Copy number estimation and genotype calling with crlmm

Rob Scharpf

May, 2009

Abstract

This vignette estimates copy number for HapMap samples on the Affymetrix 6.0 platform. See [3] for the working paper.

1 Simple usage

CRLMM supports the following platforms:

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

```
[1] "genomewidesnp6"   "genomewidesnp5"   "human370v1c"
[4] "human370quadv3c" "human550v3b"      "human650v3a"
[7] "human610quadv1b" "human660quadv1a" "human1mduov3b"
```

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.

```
R> myPath <- "/thumper/ctsa/snpmicroarray/hapmap/raw/affy/1m"
R> celFiles <- list.celfiles(myPath, full.names = TRUE,
     pattern = ".CEL")
R> outdir <- "/thumper/ctsa/snpmicroarray/rs/data/hapmap/1m/crlmm"
R> if (!file.exists(outdir)) dir.create(outdir)
```

Split the list of files by plate. For this HapMap data, the different ancestries were run on separate plates.

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

```
plate
 A  C  Y
90 90 90
```

```
R> celFiles <- split(celFiles, plate)
```

The next code chunk quantile normalizes the samples to a target reference distribution and uses the crlmm algorithm to genotype. See [1] for details regarding the crlmm genotyping algorithm.

```
R> for (i in seq(along = celFiles)) {
     platedir <- file.path(outdir, names(celFiles)[i])
     if (!file.exists(platedir))
         dir.create(platedir)
     crlmmWrapper(celFiles[[i]], cdfName = "genomewidesnp6",
         save.it = TRUE, load.it = TRUE, intensityFile = file.path(platedir,
             "normalizedIntensities.rda"), crlmmFile = file.path(platedir,
             "snpsetObject.rda"))
 }
```

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

```
R> fns <- list.files(file.path(outdir, names(celFiles[1])),
     pattern = "crlmmSetList", full.name = TRUE)
```

To estimate copy number, run the update function on the vector of filenames. Here we run update on the chromosomes 1-5 in batch 'C'.

Alternatively, load the crlmmSetList object for a particular chromosome and compute copy number.

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

 Elements in CrlmmSetList object:

class:  ABset
assayData elements:  A B
Dimensions:  23991 90

class:  SnpSet
assayData elements:  call callProbability
Dimensions:  23991 90

R> crlmmSetList <- update(crlmmSetList)

....

R> show(crlmmSetList)
```

```
 Elements in CrlmmSetList object:

class:  ABset
assayData elements:  A B
Dimensions:  23989 90

class:  SnpSet
assayData elements:  call callProbability
Dimensions:  23989 90

class:  CopyNumberSet
assayData elements:  CA CB
Dimensions:  23989 90
```

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. As discussed in citeauthorKorn2008, emission probabilities for the hidden markov model can be computed directly from the bivariate normal scatter plots of the log A versus log B intensities [2]. (By contrast, a nonparamemtric segmentation would be applied to intensities on the scale of copy number.)

```
R> library(VanillaICE)
R> copyNumberStates <- 0:3
R> if (!exists("emission.cn")) {
     emission.cn <- suppressWarnings(crlmm:::computeEmission(crlmmSetList,
         copyNumberStates))
     emission.cn[emission.cn < -10] <- -10
     dim(emission.cn)
 }
```

```
Computing emission probabilities
[1] 23989    90     4
```

*Before smoothing the estimates as a function of physical position by segmentation hidden Markov models, one should consider how to handle outliers. In particular, samples with low signal to noise ratios are likely to have many copy number outliers. See suggested visualizations.

Initial state probabilities and transition probabilities for the HMM:

```
R> initialP <- rep(1/length(copyNumberStates), length(copyNumberStates))
R> 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:

```
R> if (!exists("hmmPredictions")) {
     hmmPredictions <- viterbi(emission = emission.cn,
         initialStateProbs = log(initialP), tau = tau[,
             "transitionPr"], arm = tau[, "arm"],
         normalIndex = 3, normal2altered = 0.1,
         altered2normal = 1, altered2altered = 0.01)
 }
R> table(as.integer(hmmPredictions) - 1)

      0       1       2       3 
    514    8697 2147464    2335 

R> brks <- breaks(x = hmmPredictions, states = copyNumberStates,
     position = tau[, "position"], chromosome = tau[,
         "chromosome"])
R> str(brks)

'data.frame':        397 obs. of  7 variables:
 $ sample : chr  "NA06985_GW6_C.CEL" "NA06985_GW6_C.CEL" "NA06985_GW6_C.CEL" "NA06985_GW6_C.
 $ chr    : chr  "22" "22" "22" "22" ...
 $ start  : int  14432516 14459243 19872603 20051591 21341461 21408678 21500180 21513110 215
 $ end    : int  14457129 19795835 20038096 21334910 21401672 21498767 21512985 21553233 239
 $ nbases : num  24614 5336593 165494 1283320 60212 ...
 $ nprobes: int  8 2937 10 918 37 50 12 28 1607 53 ...
 $ state  : int  3 2 3 2 1 0 1 0 2 3 ...
```

Circular binary segmentation

```
R> library(DNAcopy)
R> library(genefilter)
R> ratioset <- crlmmSetList[[3]]
R> meds <- rowMedians(copyNumber(ratioset), na.rm = TRUE)
R> ratios <- log2(copyNumber(ratioset)) - log2(meds)
R> ratioset <- new("SnpCopyNumberSet", copyNumber = ratios,
     phenoData = phenoData(ratioset), featureData = featureData(ratioset),
     annotation = annotation(ratioset))
```

The following code chunk (not evaluated) takes a while to run:

```
R> CNA.object <- CNA(copyNumber(ratioset), chromosome(ratioset),
     position(ratioset), data.type = "logratio",
     sampleid = sampleNames(ratioset))
R> smoothed.CNA.object <- smooth.CNA(CNA.object)
R> segment.cna <- segment(smoothed.CNA.object, verbose = 1)
R> save(segment.cna, file = file.path(outdir, paste("segment.cna_",
     CHR, ".rda", sep = "")))
```

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:

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

```
[1] 23989    90
```

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:

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

```
[1] 11537    90
```

For the nonpolymorphic loci:

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

```
[1] 12452    90
```

Quantile normalized intensities for the B allele at polymorphic loci:

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

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

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

```
[1] TRUE
```

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

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

Confidence scores of the genotype calls:

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

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

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

Total copy number at nonpolymorphic loci:

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

[1] 0.05 5.00

R> median(cn.nonpolymorphic, na.rm = TRUE)

[1] 2
```

Total copy number at both polymorphic and nonpolymorphic loci:

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

2.2 Other accessors

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

```
R> xx <- position(crlmmSetList)
R> 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

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

 [1] "chromosome" "position"   "tau2A_A"    "tau2B_A"
 [5] "sig2A_A"    "sig2B_A"    "nuA_A"      "nuB_A"
 [9] "phiA_A"     "phiB_A"     "corr_A"     "corrA.BB_A"
[13] "corrB.AA_A"
```

3 Suggested visualizations

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

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

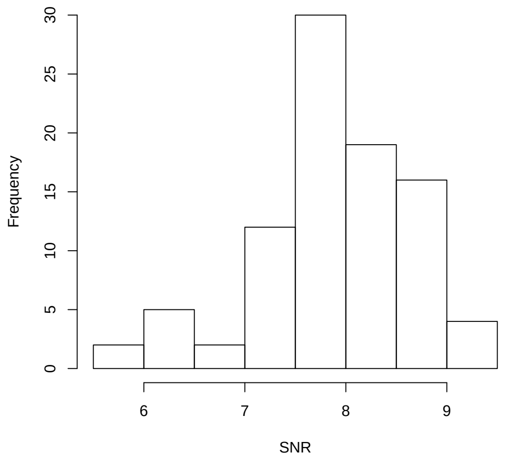

Figure 1: Signal to noise ratios for the HapMap samples.

For Affymetrix 6.0, we suggest excluding samples with a signal to noise ratio less than 5. Adjusting by date or chemistry plate can be helpful for limiting the influence of batch effects. 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`:

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

The following code chunk calculates the frequency of amplifications and deletions at each locus. Shaded regions above the zero line in Figure 2 display the frequency of amplifications, whereas shaded regions below the zero line graphically display the frequency of hemizygous or homozygous deletions.

```
R> require(SNPchip)
R> library(RColorBrewer)
R> numberUp <- rowSums(hmmPredictions > 3, na.rm = TRUE)
R> numberDown <- -rowSums(hmmPredictions < 3, na.rm = TRUE)
R> poly.cols <- brewer.pal(7, "Accent")
R> alt.brks <- brks[brks[, "state"] != "copy.number_2",
     ]
R> op <- par(ask = FALSE)
R> ylim <- c(min(numberDown) - 5, max(numberUp) +
     5)
R> xlim <- c(10 * 1e+06, max(position(crlmmSetList)))
R> 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")
R> axis(1, at = pretty(xlim), labels = pretty(xlim)/1e+06)
R> 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])
R> 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])
R> medLength <- round(median(alt.brks[, "nbases"]),
     2)
R> medMarkers <- median(alt.brks[, "nprobes"])
R> sdMarkers <- round(mad(alt.brks[, "nprobes"]),
     2)
R> sdsLength <- round(mad(alt.brks[, "nbases"]),
     2)
R> 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)
R> legend("topleft", fill = poly.cols[c(3, 5)], legend = c("amplifications",
     "deletions"), bty = "n")
R> par(op)
R> gc()
```

```
          used  (Mb) gc trigger  (Mb) max used  (Mb)
Ncells  4133821 220.8    6193578 330.8  5348343 285.7
Vcells 40810390 311.4   94233786 719.0 94232408 719.0
```

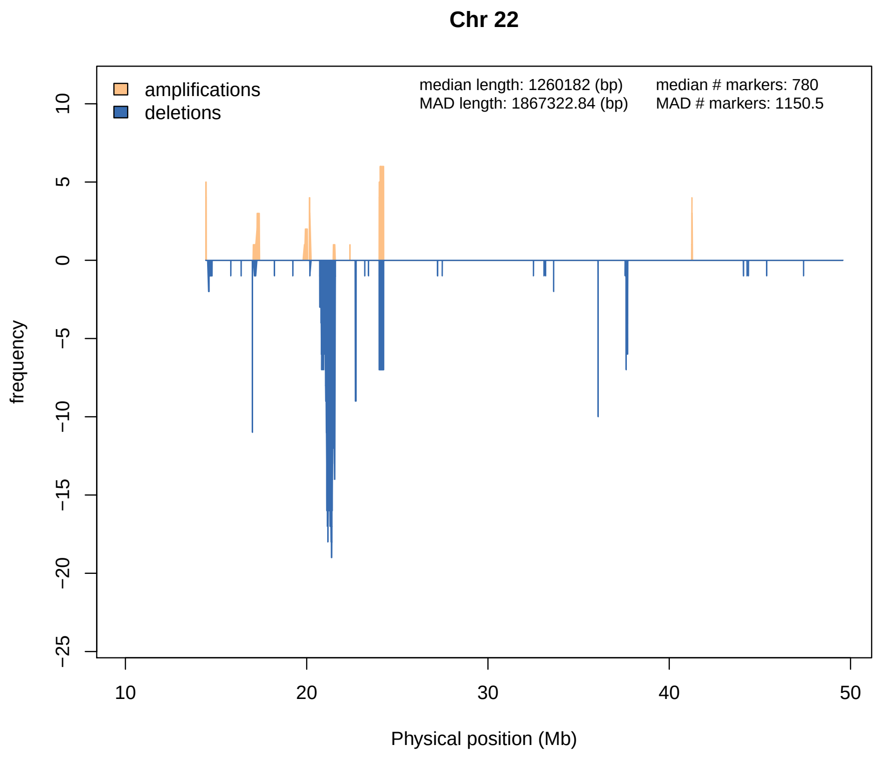

Figure 2: The frequency of amplifications in the hapmap samples is displayed above the zero line. The frequency of hemizygous or homozygous deletions are displayed below the zero line.

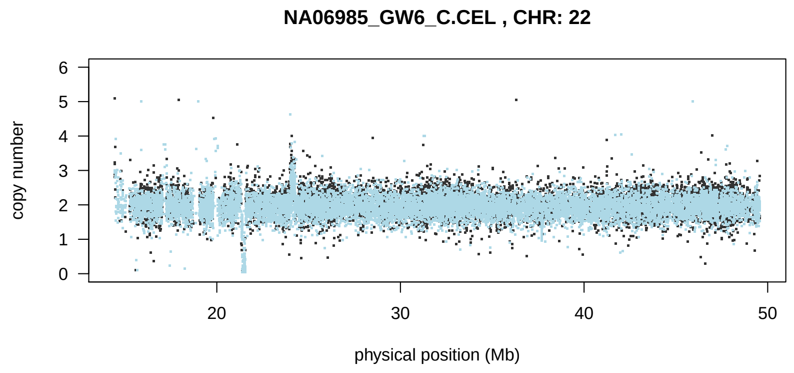

Figure 3: Total copy number (y-axis) for chromosome 22 plotted against physical position (x-axis) for one sample. Estimates at nonpolymorphic loci are plotted in light blue.

One sample at a time: locus-level estimates Figure 3 plots physical position (horizontal axis) versus copy number (vertical axis) for the first sample.

```
R> par(las = 1)
R> plot(position(crlmmSetList), copyNumber(crlmmSetList)[,
     1], pch = ".", 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))))
R> points(position(crlmmSetList)[cnIndex(crlmmSetList)],
     copyNumber(crlmmSetList)[cnIndex(crlmmSetList),
         1], pch = ".", cex = 2, col = "lightblue")
R> axis(1, at = pretty(range(position(crlmmSetList))),
     labels = pretty(range(position(crlmmSetList)))/1e+06)
```

The following code chunk plots the locus-level copy number estimates and overlays the predictions from the hidden markov model.

```
R> ask <- FALSE
R> op <- par(mfrow = c(3, 1), las = 1, mar = c(1,
     4, 1, 1), oma = c(3, 1, 1, 1), ask = ask)
R> cns <- copyNumber(crlmmSetList)
R> cnState <- hmmPredictions - as.integer(1)
R> xlim <- c(10 * 1e+06, max(position(crlmmSetList)))
R> cols <- brewer.pal(8, "Dark2")[1:4]
R> for (j in 1:3) {
     plot(position(crlmmSetList), cnState[, j],
         pch = ".", col = cols[2], xaxt = "n",
         ylab = "copy number", xlab = "Physical position (Mb)",
         type = "s", lwd = 2, ylim = c(0, 6), xlim = xlim)
     points(position(crlmmSetList), cns[, j], pch = ".",
         col = cols[3])
     lines(position(crlmmSetList), cnState[, j],
         lwd = 2, col = cols[2])
     axis(1, at = pretty(position(crlmmSetList)),
         labels = pretty(position(crlmmSetList))/1e+06)
     abline(h = c(1, 3), lty = 2, col = cols[1])
     legend("topright", bty = "n", legend = sampleNames(crlmmSetList)[j])
     legend("topleft", lty = 1, col = cols[2],
         legend = "copy number state", bty = "n",
         lwd = 2)
     plotCytoband(CHR, cytoband.ycoords = c(5,
         5.2), new = FALSE, label.cytoband = FALSE,
         xlim = xlim)
 }
R> par(op)
```

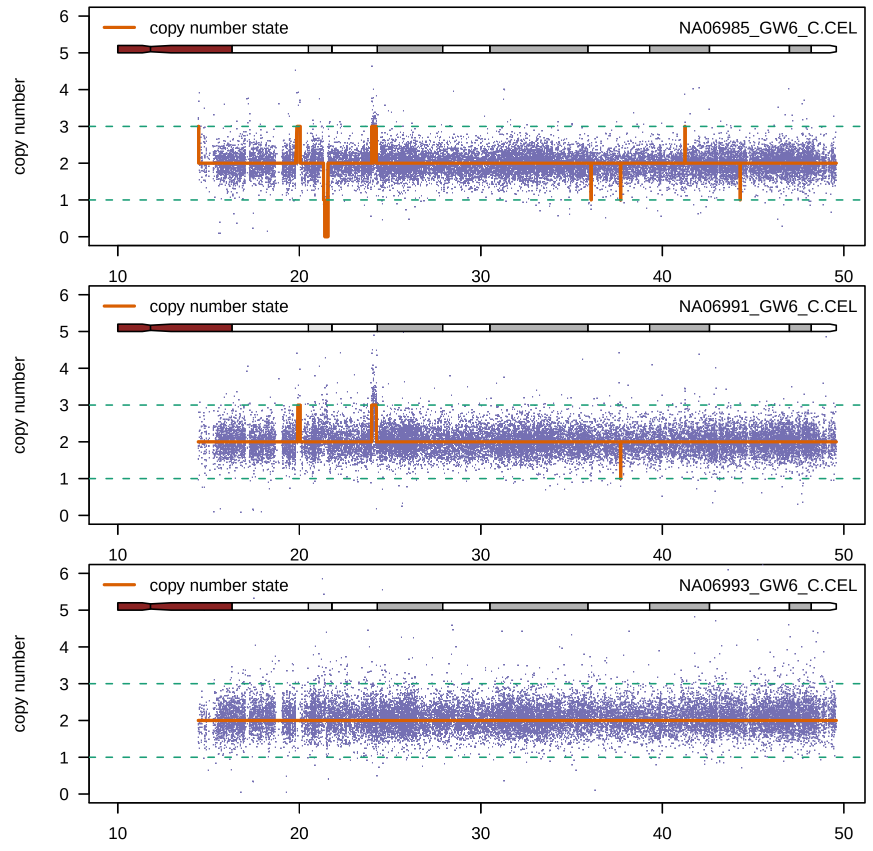

Figure 4: Total copy number (y-axis) for chromosome 22 plotted against physical position (x-axis) for three samples. Estimates at nonpolymorphic loci are plotted in light blue.

One SNP at a time Scatterplots of the A and B allele intensities (log-scale) can be useful for assessing the diallelic genotype calls. The following code chunk is displayed in Figure 5.

```
R> xlim <- ylim <- c(6.5, 13)
R> pch <- 21
R> colors <- c("red", "blue", "green3")
R> cex <- 0.6
R> par(mfrow = c(3, 3), las = 1, pty = "s", ask = FALSE,
     mar = c(2, 2, 2, 2), oma = c(2, 2, 1, 1))
R> indices <- split(snpIndex(crlmmSetList), rep(1:length(snpIndex(crlmmSetList)),
     each = 9, length.out = length(snpIndex(crlmmSetList))))
R> j <- 1
R> 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)
 }
```

4 Details for the copy number estimation procedure

We assume a linear relationship between the normalized intensities and the allele-specific copy number. SNP-specific parameters are estimated only from samples with a suitable confidence score for the diallelic genotype calls. The default confidence threshold is 0.99, but can be adjusted by passing the argument `CONF.THR` to the `update` method.

5 Session information

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

- R version 2.10.0 (2009-10-26), `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, utils

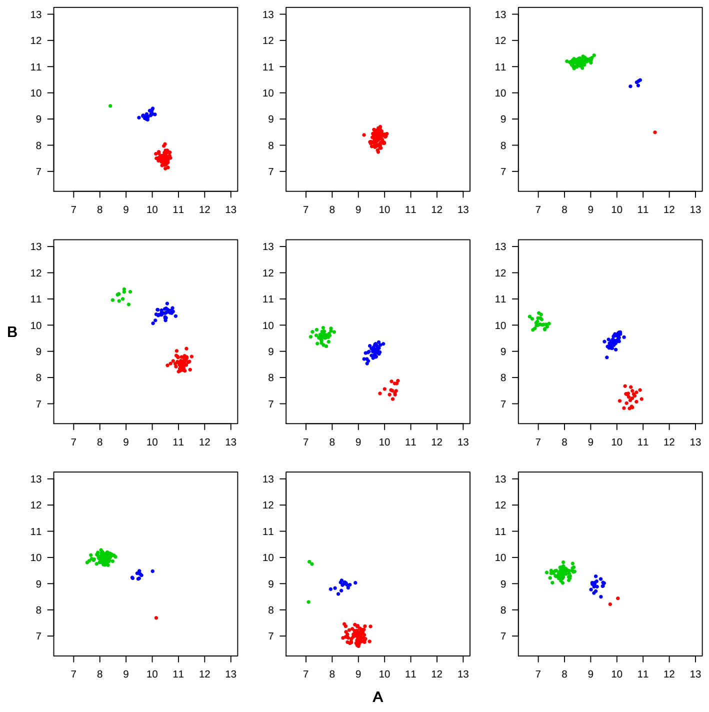

Figure 5: Scatterplots of A versus B intensities. Each panel displays a single SNP.

- Other packages: Biobase 2.6.0, crlmm 1.4.2, DNAcopy 1.20.0, genefilter 1.28.1, genomewidesnp6Crlmm 1.0.2, oligoClasses 1.8.0, RColorBrewer 1.0-2, SNPchip 1.10.0, VanillaICE 1.8.0
- Loaded via a namespace (and not attached): affyio 1.14.0, annotate 1.24.0, AnnotationDbi 1.8.1, Biostrings 2.14.8, DBI 0.2-4, ellipse 0.3-5, IRanges 1.4.8, mvtnorm 0.9-8, preprocessCore 1.8.0, RSQLite 0.7-3, splines 2.10.0, survival 2.35-7, tools 2.10.0, xtable 1.5-6

References

References

[1] Benilton Carvalho, Henrik Bengtsson, Terence P Speed, and Rafael A Irizarry. Exploration, normalization, and genotype calls of high-density oligonucleotide snp array data. *Biostatistics*, 8(2):485–499, Apr 2007.

[2] Joshua M Korn, Finny G Kuruvilla, Steven A McCarroll, Alec Wysoker, James Nemesh, Simon Cawley, Earl Hubbell, Jim Veitch, Patrick J Collins, Katayoon Darvishi, Charles Lee, Marcia M Nizzari, Stacey B Gabriel, Shaun Purcell, Mark J Daly, and David Altshuler. Integrated genotype calling and association analysis of snps, common copy number polymorphisms and rare cnvs. *Nat Genet*, 40(10):1253–1260, Oct 2008.

[3] 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. May 2009.