Analysis of Affymetrix and Nimblegen Data Using the oligo Package

Benilton Carvalho
carvalho@jhu.edu
Dept. of Biostatistics
Johns Hopkins University

BioC 2007
Seattle, WA
August, 2007

Outline

Why a package to handle feature-level data?

Better results can be achieved by starting the analysis from the feature-level data.

Working with the `affy` package

Requirements:

- CEL files;
- CDF package;
- Extra annotation for high-level explorations.

Functionality: **expression arrays.**

Advances in the Microarray World

- Not only Gene expression;
- Researchers now want:
  - Resequence;
  - Genotypes;
  - Copy-number;
- Arrays are getting denser;

The oligo Package Features

Designed to support (Affymetrix and NimbleGen arrays):

- Expression;
- Tiling;
- Exon;
- SNP.

The Structure

- Feature-level objects:
  - ExpressionFeatureSet;
  - TilingFeatureSet;
  - ExonFeatureSet;
  - SnpFeatureSet;
  - SnpCnvFeatureSet;
- Metadata (required):
  - `pdInfo`: SQLite-based;
  - `platformDesign`: data.frame-based;
- Analogy to `affy`
  - `AffyBatch` $\longrightarrow$ `FeatureSet`
  - `cdfenv` $\longrightarrow$ `pdInfo`

Data description

- 6 samples equally divided in 2 groups:
  - Brain;
  - Universal Reference.
- Human (HG18) 4-plex array:
  24K probe sets / 3 60-mers probes per probe set;
- Our objective:
  create a list of interesting units for further investigation.

Hands-on

- The first step is to load the packages;
- That will make the functions available to the user;
- Here, we're also setting a different color scheme.

Loading the packages

```
> library(oligo)
> library(maqcExpression4plex)
> library(genefilter)
> library(geneplotter)
> library(limma)
> library(RColorBrewer)
> palette(brewer.pal(8, "Dark2"))
```

Finding the Data

- We need the location where the XYS files are;
- then, we can list all this files, which will be loaded later.

Listing the files

```
> extdata <- system.file("extdata",
+     package = "maqcExpression4plex")
> xys.files <- list.xysfiles(extdata,
+     full.names = TRUE)
> basename(xys.files)

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

Reading the Data

- The next step is to read the XYS files, which contain the raw data;
- The `maqc` object is an ExpressionFeatureSet object.

Reading XYS files

```
> maqc <- read.xysfiles(xys.files)

Incompatible phenoData object. Created a new one.

> pd <- dir(extdata, pattern = "phenoData",
+     full.names = TRUE)
> phenoData(maqc) <- read.AnnotatedDataFrame(pd)
> class(maqc)

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

Exploring the Feature-level Data

The feature-level data is all available in the `exprs` slot:

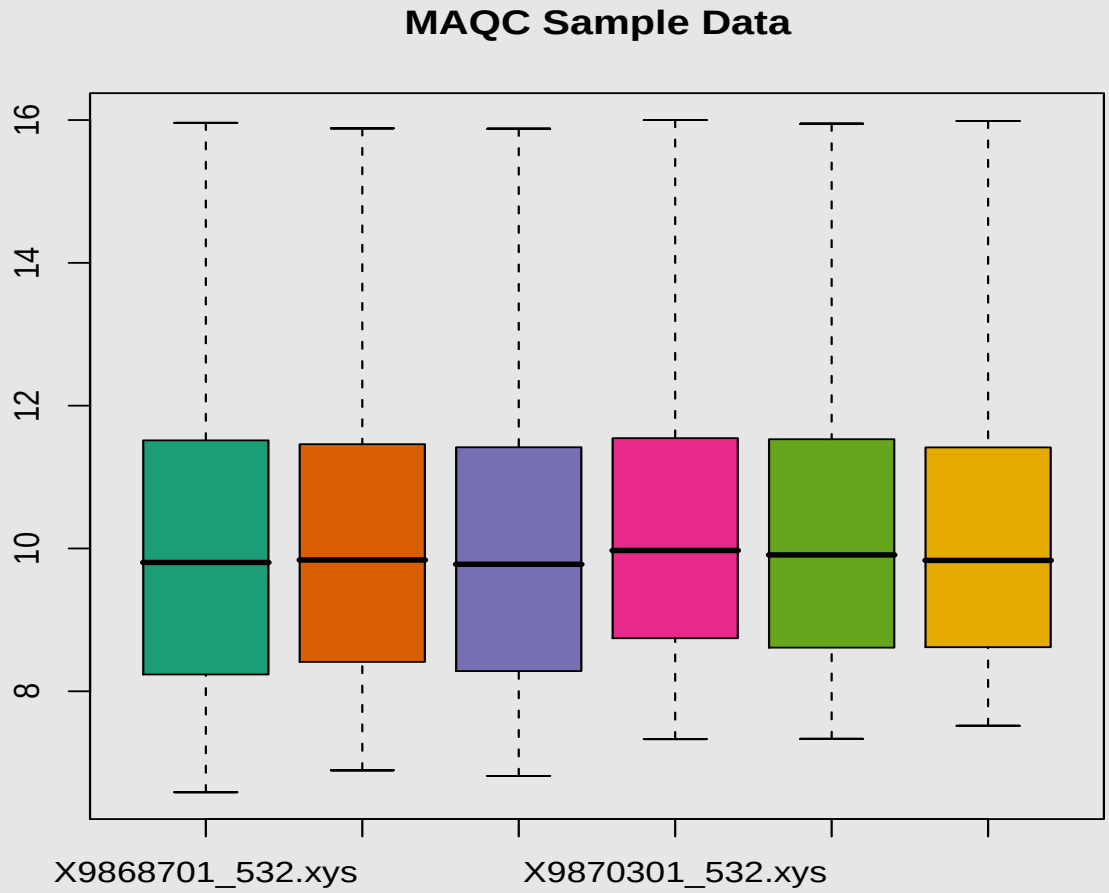
Accessing the raw data

```
> exprs(maqc)[10001:10010, 1, drop = FALSE]

      9868701_532.xys
10001            1167
10002             619
10003             753
10004             846
10005             306
10006             162
10007            3951
10008            4275
10009            1800
10010             164
```

Visualizing the Data - Boxplots

Creating boxplots

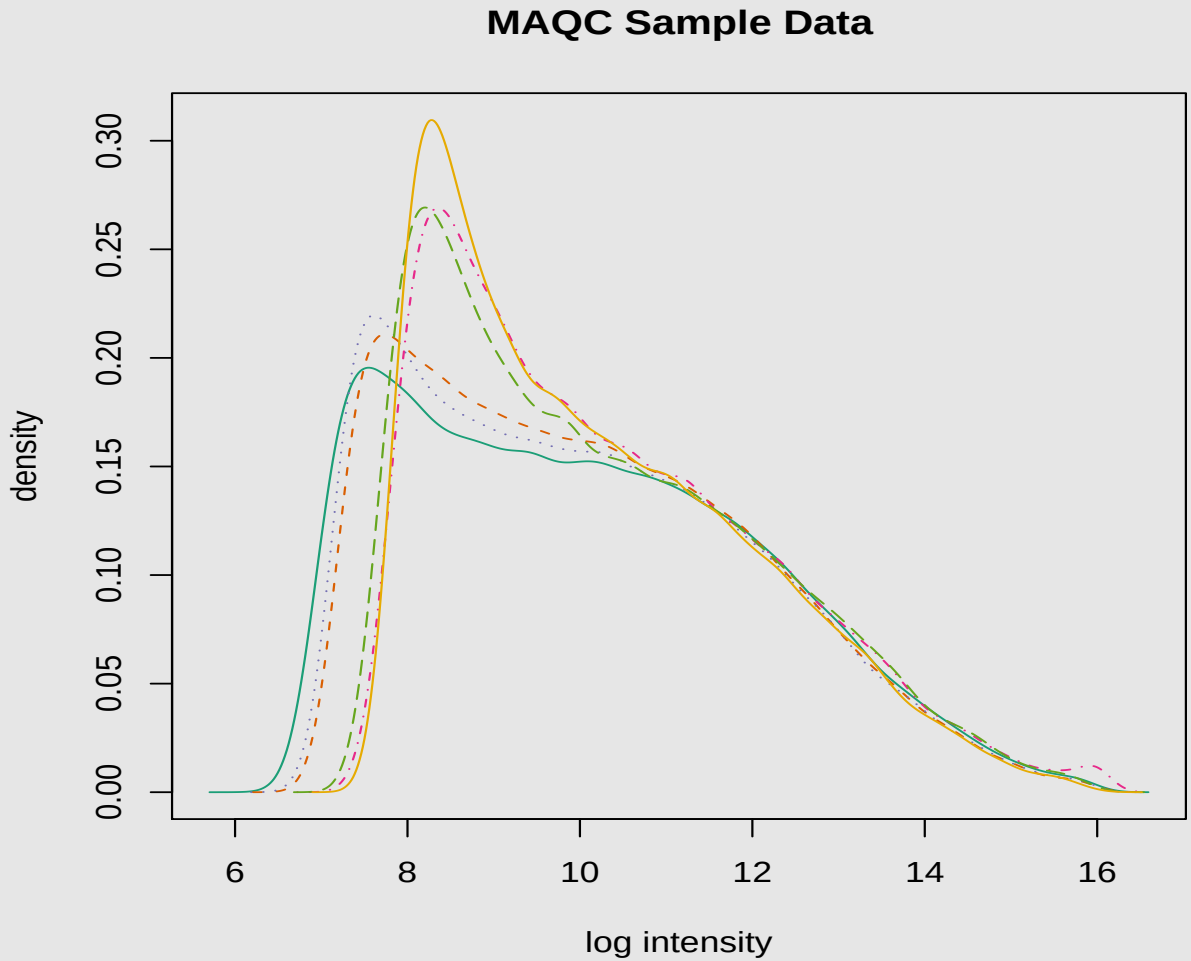
```
> boxplot(maqc, main = "MAQC Sample Data")
```



Visualizing the Data - Smoothed Histograms

Creating smoothed histograms

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



Summarizing the Data with RMA

- Background subtraction;
- Quantile normalization;
- Summarization;

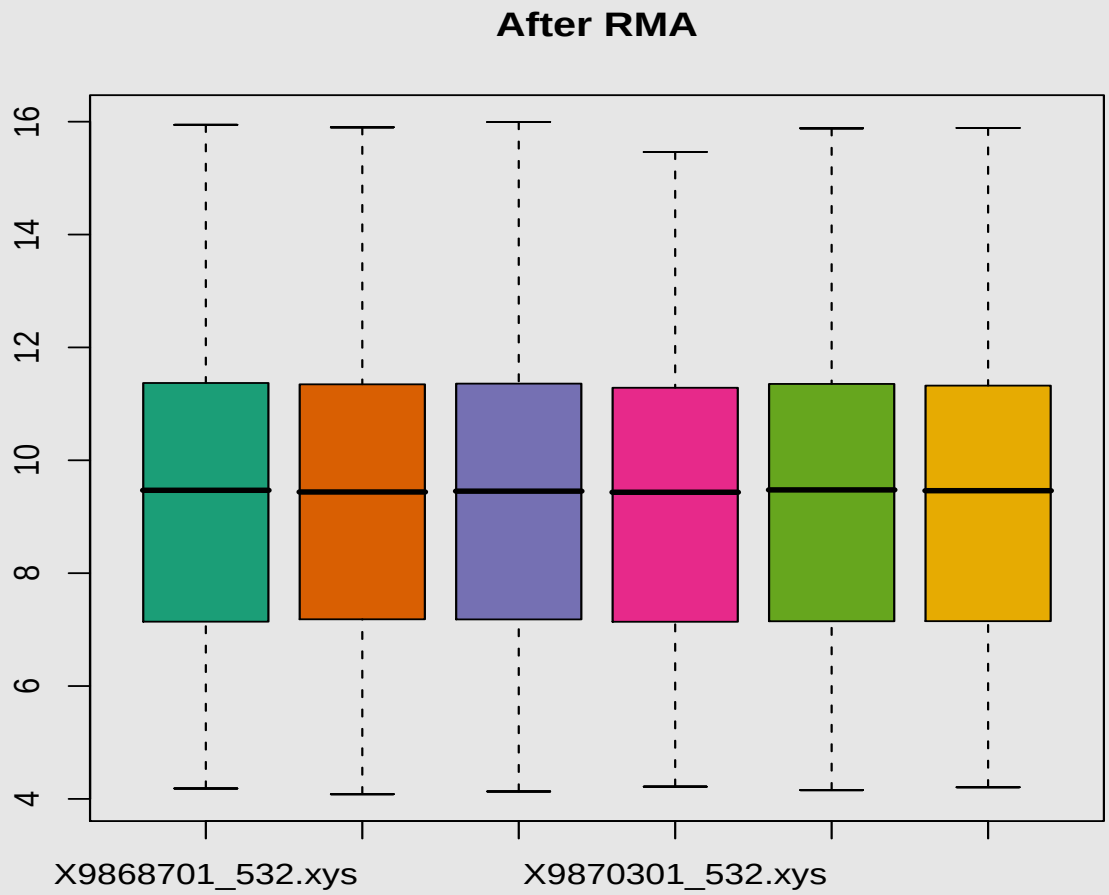
Running RMA

```
> eset <- rma(maqc)

Background correcting
Normalizing

> class(eset)

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

RMA Results - Boxplots

Visualizing the Summarized Data

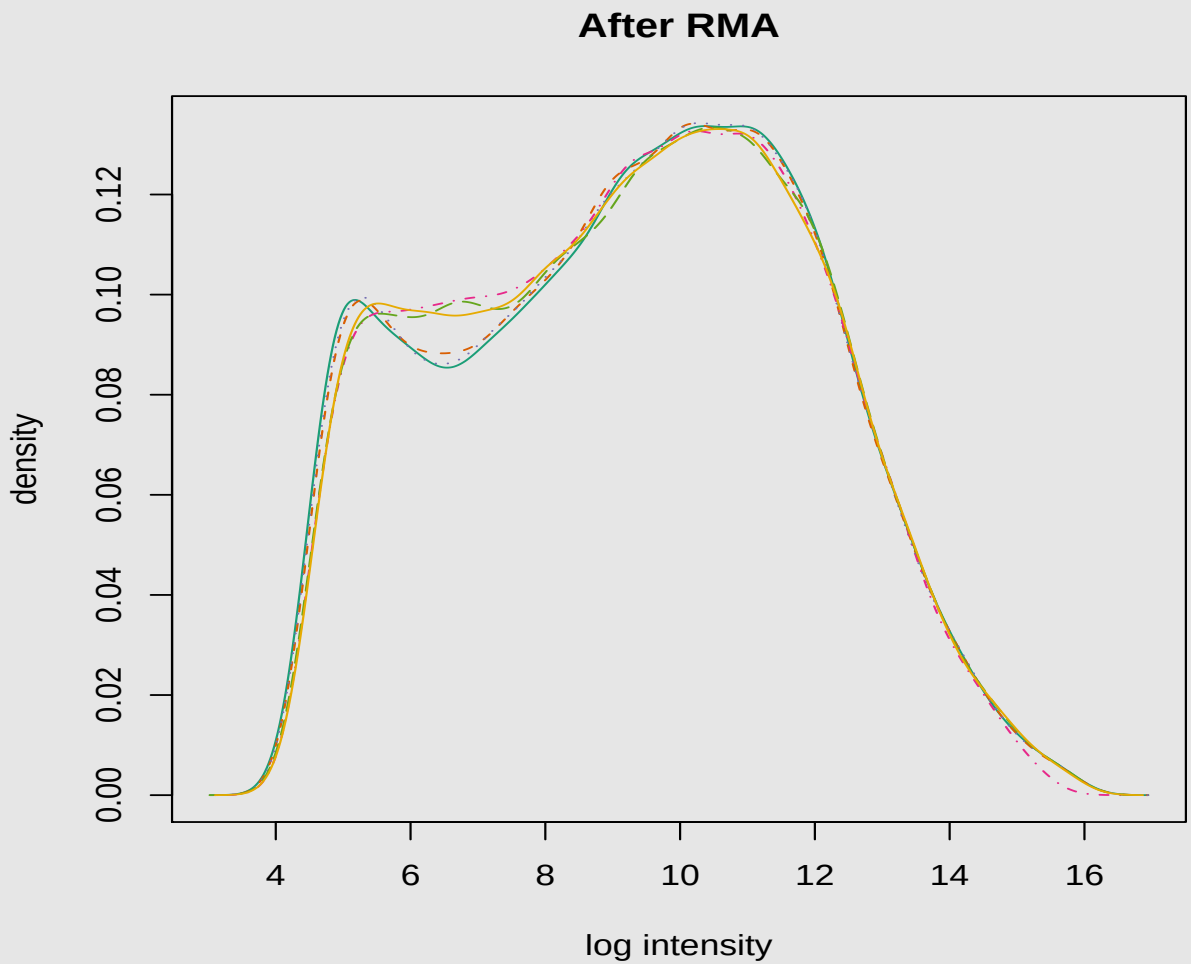
```
> boxplot(eset, main = "After RMA")
```



RMA Results - Smoothed Histogram

Visualizing the Summarized Data

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

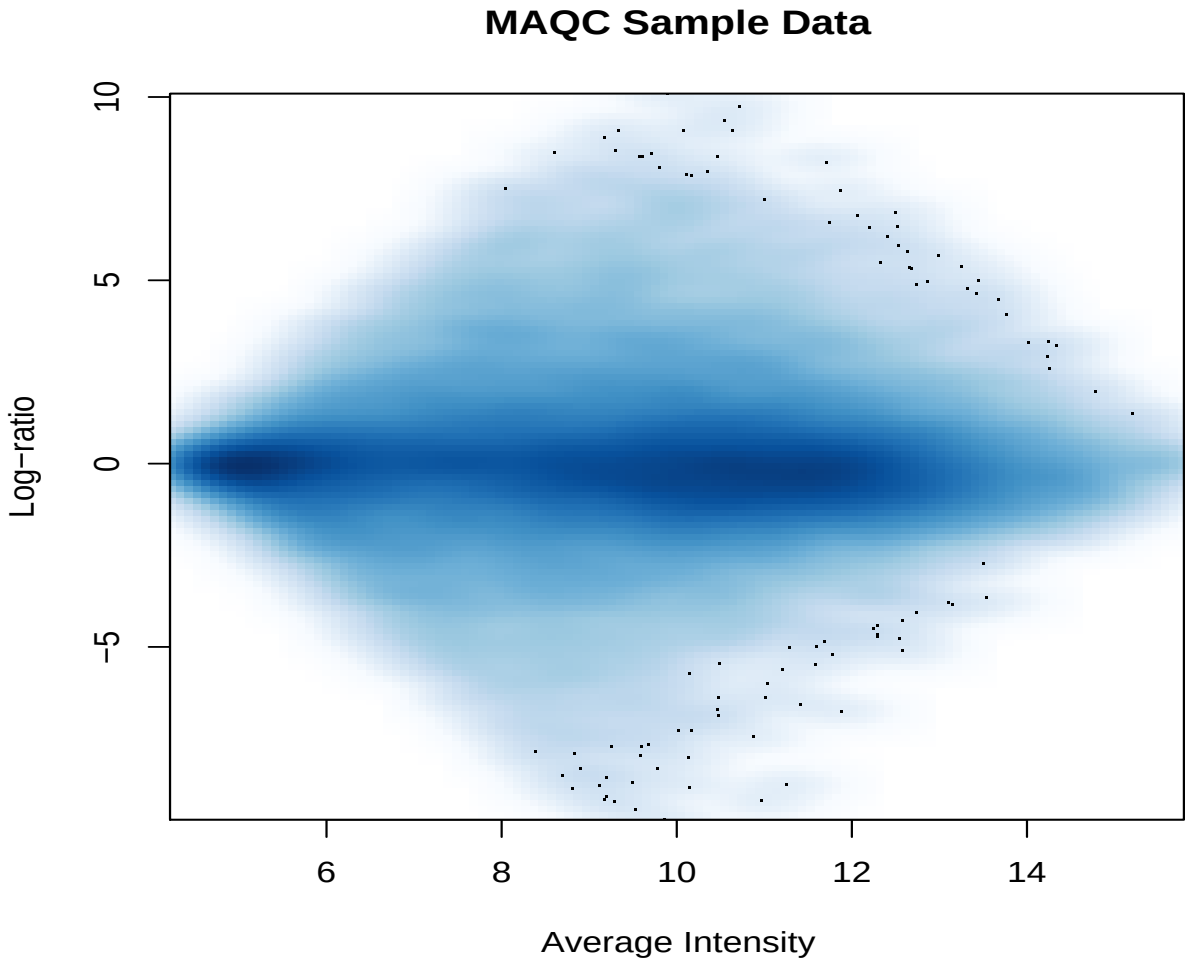


Assessing Differential Expression

- Identify the groups in the data;
- Compute the log-ratio between the groups;
- Compute the average expression per gene;
- Naïve approach, check every gene with $|d| > 1$.

Differential Expression

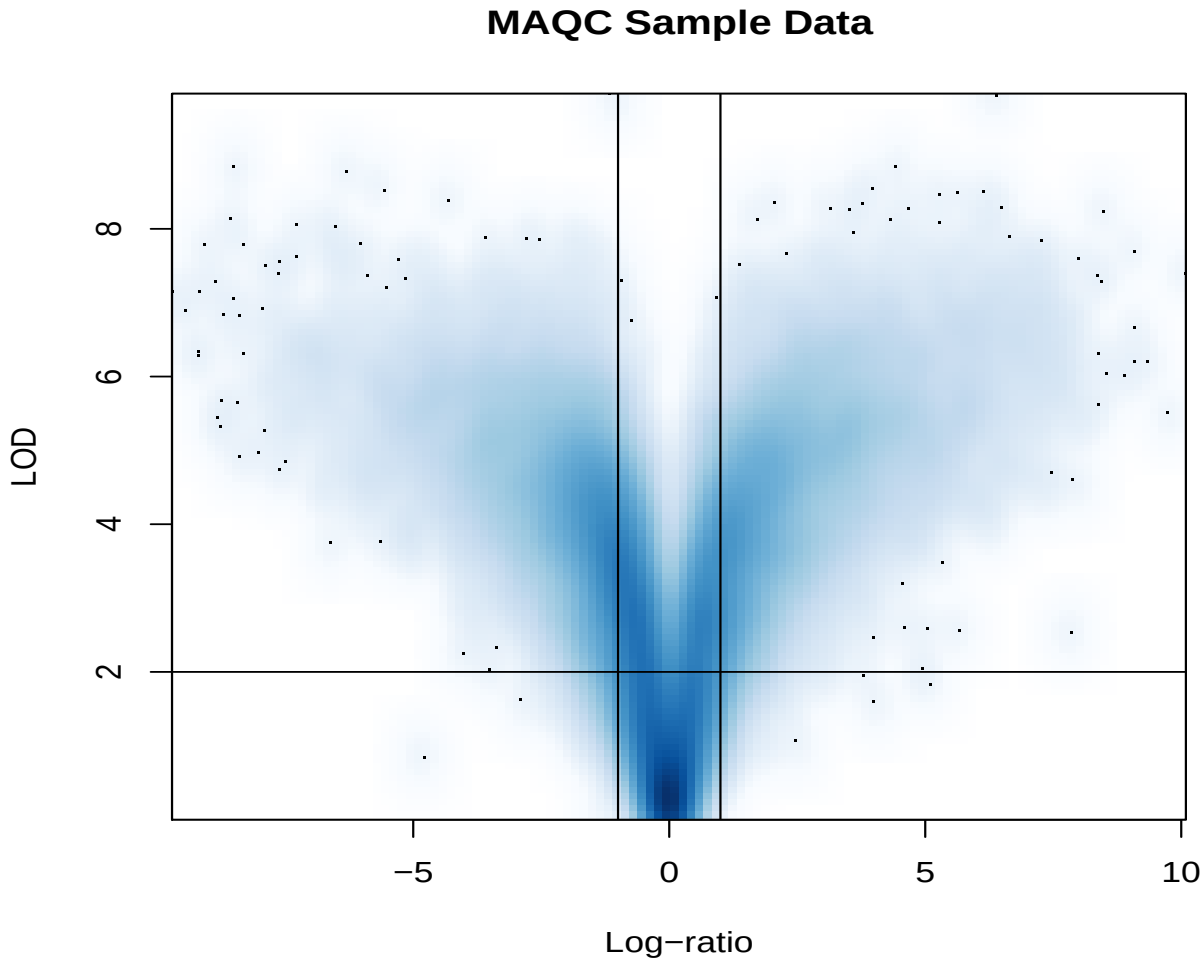
```
> e <- exprs(eset)
> dim(e)

[1] 24000     6

> index <- 1:3
> d <- rowMeans(e[, index]) -
+     rowMeans(e[, -index])
> a <- rowMeans(e)
> sum(abs(d) > 1)

[1] 10043

> smoothScatter(a, d, xlab = "Average Intensity",
+     ylab = "Log-ratio", main = "MAQC Sample Data")
```


MAQC Sample Data
Log−ratio
10
5
0
−5
6
8
10
12
14
Average Intensity

Assessing Differential Expression via *t*-tests

t-tests for Every Gene

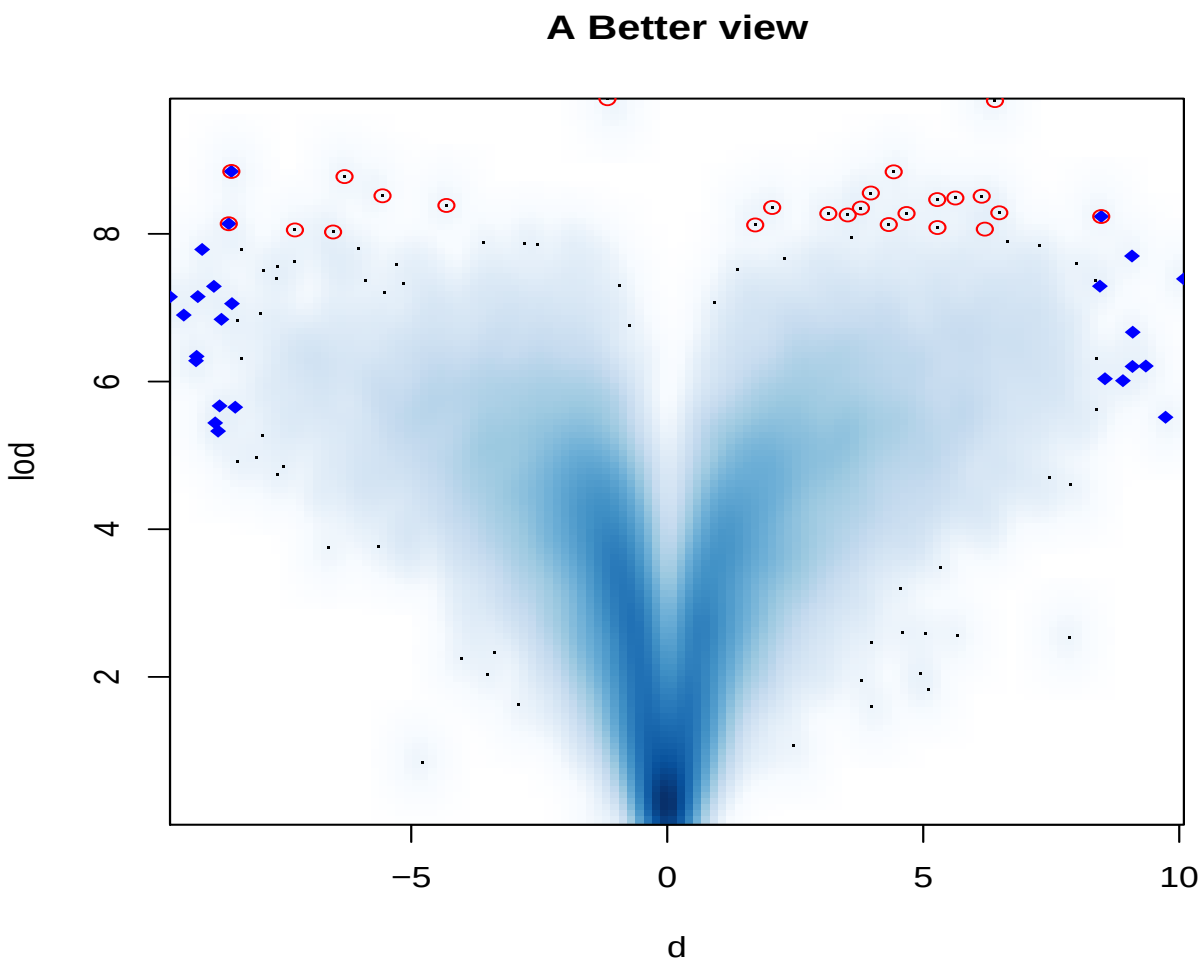
```
> tt <- rowttests(e, factor(eset$Key))
> lod <- -log10(tt$p.value)

> smoothScatter(d, lod, xlab = "Log-ratio",
+     ylab = "LOD", main = "MAQC Sample Data")
> abline(h = 2, v = c(-1, 1))
```


MAQC Sample Data
LOD
2
4
6
8
−5
0
5
10
Log−ratio

Improving the Volcano plot

Volcano plot

```
> o1 <- order(abs(d), decreasing = TRUE)[1:25]
> o2 <- order(abs(tt$statistic),
+     decreasing = TRUE)[1:25]
> o <- union(o1, o2)

> smoothScatter(d, lod, main = "A Better view")
> points(d[o1], lod[o1], pch = 18,
+     col = "blue")
> points(d[o2], lod[o2], pch = 1,
+     col = "red")
```


A Better view
lod
2
4
6
8
−5
0
5
10
d

Fitting a Linear Model Using limma

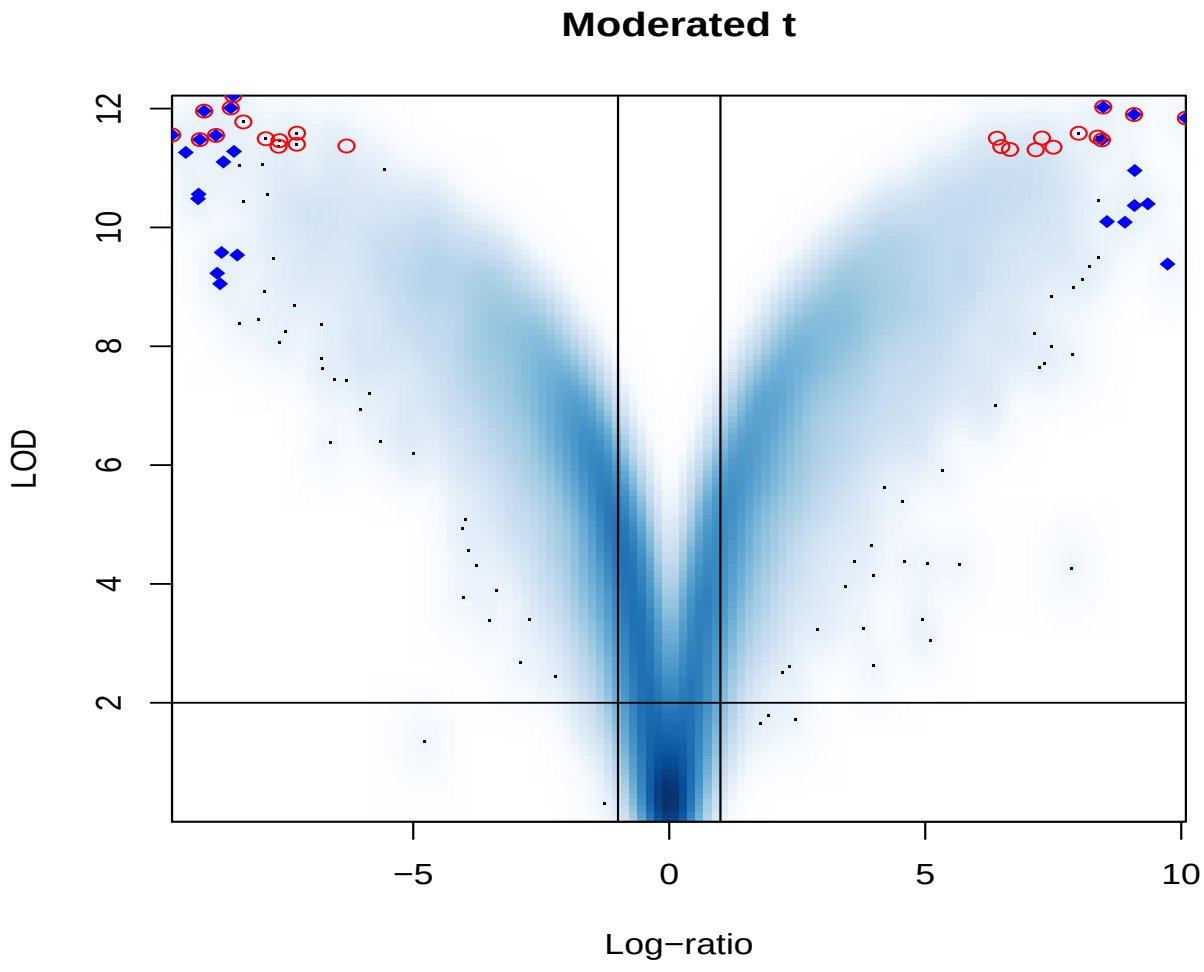
Fitting the model

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

Improving the Volcano plot

Volcano plot

```
> o1 <- order(abs(d), decreasing = TRUE)[1:25]
> o2 <- order(abs(mtstat), decreasing = TRUE)[1:25]
> o <- union(o1, o2)

> smoothScatter(d, lod, main = "Moderated t",
+     xlab = "Log-ratio", ylab = "LOD")
> points(d[o1], lod[o1], pch = 18, col = "blue")
> points(d[o2], lod[o2], pch = 1, col = "red")
> abline(h = 2, v = c(-1, 1))
```


Moderated t
LOD
12
10
8
6
4
2
−5
0
5
10
Log−ratio

Getting a list of interesting genes

Creating a top-table

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

	ID	logFC	AveExpr	t	P.Value	adj.P.Val	B
13761	NM_021871	8.5	8.7	118	6.1e-13	3.8e-09	19
746	NM_000806	-8.5	8.6	-111	9.4e-13	3.8e-09	19
169	NM_000184	8.6	9.2	111	9.8e-13	3.8e-09	19
13760	NM_021870	9.1	9.2	109	1.1e-12	3.8e-09	19
10465	NM_014841	-9.1	10.1	-107	1.3e-12	3.8e-09	19
7467	NM_005277	-10.1	9.9	-105	1.4e-12	3.8e-09	19
3286	NM_001034	8.3	8.9	103	1.7e-12	3.8e-09	18
4919	NM_002421	7.3	8.4	96	2.6e-12	3.8e-09	18
9238	NM_007325	-8.0	9.1	-96	2.6e-12	3.8e-09	18
4201	NM_001622	9.7	9.9	96	2.8e-12	3.8e-09	18

SNP Chips

Requirements to Handle SNP Arrays

`pdInfo` packages:

Array	Annotation Package	Approx. size	Version
50K XBA	pd.mapping50k.xba240	150 MB	0.3.2
50K HIND	pd.mapping50k.hind240	150 MB	0.3.2
250K STY	pd.mapping250k.sty	250 MB	0.3.2
250K NSP	pd.mapping250k.nsp	250 MB	0.3.2
SNP 6[1]	pd.genomewidesnp.6	480 MB	NA

[1] under development

Preprocessing SNP Arrays

The SNPRMA algorithm can be applied for normalization and summarization.

- **Normalization**: against a reference distribution;
- **Summarization**: via median-polish to the SNP-Allele-Strand-level.

Running SNPRMA on CEL files

We will start by identifying the CEL files of interest, then the `justSNPRMA` function will be applied.

Preparing the Data

```
> library("oligo")
> library("hapmap100kxba")
> pathCelFiles <- system.file("celFiles",
+     package = "hapmap100kxba")
> fullFilenames <- list.celfiles(path = pathCelFiles,
+     full.names = TRUE)
```

Running SNPRMA

```
> temporaryDir <- tempdir()
> preProcessedData <- justSNPRMA(fullFilenames,
+     tmpdir = temporaryDir)

Calculating Expression

> preProcessedData$gender <- c("female",
+     "female", "male")
```

Exploring Summarized Data

Average intensities and log-ratios are defined as across allele and within strand, ie:

$$A_s = \frac{\theta_{A,s} + \theta_{B,s}}{2} \quad (1)$$

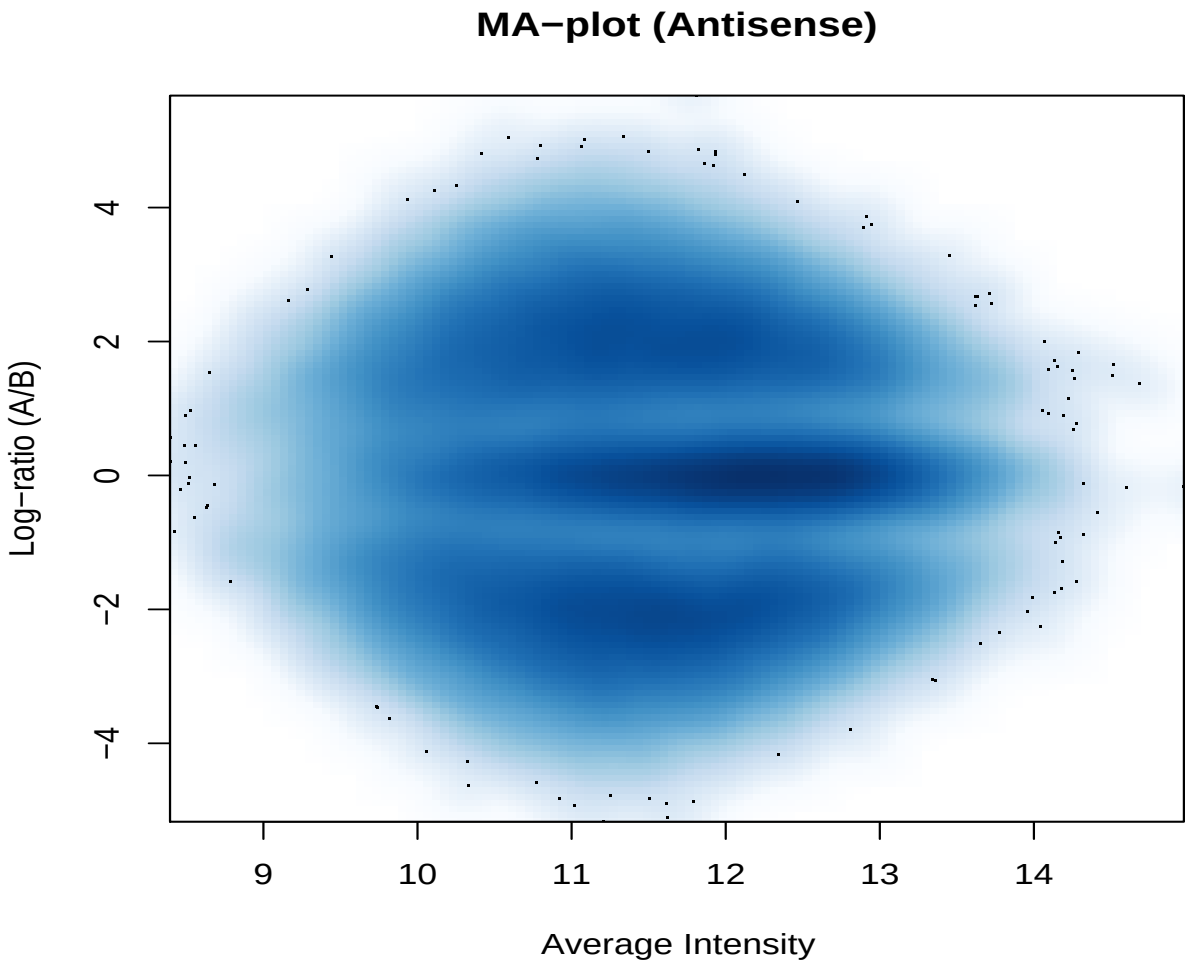
$$M_s = \theta_{A,s} - \theta_{B,s}, \quad (2)$$

where s defines the strand (antisense or sense). These quantities can be obtained via `getA` and `getM` methods, which return high-dimensional arrays with dimensions corresponding to SNP's, samples and strands, respectively.

MA-plots for SNP chips

MA-plot for SNP Chip

```
> theA <- getA(preProcessedData)
> theM <- getM(preProcessedData)
> dim(theA)

[1] 58960     3     2

> smoothScatter(theA[, 1, 1],
+     theM[, 1, 1], main = "MA-plot (Antisense)",
+     xlab = "Average Intensity",
+     ylab = "Log-ratio (A/B)")
```


MA−plot (Antisense)
Log−ratio (A/B)
4
2
0
−2
−4
9
10
11
12
13
14
Average Intensity

Considerations about CRLMM

- The SNPRMA method returns an object of class `SnpQSet`, which can be later used by the CRLMM algorithm to produce genotype calls.
- CRLMM accounts for average intensity and fragment length effects via EM, which may take long time to run.
- To save time in subsequent analyses, we must specify the name of the file that will store the results obtained with the EM algorithm using the `correctionFile` argument. If the file passed to `correctionFile` does not exist, it is created, otherwise it is loaded.

Running CRLMM

Running CRLMM

```
> crlmmOut <- crlmm(preProcessedData,
+     correctionFile = "exampleCorrection.rda",
+     verbose = FALSE)
> calls(crlmmOut)[1:3, 1]

SNP_A-1507972 SNP_A-1510136
            3             3
SNP_A-1511055
            3

> range(callsConfidence(crlmmOut))

[1] 0.49 1.00
```

Exploring the Annotation Packages

The user who is willing to make deeper investigation using the annotations provided for each SNP array can use SQL queries to access other information that might not be directly exposed.

Exploring the Annotation Packages

Checking Available Tables

```
> conn <- db(crlmmOut)
> dbListTables(conn)
 [1] "featureSet"   "mmfeature"
 [3] "pm_mm"        "pmfeature"
 [5] "qcmmfeature"  "qcpm_qcmm"
 [7] "qcpmfeature"  "sequence"
 [9] "sqlite_stat1" "table_info"
```

Exploring the Annotation Packages

Checking Available Fields

```
> dbListFields(conn, "featureSet")

 [1] "fsetid"           "man_fsetid"
 [3] "affy_snp_id"      "dbsnp_rs_id"
 [5] "chrom"            "physical_pos"
 [7] "strand"           "cytoband"
 [9] "allele_a"         "allele_b"
[11] "gene_assoc"       "fragment_length"
```

Exploring the Annotation Packages

Performing an SQL Query

```
> fields <- c("man_fsetid, chrom, physical_pos")
> cond <- c("man_fsetid LIKE 'SNP%' LIMIT 5")
> sql <- paste("SELECT", fields,
+     "FROM featureSet WHERE", cond)
> dbGetQuery(conn, sql)

     man_fsetid chrom physical_pos
1 SNP_A-1650338     2    168433267
2 SNP_A-1716667    19     40749462
3 SNP_A-1712945    19     53411226
4 SNP_A-1711654    21     31501701
5 SNP_A-1717655     1     15312743
```

Exploring the Annotation Packages

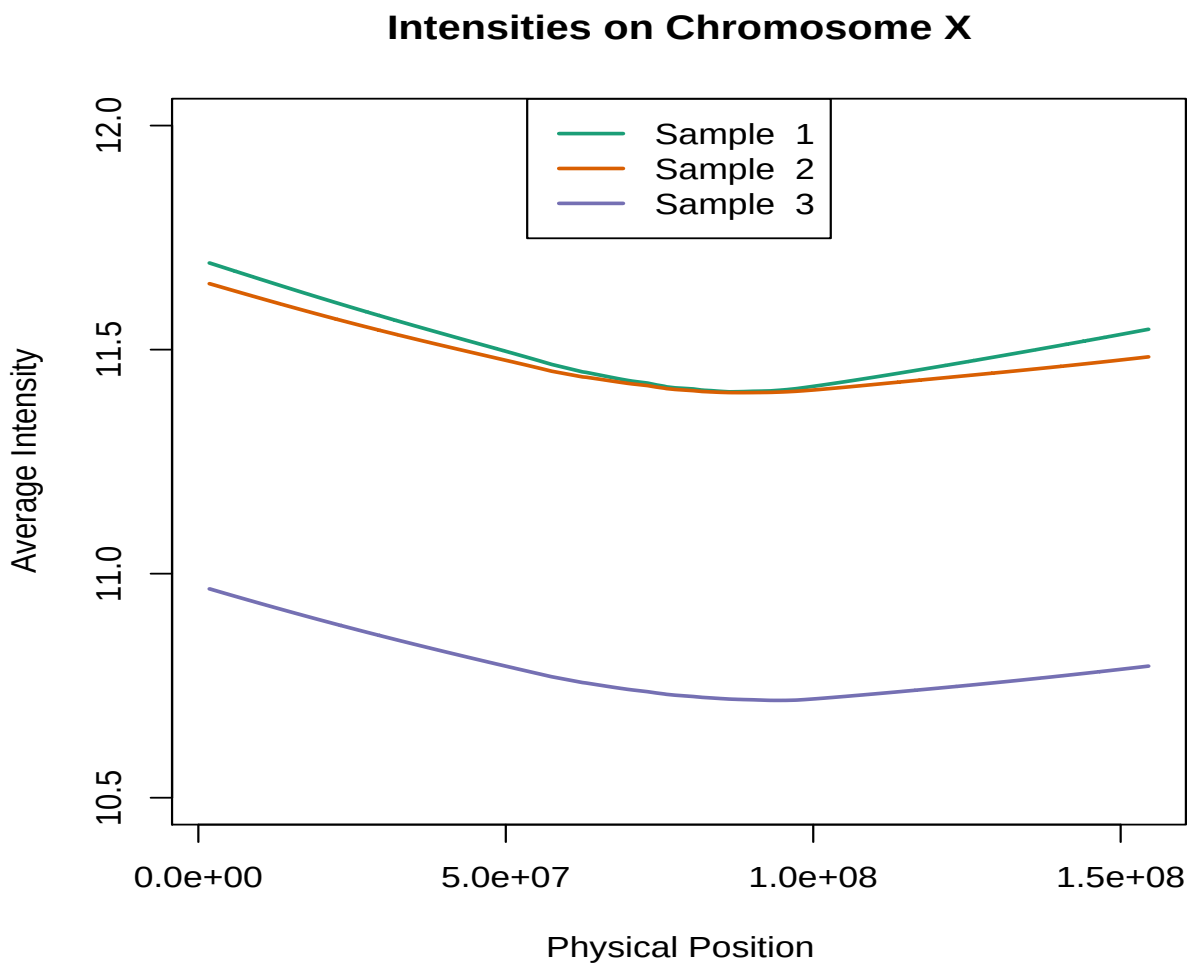
Locations for SNP on the X Chromosome

```
> p1 <- "SELECT man_fsetid, physical_pos"
> p2 <- "FROM featureSet WHERE man_fsetid"
> p3 <- "LIKE 'SNP%' AND chrom='X'"
> p4 <- "ORDER BY physical_pos"
> sql <- paste(p1, p2, p3, p4)
> x.info <- dbGetQuery(conn, sql)
```

Exploring the Annotation Packages

Locations for SNP on the X Chromosome

```
> idx <- match(x.info[, 1], rownames(theA))
> tmpA <- rowMeans(theA[idx, , ],
+     dims = 2)

> plot(1, type = "n", xlab = "Physical Position",
+     ylab = "Average Intensity",
+     main = "Intensities on Chromosome X",
+     ylim = c(10.5, 12), xlim = range(x.info[,
+         2]))
> for (i in 1:3) lines(lowess(x.info[,
+     2], tmpA[, i]), col = i, lwd = 2)
> legend("top", paste("Sample ",
+     1:3), col = 1:3, lwd = 2, lty = 1)
```


Intensities on Chromosome X
Sample 1
Sample 2
Sample 3
Average Intensity
12.0
11.5
11.0
10.5
0.0e+00
5.0e+07
1.0e+08
1.5e+08
Physical Position

Final Remarks

- The `oligo` package supports a variety of arrays;
- Code for parsing the raw data files is ready;
- The number of methods is still limited;
- As the code matures, other developers will - I hope! - contribute with more methods;
- Implementing - efficient - metadata packages for other applications;
- CRLMM was demonstrated to outperform BRLMM in a number of assessments;

```
> sessionInfo()

R version 2.6.0 Under development (unstable) (2007-07-27 r42342)
x86_64-unknown-linux-gnu

locale:
LC_CTYPE=en_US.UTF-8;LC_NUMERIC=C;LC_TIME=en_US.UTF-8;LC_COLLATE=en_US.U

attached base packages:
[1] splines   tools     stats     graphics  grDevices
[6] utils     datasets  methods   base

other attached packages:
 [1] pd.mapping50k.xba240_0.3.2  hapmap100kxba_1.1
 [3] pd.hg18.60mer.expr_1.1.1    RColorBrewer_1.0-1
 [5] limma_2.11.9                geneplotter_1.15.2
 [7] lattice_0.16-2              annotate_1.15.2
 [9] genefilter_1.15.3           survival_2.32
[11] maqcExpression4plex_1.0     oligo_1.1.10
[13] AnnotationDbi_0.0.83        preprocessCore_0.99.12
[15] BufferedMatrixMethods_1.1.4 BufferedMatrix_1.1.3
[17] RSQLite_0.6-0               DBI_0.2-3
[19] affyio_1.5.6                Biobase_1.15.23

loaded via a namespace (and not attached):
[1] affxparser_1.9.2   grid_2.6.0         KernSmooth_2.22-20
```