Estimating copy number for Affymetrix 6.0 with the crlmm Package

Rob Scharpf

February, 2009

1 Estimating copy number

At present, software for copy number estimation is provided only for the Affymetrix 6.0 platform. This vignette estimates copy number for the HapMap samples.

1.1 Preprocess and genotype the samples

We preprocess and genotype the samples as described in the CRLMM vignette.

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

Specify the complete path for the CEL files and a directory in which to store intermediate files:

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

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

```
R> if (!exists("crlmmResult")) {
    if (file.exists(file.path(outdir, "crlmmResult.rda"))) {
        load(file.path(outdir, "intensities.rda"))
        load(file.path(outdir, "crlmmResult.rda"))
    }
    else {
        crlmmResult <- crlmm(celFiles, save.it = TRUE,
            intensityFile = file.path(outdir,
                "intensities.rda"))
        save(crlmmResult, file = file.path(outdir,
            "crlmmResult.rda"))
    }
}
```

Quantile normalize the nonpolymorphic probes and save the results.

```
R> if (!exists("cnrmaResult")) {
     if (file.exists(file.path(outdir, "cnrmaResult.rda")))
         load(file.path(outdir, "cnrmaResult.rda"))
     else {
         cnrmaResult <- cnrma(celFiles)
         save(cnrmaResult, file = file.path(outdir,
             "cnrmaResult.rda"))
     }
 }
```

1.2   Copy number

We require 6 items for copy number estimation:

- quantile-normalized A intensities (I1 x J)

- quantile-normalized B intensities (I1 x J)

- quantile-normalized intensities from nonpolymorphic (NP) probes (I2 x J)

- genotype calls (I1 x J)

- confidence scores of the genotype calls (I1 x J)
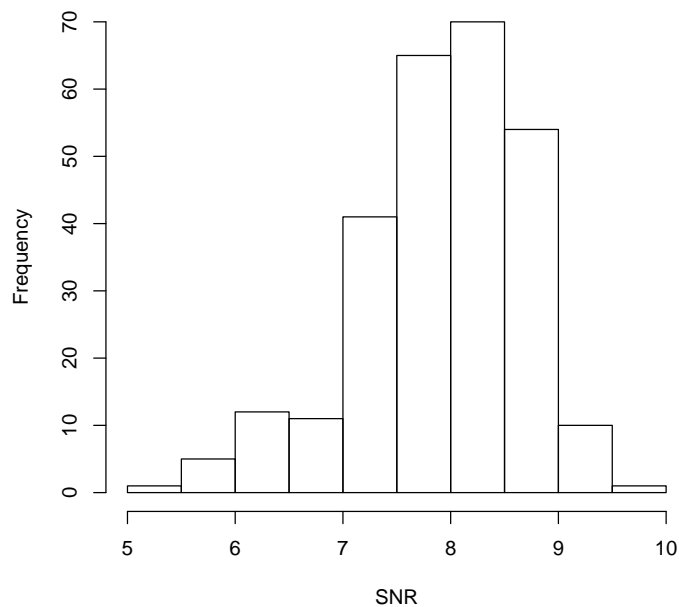
- signal to noise ratio (SNR) of the samples (J)

These items are extracted as follows:

```
R> A <- res$A
R> B <- res$B
R> calls <- crlmmResult$calls
R> conf <- crlmmResult$conf
R> SNR <- crlmmResult$SNR
R> NP <- cnrmaResult$NP
R> gc()
```

```
            used   (Mb) gc trigger   (Mb)   max used    (Mb)
Ncells    3218205  171.9    3933533  210.1    3581658   191.3
Vcells 684977131 5226.0 1020622616 7786.8 722599570 5513.0
```

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

```
R> hist(SNR, xlab = "SNR", main = "")
```

We suggest excluding samples with a signal to noise ratio less than 5. As batch effects can be very large in the quantile-normalized intensities, we suggest adjusting for date or chemistry plate.

```
R> dts <- celDates(celFiles)

..

R> table(format(dts, "%d %b %Y"))

01 Mar 2007 02 Mar 2007 03 Mar 2007 06 Mar 2007 07 Mar 2007
          1         168           9          84           4
08 Mar 2007 21 Feb 2007
          3           1
```

Ideally, one would have 70+ files in a given batch. Here we make a table of date versus ancestry:

```
R> sns <- colnames(calls)
R> sns[1]

[1] "NA06985_GW6_C.CEL"

R> plate <- substr(basename(sns), 13, 13)
R> table(plate)
```

```
plate
 A  C  Y
90 90 90

R> table(format(dts, "%d %b %Y"), plate)

            plate
             A  C  Y
  01 Mar 2007  1  0  0
  02 Mar 2007 83  0 85
  03 Mar 2007  5  0  4
  06 Mar 2007  0 84  0
  07 Mar 2007  0  4  0
  08 Mar 2007  0  2  1
  21 Feb 2007  1  0  0
```

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 to the computeCopynumber function.

```
R> if (!exists("env")) {
     CHR <- 22
     if (!file.exists(file.path(outdir, paste("env_chr",
         CHR, ".rda", sep = "")))) {
         env <- new.env()
         computeCopynumber(chrom = CHR, A = A,
             B = B, calls = calls, conf = conf,
             NP = NP, plate = plate, envir = env,
             DF.PRIOR = 50, MIN.OBS = 1, SNR = SNR,
             SNRmin = 5)
         save(env, file = file.path(outdir, paste("env_chr",
             CHR, ".rda", sep = "")))
     }
     else {
         load(file.path(outdir, paste("env_chr",
             CHR, ".rda", sep = "")))
     }
 }
```

A class representation would be useful for this sort of data (TODO). The oligoSnpSet class used below is a bit inefficient as the assay data elements are forced to be the same size. The advantage is that all of the assay data is bound to the meta-data.

```
R> require(oligoClasses)
R> data(snpProbes, package = "genomewidesnp6Crlmm")
```

4

```
R> data(cnProbes, package = "genomewidesnp6Crlmm")
R> position <- snpProbes[match(env[["snps"]], rownames(snpProbes)),
     "position"]
R> position.np <- cnProbes[match(rownames(env[["NP"]]),
     rownames(cnProbes)), "position"]
R> CA <- env[["CA"]]
R> CB <- env[["CB"]]
R> snpCT <- CA + CB
R> npCT <- env[["CT"]]
R> CT <- rbind(snpCT, npCT)
R> dimnames(CT) <- list(c(env[["snps"]], rownames(env[["NP"]])),
     env[["sns"]])
R> genotypes <- matrix(NA, nrow(CT), ncol(CT))
R> dimnames(genotypes) <- dimnames(CT)
R> genotypes[1:length(env[["snps"]]), ] <- env[["calls"]]
R> polymorphic <- c(rep(1, length(env[["snps"]])),
     rep(0, nrow(npCT)))
R> fD <- new("AnnotatedDataFrame", data = data.frame(chromosome = rep(CHR,
     nrow(CT)), polymorphic = polymorphic, position = c(position,
     position.np)), varMetadata = data.frame(labelDescription = c("chromosome",
     "polymorphic", "position")))
R> locusSet <- new("oligoSnpSet", copyNumber = CT,
     calls = genotypes, featureData = fD, phenoData = annotatedDataFrameFrom(CT,
       byrow = FALSE), annotation = "genomewidesnp6")
```

Note that an indicator for the polymorphic probes was created in the code chunk above. I use this indicator to smooth the estimates of copy number from the nonpolymorphic and polymorphic probes separately. The uncertainty estimates (work in progress) should reflect that the nonpolymorphic probes have more variance and the smoothing will take this into account. 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 SNP. For common copy number variants, this assumption may not hold. An additional iteration using a bias correction can improve the estimates. Set the `bias.adj` argument to `TRUE`.

2   Suggested plots

One sample at a time   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.

```
R> require(ellipse)
R> par(las = 1)
R> plot(position(locusSet), copyNumber(locusSet)[,
     1]/100, pch = ".", cex = 2, xaxt = "n", col = "grey70",
```

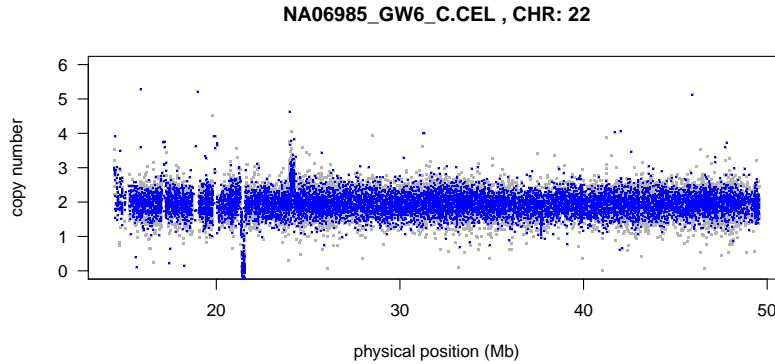**NA06985_GW6_C.CEL , CHR: 22**

Figure 1: Total copy number (y-axis) for chromosome 22 plotted against physical position (x-axis) for one sample.

```
    ylim = c(0, 6), ylab = "copy number", xlab = "physical position (Mb)",
    main = paste(sampleNames(locusSet)[1], ", CHR:",
        unique(chromosome(locusSet))))
R> axis(1, at = pretty(range(position(locusSet))),
    labels = pretty(range(position(locusSet)))/1e+06)
R> I <- fData(locusSet)$polymorphic == 0
R> points(position(locusSet)[I], copyNumber(locusSet)[I,
    1]/100, pch = ".", col = "blue", cex = 1)

R> require(SNPchip)
R> plotCytoband(22, new = FALSE, cytoband.ycoords = c(5.8,
    6), label.cytoband = FALSE)
```

In this example, the estimates of copy number for the nonpolymorphic probes appear correlated with the polymorphic probes – a good sign.

One SNP at a time This section needs to be cleaned up (TODO). The parameters needed for drawing prediction regions are plate-specific.

```
R> tau2A <- env[["tau2A"]]
R> tau2B <- env[["tau2B"]]
R> sig2A <- env[["sig2A"]]
R> sig2B <- env[["sig2B"]]
R> nuA <- env[["nuA"]]
R> nuB <- env[["nuB"]]
R> phiA <- env[["phiA"]]
R> phiB <- env[["phiB"]]
R> corr <- env[["corr"]]
R> corrA.BB <- env[["corrA.BB"]]
```

6

```
R> corrB.AA <- env[["corrB.AA"]]
R> A <- env[["A"]]
R> B <- env[["B"]]
```

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.

```
R> par(las = 1, pty = "s")
R> p <- 1
R> J <- grep(unique(plate)[p], plate)
R> ylim <- c(6.5, 13)
R> I <- which(phiA > 10 & phiB > 10)
R> i <- I[1]
R> log2(phiA[i, ])

[1] 9.071797 8.989059 8.876777

R> log2(phiB[i, ])

[1] 8.533698 8.419028 8.741665

R> plot(log2(A[i, ]), log2(B[i, ]), pch = as.character(calls[i,
      ]), col = "grey60", cex = 0.9, ylim = ylim,
      xlim = ylim, xlab = "A", ylab = "B")
R> points(log2(A[i, J]), log2(B[i, J]), col = "black",
      pch = as.character(calls[i, J]))
R> for (CT in 2) {
      if (CT == 2)
          ellipse.col <- "black"
      for (CA in 0:CT) {
          CB <- CT - CA
          A.scale <- sqrt(tau2A[i, p] * (CA == 0) +
              sig2A[i, p] * (CA > 0))
          B.scale <- sqrt(tau2B[i, p] * (CB == 0) +
              sig2B[i, p] * (CB > 0))
          scale <- c(A.scale, B.scale)
          if (CA == 0 & CB > 0)
              rho <- corrA.BB[i, p]
          if (CA > 0 & CB == 0)
              rho <- corrB.AA[i, p]
          if (CA > 0 & CB > 0)
              rho <- corr[i, p]
          lines(ellipse(x = rho, centre = c(log2(nuA[i,
              p] + CA * phiA[i, p]), log2(nuB[i,
              p] + CB * phiB[i, p])), scale = scale),
```
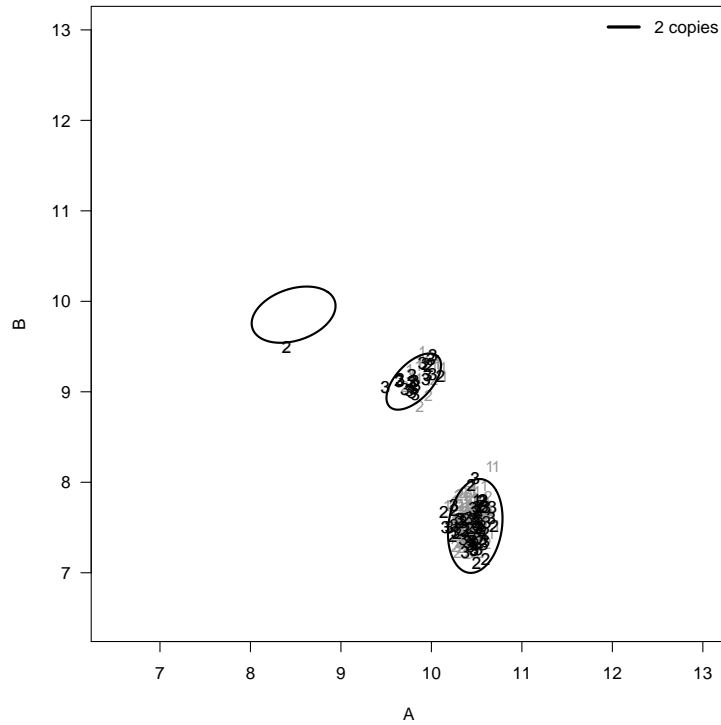
Figure 2: Prediction regions for copy number 2 for one SNP. The plate effects are negligible for this SNP and we only plot the prediction region for the first plate.

```
            col = ellipse.col, lwd = 2)
    }
 }
R> legend("topright", lwd = 3, col = "black", legend = "2 copies",
     bty = "n")
```

Look at the distribution of shifts in the predicted centers across the plates. The biggest shifts are for SNPs that have no observations in a subset of the plates – need more shrinkage here.

Now look at shifts for which we have at least 3 observations in each genotype cluster (TO DO).

3 Session information

R> sessionInfo()

```
R version 2.9.0 Under development (unstable) (2009-02-08 r47879)
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_

attached base packages:
[1] stats     graphics  grDevices utils     datasets
[6] methods   base

other attached packages:
[1] ellipse_0.3-5            oligoClasses_1.5.16
[3] Biobase_2.3.11           genomewidesnp6Crlmm_1.0.1
[5] crlmm_1.0.60

loaded via a namespace (and not attached):
 [1] affyio_1.11.3       annotate_1.21.3
 [3] AnnotationDbi_1.5.15 DBI_0.2-4
 [5] genefilter_1.23.2   mvtnorm_0.9-4
 [7] preprocessCore_1.5.3 RSQLite_0.7-1
 [9] splines_2.9.0       survival_2.34-1
[11] tools_2.9.0         xtable_1.5-4
```