Using crlmm for copy number estimation and genotype calling with Illumina platforms

Rob Scharpf

July 24, 2010

Allele-specific copy number estimation in the crlmm package is available for several Illumina platforms. As described in the `copynumber` vignette, copy number estimation in crlmm works best when there are a sufficient number of samples such that AA, AB, and BB genotypes are observed at most loci. For small studies (e.g., fewer than 50 samples), there will be a large number of SNPs that are monomorphic. For monomorphic SNPs, the estimation problem becomes more difficult and alternative strategies that estimate the relative total copy number may be preferable. In addition to installing crlmm, one must also install the appropriate annotation package for the Illumina platform. In the following code, we list the platforms for which annotation packages are currently available. Next we create a directory where output files will be stored and indicate the directory that contains the IDAT files that will be used in our analysis.

```
> library(crlmm)
> crlmm:::validCdfNames()

 [1] "genomewidesnp6"    "genomewidesnp5"    "human370v1c"
 [4] "human370quadv3c"   "human550v3b"       "human650v3a"
 [7] "human610quadv1b"   "human660quadv1a"   "human1mduov3b"
[10] "humanomni1quadv1b"

> outdir <- "/thumper/ctsa/snpmicroarray/rs/data/hapmap/crlmmVignette/release/illumina"
> dir.create(outdir, showWarnings = FALSE, recursive = TRUE)
> datadir <- "/thumper/ctsa/snpmicroarray/illumina/IDATS/370k"
```

To perform copy number analysis on the Illumina platform, several steps are required. The first step is to read in the IDAT files and create a container for storing the red and green intensities. These intensities are quantile normalized in the function `crlmmIllumina`, and then genotyped using the crlmm algorithm. Details on the crlmm genotyping algorithm are described elsewhere. It is important to specify `save.it = TRUE` and provide output files to store the quantile normalized intensities. We will make use of the normalized intensities when we estimate copy number. The object returned by `crlmmIllumina` in an instance of the `SnpSet` class, a container for storing the genotype calls and the genotype confidence scores. The genotype confidence scores are saved as an integer, and can be converted back to a $[0, 1]$ probability scale by the transformation $round(-1000 * log2(1 - p))$. At this point, one may want to extract the scan date of the arrays for later use. The scan dates can be pulled from the RG object and added to the `SnpSet` returned by `crlmmIllumina` as illustrated below.

```
> samplesheet = read.csv(file.path(datadir, "HumanHap370Duo_Sample_Map.csv"),
     header = TRUE, as.is = TRUE)
> samplesheet <- samplesheet[-c(28:46, 61:75, 78:79), ]
> arrayNames <- file.path(datadir, unique(samplesheet[,
     "SentrixPosition"]))
> grnfiles = all(file.exists(paste(arrayNames, "_Grn.idat",
     sep = "")))
> redfiles = all(file.exists(paste(arrayNames, "_Red.idat",
     sep = "")))
```

```
> RG <- readIdatFiles(samplesheet, path = dirname(arrayNames[1]),
     arrayInfoColNames = list(barcode = NULL, position = "SentrixPosition"),
     saveDate = TRUE)
> crlmmResult <- crlmmIllumina(RG = RG, cdfName = "human370v1c",
     sns = pData(RG)$ID, returnParams = TRUE, cnFile = file.path(outdir,
         "cnFile.rda"), snpFile = file.path(outdir, "snpFile.rda"),
     save.it = TRUE)
> protocolData(crlmmResult)$ScanDate <- protocolData(RG)$ScanDate
> range(protocolData(crlmmResult)$ScanDate)
> rm(RG)
> gc()
```

Finally, we load a few of the intermediate files that were created during the preprocessing and genotyping.

```
> load(file.path(outdir, "snpFile.rda"))
> res <- get("res")
> load(file.path(outdir, "cnFile.rda"))
> cnAB <- get("cnAB")
> load(file.path(outdir, "crlmmResult.rda"))
```

After running the crlmm algorithm, we construct a container for storing the quantile normalized intensities, genotype calls, and allele-specific copy number estimates. A few helper functions for facilitating the construction of this container have been added to the inst/scripts directory of this package and can be sourced as follows. Documentation for these helper functions will be available in the devel version of this package.

```
> path <- system.file("scripts", package = "crlmm")
> source(file.path(path, "helperFunctions.R"))
> fD <- constructFeatureData(c(res$gns, cnAB$gns), cdfName = "human370v1c")
> new.order <- order(fD$chromosome, fD$position)
> fD <- fD[new.order, ]
> aD <- constructAssayData(cnAB, res, crlmmResult, order.index = new.order)
> protocolData(crlmmResult)$batch <- vector("integer",
     ncol(crlmmResult))
> container <- new("CNSetLM", assayData = aD, phenoData = phenoData(crlmmResult),
     protocolData = protocolData(crlmmResult), featureData = fD,
     annotation = "human370v1c")
```

As described in the `copynumber` vignette, two R functions for copy number estimation are available: `crlmmCopynumber` and `crlmmCopynumber2`. The latter requires that the assay data elements are represented using `ff` objects. As the dataset for this vignette is small (43 arrays) and the above steps did not make use of the `ff` features, constructing `ff` objects at this point in the analysis would serve little purpose. The decision to use ordinary matrices or `ff` objects should be decided at the beginning of the analysis and then propogated to both the genotyping and copy number estimation steps. Here, we use the `crlmmCopynumber` to estimate copy number.

```
> cnSet <- crlmmCopynumber(container, verbose = TRUE)
```

Accessors for extracting the locus-level copy number estimates. As an example of how to use accessors to obtain the allele-specific CN estimates, the following code chunk extracts the allele-specific copy number for polymorphic markers on chromosome 21.

```
> marker.index <- which(chromosome(cnSet) == 21 & isSnp(cnSet))
> ca <- CA(cnSet)[marker.index, ]/100
> cb <- CB(cnSet)[marker.index, ]/100
> missing.index <- which(rowSums(is.na(ca)) == ncol(cnSet))
```

```
> ca <- ca[-missing.index, ]
> cb <- cb[-missing.index, ]
```

Negating the `isSnp` function could be used to extract the estimates at nonpolymorphic markers. For instance,

```
> cn.monomorphic <- CA(cnSet)[which(chromosome(cnSet) ==
     21 & !isSnp(cnSet)), ]/100
```

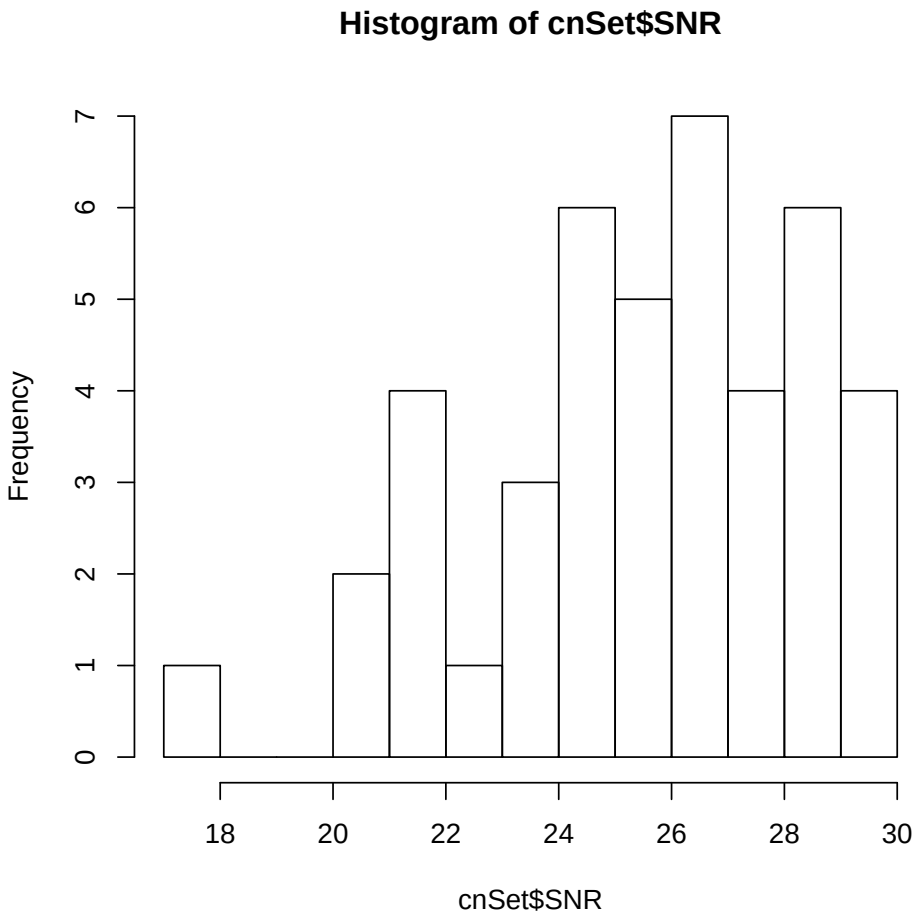
At polymorphic loci, the total copy number is the sum of the number of copies of the A allele and the number of copies for the B allele. At nonpolymorphic loci, the total copy number is the number of copies for the A allele. The helper function totalCopyNumber can be used to extract the total copy number for all polymorphic and nonpolymorphic markers. Documentation of the `totalCopyNumber` will be available in the devel version of the oligoClasses.

```
> cn.total <- ca + cb
> cn.total2 <- totalCopyNumber(cnSet, i = marker.index)
> cn.total2 <- cn.total2[-missing.index, ]
> dimnames(cn.total) <- NULL
> all.equal(cn.total2, cn.total)
```

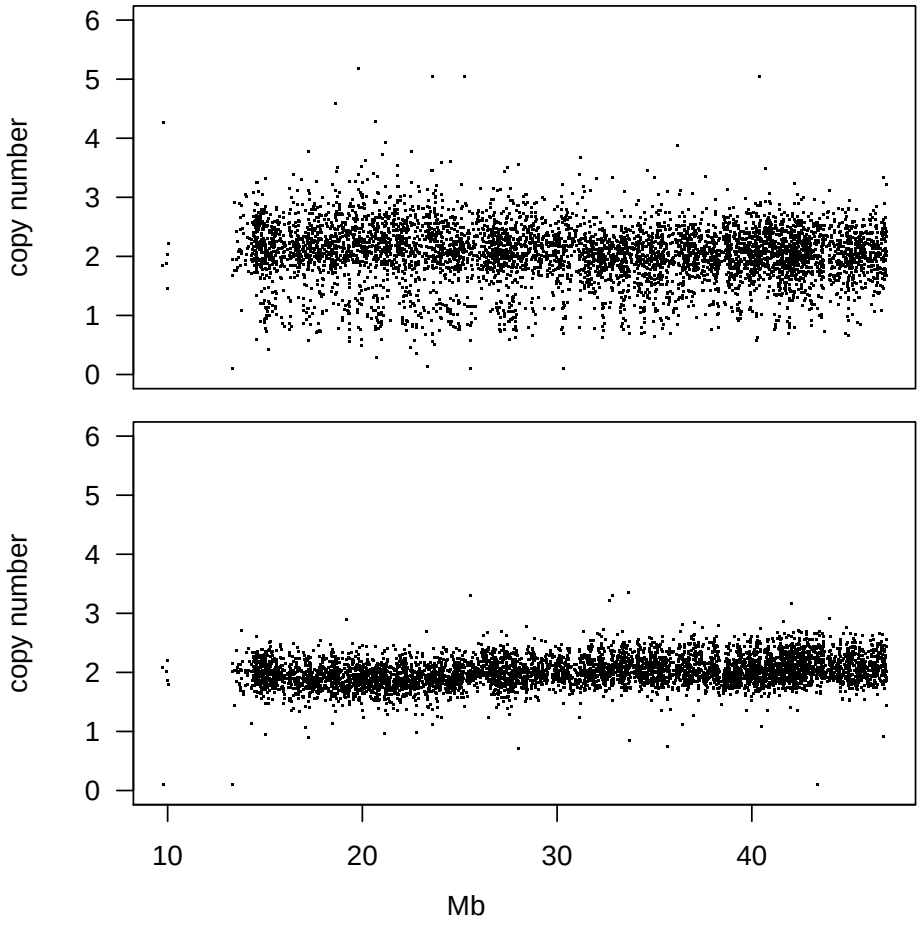
```
[1] TRUE
```

A few simple visualizations may be helpful at this point. The first plot is a histogram of the signal to noise ratio of the sample – an overall measure of how well the genotype clusters separate. (This statistic tends to be much higher for Illumina than for the Affymetrix platforms.) The second is a visualization of the total copy number estimates plotted versus physical position on chromosome 1 for the two samples with the lowest (top) and highest (bottom) signal to noise ratios.

```
> hist(cnSet$SNR, breaks = 15)
```

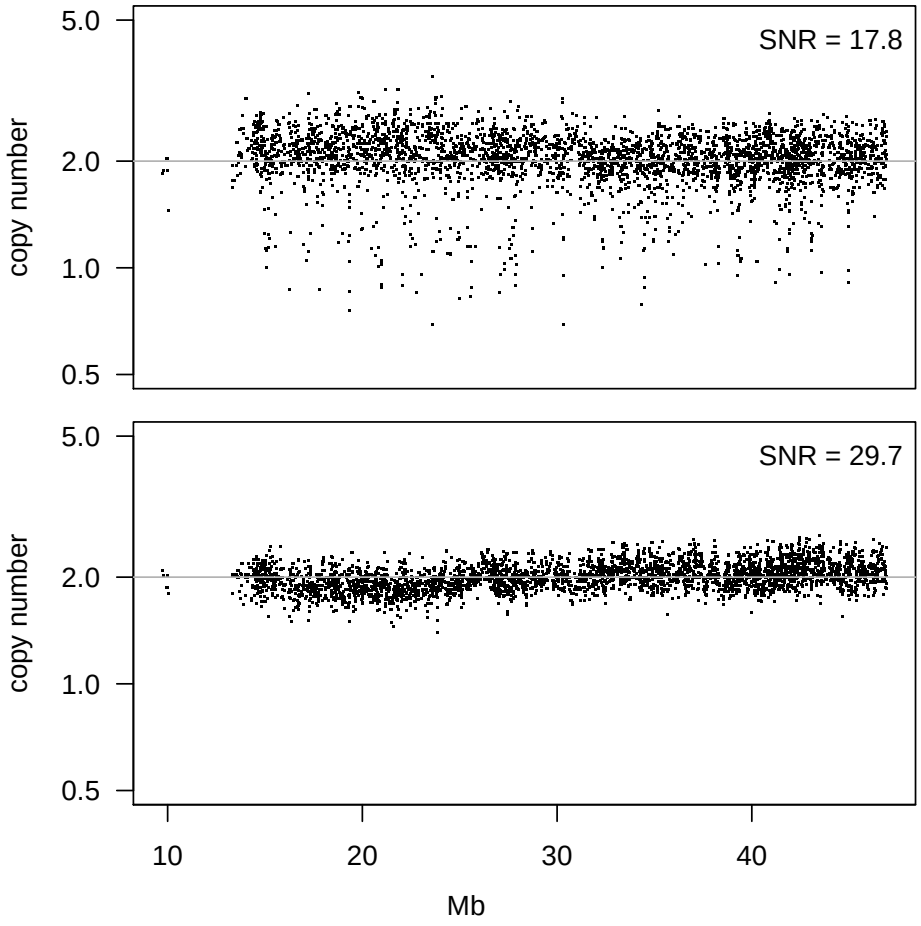



```
> low.snr <- which(cnSet$SNR == min(cnSet$SNR))
> high.snr <- which(cnSet$SNR == max(cnSet$SNR))
> x <- position(cnSet)[marker.index]
> x <- x[-missing.index]
> par(mfrow = c(2, 1), las = 1, mar = c(0.5, 4, 0.5, 0.5),
      oma = c(4, 1, 1, 1))
> for (j in c(low.snr, high.snr)) {
      cn <- cn.total[, j]
      cn[cn < 0.05] <- 0.05
      plot(x, cn, pch = ".", ylab = "copy number", xaxt = "n",
          ylim = c(0, 6))
 }
> axis(1, at = pretty(x), labels = pretty(x/1e+06))
> mtext("Mb", 1, outer = TRUE, line = 2)
```



Here's a very simple approach to handle outliers by applying a running median using a window of size 3. Following outlier removal, we suggest applying a wave correction to adjust for more global waves followed by a segmentation or hidden markov model.

```
> par(mfrow = c(2, 1), las = 1, mar = c(0.5, 4, 0.5, 0.5),
     oma = c(4, 1, 1, 1))
> for (j in c(low.snr, high.snr)) {
     cn <- cn.total[, j]
     x <- x[!is.na(cn)]
     cn <- cn[!is.na(cn)]
     y <- as.numeric(runmed(cn, k = 3))
     plot(x, y, pch = ".", ylab = "copy number", xaxt = "n",
         log = "y", ylim = c(0.5, 5))
     legend("topright", bty = "n", legend = paste("SNR =",
         round(cnSet$SNR[j], 1)))
     abline(h = 2, col = "grey70")
 }
> axis(1, at = pretty(x), labels = pretty(x/1e+06))
> mtext("Mb", 1, outer = TRUE, line = 2)
```


SNR = 17.8
SNR = 29.7
copy number
copy number
5.0
2.0
1.0
0.5
5.0
2.0
1.0
0.5
10
20
30
40
Mb