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. 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.

We will start by loading the oligo package and importing the CEL files available on the hapmap100kxba package. An output directory should also be defined and that is the location where the summary files, including genotype calls and confidences are stored.

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

The density of the SNP arrays increased in a way that it is not interesting to store the intensity matrix in memory; efficient methods to handle this situations have been developed and batches of SNPs are analyzed one at a time.

The genotyping algorithm implemented in oligo is described in [1]. The whole process can be described in three steps:

1. Normalization against a reference distribution;

2. Summarization via SNPRMA;

3. Genotype calling via CRLMM.

The normalization step is done by equalizing the observed intensities accordingly to a reference distribution, based on the 270 Hapmap samples. These samples are available to the public at http://www.hapmap.org. The normalization step will remove systematic biases, which are not due to biological factors.

The SNPRMA algorithm is responsible for summarizing the data. The design of the SNP arrays is such that up to the 250K density, there are probes for both alleles on both strands. On the SNP 6.0 platform, given a SNP, there are probes only on one strand.

Therefore, for the designs up to 250K, SNPRMA will create summaries at the SNP-Allele-Strand-level. 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.

For the SNP 6.0 array, a similar approach is taken, but the summaries are given at the SNP-Allele-level and there will be only (θ_A, θ_B) estimates for each SNP. An object of class *SnpCnvQSet* is returned and contains two matrices: `thetaA` and `thetaB`. Accessors with the same name are provided.

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. These measures are later used for genotyping.

The CRLMM algorithm can be applied on a *SnpQSet* or *SnpCnvQSet* 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.

```
R> crlmm(fullFilenames, outputDir, verbose = FALSE)

Removing temporary files ... OK.
```

The `crlmm` method does not return an object to the R session. Instead, it saves the objects to disk, as not all systems are guaranteed to meet the memory requirements that *SnpCallSetPlus* (for 100K and 500K arrays) or *SnpCnvCallSetPlus* (for SNP 5.0 and SNP 6.0 arrays) objects might need. For convenience, the `getCrlmmSummaries` will read the information from disk and make a *SnpCallSetPlus* or *SnpCnvCallSetPlus* object available to the user.

```
R> crlmmOut <- getCrlmmSummaries(outputDir)
R> calls(crlmmOut)[1:5, 1:2]

             NA06985.CEL NA06991.CEL
SNP_A-1507972           3           3
SNP_A-1510136           3           3
SNP_A-1511055           3           3
SNP_A-1518245           2           3
SNP_A-1641749           3           3

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

             NA06985.CEL NA06991.CEL
SNP_A-1507972   0.9999254   0.9998893
SNP_A-1510136   0.9999254   0.9999254
SNP_A-1511055   0.9999254   0.9999254
SNP_A-1518245   0.9997851   0.9999254
SNP_A-1641749   0.9997838   0.9997551

R> crlmmCalls <- readSummaries("calls", outputDir)
R> crlmmConf <- readSummaries("conf", outputDir)
R> crlmmCalls[1:5, 1:2]

             NA06985.CEL NA06991.CEL
SNP_A-1507972           3           3
SNP_A-1510136           3           3
SNP_A-1511055           3           3
SNP_A-1518245           2           3
SNP_A-1641749           3           3

R> crlmmConf[1:5, 1:2]

             NA06985.CEL NA06991.CEL
SNP_A-1507972   0.9999254   0.9998893
SNP_A-1510136   0.9999254   0.9999254
SNP_A-1511055   0.9999254   0.9999254
SNP_A-1518245   0.9997851   0.9999254
SNP_A-1641749   0.9997838   0.9997551
```

The genotype calls are represented by 1 (AA), 2 (AB) and 3 (BB). The confidence is the predicted probability that the algorithm made the right call.

Summaries generated by the algorithm can also be accessed from the R session. The options for summaries are "alleleA", "alleleB", "alleleA-sense", "alleleA-antisense", "alleleB-sense", "alleleB-antisense". The options "alleleA" and "alleleB" must be used **only** with SNP 5.0 and SNP 6.0 platforms. The remaining options must be used with 50K and 250K arrays.

```
R> alleleAsense <- readSummaries("alleleA-sense",
     outputDir)[1:5, ]
R> alleleBsense <- readSummaries("alleleB-sense",
     outputDir)[1:5, ]
R> log.ratios.sense <- alleleAsense - alleleBsense
R> log.ratios.sense[, 1:2]

              NA06985.CEL NA06991.CEL
SNP_A-1507972  -2.1105518   -2.074544
SNP_A-1510136  -2.3079444   -2.332064
SNP_A-1511055  -1.5782634   -1.495551
SNP_A-1518245   0.3572502   -1.528293
SNP_A-1641749  -1.6593009   -1.441162
```

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(pd.mapping50k.xba240)
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"
[13] "dbsnp"           "cnv"

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 2.9.1 (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;I

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

other attached packages:
[1] pd.mapping50k.xba240_0.4.1 RSQLite_0.7-2
[3] DBI_0.2-4                  hapmap100kxba_1.3.2
[5] oligo_1.8.3                preprocessCore_1.6.0
[7] oligoClasses_1.6.0         Biobase_2.4.1

loaded via a namespace (and not attached):
[1] affxparser_1.16.0 affyio_1.12.0     Biostrings_2.12.8
[4] IRanges_1.2.3     splines_2.9.1
```