Copy number estimation

Rob Scharpf

May, 2009

Abstract

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

1 Simple usage

The following packages are required:

```
R> library(crlmm)
R> 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.

```
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/affy"
```

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

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

```
[1] "load.it is TRUE and 'crlmmSetList_<CHR>.rda' objects found. Nothing to do..."
```

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

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

```
 [1] "crlmmSetList_10.rda" "crlmmSetList_11.rda"
 [3] "crlmmSetList_12.rda" "crlmmSetList_13.rda"
 [5] "crlmmSetList_14.rda" "crlmmSetList_15.rda"
 [7] "crlmmSetList_16.rda" "crlmmSetList_17.rda"
```

```
 [9] "crlmmSetList_18.rda" "crlmmSetList_19.rda"
[11] "crlmmSetList_1.rda"  "crlmmSetList_20.rda"
[13] "crlmmSetList_21.rda" "crlmmSetList_22.rda"
[15] "crlmmSetList_23.rda" "crlmmSetList_24.rda"
[17] "crlmmSetList_2.rda"  "crlmmSetList_3.rda"
[19] "crlmmSetList_4.rda"  "crlmmSetList_5.rda"
[21] "crlmmSetList_6.rda"  "crlmmSetList_7.rda"
[23] "crlmmSetList_8.rda"  "crlmmSetList_9.rda"
```

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

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

 Elements in CrlmmSetList object:

class:  ABset
assayData elements:  A B
Dimensions:  23989 15

class:  SnpSet
assayData elements:  call callProbability
Dimensions:  23989 15

class:  CopyNumberSet
assayData elements:  CA CB
Dimensions:  23989 15

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

 Elements in CrlmmSetList object:

class:  ABset
assayData elements:  A B
Dimensions:  23989 15

class:  SnpSet
assayData elements:  call callProbability
Dimensions:  23989 15

class:  CopyNumberSet
```

```
assayData elements:  CA CB
Dimensions:  23989 15
```

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.)

```
R> library(VanillaICE)
R> copyNumberStates <- 0:5
R> 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:

```
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 = 0.01,
         altered2normal = 1, altered2altered = 0.001)
 }
R> table(as.integer(hmmPredictions) - 1)

     0      1      2      3      4      5
   298   1542 357049    774    125     47

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

```
'data.frame':        1211 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 14457129 14805133 14824855 16567216 16568608 16939972 16940248 171
 $ end    : int  14441342 14795579 14805814 16564169 16567216 16939950 16939972 17150515 172
 $ nbases : num  8827 338451 682 1739315 1 ...
 $ nprobes: int  7 62 3 851 1 276 1 95 6 276 ...
 $ state  : int  3 2 3 2 1 2 1 2 3 2 ...
```

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    15
```

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    15
```

For the nonpolymorphic loci:

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

```
[1] 12452    15
```

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 = "")
```

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. 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

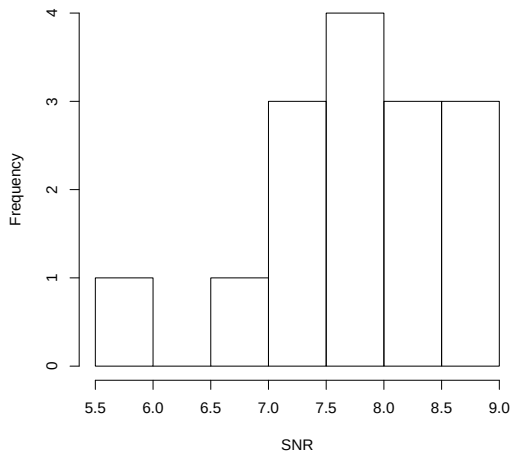

Figure 1: Signal to noise ratios for the HapMap samples.

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  3199894 170.9    4679654 250.0  4679654 250.0
Vcells 16671516 127.2   33050885 252.2 33048207 252.2
```

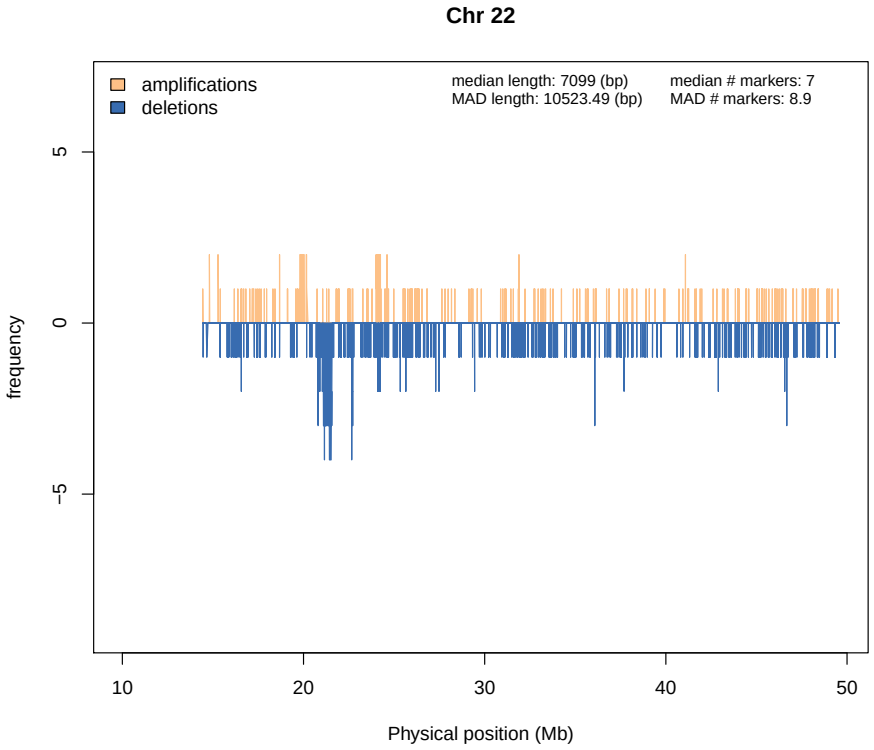

Figure 2: The frequency of amplifications in the hapmap samples is displayed above the zero line. The frequency of hemizygous or homozygous deletions are displayed below the zero line.

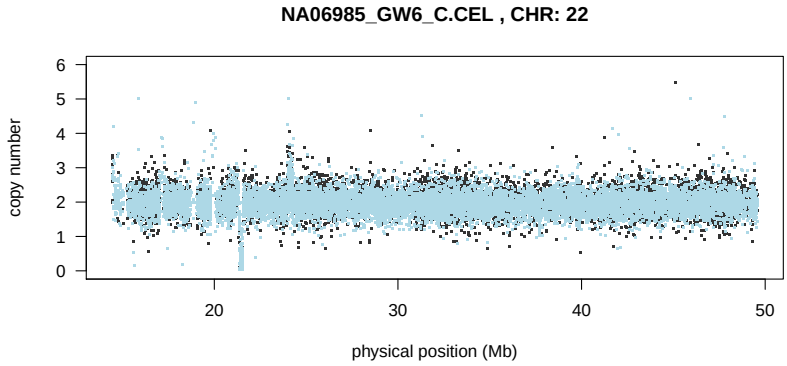

Figure 3: Total copy number (y-axis) for chromosome 22 plotted against physical position (x-axis) for one sample. Estimates at nonpolymorphic loci are plotted in light blue.

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)
```

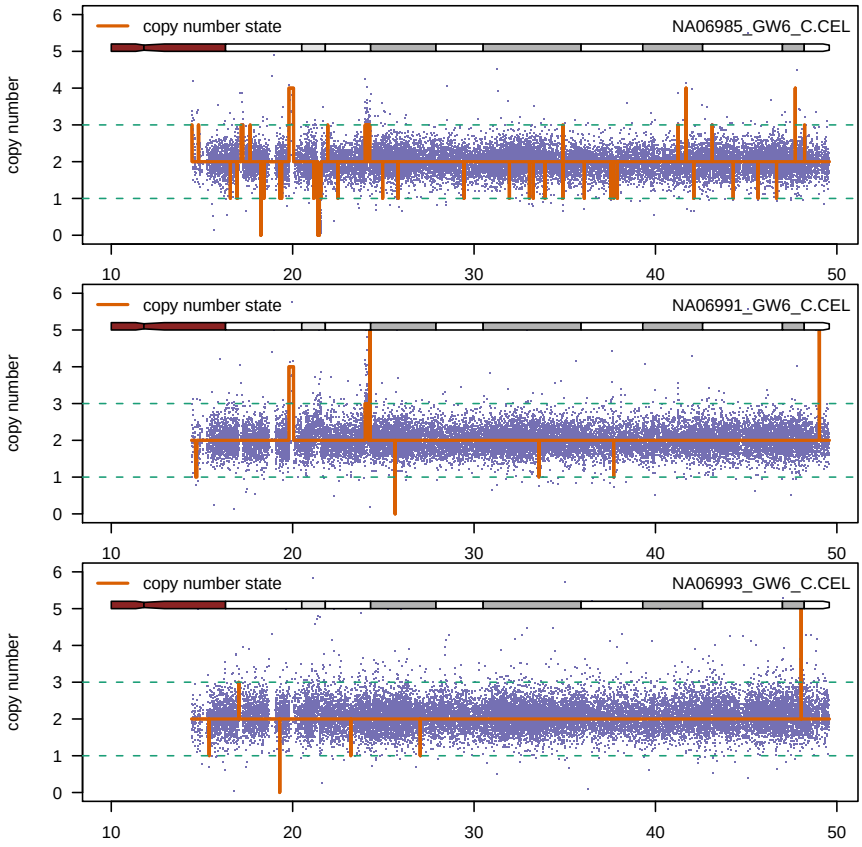

Figure 4: Total copy number (y-axis) for chromosome 22 plotted against physical position (x-axis) for three samples. Estimates at nonpolymorphic loci are plotted in light blue.

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)
 }
```

TODO: 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.

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. TODO: illustrate this approach with boxplots of the A and B intensities stratified by genotype.

5 Session information

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

- R version 2.10.0 Under development (unstable) (2009-08-03 r49060), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.iso885915`, `LC_NUMERIC=C`, `LC_TIME=en_US.iso885915`, `LC_COLLATE=en_US.iso885915`,

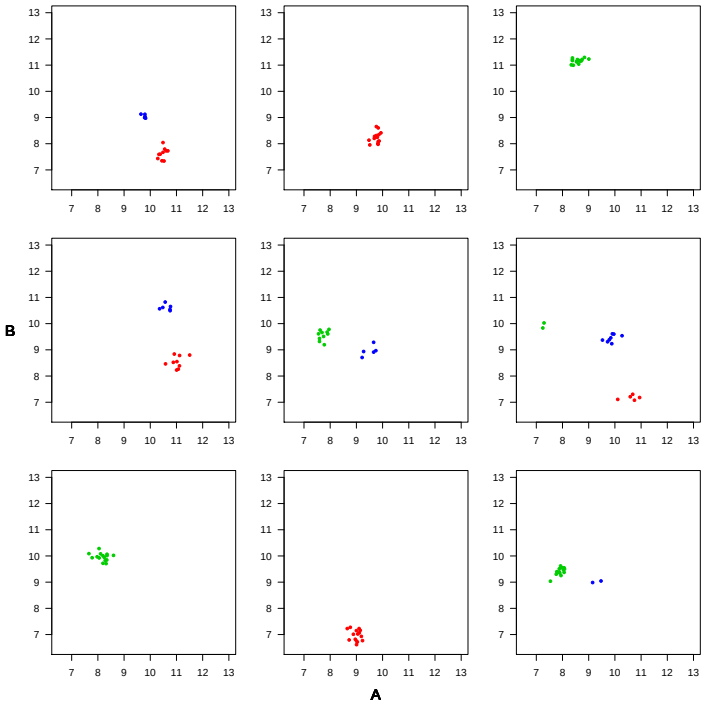

Figure 5: Scatterplots of A versus B intensities. Each panel displays a single SNP.

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
- Other packages: Biobase 2.5.5, crlmm 1.3.19, DNAcopy 1.19.2, genefilter 1.25.2, genomewidesnp6Crlmm 1.0.4, oligoClasses 1.7.10, RColorBrewer 1.0-2, SNPchip 1.9.0, VanillaICE 1.7.8
- Loaded via a namespace (and not attached): affyio 1.13.3, annotate 1.23.1, AnnotationDbi 1.7.8, Biostrings 2.13.30, DBI 0.2-4, ellipse 0.3-5, IRanges 1.3.49, mvtnorm 0.9-7, preprocessCore 1.7.4, RSQLite 0.7-1, splines 2.10.0, survival 2.35-4, tools 2.10.0, xtable 1.5-5

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. May 2009.