Preprocessing & Genotyping Affymetrix Arrays for Copy Number Analysis

Rob Scharpf

May 30, 2012

Abstract

This vignette describes the setup needed to analyze Affymetrix 6.0 (or 5.0) CEL files and the steps for preprocessing and genotyping. These steps must be completed prior to copy number analyses in crlmm. After completing these steps, users can refer to the `copynumber` vignette.

1 Set up

```
> library(oligoClasses)
> library2(crlmm)
> library2(ff)
> library2(cacheSweave)
```

This vignette analyzes HapMap samples assayed on the Affymetrix 6.0 platform. The annotation package for this platform is genomewidesnp6Crlmm. We assign the name of the annotation package without the `Crlmm` postfix to the name `cdfName`. We use the R package cacheSweave to cache long computations in this vignette. Users should refer to the cacheSweave package for additional details regarding cacheing.

```
> cdfName <- "genomewidesnp6"
```

The HapMap CEL files are stored in a local directory assigned to `pathToCels` in the following code. The genotyping step will create several files with `ff` extensions. The ff objects contain the low-level, normalized intensities as well as parameters used to subsequently estimate copy number and B allele frequencies. These files should not be deleted or moved. We will store these files to the path indicated by `outdir`.

```
> pathToCels <- "/thumper/ctsa/snpmicroarray/hapmap/raw/affy/1m"
> outdir <- paste("/local_data/r00/crlmm/", getRversion(), "/affy_vignette", sep="")
> dir.create(outdir, recursive=TRUE, showWarnings=FALSE)
```

By providing the path in `outdir` as an argument to the R function `ldPath`, all of the `ff` files created during the genotyping step will be stored in `outdir`.

```
> ldPath(outdir)
```

The R functions `ocProbesets` and `ocSamples` manage the RAM required for our analysis. See the documentation for these functions and the `CopyNumberOverview` vignette for additional details.

```
> ocProbesets(100000)
> ocSamples(200)
```

Next we indicate the local directory that contains the CEL files. For the purposes of this vignette, we only analyze the CEPH ('C') and Yoruban ('Y') samples.

```
> celFiles <- list.celfiles(pathToCels, full.names=TRUE, pattern=".CEL")
> celFiles <- celFiles[substr(basename(celFiles), 13, 13) %in% c("C", "Y")]
> ##celFiles <- celFiles[substr(basename(celFiles), 13, 13) %in% "C"]
```

Finally, copy number analyses using crlmm require specification of a batch variable that is used to indicate which samples were processed together. For example, if some of the samples were processed in April and another set of samples were processed in June, we could name the batches 'April' and 'June', respectively. A useful surrogate for batch is often the chemistry plate or the scan date of the array. For the HapMap CEL files analyzed in this vignette, the CEPH (C) and Yoruban (Y) samples were prepared on separate chemistry plates. In the following code chunk, we extract the population identifier from the CEL file names and assign these identifiers to the variable `plate`.

```
> plates <- substr(basename(celFiles), 13, 13)
```

2 Preprocessing and genotyping.

The preprocessing steps for copy number estimation includes quantile normalization of the raw intensities for each probe and a step that summarizes the intensities of multiple probes at a single locus. For example, the Affymetrix 6.0 platform has 3 or 4 identical probes at each polymorphic locus and the normalized intensities are summarized by a median. For the nonpolymorphic markers on Affymetrix 6.0, only one probe per locus is available and the summarization step is not needed. After preprocessing the arrays, the crlmm package estimates the genotype using the CRLMM algorithm and provides a confidence score for the genotype calls. The function `genotype` performs both the preprocessing and genotyping.

```
> cnSet <- genotype(celFiles, batch=plates, cdfName=cdfName, genome="hg19")
```

Segment faults that occur with the above step can often be traced to a corrupt cel file. To check if any of the files are corrupt, try reading the files in one at a time:

```
> validCEL(celFiles)
```

The value returned by genotype is an instance of the class `CNSet`. The normalized intensities, genotype calls, and confidence scores are stored as `ff` objects in the `assayData` slot. A concise summary of this object can be obtained throught the `print` or `show` methods.

```
> print(cnSet)
```

```
CNSet (assayData/batchStatistics elements: ff_matrix)
CNSet (storageMode: lockedEnvironment)
assayData: 1852426 features, 180 samples
  element names: alleleA, alleleB, call, callProbability
protocolData
  rowNames: NA06985_GW6_C.CEL NA06991_GW6_C.CEL ...
    NA19240_GW6_Y.CEL (180 total)
  varLabels: ScanDate
  varMetadata: labelDescription
phenoData
  sampleNames: NA06985_GW6_C.CEL NA06991_GW6_C.CEL ...
    NA19240_GW6_Y.CEL (180 total)
  varLabels: SKW SNR gender
  varMetadata: labelDescription
featureData
  featureNames: SNP_A-2131660 SNP_A-1967418 ... CN_929945
```

```
    (1852426 total)
  fvarLabels: isSnp position chromosome
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation: genomewidesnp6
genome:  hg19
batch:    C:90, Y:90, grandMean:90
batchStatistics:  29  elements,  1852426  features,  2  batches
```

Note that the object is relatively small as the intensities and genotype calls are stored on disk rather than in active memory.

```
> print(object.size(cnSet), units="Mb")
```

```
134.3 Mb
```

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 (**?**) 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`.

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 ?, 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 (**??**). 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] 1852426       2

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

[1] 43000   180
```

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 for the first 2 samples.

```
> snp.index <- which(isSnp(cnSet) & !is.na(chromosome(cnSet)))
> ca <- CA(cnSet, i=snp.index, j=1:2)
> cb <- CB(cnSet, i=snp.index, j=1:2)
```

4.2 Container for log R ratios and B allele frequencies

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

```
> library(VanillaICE)
> open(cnSet3)

[1] TRUE

> oligoSetList <- constructOligoSetListFrom(cnSet3, batch.name="C")
> close(cnSet3)

NULL

> show(oligoSetList)

oligoSetList of length 1
```

```
> class(oligoSetList)

[1] "oligoSetList"
attr(,"package")
[1] "oligoClasses"

> ## oligoSnpSet of first chromosome
> oligoSetList[[1]]

oligoSnpSet (storageMode: lockedEnvironment)
assayData: 144293 features, 90 samples
  element names: baf, call, callProbability, copyNumber
protocolData: none
phenoData
  sampleNames: NA06985_GW6_C.CEL NA06991_GW6_C.CEL ...
    NA12892_GW6_C.CEL (90 total)
  varLabels: SKW SNR gender
  varMetadata: labelDescription
featureData
  featureNames: CN_473963 CN_473964 ... CN_479920 (144293
    total)
  fvarLabels: isSnp position chromosome
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation: genomewidesnp6
genome:  hg19
```

Note that log R ratios stored in the `oligoSnpSet` object can be retrieved by the `copyNumber` accessor. B allele frequences are retrieved by the `baf` accessor.

```
> lrrList <- copyNumber(oligoSetList)
> class(lrrList)

[1] "list"

> dim(lrrList[[1]]) ## log R ratios for chromosome 1.

[1] 144293     90

> bafList <- baf(oligoSetList)
> dim(bafList[[1]]) ## B allele frequencies for chromosome 1

[1] 144293     90
```

5 Session information

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

- R version 2.15.0 Patched (2012-04-25 r59178), `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: bit 1.1-8, cacheSweave 0.6-1, crlmm 1.15.4, ff 2.2-7, filehash 2.2-1, genomewidesnp6Crlmm 1.0.6, lattice 0.20-6, oligoClasses 1.19.15, stashR 0.3-5, VanillaICE 1.19.11
- Loaded via a namespace (and not attached): affyio 1.24.0, annotate 1.34.0, AnnotationDbi 1.18.0, Biobase 2.16.0, BiocGenerics 0.3.0, BiocInstaller 1.5.7, Biostrings 2.24.1, codetools 0.2-8, compiler 2.15.0, DBI 0.2-5, digest 0.5.2, ellipse 0.3-7, foreach 1.4.0, genefilter 1.38.0, GenomicRanges 1.9.13, grid 2.15.0, IRanges 1.15.9, iterators 1.0.6, msm 1.1.1, mvtnorm 0.9-9992, preprocessCore 1.18.0, RSQLite 0.11.1, splines 2.15.0, stats4 2.15.0, survival 2.36-14, xtable 1.7-0, zlibbioc 1.2.0

plain

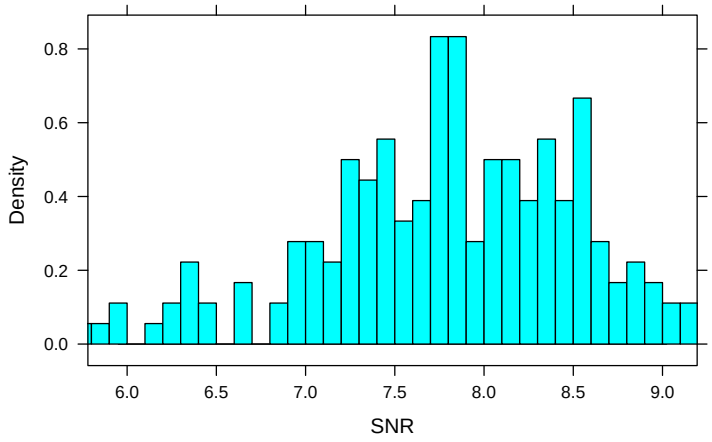

Figure 1: The signal to noise ratio (SNR) for 180 HapMap samples. For Affymetrix platforms, SNR values below 5 can indicate possible problems with sample quality. In some circumstances, it may be more helpful to exclude samples with poor DNA quality.