Copy number estimation and genotype calling with crlmm

Rob Scharpf

May 27, 2010

Allele-specific copy number estimation in the crlmm package is available for several Illumina platforms. As described in the `copynumber.pdf` vignette, this algorithm 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 (as to absolute allele-specific copy number) may be preferable. In the following code, we list the platforms for which annotation packages are available for computations performed by crlmm. 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 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.

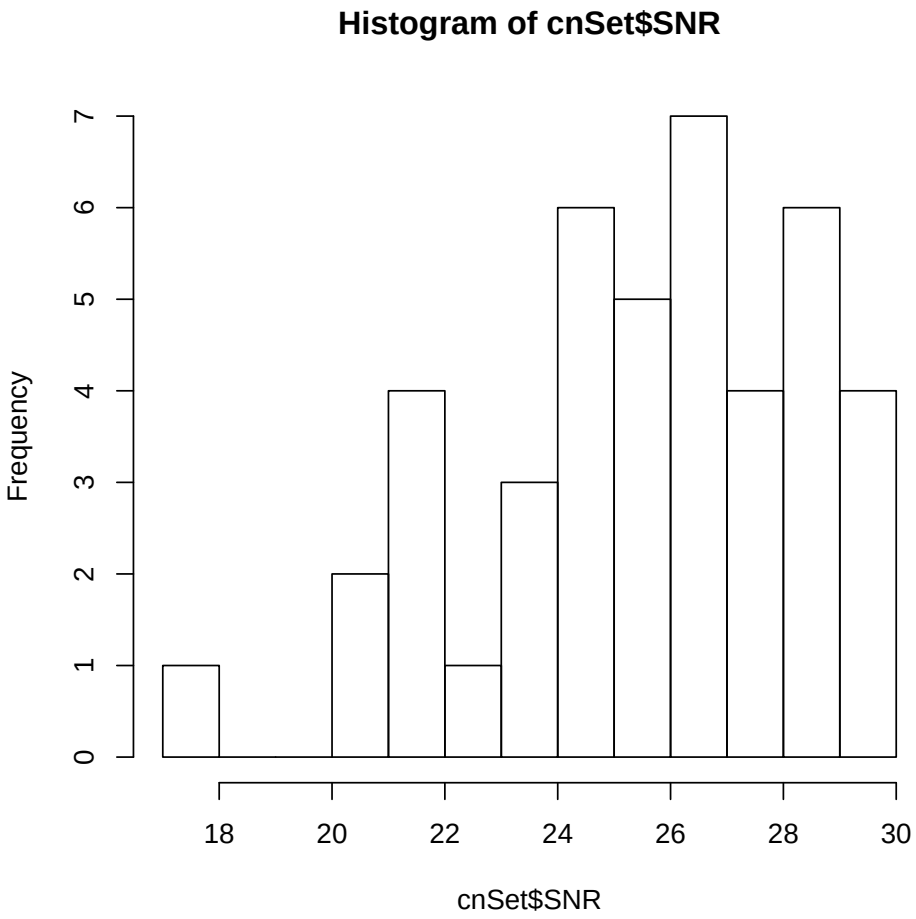
```
> samplesheet = read.csv(file.path(datadir, "HumanHap370Duo_Sample_Map.csv"),
     header = TRUE, as.is = TRUE)
> samplesheet <- samplesheet[-c(28:46, 61:75, 78:79), ]
> arrayNames <- file.path(datadir, unique(samplesheet[,
     "SentrixPosition"]))
> grnfiles = all(file.exists(paste(arrayNames, "_Grn.idat",
     sep = "")))
> redfiles = all(file.exists(paste(arrayNames, "_Red.idat",
     sep = "")))
> if (!exists("crlmmResult")) {
     if (!file.exists(file.path(outdir, "crlmmResult.rda"))) {
         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)
        save(crlmmResult, file = file.path(outdir, "crlmmResult.rda"))
        rm(RG)
        gc()
    }
    else {
        message("Loading previously saved crlmm results")
        load(file.path(outdir, "snpFile.rda"))
        load(file.path(outdir, "cnFile.rda"))
        load(file.path(outdir, "crlmmResult.rda"))
    }
}
```

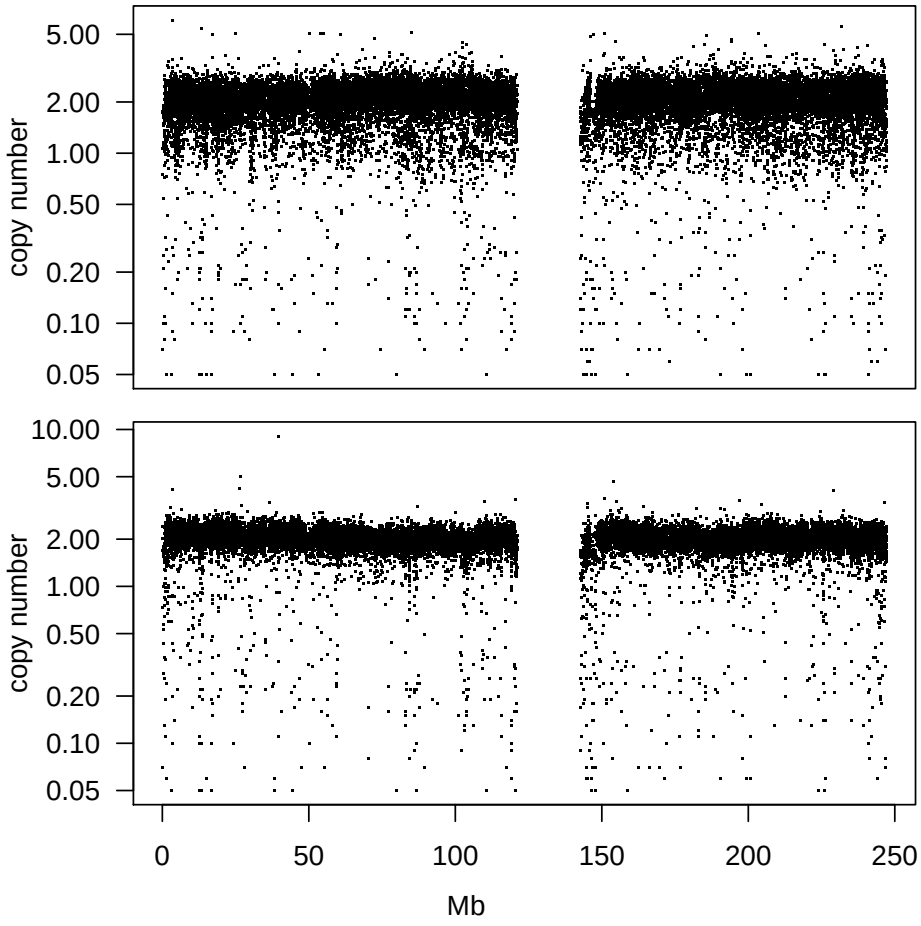
After running the crlmm algorithm, the only thing we need to do is pull together the various intermediate files and genotype calls for use by the copy number estimation algorithm. As described in the `copynumber.pdf` vignette, two R functions for copy number estimation are available: `crlmmCopynumber` 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 have little purpose. The decision to use ordinary matrices or `ff` objects should be decided at the very beginning of the analysis and then propogated to both the genotyping and copy number estimation steps. In the following code, we define helper functions to construct a `featureData` object that chromosome and physical position annotation for each marker. We then define a constructor for initializing the assay data – the copy number estimation algorithm requires normalized intensities, genotype calls, and genotype confidence scores. Note that the order of the construction is important. In particular, the validity method for the `CNSetLM` object requires a 'batch' label in the `protocolData` slot and that 'chromosome', 'position', and 'isSnp' labels are present in the `featureData` slot. With the object `cnSet` in hand, one can proceed with the copy number analysis as outlined in the `copynumber.pdf` vignette. Useful accessors and visualizations of the locus-level estimates are also discussed in the `copynumber.pdf` vignette.

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)
```

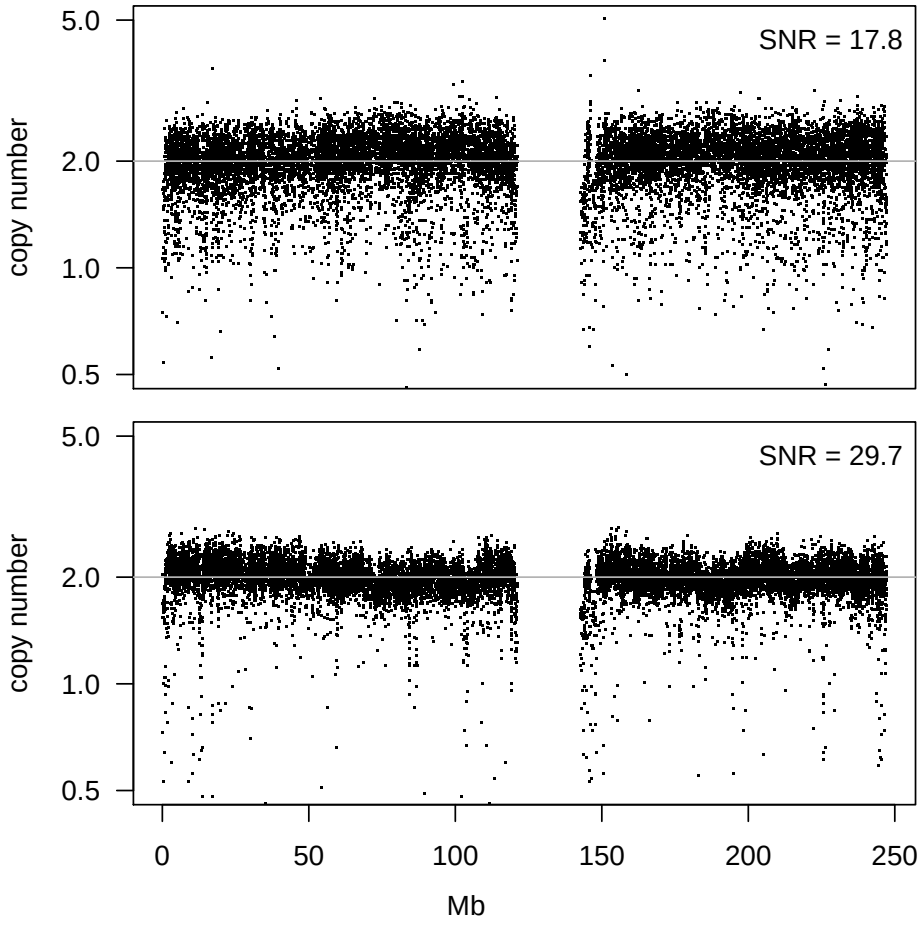



```
> low.snr <- which(cnSet$SNR == min(cnSet$SNR))
> high.snr <- which(cnSet$SNR == max(cnSet$SNR))
> x <- position(cnSet)[chromosome(cnSet) == 1]
> par(mfrow = c(2, 1), las = 1, mar = c(0.5, 4, 0.5, 0.5),
     oma = c(4, 1, 1, 1))
> for (j in c(low.snr, high.snr)) {
     cn <- copyNumber(cnSet)[, j]/100
     cn[cn < 0.05] <- 0.05
     plot(x, cn[chromosome(cnSet) == 1], pch = ".", ylab = "copy number",
         xaxt = "n", log = "y")
 }
> 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)) {
     x <- position(cnSet)[chromosome(cnSet) == 1]
     cn <- copyNumber(cnSet)[, j]/100
     cn[cn < 0.05] <- 0.05
     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
0
50
100
150
200
250
Mb