An Introduction to the Oligo Package

Benilton Carvalho

March, 2007

1 Introduction

The oligo package is designed to support all microarray designs provided by Affymetrix and NimbleGen: expression, tiling, SNP and exon arrays. With the increase in the density of the current technologies, oligo uses the resources offered by the BufferedMatrix packages to handle the feature-level information. As of now, chip-specific packages are built via makePlatformDesign and transitioning to the pdInfoBuilder package, which creates the data packages for the Affymetrix SNP arrays.

2 Analyzing Affymetrix SNP Arrays

Genotyping can be performed using oligo and you will need:

- oligo and its dependencies;
- Chip specific data package, eg. pd.mapping50k.xba240: package that contains the array specifications and SNP annotation.
- CEL files.

Figure 1 shows the general workflow for genotyping using the oligo package.

We will start by loading the oligo package and importing the CEL files available on the sampleDataAffy100K. The intensity matrix will be a *BufferedMatrix* object and this will require the use of temporary files in order to reduce the RAM usage. Although the temporary files can be stored anywhere, a better approach will be to use a local directory rather than using a directory on the network. The `tmpdir` in the `read.celfiles()` sets the directory where the temporary files are going to be stored.

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

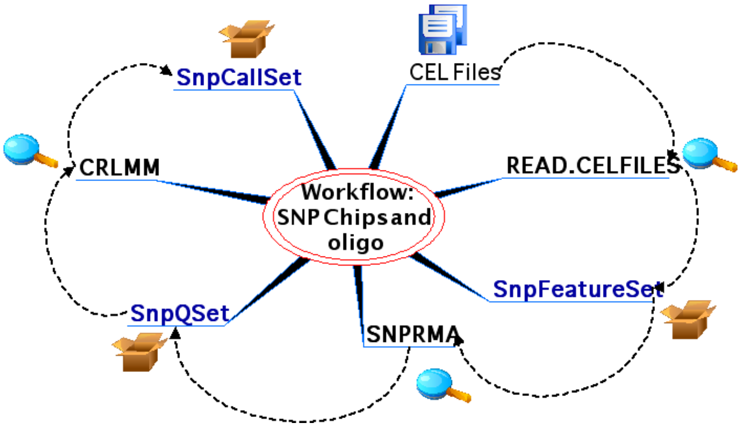


Figure 1: Genotyping workflow using the oligo package.

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

```
Calculating Expression
```

The SNPRMA approach described in [1] is implemented in the `justSNPRMA` methods (see Figure 2), which preprocessess the CEL files in a more effective way in terms of memory management. It returns a *SnpQSet* object without creating a *SnpFeatureSet object*. The *SnpQSet* object contains the summarized data.

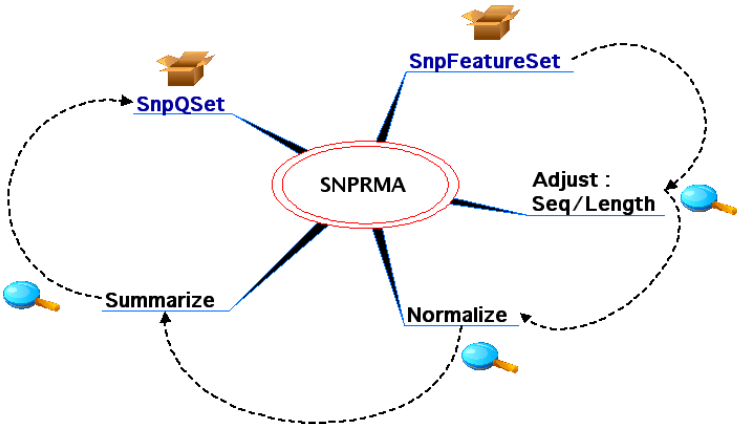


Figure 2: SNPRMA overview

For each SNP there are four numbers $(\theta_{A-}, \theta_{B-}, \theta_{A+}, \theta_{B+})$, which are proportional to the log-intensities in each of these combinations of allele and strand

(−: antisense; +: sense). They are represented by four matrices: `antisenseThetaA`, `antisenseThetaB`, `senseThetaA` and `senseThetaB`, which are the components of the *SnpQSet* object. One can extract these objects using accessors of the same name.

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

$$\begin{aligned} A_s &= \frac{\theta_{A,s} + \theta_{B,s}}{2} & (1) \\ M_s &= \theta_{A,s} - \theta_{B,s}, & (2) \end{aligned}$$

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.

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

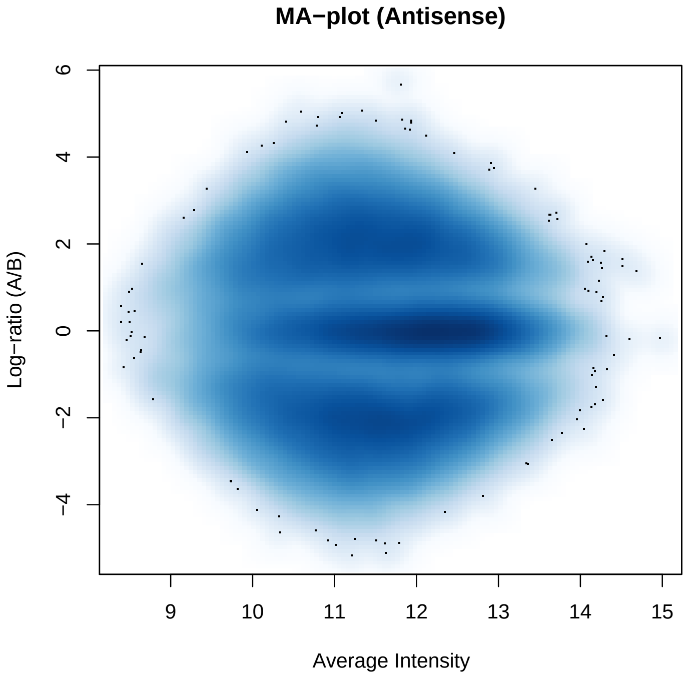
```
[1] 58960     3     2
```

```
R> str(theM)
```

```
 num [1:58960, 1:3, 1:2] -2.482 -3.525 -2.867 -0.144 -1.528 ...
 - attr(*, "dimnames")=List of 3
  ..$ : chr [1:58960] "SNP_A-1507972" "SNP_A-1510136" "SNP_A-1511055" "SNP_A-1518245" ...
  ..$ : chr [1:3] "NA06985.CEL.gz" "NA06991.CEL.gz" "NA06993.CEL.gz"
  ..$ : chr [1:2] "antisense" "sense"
```

These measures can be used, for example, to create an MA-plot and are later used for genotyping. The example below generates an MA-plot for the first sample using only the antisense strand data:

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



The CRLMM algorithm [1] can be applied on a *SnpQSet* object in order to produce genotype calls. It involves running a mixture of regressions via EM algorithm to adjust for average intensity and fragment length in the log-ratio scale. These adjustments may take long time to run, depending on the combination of number of samples and computer resources available. 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. Figure 3 presents a diagram of the CRLMM algorithm:

A word of warning: the `crlmm()` method searches for a variable `gender` in the *phenoData* slot of the *SnpQSet* object. If it fails to find that variable, it will try estimate the gender from the data. If there is not enough discrimination power to estimate the gender, the following error message will be returned:

empty cluster: try a better set of initial centers

Increasing the sample size is one of the possible solutions, although the preferred one is to have `gender` already defined in the *phenoData* slot.

The *phenoData* slot includes covariates about the samples. Genotyping and copy number analyses often make use of gender information in order to provide more precise inferences. The code below exemplifies the creation of the *phenoData* object.

```
R> aboutSamples <- data.frame(gender = c("female",
     "female", "male"))
```

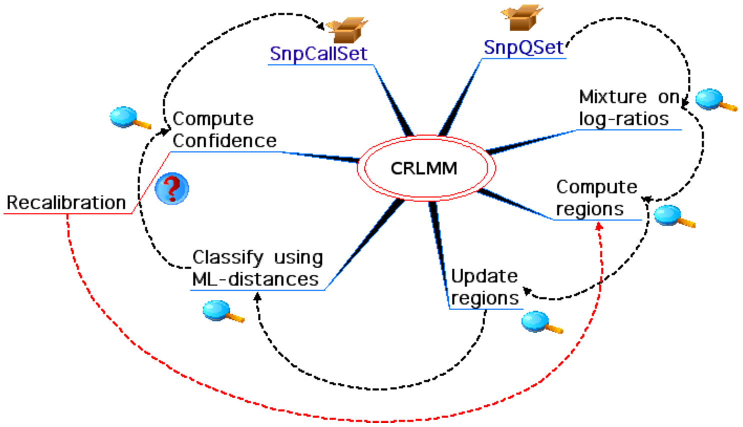


Figure 3: CRLMM Algorithm diagram

```
R> rownames(aboutSamples) <- sampleNames(preProcessedData)
R> aboutVars <- data.frame(labelDescription = "male/female")
R> rownames(aboutVars) <- "gender"
R> phenoData(preProcessedData) <- new("AnnotatedDataFrame",
     data = aboutSamples, varMetadata = aboutVars)
```

```
R> crlmmOut <- crlmm(preProcessedData, correctionFile = "exampleCorrection.rda",
     verbose = FALSE)
```

The `crlmmOut` object above belongs to the *SnpCallSet* class and contains the genotype calls and confidence measures associated to the calls, represented respectively by the `calls` and `callsConfidence` matrices. These matrices can be accessed using the methods of the same name as demonstrated below:

```
R> calls(crlmmOut)[1:5, 1:2]
```

```
  NA06985.CEL.gz NA06991.CEL.gz
1              3              3
2              3              3
3              3              3
4              2              3
5              3              3
```

```
R> callsConfidence(crlmmOut)[1:5, 1:2]
```

```
  NA06985.CEL.gz NA06991.CEL.gz
1      0.9999254      0.9998909
2      0.9999254      0.9999254
3      0.9999254      0.9999254
4      0.9997885      0.9999254
5      0.9997851      0.9997564
```

The genotype calls are represented by 1 (AA), 2 (AB) and 3 (BB). The confidence is the log-likelihood ratio of the two most likely calls.

2.1 Exploring the Annotation Package

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

The example below demonstrates how to see the available tables, fields and extract chromosome, physical location and cytoband for the first five SNP's (probes querying specific SNP's have names starting with the string "SNP").

```
R> conn <- db(preProcessedData)
R> dbListTables(conn)

 [1] "featureSet"   "mmfeature"    "pm_mm"
 [4] "pmfeature"    "qcmmfeature"  "qcpm_qcmm"
 [7] "qcpmfeature"  "sequence"     "sqlite_stat1"
[10] "table_info"

R> dbListFields(conn, "featureSet")

 [1] "fsetid"          "man_fsetid"      "affy_snp_id"
 [4] "dbsnp_rs_id"     "chrom"           "physical_pos"
 [7] "strand"          "cytoband"        "allele_a"
[10] "allele_b"        "gene_assoc"      "fragment_length"

R> sql <- "SELECT man_fsetid, chrom, physical_pos FROM featureSet WHERE man_fsetid LIKE 'SNP
R> 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
```

References

[1] Benilton Carvalho, Henrik Bengtsson, Terence P Speed, and Rafael A Irizarry. Exploration, normalization, and genotype calls of high density oligonucleotide snp array data. *Biostatistics*, Dec 2006.

3 Details

This document was written using:

```
R> sessionInfo()

R version 2.6.0 alpha (2007-09-05 r42787)
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.UTF-8;LC_MONETARY=en_

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

other attached packages:
 [1] geneplotter_1.15.9          lattice_0.16-4
 [3] annotate_1.15.7             pd.mapping50k.xba240_0.3.2
 [5] hapmap100kxba_1.1           oligo_1.1.17
 [7] oligoClasses_0.99.0         affxparser_1.9.4
 [9] AnnotationDbi_0.1.13        preprocessCore_0.99.12
[11] BufferedMatrixMethods_1.1.4 BufferedMatrix_1.1.4
[13] RSQLite_0.6-0               DBI_0.2-3
[15] affyio_1.5.9                Biobase_1.15.33

loaded via a namespace (and not attached):
[1] grid_2.6.0         KernSmooth_2.22-21 RColorBrewer_1.0-1
```