001034	8.3	8.9	103	1.7e-12
4919	NM_002421	7.3	8.4	96	2.6e-12
9238	NM_007325	-8.0	9.1	-96	2.6e-12
4201	NM_001622	9.7	9.9	96	2.8e-12
	adj.P.Val	B			
13761	3.8e-09	19			
746	3.8e-09	19			
169	3.8e-09	19			

```
13760 3.8e-09 19
10465 3.8e-09 19
7467 3.8e-09 19
3286 3.8e-09 18
4919 3.8e-09 18
9238 3.8e-09 18
4201 3.8e-09 18
```

6 Session Info

This document was created using the following:

```
R> sessionInfo()
```

```
R version 2.9.1 (2009-06-26)
x86_64-unknown-linux-gnu
```

```
locale:
```

```
LC_CTYPE=en_US;LC_NUMERIC=C;LC_TIME=en_US;LC_COLLATE=en_US;LC_MONETARY=C;LC_MESSAGES=en_US;L
```

```
attached base packages:
```

```
[1] tools      stats      graphics  grDevices
[5] utils      datasets  methods   base
```

```
other attached packages:
```

```
[1] pd.hg18.60mer.expr_2.4.1
[2] RColorBrewer_1.0-2
[3] limma_2.18.2
[4] genefilter_1.24.2
[5] maqcExpression4plex_1.2
[6] pd.mapping50k.xba240_0.4.1
[7] RSQLite_0.7-2
[8] DBI_0.2-4
[9] hapmap100kxba_1.3.2
[10] oligo_1.8.3
[11] preprocessCore_1.6.0
[12] oligoClasses_1.6.0
[13] Biobase_2.4.1
```

```
loaded via a namespace (and not attached):
```

```
[1] affxparser_1.16.0  affyio_1.12.0
[3] annotate_1.22.0    AnnotationDbi_1.6.1
[5] Biostrings_2.12.8 IRanges_1.2.3
[7] KernSmooth_2.23-2 splines_2.9.1
[9] survival_2.35-4   xtable_1.5-5
```