Preprocessing and Genotyping Illumina Arrays for Copy Number Analysis

Rob Scharpf

January 17, 2012

Abstract

This vignette illustrates the steps required prior to copy number analysis for Infinium platforms. Specifically, we require construction of a container to store processed forms of the raw data, preprocessing to normalize the arrays, and genotyping using the CRLMM algorithm. After completing these steps, users can refer to the `copynumber` vignette.

1 Set up

The following codechunk declares a directory for saving `ff` files that will contain the normalized intensities and the genotype calls.

```
> library(ff)
> if(getRversion() < "2.13.0"){
         rpath <- getRversion()
 } else rpath <- "trunk"
> outdir <- paste("/thumper/ctsa/snpmicroarray/rs/ProcessedData/crlmm/", rpath, "/illumina_vignette", se
> ldPath(outdir)
> dir.create(outdir, recursive=TRUE, showWarnings=FALSE)
```

We will also store cached computations in the directory `outdir`.

We declare that crlmm should process 150,000 markers at a time and/or 500 samples at a time (when possible) to reduce the memory footprint. As our example dataset in this vignette contains fewer than 500 samples, all samples will be processed simultaneously.

```
> ocProbesets(150e3)
> ocSamples(500)
```

We begin by specifying the path containing the raw IDAT files for a set of samples from the Infinium 370k platform.

```
> datadir <- "/thumper/ctsa/snpmicroarray/illumina/IDATS/370k"
```

For Infinium platforms, an Illumina sample sheet containing information for reading the raw IDAT files is required. Please refer to the BeadStudio Genotyping guide, Appendix A, for additional information. The following code reads in the samplesheet for the IDAT files on our local server.

```
> samplesheet = read.csv(file.path(datadir, "HumanHap370Duo_Sample_Map.csv"), header=TRUE, as.is=TRUE)
```

For the purposes of this vignette, we indicate that we only wish to process a subset of the arrays. For our dataset, the file extensions are 'Grn.dat' and 'Red.idat'. We store the complete path to the filename without the file extension in the `arrayNames` and check that all of the green and red IDAT files exists.

```
> samplesheet <- samplesheet[-c(28:46,61:75,78:79), ]
> arrayNames <- file.path(datadir, unique(samplesheet[, "SentrixPosition"]))
> all(file.exists(paste(arrayNames, "_Grn.idat", sep="")))
```

```
[1] TRUE
```

```
> all(file.exists(paste(arrayNames, "_Red.idat", sep="")))
```

```
[1] TRUE
```

```
> arrayInfo <- list(barcode=NULL, position="SentrixPosition")
```

All supported platforms have a corresponding annotation package. The appropriate annotation package is specified by the platform identifier without the `Crlmm` postfix.

```
> cdfName <- "human370v1c"
```

Next, we construct a character vector that specifies the batch for each of the 43 arrays. Here, we have a small dataset and process the samples in a single batch. Processing the samples as a single batch is generally reasonable if the samples were processed at similar times (e.g., within a few weeks).

```
> batch <- rep("1", nrow(samplesheet))
```

2 Preprocessing and genotyping

The raw intensities from the Infinium IDAT files are read and normalized using the function `preprocessInf`. The function `preprocessInf` returns a `ff` object containing the parameters for the mixture model used by the CRLMM genotyping algorithm. Following preprocessing, the `genotypeInf` genotypes the samples. The function `genotype.Illumina` is a wrapper to the above functions and returns an object of class *CNSet*.

```
> cnSet <- genotype.Illumina(sampleSheet=samplesheet,
                             arrayNames=arrayNames,
                             arrayInfoColNames=arrayInfo,
                             cdfName="human370v1c",
                             batch=batch)
```

Note, to fully remove the data associated with the `cnSet2` object, one should use the `delete` function in the ff package followed by the `rm` function. The following code is not evaluated is it would change the results of the cached computations in the previous code chunk.

```
> lapply(assayData(cnSet), delete)
> lapply(batchStatistics(cnSet), delete)
> delete(cnSet$gender)
> delete(cnSet$SNR)
> delete(cnSet$SKW)
> rm(cnSet)
```

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/crlmm

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.

3 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 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.

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

```
[1] TRUE
```

```
> print(histogram(~snr,
                  panel=function(...){
                          panel.histogram(...)},
                  breaks=25, xlim=range(snr), xlab="SNR"))
```

4 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. Note that the data structure 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)
> for(i in seq_along(nms)) cls[i] <- class(batchStatistics(cnSet2)[[nms[i]]])[1]
> all(cls == "matrix")

[1] TRUE

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

4.1 Marker-specific estimates

Raw total 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=seq_len(nrow(cnSet)), j=1:2)
> dim(tmp)

[1] 370024      2

> tmp2 <- totalCopynumber(cnSet, i=which(chromosome(cnSet) == 20), j=seq_len(ncol(cnSet)))
> dim(tmp2)

[1] 8659   43
```

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=seq_len(ncol(cnSet)))
> cb <- CB(cnSet, i=snp.index, j=seq_len(ncol(cnSet)))
```

4.2 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)

[1] TRUE

> 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=seq_len(ncol(cnSet3)))
> all.equal(copyNumber(oligoSet), total.cn3)

[1] "Mean relative difference: 1.03623"
```

5 Session information

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

- R Under development (unstable) (2012-01-17 r58125), `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=en_US.iso885915`, `LC_MESSAGES=en_US.iso885915`, `LC_PAPER=C`, `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 2.15.3, BiocGenerics 0.1.4, BiocInstaller 1.3.5, bit 1.1-7, cacheSweave 0.6, crlmm 1.13.7, ff 2.2-4, filehash 2.2, lattice 0.20-0, oligoClasses 1.17.15, stashR 0.3-4
- Loaded via a namespace (and not attached): affyio 1.23.1, annotate 1.33.1, AnnotationDbi 1.17.14, Biostrings 2.23.5, DBI 0.2-5, digest 0.5.1, ellipse 0.3-5, genefilter 1.37.0, grid 2.15.0, IRanges 1.13.19, mvtnorm 0.9-9991, preprocessCore 1.17.1, RSQLite 0.11.1, splines 2.15.0, survival 2.36-10, xtable 1.6-0, zlibbioc 1.1.0

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.

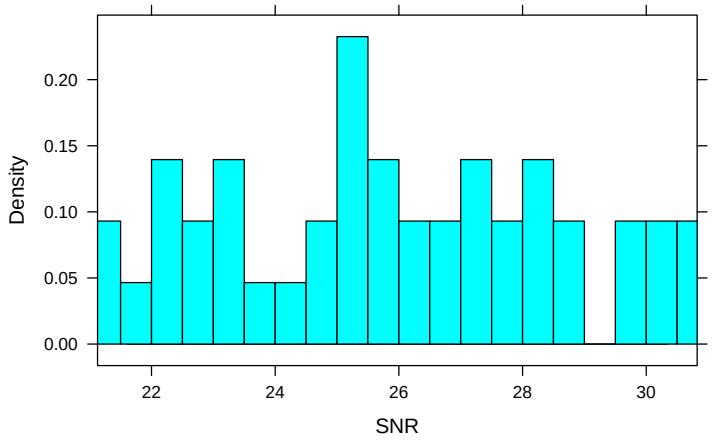

Figure 1: The signal to noise ratio (SNR) for 180 HapMap samples. For Affymetrix platforms, SNR values below 5 can indicate possible problems with sample quality. In some circumstances, it may be more helpful to exclude samples with poor DNA quality.