Copy number estimation and genotype calling with crlmm

Rob Scharpf

August 30, 2010

Abstract

This vignette estimates copy number for HapMap samples on the Affymetrix 6.0 platform. See [1] for the working paper.

1 Simple usage

CRLMM supports the following platforms:

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

 [1] "genomewidesnp6"        "genomewidesnp5"
 [3] "human370v1c"           "human370quadv3c"
 [5] "human550v3b"           "human650v3a"
 [7] "human610quadv1b"       "human660quadv1a"
 [9] "human1mduov3b"         "humanomni1quadv1b"
[11] "humanomniexpress12v1b"

> cdfName <- "genomewidesnp6"
```

Preprocess and genotype. In the following code chunk, we provide the complete path to the Affymetrix CEL files, assign a path to store the normalized intensities and genotype data, and define a 'batch' variable. The batch variable will later be useful when we estimate copy number. We typically use the 96 well chemistry plate or the scan date of the array as a surrogate for batch. Adjusting by date or chemistry plate can be helpful for limiting the influence of batch effects.

```
> if (getRversion() < "2.12.0") {
     rpath <- getRversion()
 } else rpath <- "trunk"
> outdir <- paste("/thumper/ctsa/snpmicroarray/rs/ProcessedData/crlmm/",
     rpath, "/affy_vignette", sep = "")
> datadir <- "/thumper/ctsa/snpmicroarray/hapmap/raw/affy/1m"
> celFiles <- list.celfiles(datadir, full.names = TRUE,
     pattern = ".CEL")
> dir.create(outdir, showWarnings = FALSE)
> batch <- substr(basename(celFiles), 13, 13)
> celFiles <- celFiles[batch %in% c("C", "Y")]
> batch <- batch[batch %in% c("C", "Y")]
> stopifnot(length(batch) == length(celFiles))
```

The preprocessing steps for copy number estimation includes quantile normalization of the polymorphic and nonpolymorphic markers and a summarization step whereby the quantile normalized intenities are summarized for each locus. As the markers at polymorphic loci on the Affymetrix 6.0 platform are identical,

we summarize the intensities by the median. For the nonpolymorphic markers, no summarization step is performs. Next, the crlmm package provides a genotype call and a confidence score at each polymorphic locus. Unless the dataset is small (e.g., fewer than 50 samples), we suggest installing and loading the R package ff. The function genotype checks to see whether the ff is loaded. If loaded, the normalized intensities and genotype are stored as ff objects on disk. Otherwise, the genotypes and normalized intensities are stored in matrices. A word of caution: the genotype function requires a potentially large amount of RAM even when the R package ff is loaded. Users with large datasets may want to skip this part. For the 180 HapMap samples processed in this vignette, the genotype function required 25MB RAM. The next two code chunks should be submitted as batch jobs.

```
> if (!exists("cnSet")) {
     if (!file.exists(file.path(outdir, "cnSet.rda"))) {
         cnSet <- genotype(celFiles[1:50], cdfName = cdfName,
             copynumber = TRUE, batch = batch[1:50])
         save(cnSet, file = file.path(outdir, "cnSet.rda"))
     }
     else {
         message("Loading cnSet container")
         load(file.path(outdir, "cnSet.rda"))
     }
 }
```

A more memory efficient approach to preprocessing and genotyping is implemented in the R function genotype2. The arguments to this function are similar to genotype but requires explicit loading of the R package ff prior to calling the function. The functions ocProbesets and ocSamples can be used to manage how many probesets and samples are to be loaded into memory and processed. Using the default settings, the following code required 1.9Mb RAM to process 180 CEL files.

```
> library(ff)
> ld.path <- file.path(outdir, "ffobjs")
> if (!file.exists(ld.path)) dir.create(ld.path)
> ldPath(ld.path)
> ocProbesets(1e+05)
> ocSamples(300)
> if (!exists("cnSet2")) {
     if (!file.exists(file.path(outdir, "cnSet2.rda"))) {
         cnSet2 <- genotype2(celFiles, cdfName = cdfName,
             copynumber = TRUE, batch = batch)
         save(cnSet2, file = file.path(outdir, "cnSet2.rda"))
     }
     else {
         message("Loading cnSet2 container")
         load(file.path(outdir, "cnSet2.rda"))
     }
 }
```

The objects returned by the genotype and genotype2 differ in size as the former returns an object with matrices in the assay data slot, whereas the latter return an object with pointers to files on disk. The functions open and close can be used to open and close the connections stored in the assay data of the cnSet2 object, respectively. Subsetting the ff object pulls the data from disk into memory. As subsetting operations pull the data from disk to memory, these operations must be performed with care. In particular, subsetting the cnSet2 will subset each assay data element and this can be slow. The preferred approach would be to extract the assay data element that is needed, and then subset this function object as illustrated below for the genotype calls.

```
> class(calls(cnSet))
```

```
[1] "matrix"

> dims(cnSet)

         alleleA alleleB      CA    call callProbability      CB
Features 1852406 1852406 1852406 1852406         1852406 1852406
Samples       50      50      50      50              50      50

> class(calls(cnSet2))

[1] "ff_matrix" "ff_array"  "ff"

> dims(cnSet2)

         alleleA alleleB      CA    call callProbability      CB
Features 1852406 1852406 1852406 1852406         1852406 1852406
Samples      180     180     180     180             180     180

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

134.3 Mb

> print(object.size(cnSet2), units = "Mb")

134.3 Mb

> snp.index <- which(isSnp(cnSet))
> if (FALSE) {
     replicate(5, system.time(gt1 <- calls(cnSet)[, 1:50])[[1]])
     open(calls(cnSet2))
     replicate(5, system.time(gt2 <- calls(cnSet2)[, 1:50])[[1]])
 }
> gt1 <- calls(cnSet)[snp.index, 1:50]
> gt2 <- calls(cnSet2)[snp.index, 1:50]
> all.equal(gt1, gt2)

[1] "Mean relative difference: 0.5603154"
```

Note that the genotype calls and call probabilities may differ slightly as the two approaches used a different set of samples. With `ff` objects the additional I/O time required for extracting data is substantial and will increase for larger datasets. Patience is required.

Note that for the Affymetrix 6.0 platform the assay data elements each have a row dimension corresponding to the total number of polymorphic and nonpolymorphic markers interrogated by the Affymetrix 6.0 platform. A consequence of keeping the rows of the assay data elements the same for all of the statistical summaries is that the matrix used to store genotype calls is larger than necessary. Also, note the additional overhead of some operations when using `ff` objects. For instance, the posterior probabilities for the CRLMM genotype calls are represented as integers. The accessor `snpCallProbability` can be used to access these confidence scores. When stored as matrices, converting the integer representation back to the probability scale is straightforward as shown below. However, for the `ff` objects we must first bring the data into memory. To bring all the scores into memory, one could use the function `[,]` but this could be slow and require a lot of RAM depending on the size of the dataset. Instead, we suggest pulling only the needed rows and columns from memory. In the following example, we convert the integer scores to probabilities for the CEPH samples. As genotype confidence scores are not applicable to the nonpolymorphic markers, we extract only the polymorphic markers using the `isSnp` function. We observe a slight difference in the posterior probabilities. The difference arises because `cnSet2` was processed with a larger number of samples – for some SNPs a larger sample size can improve the genotype calling algorithm as the AA, AB, and BB clusters are better defined.

```
> integerScoreToProbability <- function(x) 1 - 2^(-x/1000)
> rows <- which(isSnp(cnSet))
> cols <- 1:50
> posterior.prob <- tryCatch(integerScoreToProbability(snpCallProbability),
    error = function(e) print("This will not work for an ff object."))

[1] "This will not work for an ff object."

> posterior.prob1 <- integerScoreToProbability(snpCallProbability(cnSet)[rows,
    cols])
> invisible(open(snpCallProbability(cnSet2)))
> posterior.prob2 <- integerScoreToProbability(snpCallProbability(cnSet2)[rows,
    cols])
> invisible(close(snpCallProbability(cnSet2)))
> all.equal(posterior.prob1, posterior.prob2)

[1] "Mean relative difference: 0.001155494"
```

Next, we obtain locus-level estimates of copy number by fitting a linear model to each SNP. A variable named 'batch' must be indicated in the `protocolData` of the `cnSet` object. As the inverse variance of the within-genotype normalized intensities are used as weights in the linear model (and hence the design matrix in the regression changes for each SNP), the time is linear with the number of markers on the array. Copy number estimation at nonpolymorphic markers and polymorphic markers with unobserved genotypes is more difficult. We refer the interested reader to the technical report [**?**]. Again, we peform the copy number estimation using the matrix version and the ff version in parallel and encourage users with large datasets to explore the latter. As with the preprocessing and genotyping, the following code should be submitted as part of the batch job as it is too slow for interactive analysis.

```
> tryCatch(print(paste("Time for matrix version:", time1)),
    error = function(e) print("timing for CN estimation not available"))
> tryCatch(print(paste("Time for ff version:", time2)),
    error = function(e) print("timing for CN estimation not available"))
> rm(cnSet2)
> gc()
```

2 Accessors

For the remainder of this vignette, we illustrate accessors and visualizations using the sample object provided in the crlmm package.

```
> data(sample.CNSetLM)
> x <- sample.CNSetLM
```

2.1 Quantile-normalized intensities

Accessors for the quantile normalized intensities for the A allele at polymorphic loci:

```
> snp.index <- which(isSnp(x))
> np.index <- which(!isSnp(x))
> a <- (A(x))[snp.index, ]
> dim(a)

[1] 1139  180
```

The extra set of parentheses surrounding `A(cnSet2)` above is added to emphasize the appropriate order of operations. Subsetting the entire `x` object in the following code should be avoided for large datasets.

```
> a <- A(x[snp.index, ])
```

The quantile-normalized intensities for nonpolymorphic loci are obtained by:

```
> npIntensities <- (A(x))[np.index, ]
```

Quantile normalized intensities for the B allele at polymorphic loci:

```
> b.snps <- (B(x))[snp.index, ]
```

Note that NAs are recorded in the 'B' assay data element for nonpolymorphic loci:

```
> all(is.na((B(x))[np.index, ]))
```

```
[1] TRUE
```

```
            used   (Mb) gc trigger   (Mb)  max used   (Mb)
Ncells   3137861  167.6    4679654  250.0   4122692  220.2
Vcells 456898770 3485.9  861801150 6575.1 861140416 6570.0
```

SnpSet**: Genotype calls and confidence scores** Genotype calls:

```
> genotypes <- (snpCall(x))[snp.index, ]
```

Confidence scores of the genotype calls:

```
> genotypeConf <- integerScoreToProbability(snpCallProbability(x)[snp.index[1:10],
    ])
```

CopyNumberSet**: allele-specific copy number** Allele-specific copy number at polymorphic loci:

```
> ca <- CA(x[snp.index, ])/100
> cb <- CB(x[snp.index, ])/100
> ct <- ca + cb
```

Total copy number at nonpolymorphic loci:

```
> cn.nonpolymorphic <- CA(x[np.index, ])/100
```

Total copy number at both polymorphic and nonpolymorphic loci:

```
> cn <- crlmm:::totalCopyNumber(x, i = c(snp.index, np.index))
> apply(cn, 2, median, na.rm = TRUE)
```

```
  [1] 1.990 1.990 2.110 2.020 1.950 2.050 2.020 2.040 1.990 2.050
 [11] 1.960 1.970 1.980 2.060 2.060 2.070 2.010 2.040 2.070 2.070
 [21] 2.080 2.020 1.990 2.060 1.970 2.050 2.050 2.140 2.010 1.980
 [31] 2.020 2.070 2.010 2.070 2.010 2.030 1.990 2.080 1.950 2.030
 [41] 1.990 1.990 1.990 2.020 1.990 2.110 1.970 2.030 1.980 2.020
 [51] 1.970 2.020 1.990 2.070 2.080 2.040 2.000 2.040 1.970 2.010
 [61] 1.990 2.000 2.000 2.060 2.060 1.980 2.000 2.080 1.970 2.020
 [71] 1.970 2.030 2.030 2.060 2.030 2.040 2.020 2.000 2.000 2.000
 [81] 1.970 2.000 2.060 2.020 2.020 2.050 2.000 2.000 2.040 2.000
 [91] 2.050 1.970 1.990 2.040 2.130 1.970 2.040 2.090 2.000 2.020
[101] 2.060 2.040 2.060 2.180 1.990 2.000 2.020 2.030 2.020 2.040
[111] 2.030 1.970 2.070 2.080 1.970 2.010 2.050 2.000 2.020 2.060
[121] 1.960 2.030 2.030 2.020 1.990 2.030 2.030 1.980 1.950 2.010
[131] 1.980 2.070 2.020 2.000 2.070 2.020 2.010 2.010 2.020 1.990
[141] 2.020 1.960 2.020 2.020 2.030 2.040 2.020 2.010 2.080 2.040
[151] 1.990 2.040 2.050 2.080 2.050 2.040 2.080 2.040 2.020 2.020
[161] 1.990 1.990 2.040 2.020 2.010 2.030 2.030 2.020 2.030 2.010
[171] 1.990 2.005 2.060 2.060 1.980 1.970 2.060 1.930 2.045 1.970
```

(Note: the accessor totalCopyNumber method is sourced from a few helper functions. Documentation will be available in the devel version of the oligoClasses package.)

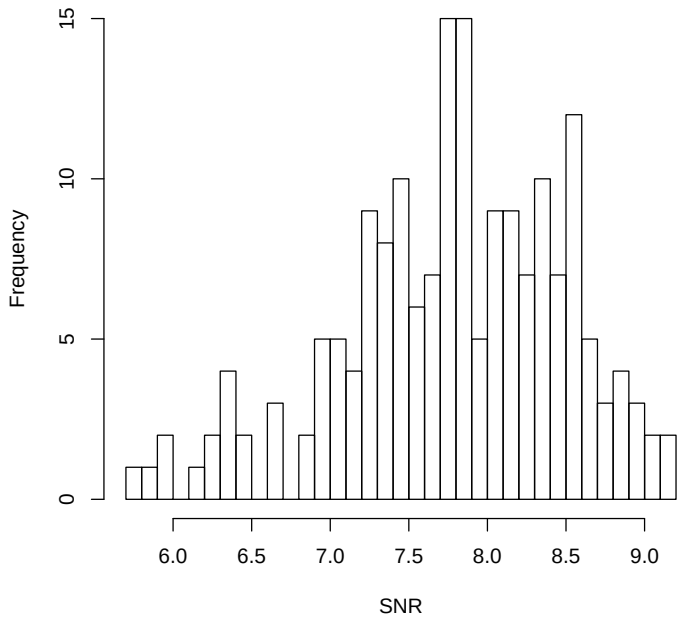

Figure 1: Signal to noise ratios for the HapMap samples.

2.2 Other accessors

Information on physical position and chromosome can be accessed as follows:

```
> xx <- position(x)
> yy <- chromosome(x)
```

Parameters from the linear model used to estimate copy number are stored in the slot `lM`.

```
> names(lM(x))
```

```
 [1] "tau2A"     "tau2B"     "sig2A"     "sig2B"     "nuA"
 [6] "nuB"       "phiA"      "phiB"      "phiPrimeA" "phiPrimeB"
[11] "corrAB"    "corrBB"    "corrAA"
```

```
> dim(lM(x)[[1]])
```

```
[1] 2119    2
```

3 Suggested visualizations

SNR. A histogram of the signal to noise ratio for the HapMap samples:

```
> hist(x$SNR, xlab = "SNR", main = "", breaks = 25)
```

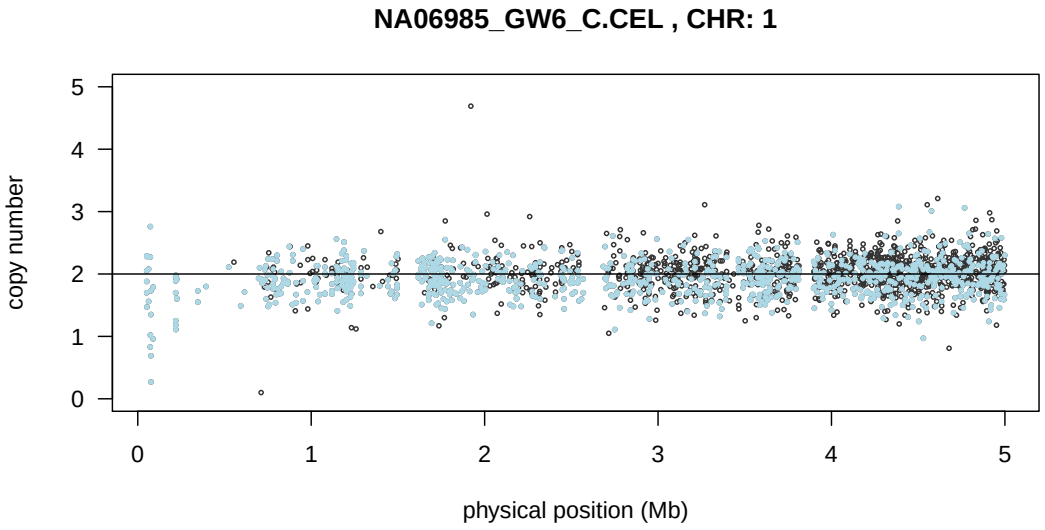

Figure 2: Total copy number (y-axis) for chromosome 1 plotted against physical position (x-axis) for one sample. Estimates at nonpolymorphic loci are plotted in light blue.

One sample at a time: locus-level estimates Figure 2 plots physical position (horizontal axis) versus copy number (vertical axis) for the first sample. There is less information to estimate copy number at nonpolymorphic loci; improvements to the univariate prediction regions at nonpolymorphic loci are a future area of research.

```
> cn <- crlmm:::totalCopyNumber(x, j = 1)
> par(las = 1, mar = c(4, 5, 4, 2))
> plot(position(x), cn, pch = 21, cex = 0.4, xaxt = "n",
     col = "grey20", ylim = c(0, 5), ylab = "copy number",
     xlab = "physical position (Mb)", main = paste(sampleNames(x)[1],
         ", CHR:", unique(chromosome(x))))
> points(position(x)[!isSnp(x)], cn[!isSnp(x)], pch = 21,
     cex = 0.4, col = "lightblue", bg = "lightblue")
> axis(1, at = pretty(range(position(x))), labels = pretty(range(position(x)))/1e+06)
> abline(h = 2)
```

One SNP at a time Scatterplots of the A and B allele intensities (log-scale) can be useful for assessing the biallelic genotype calls. This section of the vignette is currently under development.

4 Session information

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

- R version 2.11.1 RC (2010-05-27 r52117), `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 2.8.0, bit 1.1-4, crlmm 1.6.5, ff 2.1-4, oligoClasses 1.10.0
- Loaded via a namespace (and not attached): affyio 1.16.0, annotate 1.26.0, AnnotationDbi 1.10.1, Biostrings 2.16.5, DBI 0.2-5, ellipse 0.3-5, genefilter 1.30.0, IRanges 1.6.6, mvtnorm 0.9-9, preprocessCore 1.10.0, RSQLite 0.9-1, splines 2.11.1, survival 2.35-8, xtable 1.5-6

References

References

[1] Robert B Scharpf, Ingo Ruczinski, Benilton Carvalho, Betty Doan, Aravinda Chakravarti, and Rafael Irizarry. A multilevel model to address batch effects in copy number estimation using snp arrays. *Biostatistics*, July 2010.