Copy number estimation

Rob Scharpf

May, 2009

Abstract

This vignette estimates copy number for HapMap samples on the Affymetrix 6.0 platform.

1 Simple usage

The following packages are required:

```
> library(crlmm)
> library(genomewidesnp6Crlmm)
```

Preprocess and genotype. Specify the coordinates of Affymetrix cel files and where to put intermediate files generated during the course of preprocessing / copy number estimation.

```
> myPath <- "/thumper/ctsa/snpmicroarray/hapmap/raw/affy/1m"
> celFiles <- list.celfiles(myPath, full.names = TRUE, pattern = ".CEL")
> outdir <- "/thumper/ctsa/snpmicroarray/rs/data/hapmap/1m/affy"
```

Preprocess and genotype (for more info see the crlmm vignette):

```
> crlmmWrapper(celFiles[1:15], save.it = TRUE, load.it = TRUE,
+     intensityFile = file.path(outdir, "normalizedIntensities.rda"),
+     crlmmFile = file.path(outdir, "snpsetObject.rda"))
```

As a result of the above wrapper, the following R objects are now created:

```
> list.files(outdir)[grep("crlmmSetList", list.files(outdir))]
```

Locus- and allele-specific estimates of copy number. Load the object for chromosome 22 and compute copy number:

```
> CHR <- 22
> if (!exists("crlmmSetList")) load(file.path(outdir, paste("crlmmSetList_",
+     CHR, ".rda", sep = "")))
```

```
> show(crlmmSetList)
> if (length(crlmmSetList) == 2) {
+     crlmmSetList <- update(crlmmSetList, CHR = CHR)
+ }
> show(crlmmSetList)
```

See the help file for `computeCopynumber` for arguments to `update`. Provided that the assumption of integer copy number is reasonable, one can fit a hidden Markov model to the locus-level estimates of copy number and uncertainty.

A hidden Markov model. Emission probabilities for the hidden markov model can be computed from the copy number prediction regions based on bivariate normal scatter plots of the log A versus log B intensities. (By contrast, a nonparamemtric segmentation would be applied to intensities on the scale of copy number.)

```
> library(VanillaICE)
> copyNumberStates <- 0:5
> if (!exists("emission.cn")) {
+     emission.cn <- suppressWarnings(crlmm:::computeEmission(crlmmSetList,
+         copyNumberStates))
+     dim(emission.cn)
+ }
```

Warning: more can be done than currently implemented to protect against outliers. In addition, improved estimates of uncertainty for the copy number prediction regions will also help.

Initial state probabilities and transition probabilities for the HMM:

```
> initialP <- rep(1/length(copyNumberStates), length(copyNumberStates))
> tau <- transitionProbability(chromosome = chromosome(crlmmSetList),
+     position = position(crlmmSetList), TAUP = 1e+08)
```

The viterbi algorithm is used to identify the sequence of states that maximizes the likelihood:

```
> if (!exists("hmmPredictions")) {
+     hmmPredictions <- viterbi(emission = emission.cn, initialStateProbs = log(initialP),
+         tau = tau[, "transitionPr"], arm = tau[, "arm"], normalIndex = 3,
+         normal2altered = 0.01, altered2normal = 1, altered2altered = 0.001)
+ }
> table(as.integer(hmmPredictions) - 1)
> brks <- breaks(x = hmmPredictions, states = copyNumberStates,
+     position = tau[, "position"], chromosome = tau[, "chromosome"])
> str(brks)
```

Circular binary segmentation TO DO.

2 Accessors

2.1 Assay data accessors

`ABset`: **quantile normalized intensities** An object of class `ABset` is stored in the first element of the `crlmmSetList` object. The following accessors may be of use:

Accessors for the quantile normalized intensities for the A allele:

```
> a <- A(crlmmSetList)
> dim(a)
```

The quantile normalized intensities for the nonpolymorphic probes are also stored in the 'A' assay data element. To retrieve the quantile normalized intensities for the A allele only at polymorphic loci:

```
> a.snps <- A(crlmmSetList[snpIndex(crlmmSetList), ])
> dim(a.snps)
```

For the nonpolymorphic loci:

```
> a.nps <- A(crlmmSetList[cnIndex(crlmmSetList), ])
> dim(a.nps)
```

Quantile normalized intensities for the B allele at polymorphic loci:

```
> b.snps <- B(crlmmSetList[snpIndex(crlmmSetList), ])
```

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

```
> all(is.na(B(crlmmSetList[cnIndex(crlmmSetList), ])))
```

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

```
> genotypes <- calls(crlmmSetList)
```

Confidence scores of the genotype calls:

```
> genotypeConf <- confs(crlmmSetList)
```

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

```
> ca <- CA(crlmmSetList[snpIndex(crlmmSetList), ])
```

Total copy number at nonpolymorphic loci:

```
> cn.nonpolymorphic <- CA(crlmmSetList[cnIndex(crlmmSetList), ])
> range(cn.nonpolymorphic, na.rm = TRUE)
> median(cn.nonpolymorphic, na.rm = TRUE)
```

Total copy number at both polymorphic and nonpolymorphic loci:

```
> cn <- copyNumber(crlmmSetList)
```

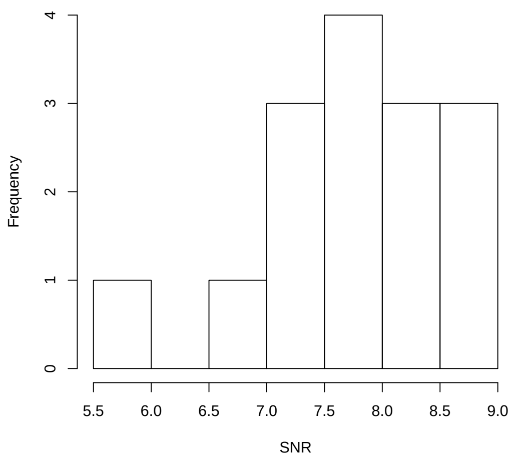

Figure 1: Signal to noise ratios for the HapMap samples.

2.2 Other accessors

After running `update` on the `crlmmSetList` object, information on physical position and chromosome can be accessed by the following accessors:

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

There are many parameters computed during copy number estimation that are at present stored in the `featureData` slot of the `CopyNumberSet` element. TO DO: Accessors for these parameters, as well as better containers for storing these parameters. See

```
> fvarLabels(crlmmSetList[[3]])
```

3 Suggested visualizations

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

```
> hist(crlmmSetList[[2]]$SNR, xlab = "SNR", main = "")
```

We suggest excluding samples with a signal to noise ratio less than `<<SNRmin>>`. As batch effects can be very large in the quantile-normalized intensities, we suggest adjusting for date or chemistry plate. Ideally, one would have 70+ files in a given batch. Here we make a table of date versus ancestry:

As all of these samples were run on the first week of March, we would expect that any systematic artifacts to the intensities that develop over time to be minimal (a best case scenario). As this is typically not the case, we illustrate how one may adjust for batch using the chemistry plate as an argument for `batch` in the `computeCopynumber` function.

**Note: the number of samples in the `CrlmmSetList` object after copy number estimation may be fewer than the number of samples in the `CrlmmSetList` object after preprocessing/genotyping. This occurs when 1 or more samples have a signal-to-noise ratio less than value passed to `SNRmin`. By default, intermediate forms of the data are stored in one object to ensure that each element in the `CrlmmSetList` have the same ordering of probes and samples. The object returned by `computeCopynumber` is ordered by chromosome and physical position (useful for downstream methods that smooth the copy number as a function of the physical position). **

The above algorithm for estimating copy number is predicated on the assumption that most samples within a batch have copy number 2 at any given locus. For common copy number variants, this assumption may not hold. An additional iteration using a bias correction provides additional robustness to this assumption. Set the `bias.adj` argument to `TRUE`:

```
> update(crlmmSetList, CHR = 22, bias.adj = TRUE)
```

Calculate the frequency of amplifications and deletions at each locus.

```
> require(SNPchip)
> library(RColorBrewer)
> numberUp <- rowSums(hmmPredictions > 3, na.rm = TRUE)
> numberDown <- -rowSums(hmmPredictions < 3, na.rm = TRUE)
> poly.cols <- brewer.pal(7, "Accent")
> alt.brks <- brks[brks[, "state"] != "copy.number_2", ]
> op <- par(ask = FALSE)
> ylim <- c(min(numberDown) - 5, max(numberUp) + 5)
> xlim <- c(10 * 1e+06, max(position(crlmmSetList)))
> plot(position(crlmmSetList), rep(0, nrow(crlmmSetList[[1]])),
+     type = "n", xlab = "Physical position (Mb)", ylim = ylim,
+     xlim = xlim, ylab = "frequency", main = "Chr 22", xaxt = "n",
+     xaxs = "r")
> axis(1, at = pretty(xlim), labels = pretty(xlim)/1e+06)
> polygon(x = c(position(crlmmSetList), rev(position(crlmmSetList))),
+     y = c(rep(0, nrow(crlmmSetList[[1]])), rev(numberUp)), col = poly.cols[3],
+     border = poly.cols[3])
> polygon(x = c(position(crlmmSetList), rev(position(crlmmSetList))),
+     y = c(rep(0, nrow(crlmmSetList[[1]])), rev(numberDown)),
+     col = poly.cols[5], border = poly.cols[5])
> medLength <- round(median(alt.brks[, "nbases"]), 2)
> medMarkers <- median(alt.brks[, "nprobes"])
> sdMarkers <- round(mad(alt.brks[, "nprobes"]), 2)
```

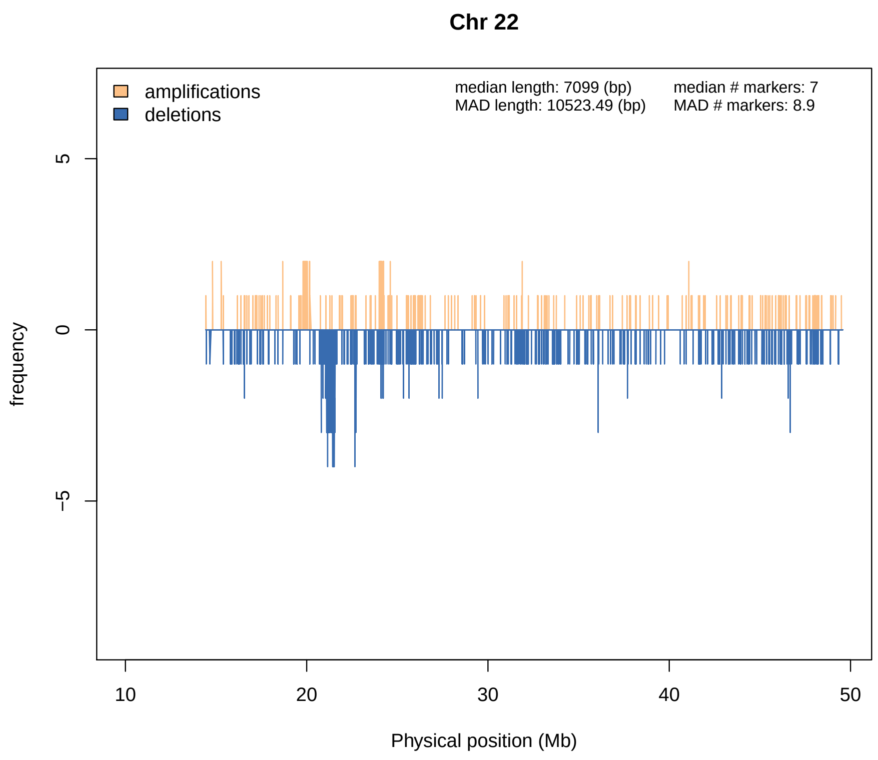

Figure 2:

```
> sdsLength <- round(mad(alt.brks[, "nbases"]), 2)
> legend("topright", bty = "n", legend = c(paste("median length:",
+     medLength, "(bp)"), paste("MAD length:", sdsLength, "(bp)"),
+     paste("median # markers:", medMarkers), paste("MAD # markers:",
+         sdMarkers)), cex = 0.8, ncol = 2)
> legend("topleft", fill = poly.cols[c(3, 5)], legend = c("amplifications",
+     "deletions"), bty = "n")
> par(op)
> gc()
```

One sample at a time: locus-level estimates Plot physical position versus copy number for the first sample. Recall that the copy number estimates were multiplied by 100 and stored as an integer.

```
> par(las = 1)
> plot(position(crlmmSetList), copyNumber(crlmmSetList)[, 1], pch = ".",
```

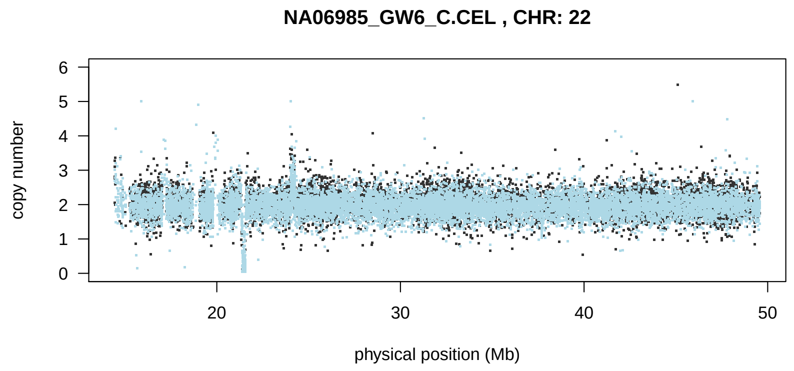

Figure 3: Total copy number (y-axis) for chromosome 22 plotted against physical position (x-axis) for one sample. Estimates at nonpolymorphic loci are plotted in light blue.

```
+     cex = 2, xaxt = "n", col = "grey20", ylim = c(0, 6), ylab = "copy number",
+     xlab = "physical position (Mb)", main = paste(sampleNames(crlmmSetList)[1],
+         ", CHR:", unique(chromosome(crlmmSetList))))
> points(position(crlmmSetList)[cnIndex(crlmmSetList)], copyNumber(crlmmSetList)[cnIndex(crl
+     1], pch = ".", cex = 2, col = "lightblue")
> axis(1, at = pretty(range(position(crlmmSetList))), labels = pretty(range(position(crlmmSe
```

One SNP at a time Plot the prediction regions for total copy number 2 and 3 for the first plate. Plotting symbols are the genotype calls (1=AA, 2=AB, 3=BB); light grey points are from other plates. One could also add the prediction regions for 0-4 copies, but it gets crowded.

```
> xlim <- ylim <- c(6.5, 13)
> pch <- 21
> colors <- c("red", "blue", "green3")
> cex <- 0.6
> par(mfrow = c(3, 3), las = 1, pty = "s", ask = FALSE, mar = c(2,
+     2, 2, 2), oma = c(2, 2, 1, 1))
> indices <- split(snpIndex(crlmmSetList), rep(1:length(snpIndex(crlmmSetList)),
+     each = 9, length.out = length(snpIndex(crlmmSetList))))
> par(ask = FALSE)
> j <- 1
> for (i in indices[[j]]) {
+     gt <- calls(crlmmSetList)[i, ]
+     plot(crlmmSetList[i, ], pch = pch, col = colors[gt], bg = colors[gt],
+         cex = cex, xlim = xlim, ylim = ylim)
+     mtext("A", 1, outer = TRUE, line = 1)
+     mtext("B", 2, outer = TRUE, line = 1)
+ }

> require(RColorBrewer)
> library(ellipse)
> greens <- brewer.pal(9, "Greens")
> J <- split(1:ncol(crlmmSetList), batch(crlmmSetList))
> colors <- c("red", "blue", "green3")
> cex <- 0.6
> colors <- c("blue", greens[8], "red")
> pch.col <- c("grey40", "black", "grey40")
> xlim <- ylim <- c(6.5, 13)
> plotpoints <- FALSE
> lwd <- 2
> ask <- FALSE
> par(mfrow = c(3, 3), las = 1, pty = "s", ask = ask, mar = c(2,
+     2, 2, 2), oma = c(2, 2, 1, 1))
> indices <- split(snpIndex(crlmmSetList), rep(1:length(snpIndex(crlmmSetList)),
+     each = 9, length.out = length(snpIndex(crlmmSetList))))
> j <- 1
> cat(j, "\n")
> k <- 1
> for (i in indices[[j]]) {
+     gt <- calls(crlmmSetList)[i, ]
+     pch <- as.character(gt)
+     cex <- 0.9
+     plot(crlmmSetList[i, ], pch = pch, col = pch.col[gt], cex = cex,
+         xlim = xlim, ylim = ylim, type = "n")
+     if (plotpoints) {
+         for (b in seq(along = unique(batch(crlmmSetList)))) {
+             points(crlmmSetList[i, J[[b]]], pch = pch, col = colors[b],
+                 bg = colors[b], cex = cex, xlim = xlim, ylim = ylim)
+         }
+     }
+     for (b in seq(along = unique(batch(crlmmSetList)))) {
+         ellipse(crlmmSetList[i, J[[b]]], copynumber = 2, col = colors[b],
+             lwd = lwd)
+     }
+     if (k == 1) {
+         legend("bottomleft", bty = "n", fill = colors, legend = c("CEPH",
+             "Yoruba", "Asian"))
+         mtext("A", 1, outer = TRUE, line = 1)
+         mtext("B", 2, outer = TRUE, line = 0)
+     }
+     k <- k + 1
+ }
```

4 Details for the copy number estimation procedure

TO DO.

5 Session information

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