Copy number estimation with crlmm

Rob Scharpf

March 31, 2011

Abstract

Copy number routines in the crlmm package are available for Affymetrix 5.0 and 6.0 platforms, as well as several Illumina platforms. This vignette assumes that the arrays have already been successfully preprocessed and genotyped as per the instructions in the `AffymetrixPreprocessCN` and `IlluminaPreprocessCN` vignettes for the Affymetrix and Illumina platforms, respectively. While this vignette uses Affymetrix 6.0 arrays for illustration, the steps at this point are identical for both platforms. See [1] for details regarding the methodology implemented in crlmm for copy number analysis. In addition, a compendium describing copy number analysis using the crlmm package is available from the author's website: `http://www.biostat.jhsph.edu/~rscharpf/crlmmCompendium/index.html`.

1 Set up

```
> library(ff)
> library(crlmm)
> library(lattice)
> library(cacheSweave)
> if (getRversion() < "2.13.0") {
      rpath <- getRversion()
  } else rpath <- "trunk"
> outdir <- paste("/thumper/ctsa/snpmicroarray/rs/ProcessedData/crlmm/",
      rpath, "/copynumber_vignette", sep = "")

> ldPath(outdir)
> setCacheDir(outdir)
> ocProbesets(150000)
> ocSamples(200)
```

We begin by loading the `cnSet` object created by the `AffymetrixPreprocessCN` vignette.

```
> if (!exists("cnSet")) load(file.path(outdir, "cnSet.rda"))
```

Limitations: While a minimum number of samples is not required for preprocessing and genotyping, copy number estimation in the crlmm package currently requires at least 10 samples per batch. The parameter estimates for copy number and the corresponding estimates of raw copy number will tend to be more noisy for batches with small sample sizes (e.g., < 50). Chemistry plate or scan date are often useful surrogates for batch. Samples that were processed at similar times (e.g., in the same month) can be grouped together in the same batch.

2 Quality control

The signal to noise ratio (SNR) estimated by the CRLMM genotyping algorithm is an overall measure of the separation of the diallelic genotype clusters at polymorphic loci and can be a useful measure of array quality. Small SNR values can indicate possible problems with the DNA. Depending on the size of the dataset

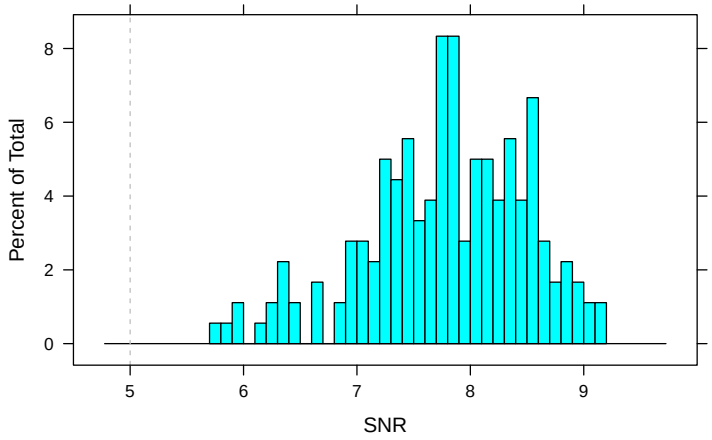

Figure 1: The signal to noise ratio (SNR) for 180 HapMap samples. For Affymetrix platforms, SNR values below 5 can indicate possible problems with sample quality. In some circumstances, it may be more helpful to exclude samples with poor DNA quality.

and the number of samples with low SNR, users may wish to rerun the preprocessing and genotyping steps after excluding samples with low SNR. The SNR is stored in the `phenoData` slot of the `CNSet` object and is available after preprocessing and genotyping. SNR values below 5 for Affymetrix or below 25 for Illumina may indicate poor sample quality. The following code chunk makes a histogram of the SNR values for the HapMap samples.

```
> invisible(open(cnSet$SNR))
> snr <- cnSet$SNR[]
> close(cnSet$SNR)
```

```
[1] TRUE
```

```
> print(histogram(~snr, panel = function(...) {
     panel.histogram(...)
     panel.abline(v = 5, col = "grey", lty = 2)
 }, breaks = 25, xlim = c(4.5, 10), xlab = "SNR"))
```

3 Copy number estimation

As described in [1], the CRLMM-CopyNumber algorithm fits a linear model to the normalized intensities stratified by the diallic genotype call. The intercept and slope from the linear model are both SNP- and batch-specific. The implementation in the `crlmm` package is encapsulated by the function `crlmmCopynumber` that, using the default settings, can be called by passing a single object of class *CNSet*. See the appropriate preprocessing/genotyping vignette for the construction of an object of class *CNSet*.

```
> (cnSet.updated <- crlmmCopynumber(cnSet))
```

The following steps were performed by the `crlmmCopynumber` function:

- sufficient statistics for the genotype clusters for each batch
- unobserved genotype centers imputed

- posterior summaries of sufficient statistics
- intercept and slope for linear model

Depending on the value of `ocProbesets()`, these summaries are computed for subsets of the markers to reduce the required RAM. Note that the value returned by the `crlmmCopynumber` function in the above example is `TRUE`. The reason the function returns `TRUE` in the above example is that the elements of the `batchStatistics` slot have the class *ff_matrix*. Rather than keep the statistical summaries in memory, the summaries are written to files on disk using protocols described in the ff package. Hence, while the `cnSet` object itself is unchanged as a result of the `crlmmCopynumber` function, the data on disk is updated accordingly. Users that are interested in accessing these low-level summaries can refer to the `Infrastructure` vignette. Computation of the raw copy number estimates for each allele is described in the following section.

For users that are interested in the analysis of a specific chromosome (subset of markers) or a s

pointers to files on disk, are stored in the `batchStatistics` slot of the class *CNSet*. Using the default settings for the `crlmmCopynumber` function, only an object of class *CNSet* is required.

Note that depends on whether the elements of the `batchStatistics` slot are `ff` objects or ordinary matrices. In this example, the elements of `batchStatistics` have the class *ff_matrix*.

```
> nms <- ls(batchStatistics(cnSet))
> cls <- rep(NA, length(nms))
> for (i in seq_along(nms)) cls[i] <- class(batchStatistics(cnSet)[[nms[i]]])[1]
> all(cls == "ff_matrix")

[1] TRUE
```

The batch-specific statistical summaries computed by `crlmmCopynumber` are written to files on disk using protocols described in the R package ff. The value returned by `crlmmCopynumber` is `TRUE`, indicating that the files on disk have been successfully updated. Note that while the `cnSet` object is unchanged, the values on disk are different.

On the other hand, subsetting the `cnSet` with the '[' method coerces all of the elements to class *matrix*. The batch-specific summaries are now ordinary matrices stored in RAM. The object returned by `crlmmCopynumber` is an object of class *CNSet* with the matrices in the `batchStatistics` slot updated.

```
> chr1.index <- which(chromosome(cnSet) == 1)
> open(cnSet)

[1] TRUE

> cnSet2 <- cnSet[chr1.index, ]

> close(cnSet)

NULL

> for (i in seq_along(nms)) cls[i] <- class(batchStatistics(cnSet2)[[nms[i]]])[1]
> all(cls == "matrix")

[1] TRUE

> cnSet3 <- crlmmCopynumber(cnSet2)
> class(cnSet3)
```

3.1 Raw copy number

Several functions are available that will compute relatively quickly the allele-specific, *raw* copy number estimates. At allele k, marker i, sample j, and batch p, the estimate of allele-specific copy number is

computed by subtracting the estimated background from the normalized intensity and scaling by the slope coefficient. More formally,

$$\hat{c}_{k,ijp} \quad = \quad \max\left\{\frac{1}{\hat{\phi}_{k,ip}}\left(I_{k,ijp} - \hat{\nu}_{k,ip}\right),\ 0\right\} \text{ for } k \in \{A, B\}. \tag{1}$$

See [**?**] for details.

The function `totalCopynumber` translates the normalized intensities to an estimate of raw copy number by adding the allele-specific summaries in Equation (1). For large datasets, the calculation will not be instantaneous as the I/O can be substantial. Users should specify either a subset of the markers or a subset of the samples to avoid using all of the available RAM. For example, in the following code chunk we compute the total copy number at all markers for the first 2 samples, and the total copy number for chromosome 20 for the first 50 samples.

```
> tmp <- totalCopynumber(cnSet, i = 1:nrow(cnSet), j = 1:2)
> dim(tmp)

[1] 1852215       2

> tmp2 <- totalCopynumber(cnSet, i = which(chromosome(cnSet) ==
     20), j = 1:50)
> dim(tmp2)

[1] 43002    50
```

Alternatively, the functions `CA` and `CB` compute the allele-specific copy number. For instance, the following code chunk computes the allele-specific summaries at all polymorphic loci.

```
> snp.index <- which(isSnp(cnSet) & !is.na(chromosome(cnSet)))
> ca <- CA(cnSet, i = snp.index, j = 1:5)
> cb <- CB(cnSet, i = snp.index, j = 1:5)
```

Note the equivalence of the following calculations.

```
> ct <- ca + cb
> ct2 <- totalCopynumber(cnSet, i = snp.index, j = 1:5)
> stopifnot(all.equal(ct, ct2))
```

At nonpolymorphic loci, `CA` function returns the total copy number and, by construction, the `CB` function returns 0.

```
> marker.index <- which(!isSnp(cnSet))
> ct <- CA(cnSet, i = marker.index, j = 1:5)
> stopifnot(all(CB(cnSet, i = marker.index, j = 1:5) ==
     0))
> ct2 <- totalCopynumber(cnSet, i = marker.index, j = 1:5)
> stopifnot(all.equal(ct, ct2))
```

In the following code chunk, we extract estimates of the total copy number at nonpolymorphic markers on chromosome X.

```
> set.seed(123)
> npx.index <- which(chromosome(cnSet) == 23 & !isSnp(cnSet))
> M <- sample(which(cnSet$gender[] == 1), 5)
> F <- sample(which(cnSet$gender[] == 2), 5)
> cn.M <- CA(cnSet, i = npx.index, j = M)
> cn.F <- CA(cnSet, i = npx.index, j = F)
```

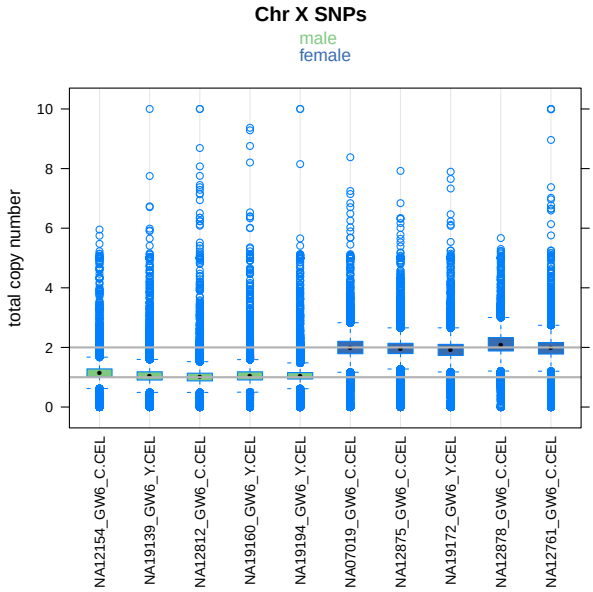

Figure 2: Copy number estimates for polymorphic markers on chromosome X. crlmm assumes that the median copy number across samples at a given marker on X is 1 for men and 2 for women.

Again, the function `totalCopynumber` is equivalent.

```
> cnX <- cbind(cn.M, cn.F)
> cnX2 <- totalCopynumber(cnSet, i = npx.index, j = c(M,
      F))
> stopifnot(all.equal(cnX, cnX2))
```

Polymorphic markers on chromosome X:

```
> library(RColorBrewer)
> cols <- brewer.pal(8, "Accent")[c(1, 5)]
> X.markers <- which(isSnp(cnSet) & chromosome(cnSet) ==
      23)
> cnX <- totalCopynumber(cnSet, i = X.markers, j = c(M,
      F))
> df <- data.frame(cn = as.numeric(cnX), id = factor(rep(sampleNames(cnSet)[c(M,
      F)], each = length(X.markers)), levels = sampleNames(cnSet)[c(M,
      F)], ordered = T))
> mykey <- simpleKey(c("male", "female"), points = FALSE,
      col = cols)
> print(bwplot(cn ~ id, df, panel = function(x, y, ...) {
      panel.grid(v = -10, h = 0)
      panel.bwplot(x, y, ...)
      panel.abline(h = 1:2, col = "grey70", lwd = 2)
  }, scales = list(x = list(rot = 90)), cex = 0.5, ylab = "total copy number",
      main = "Chr X SNPs", fill = cols[cnSet$gender[c(M,
          F)]], key = mykey))
```

3.2 A container for raw copy number

A useful container for storing the crlmm genotypes, genotype confidence scores, and the total copy number at each marker is the *oligoSnpSet* class. Coercion of a *CNSet* object to a `oligoSnpSet` object can be acheived by using the method `as` (as illustrated below). Users should note that if the `assayData` elements in the *CNSet* instance are ff objects, the `assayData` elements of the instantiated *oligoSnpSet* will also be ff-dervied objects (a new `total_cn*.ff` file will be created in the `ldPath()` directory).

```
> open(cnSet3)
```

```
NULL
```

```
> oligoSet <- as(cnSet3, "oligoSnpSet")
> close(cnSet3)
```

```
NULL
```

```
> class(copyNumber(oligoSet))
```

```
[1] "matrix"
```

Note that the raw copy number estimates stored in the `oligoSnpSet` object can be retrieved by the `copyNumber` accessor and is equivalent to that returned by the `totalCopynumber` function defined over the same row and column indices.

```
> total.cn3 <- totalCopynumber(cnSet3, i = 1:nrow(cnSet3),
     j = 1:ncol(cnSet3))
> all.equal(copyNumber(oligoSet), total.cn3)
```

```
[1] TRUE
```

4 Session information

```
> toLatex(sessionInfo())
```

- R version 2.14.0 Under development (unstable) (2011-03-31 r55220), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.iso885915`, `LC_NUMERIC=C`, `LC_TIME=en_US.iso885915`, `LC_COLLATE=en_US.iso885915`, `LC_MONETARY=C`, `LC_MESSAGES=en_US.iso885915`, `LC_PAPER=en_US.iso885915`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.iso885915`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, methods, stats, tools, utils
- Other packages: Biobase 2.11.9, bit 1.1-6, cacheSweave 0.4-5, crlmm 1.9.25, ff 2.2-1, filehash 2.1-1, lattice 0.19-17, oligoClasses 1.13.22, RColorBrewer 1.0-2, stashR 0.3-3
- Loaded via a namespace (and not attached): affyio 1.19.2, annotate 1.29.3, AnnotationDbi 1.13.18, Biostrings 2.19.12, DBI 0.2-5, digest 0.4.2, ellipse 0.3-5, genefilter 1.33.1, grid 2.14.0, IRanges 1.9.27, mvtnorm 0.9-96, preprocessCore 1.13.6, RSQLite 0.9-4, splines 2.14.0, survival 2.36-5, xtable 1.5-6

References

[1] Robert B Scharpf, Ingo Ruczinski, Benilton Carvalho, Betty Doan, Aravinda Chakravarti, and Rafael A Irizarry. A multilevel model to address batch effects in copy number estimation using snp arrays. *Biostatistics*, 12(1):33–50, Jan 2011.