Extending *oligo* with *SNPchip*

Robert Scharpf

March 10, 2008

Introduction

This vignette describes a pipeline for preprocessing and visualizing SNP-level summaries using the packages *oligo* and *SNPchip*. We use a set of unprocessed Affymetrix files (CEL files) available as experimental data packages on Bioconductor. A minimal set of commands to perform pre-processing with *oligo* are provided here, though one should consult the *oligo* vignette for additional information. An object of the processed data is provided with this package to reduce the time of computation – the code chunks for the preprocessing steps are not evaluated in the vignette. An example of using *oligo* to process a batch of 209 Affymetrix 100k CEL files and *VanillaICE* to identify regions of alterations are provided in the `hapmap100k` vignette in the directory `inst/testing` of the *VanillaICE* package. The `hapmap100k` vignette is not reproducible as it depends on access to the 209 CEL hapmap CEL files that are not provided with the *VanillaICE* package, but may be useful as a guideline when performing your own analyses. Comparable vignettes for hapmapAffy500k, hapmapAffy5.0, hapmapAffy6.0, and Illumina will also be added to *VanillaICE* in the near future.

1 Creating an instance of `oligoSnpSet`

The *oligo* vignette creates an instance of `SnpCallSetPlus`, `crlmmOut`, from the call to the function `crlmm`. For purposes of illustration, I subset the object to only include SNPs on chromosome 1. I also took the liberty of adding chromosome and physical position to the featureData slot. This object can be loaded by

```
> library(SNPchip)
> data(crlmmOut)
> class(crlmmOut)
```

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

The elements in the `assayData` for instances of `SnpCallSetPlus` is dependent on the Affymetrix platform.

```
> annotation(crlmmOut)
```

```
[1] "pd.mapping50k.xba240"
```

```
> ls(assayData(crlmmOut))
```

```
[1] "antisenseThetaA" "antisenseThetaB" "calls"
[4] "callsConfidence" "senseThetaA"     "senseThetaB"
```

```
> callset <- crlmmOut
```

1.1 Estimating copy number

The simplest way to obtain copy number estimates from `crlmmOut` is to coerce to an `oligoSnpSet`

```
> snpset <- as(callset, "oligoSnpSet")
```

The details for this coercion are provided below.
We expect that the intensities for the A and B alleles averaged over the sense and antisense strands will be proportional to the total copy number. Because the fluorescence at a SNP is very much SNP-dependent, we'll center each SNP at the median value and then recenter at the 'normal' copy number. Here, we define normal copy number to be two for autosomes, 1 for the male X, and 2 for the female X. Because we've noticed that homozygous genotype calls appear to have overall less fluorescence than the heterozygous genotype calls, we'll recenter the median intensities for the homozygous and heterozygous genotypes to be equivalent.
We begin by extracting an array of average values for the sense (A and B alleles) and antisense (A and B) strands.

```
> if (require(oligo)) {
+     A <- getA(callset)
+     dim(A)
+ } else {
+     stop("R package oligo is not available")
+ }
```

```
[1] 4669    3    2
```

We then average the sense and antisense values:

```
> log2cn <- rowMeans(A, dims = 2, na.rm = TRUE)
> dim(log2cn)
```

```
[1] 4669    3
```

The homozygous and heterozygous genotypes are centered separately since the average intensities differ:

```
> chr.matrix <- matrix(chromosome(callset), ncol = ncol(log2cn),
+     nrow = nrow(log2cn))
> median.hom <- median(log2cn[(calls(callset) ==
+     1 | calls(callset) == 3) & chr.matrix != "X"],
+     na.rm = TRUE)
> median.het <- median(log2cn[calls(callset) ==
+     2 & chr.matrix != "X"], na.rm = TRUE)
> recenterByGenotype <- function(x, callset, recenter.hom,
+     recenter.het) {
+     calls <- as.vector(calls(callset))
+     x[calls == 1 | calls == 3] <- x[calls == 1 |
+         calls == 3] - recenter.hom
+     x[calls == 2] <- x[calls == 2] - recenter.het
+     x
+ }
> for (j in 1:ncol(log2cn)) {
+     log2cn[, j] <- recenterByGenotype(log2cn[,
+         j], callset[, j], recenter.hom = median.hom,
+         recenter.het = median.het)
+ }
```

Next, we sweep out a robust estimate of the median from the samples (tries to put fluorescence intensities on a similar scale for each of the samples)

```
> f <- function(x, chromosome) {
+     tmp2 <- split(x, chromosome)
+     if (length(tmp2) > 15) {
+         idx <- order(sapply(tmp2, "median"))
+         tmp2 <- tmp2[idx]
+         tmp3 <- tmp2[-c(1:5, (length(tmp2) - 4):length(tmp2))]
+         med <- median(unlist(tmp3))
+     }
+     else {
+         med <- median(sapply(tmp2, "median"))
+     }
+     return(med)
+ }
> robust.median <- apply(log2cn, 2, f, chromosome(callset))
> log2cn <- sweep(log2cn, 2, robust.median)
```

We then sweep out the SNP-specific median intensities and recenter to the expected copy number (depends on chromosome):

```
> rowSweep <- function(callset, X, value, recenter,
+     j) {
+     if (length(value) == 1) {
+         i <- chromosome(callset) == value
+     }
+     else {
+         i <- chromosome(callset) %in% value
+     }
+     i[is.na(i)] <- FALSE
+     if (sum(i) > 1) {
+         if (!missing(j)) {
+             if (sum(j) < 5)
+                 warning("very few samples for calculating a robust average")
+             avg <- rowMedians(X[i, j], na.rm = TRUE)
+             X[i, j] <- sweep(X[i, j], 1, avg) +
+                 recenter
+         }
+         else {
+             avg <- rowMedians(X[i, ], na.rm = TRUE)
+             X[i, ] <- sweep(X[i, ], 1, avg) +
+                 recenter
+         }
+     }
+     X
+ }
> male <- callset$gender == 1
> female <- callset$gender == 2
> chromosome(callset)[is.na(chromosome(callset))] <- "NA"
> log2cn <- rowSweep(callset, log2cn, "NA", log2(2))
> log2cn <- rowSweep(callset, log2cn, "M", log2(2))
```

```
> log2cn <- rowSweep(callset, log2cn, "X", log2(1),
+     male)
> log2cn <- rowSweep(callset, log2cn, "X", log2(2),
+     female)
> log2cn <- rowSweep(callset, log2cn, "Y", log2(1),
+     male)
> log2cn <- rowSweep(callset, log2cn, "Y", log2(0.5),
+     female)
> log2cn <- rowSweep(callset, log2cn, as.character(1:22),
+     log2(2))
> chromosome(callset)[chromosome(callset) == "NA"] <- NA
> copyNumber <- 2^log2cn
```

The proceding codechunks may be replaced by the function `calculateCopyNumber`.

```
> cn <- calculateCopyNumber(callset, center.autosomes = 2,
+     center.X.male = 1, center.X.female = 2, center.Y.male = 1,
+     center.Y.female = 0.4)
> identical(cn, copyNumber)
```

```
[1] TRUE
```

We may now create an object of class `oligoSnpSet` that contains `assayData` elements for SNP-level summaries of genotype calls and copy number. For now, we will assign a matrix of missing values for copy number confidence estimates.

```
> cnConfidence <- matrix(NA, nrow = nrow(callset),
+     ncol = ncol(callset))
> rownames(cnConfidence) <- featureNames(callset)
> colnames(cnConfidence) <- sampleNames(callset)
> if (identical(rownames(copyNumber), featureNames(callset))) {
+     snpset <- new("oligoSnpSet", copyNumber = copyNumber,
+         cnConfidence = cnConfidence, calls = calls(callset),
+         callsConfidence = callsConfidence(callset),
+         experimentData = experimentData(callset),
+         featureData = featureData(callset), phenoData = phenoData(callset),
+         annotation = annotation(callset))
+ }
```

Estimating copy number from a `SnpCallSetPlus` object and creating an instance of the class `oligoSnpSet` are encapsulated in the method for coercing an object between the two classes:

```
> snpset <- as(callset, "oligoSnpSet")
```

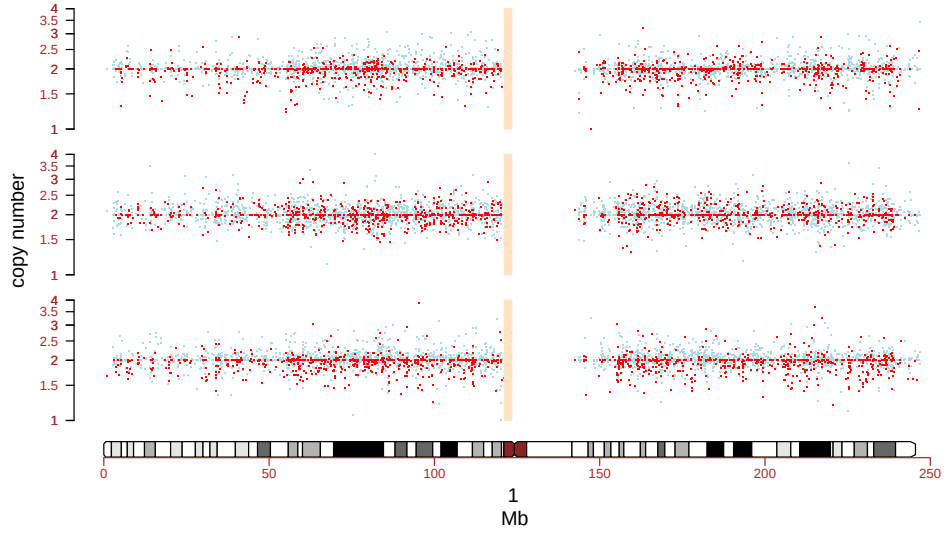
A plot of chromosome 1:

```
> gp <- new("ParSnpSet")
> gp <- getPar(gp, snpset)
```

```
[1] "one.ylim is FALSE. Calculating ylim based on the percentiles of the copy number distribution"
```

```
> gp$ylab <- "copy number"
> gp$ylim <- c(1, 4)
> plotSnp(gp, snpset)
```

```
NULL
```



A hidden Markov model can be used to identify chromosomal alterations using genotype and copy number estimates as described in the *VanillaICE* vignette.

2 Combining objects that use different annotation packages

Here we illustrate how one may combine two objects of class `SnpCallSetPlus` that use different annotation packages: e.g., *pd.mapping50k.hind240* and *pd.mapping50k.xba240*. Following the *oligo* vignette, I created `hind` and `xba` instances of `SnpCallSetPlus`. The following code is not evaluated due to time constraints.

```
> library("oligo")
> library("hapmap100kxba")
> pathCelFiles <- system.file("celFiles", package = "hapmap100kxba")
> fullFilenames <- list.celfiles(path = pathCelFiles,
+     full.names = TRUE)
> aboutSamples <- data.frame(gender = c("female",
+     "female", "male"))
> rownames(aboutSamples) <- basename(fullFilenames)
> aboutVars <- data.frame(labelDescription = "male/female")
> rownames(aboutVars) <- "gender"
> pd <- new("AnnotatedDataFrame", data = aboutSamples,
+     varMetadata = aboutVars)
> xba <- justCRLMM(fullFilenames, phenoData = pd,
+     verbose = FALSE)
> library("hapmap100khind")
> pathCelFiles <- system.file("celFiles", package = "hapmap100khind")
> fullFilenames <- list.celfiles(path = pathCelFiles,
+     full.names = TRUE)
> aboutSamples <- data.frame(gender = c("female",
+     "female", "male"))
```

```
> rownames(aboutSamples) <- basename(fullFilenames)
> aboutVars <- data.frame(labelDescription = "male/female")
> rownames(aboutVars) <- "gender"
> pd <- new("AnnotatedDataFrame", data = aboutSamples,
+     varMetadata = aboutVars)
> hind <- justCRLMM(fullFilenames, phenoData = pd,
+     verbose = FALSE)
```

To combine into one object, simply

```
> callset <- combine(xba, hind)
```

3 Session Information

- R version 2.7.0 Under development (unstable) (2008-01-28 r44219), `powerpc-apple-darwin8.11.0`
- Locale: `C`
- Base packages: base, datasets, grDevices, graphics, methods, splines, stats, tools, utils
- Other packages: AnnotationDbi 0.99.23, Biobase 1.99.0, DBI 0.2-3, RColorBrewer 1.0-1, RSQLite 0.6-4, SNPchip 1.3.17, VanillaICE 1.1.15, affxparser 1.9.5, oligo 1.3.15, oligoClasses 1.1.15, pd.mapping50k.xba240 0.3.4, preprocessCore 0.99.22