

Using **crlmm** for copy number estimation and genotype calling with Illumina platforms

Rob Scharpf

October 20, 2010

Abstract

This vignette illustrates the steps necessary for obtaining marker-level estimates of allele-specific copy number from the raw Illumina IDAT files. Hidden markov models or segmentation algorithms can be applied to the marker-level estimates. Examples of both are illustrated.

1 About this vignette

Allele-specific copy number estimation in the **crlmm** package is available for several Illumina platforms. As described in the **copynumber** vignette, copy number estimation in **crlmm** works best when there are a sufficient number of samples such that AA, AB, and BB genotypes are observed at most loci. For small studies (e.g., fewer than 50 samples), there will be a large number of SNPs that are monomorphic. For monomorphic SNPs, the estimation problem becomes more difficult and alternative strategies that estimate the relative total copy number may be preferable. In addition to installing **crlmm**, one must also install the appropriate annotation package for the Illumina platform. In the following code, we list the platforms for which annotation packages are currently available. Next we create a directory where output files will be stored and indicate the directory that contains the IDAT files that will be used in our analysis.

```
> library(ff)
> pkgs <- annotationPackages()
> pkgs[grep("Crlmm", pkgs)]
[1] "genomewidesnp6Crlmm"      "genomewidesnp5Crlmm"
[3] "human370v1cCrlmm"        "human370quadv3cCrlmm"
[5] "human550v3bCrlmm"        "human650v3aCrlmm"
[7] "human610quadv1bCrlmm"    "human660quadv1aCrlmm"
[9] "human1mduov3bCrlmm"      "humanomni1quadv1bCrlmm"

> if (getRversion() < "2.12.0") {
  rpath <- getRversion()
} else rpath <- "trunk"
> outdir <- paste("/thumper/ctsa/snpmicroarray/rs/ProcessedData/crlmm/",
  rpath, "/illumina_vignette", sep = "")
> datadir <- "/thumper/ctsa/snpmicroarray/illumina/IDATS/370k"

> ldPath(outdir)
> ocProbesets(150000)
> ocSamples(200)
```

This vignette was created using Illumina IDAT files that are located in a specific directory on my computer (**pathToCels**). Long computations are saved in the output directory **outdir**. Towards this end, we make repeated use of the **checkExists** – an ad-hoc approach for storing long computations. This function simply checks that a variable is in the workspace. If not, the variable is loaded from the indicated path. If the file

(with '.rda' extension) does not exist in this path, the function indicated by the .FUN argument is executed. Alternatively, if .load.it is FALSE the function will be executed regardless of whether the file exists. Users should see the `weaver` package for a more 'correct' approach for cacheing long computations. The following code chunk checks that that the variables specific to the directory structure on my computer exist. Users should modify the `outdir` and `datadir` variables as appropriate.

```
> if (!file.exists(outdir)) stop("Please specify valid directory for storing output")
> if (!file.exists(datadir)) stop("Please specify the correct path to the CEL files")
```

2 Preprocessing Illumina IDAT files, genotyping, and copy number estimation

To perform copy number analysis on the Illumina platform, several steps are required. The first step is to read in the IDAT files and create a container for storing the red and green intensities. These intensities are quantile normalized in the function `crlmmIllumina`, and then genotyped using the `crlmm` algorithm. Details on the `crlmm` genotyping algorithm are described elsewhere. We will make use of the normalized intensities when we estimate copy number. The object returned by the `genotype.Illumina` function is an instance of the `CNSet` class, a container for storing the genotype calls, genotype confidence scores, the normalized intensities for the A and B channels, and summary statistics on the batches. The batch variable can be the date on which the array was scanned or the 96 well microarray chemistry plate on which the samples were processed. Both date and chemistry plate are useful surrogates for experimental factors that effect subgroups of the probe intensities. (Quantile normalization does not remove batch effects.) The genotype confidence scores are saved as an integer, and can be converted back to a [0, 1] probability scale by the transformation $round(-1000 * \log_2(1 - p))$.

```
> samplesheet = read.csv(file.path(datadir, "HumanHap370Duo_Sample_Map.csv"),
  header = TRUE, as.is = TRUE)
> samplesheet <- samplesheet[-c(28:46, 61:75, 78:79), ]
> arrayNames <- file.path(datadir, unique(samplesheet[,
  "SentrixPosition"]))
> grnfiles = all(file.exists(paste(arrayNames, "_Grn.idat",
  sep = "")))
> redfiles = all(file.exists(paste(arrayNames, "_Red.idat",
  sep = "")))
> load.it <- TRUE
> if (!load.it) {
  rmFiles <- list.files(outdir, pattern = ".ff", full.names = TRUE)
  unlink(rmFiles)
}
> container <- checkExists("container", .path = outdir,
  .FUN = crlmm:::genotype.Illumina, sampleSheet = samplesheet,
  path = dirname(arrayNames[1]), arrayInfoColNames = list(barcode = NULL,
  position = "SentrixPosition"), cdfName = "human370v1c",
  .load.it = load.it)
> cnSet <- checkExists("cnSet", .path = outdir, .FUN = crlmmCopynumber,
  object = container, .load.it = load.it)
```

The `cnSet` returned by the `crlmmCopynumber` function contains all of the parameters used to compute allele-specific copy number. In order to reduce I/O and speed computation, the allele-specific copy number estimates are not written to file nor saved in the `CNSet` object. Rather, the estimates are computed on the fly when the user calls one of the helper functions that uses this information. For instance, often it is of interest to smooth the total copy number estimates (the sum of the allele-specific copy number) as a function of the physical position. The following code chunk can be used to create an instance of `CopyNumberSet` that contains the CA + CB at polymorphic markers and 'CA' at nonpolymorphic markers.

```

> marker.index <- which(chromosome(cnSet) <= 22)
> sample.index <- 1:5
> invisible(open(cnSet))
> copynumberSet <- as(cnSet[marker.index, 1:5], "CopyNumberSet")
> invisible(close(cnSet))
> copynumberSet <- copynumberSet[order(chromosome(copynumberSet),
  position(copynumberSet)), ]
> indices <- split(1:nrow(copynumberSet), chromosome(copynumberSet))
> dup.index <- unlist(sapply(indices, function(i, position) i[duplicated(position[i])],
  position = position(copynumberSet)))
> if (length(dup.index) > 0) copynumberSet <- copynumberSet[-dup.index,
  ]
> missing.index <- which(rowSums(is.na(copyNumber(copynumberSet))) ==
  ncol(copynumberSet))
> copynumberSet <- copynumberSet[-missing.index, ]

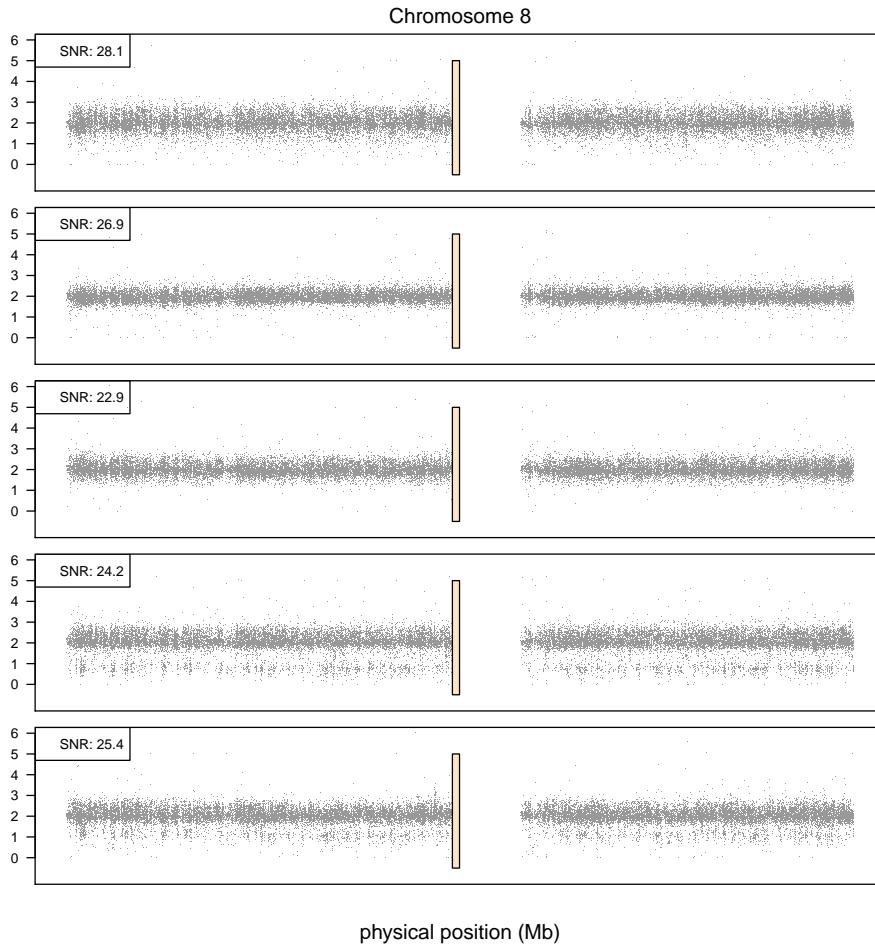
```

We suggest plotting the total copy number as a function of the physical position to evaluate whether a wave-correction is needed.

```

> require(SNPchip)
> data(chromosomeAnnotation)
> chrAnn <- chromosomeAnnotation[1, ]
> layout(matrix(1:5, 5, 1), heights = c(1, 1, 1, 1, 1))
> par(mar = c(0.5, 0.5, 0.5, 0.5), las = 1, oma = c(4,
  4, 4, 4))
> i <- which(chromosome(copynumberSet) == 1)
> for (j in 1:5) {
  plot(position(copynumberSet)[i], copyNumber(copynumberSet)[i,
    j], pch = ".", cex = 0.6, col = "grey60", xaxt = "n",
    ylim = c(-1, 6), ylab = "total copy number",
    yaxt = "n", xlim = c(0, max(position(copynumberSet)[i])))
  legend("topleft", legend = paste("SNR:", round(copynumberSet$SNR[j],
    1)))
  axis(2, at = 0:6, labels = 0:6)
  if (j == 5) {
    mtext("physical position (Mb)", 1, outer = TRUE,
      line = 2)
    mtext("Chromosome 8", 3, line = 0, outer = TRUE)
    at <- pretty(position(copynumberSet)[i], n = 8)
  }
  xx <- c(chrAnn[1:2], chrAnn[2:1])
  yy <- c(-0.5, -0.5, 5, 5)
  polygon(xx, yy, col = "bisque")
}

```



3 Smoothing marker-level estimates of total copy number

In this section, we show how the total copy number can be smoothed using either the hidden Markov model implemented in the R package `VanillaICE` or circular binary segmentation implemented in the R package `DNAcopy`.

Smoothing via a hidden Markov model.

```
> if (require(VanillaICE)) {
  hmmOpts <- hmm.setup(copynumberSet, c("hom-del",
    "hem-del", "normal", "amp"), copynumberStates = 0:3,
    normalIndex = 3, log.initialP = rep(log(1/4),
    4))
  timing <- system.time(fit.cn <- hmm(copynumberSet,
    hmmOpts, verbose = FALSE))
  hmm.df <- as.data.frame(fit.cn)
  print(hmm.df[1:5, c(2:4, 7:9)])
}
      start       end     width state numMarkers LLR
1 7764 52495853 52488090      3      6427    0
2 18285782 100320863 82035082      3      10394    0
```

```

3      835  35141900 35141066      3      4376   0
4    11799  26255797 26243999      3      4162   0
5 28043606  62426157 34382552      3      4383   0

```

Smoothing via a hidden Markov model.

```

> library("DNAcopy")
> CNA.object <- CNA(genomdat = copyNumber(copynumberSet),
+                      chrom = chromosome(copynumberSet), maploc = position(copynumberSet),
+                      data.type = "logratio", sampleid = sampleNames(copynumberSet))
> smu.object <- smooth.CNA(CNA.object)
> if (!load.it) unlink(file.path(outdir, "cbs.segments.rda"))
> cbs.segments <- checkExists("cbs.segments", .path = outdir,
+                               .FUN = segment, x = smu.object, .load.it = load.it)
> cbs.segments <- cbind(cbs.segments$output, cbs.segments$segRows)
> cbs.segments$call <- rep(3, nrow(cbs.segments))
> cbs.segments$call[cbs.segments$seg.mean > 2.5] <- 4
> cbs.segments$call[cbs.segments$seg.mean < 1.25 & cbs.segments$seg.mean >
+ 0.75] <- 2
> cbs.segments$call[cbs.segments$seg.mean < 0.75] <- 1
> cbs.ir <- RangedData(IRanges(cbs.segments$loc.start,
+                                cbs.segments$loc.end), chrom = cbs.segments$chrom,
+                                numMarkers = cbs.segments$num.mark, seg.mean = cbs.segments$seg.mean,
+                                cnCall = cbs.segments$call, sampleId = substr(cbs.segments$ID,
+                                2, 13))

```

Visualizing inferred CNV. In the following code chunk, we plot the total copy number versus physical position for chromosome 1 and overlay the copy number predictions from the HMM and a CNV call from the segmentation obtained by thresholding the segment means. We restrict our focus to regions that have 3 or more markers.

```

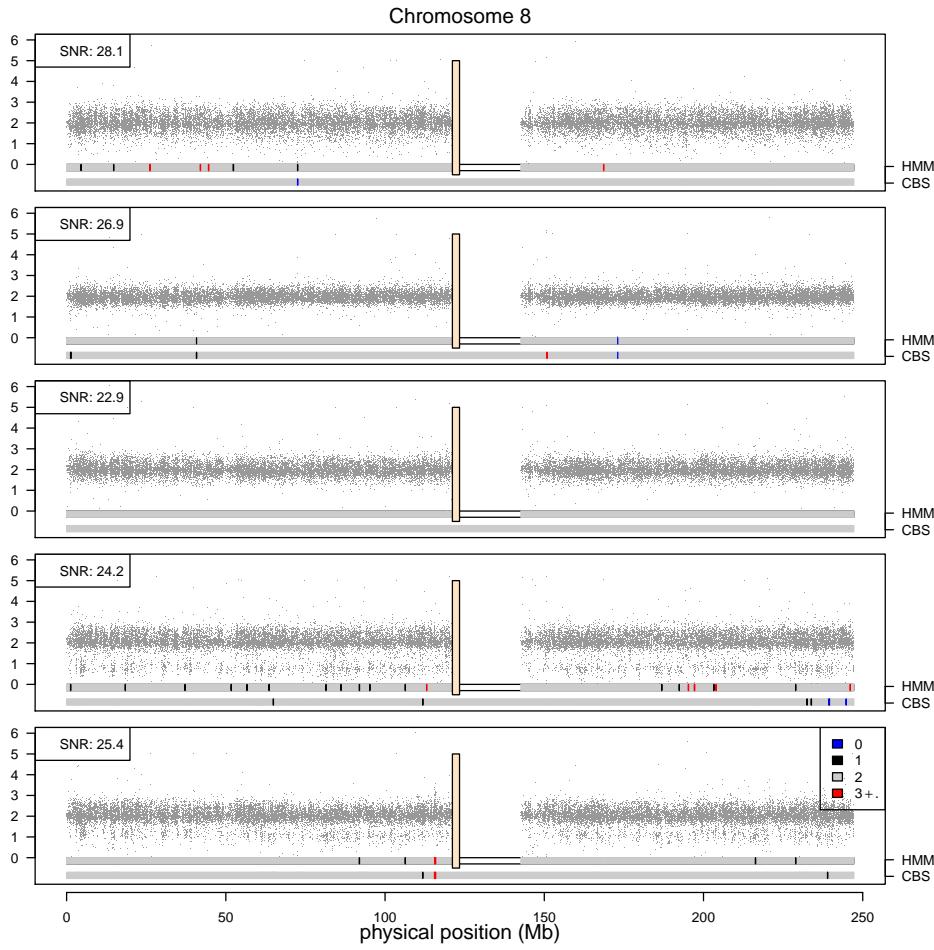
> require(SNPchip)
> CHR <- 1
> cnSet2 <- copynumberSet
> fit.cn2 <- fit.cn[fit.cn$numMarkers >= 3, ]
> cbs.ir2 <- cbs.ir[cbs.ir$numMarkers >= 3, ]
> copynumberSet2 <- copynumberSet[which(chromosome(copynumberSet) ==
+ CHR), ]
> layout(matrix(1:5, 5, 1), heights = c(1, 1, 1, 1, 1))
> par(mar = c(0.5, 0.5, 0.5, 0.5), las = 1, oma = c(4,
+ 4, 4, 4))
> for (j in 1:5) {
+   fit.cn3 <- fit.cn2[fit.cn2$chrom == CHR & fit.cn2$sampleId ==
+     sampleNames(copynumberSet2)[j], ]
+   cbs.ir3 <- cbs.ir2[cbs.ir2$chrom == CHR & cbs.ir2$sampleId ==
+     sampleNames(copynumberSet2)[j], ]
+   plot(position(copynumberSet2), copyNumber(copynumberSet2)[,
+     j], pch = ".", cex = 0.6, col = "grey60", xaxt = "n",
+     ylim = c(-1, 6), ylab = "total copy number",
+     yaxt = "n", xlim = c(0, max(position(copynumberSet2))))
+   legend("topleft", legend = paste("SNR:", round(copynumberSet2$SNR[j],
+     1)))
+   axis(2, at = 0:6, labels = 0:6)
+   if (j == 5) {

```

```

mtext("physical position (Mb)", 1, outer = TRUE,
      line = 2)
mtext("Chromosome 8", 3, line = 0, outer = TRUE)
at <- pretty(position(copynumberSet2))
}
w <- width(fit.cn3)
fit.cn3 <- fit.cn3[order(w, decreasing = TRUE), ]
col <- c("blue", "black", "grey80", "red")
x <- c(min(start(fit.cn3)), max(end(fit.cn3)))
xx <- c(x, rev(x))
y <- rep(c(-0.3, 0), each = 2)
polygon(xx, y, col = "white")
for (i in seq_along(w)) {
  x <- c(start(fit.cn3)[i], end(fit.cn3)[i])
  xx <- c(x, rev(x))
  statecolor <- col[fit.cn3$state[i]]
  polygon(xx, y, col = statecolor, border = statecolor)
}
w <- width(cbs.ir3)
cbs.ir3 <- cbs.ir3[order(w, decreasing = TRUE), ]
y <- rep(c(-1, -0.7), each = 2)
for (i in seq_along(w)) {
  x <- c(start(cbs.ir3)[i], end(cbs.ir3)[i])
  xx <- c(x, rev(x))
  statecolor <- col[cbs.ir3$cnCall[i]]
  polygon(xx, y, col = statecolor, border = statecolor)
}
axis(4, at = c(-0.9, -0.1), c("CBS", "HMM"), cex = 0.8)
if (j == 5) {
  legend("topright", fill = col, legend = c(0,
                                             1, 2, expression(3 + .)))
  labels <- at/1e+06
  axis(1, at = at, labels = labels, outer = T,
        line = 0)
}
xx <- c(chrAnn[1:2], chrAnn[2:1])
yy <- c(-0.5, -0.5, 5, 5)
polygon(xx, yy, col = "bisque")
}

```



TODO: Samples 4 and 5 have a sparse band of CN-estimates near 1 that is uniformly distributed across chromosome 1. This suggests markers that are either not well fit by the linear model, or a problem with the genotyping. Need to check.

4 Accessors for extracting the locus-level copy number estimates.

As an example of how to use accessors to obtain the allele-specific CN estimates, the following code chunk extracts the allele-specific copy number for polymorphic markers on chromosome 21.

```
> marker.index <- which(chromosome(cnSet) == 21 & isSnp(cnSet))
> ca <- CA(cnSet, i = marker.index)
> cb <- CB(cnSet, i = marker.index)
> missing.index <- which(rowSums(is.na(ca)) == ncol(cnSet))
> ca <- ca[-missing.index, ]
> cb <- cb[-missing.index, ]
```

Negating the `isSnp` function could be used to extract the estimates at nonpolymorphic markers. For instance,

```
> cn.monomorphic <- CA(cnSet, i = which(chromosome(cnSet) ==
  21 & !isSnp(cnSet)))
```

At polymorphic loci, the total copy number is the sum of the number of copies of the A allele and the number of copies for the B allele. At nonpolymorphic loci, the total copy number is the number of copies

for the A allele. The helper function `totalCopyNumber` can be used to extract the total copy number for all polymorphic and nonpolymorphic markers. Documentation of the `totalCopyNumber` will be available in the devel version of the `oligoClasses`.

```
> cn.total <- ca + cb
> cn.total2 <- totalCopyNumber(cnSet, i = marker.index)
> cn.total2 <- cn.total2[-missing.index, ]
> all.equal(cn.total2, cn.total)

[1] TRUE
```

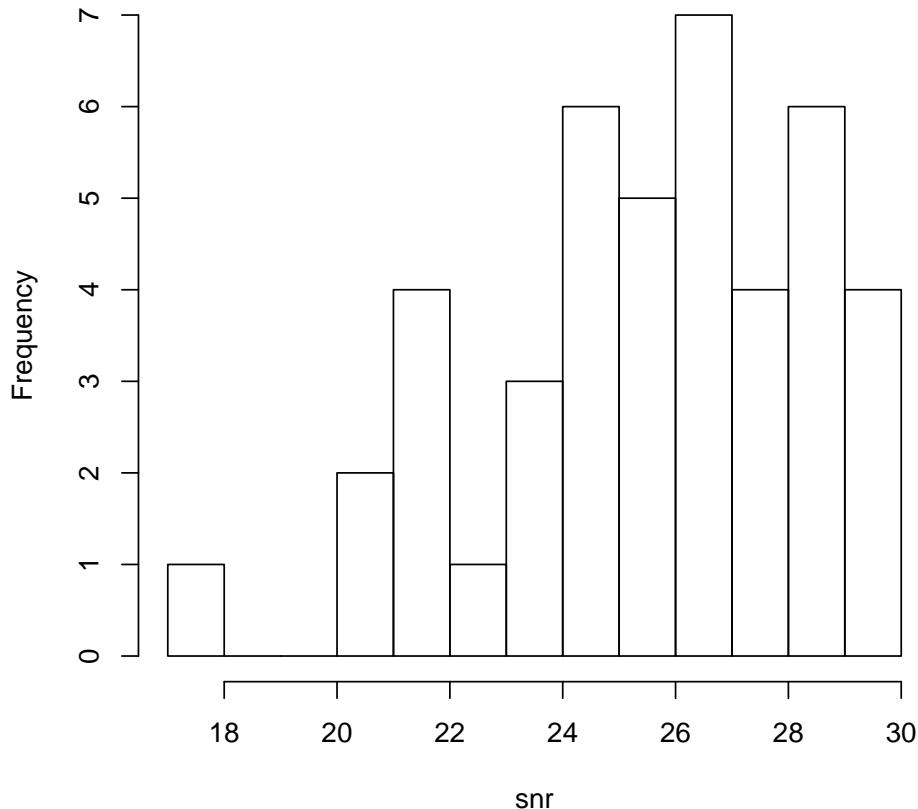
A few simple visualizations may be helpful at this point. The first plot is a histogram of the signal to noise ratio of the sample – an overall measure of how well the genotype clusters separate. (This statistic tends to be much higher for Illumina than for the Affymetrix platforms.) The second is a visualization of the total copy number estimates plotted versus physical position on chromosome 1 for the two samples with the lowest (top) and highest (bottom) signal to noise ratios.

```
> open(cnSet$SNR)

[1] FALSE

> snr <- cnSet$SNR[, ]
> hist(snr, breaks = 15)
```

Histogram of snr



```

> low.snr <- which(snr == min(snr))
> high.snr <- which(snr == max(snr))
> x <- position(cnSet)[marker.index]
> x <- x[-missing.index]
> par(mfrow = c(2, 1), las = 1, mar = c(0.5, 4, 0.5, 0.5),
      oma = c(4, 1, 1, 1))
> for (j in c(low.snr, high.snr)) {
  cn <- cn.total[, j]
  cn[cn < 0.05] <- 0.05
  plot(x, cn, pch = ".", ylab = "copy number", xaxt = "n",
        ylim = c(0, 6))
}
> axis(1, at = pretty(x), labels = pretty(x/1e+06))
> mtext("Mb", 1, outer = TRUE, line = 2)

```

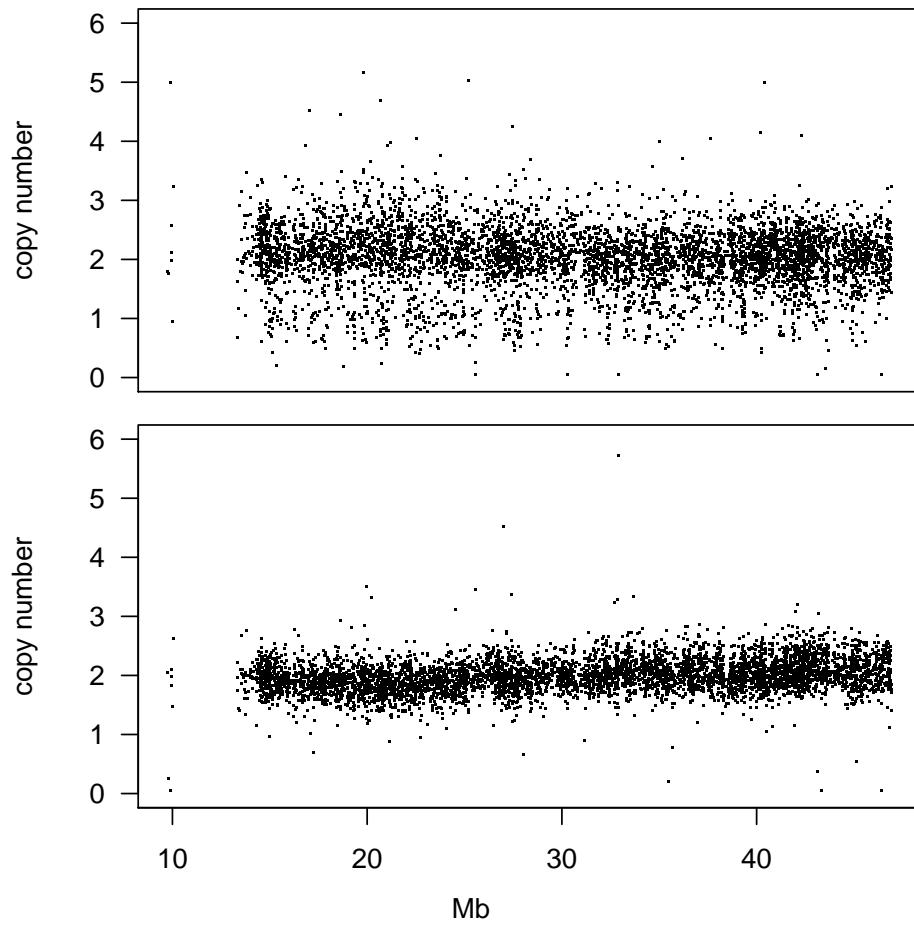


Figure 1: Copy number estimates for samples with low signal to noise ratio (top) and high signal to noise ratio (bottom). Note the slight wave in the bottom figure. Such waves are typically correlated with GC-content and can be removed by scatterplot smoothers or any of the published wave-correction tools for genotyping arrays.

5 Session information

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

- R version 2.13.0 Under development (unstable) (2010-10-20 r53379), 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 2.9.1, bit 1.1-4, crlmm 1.9.1, DNAcopy 1.23.8, ff 2.1-4, IRanges 1.7.37, oligoClasses 1.11.11, SNPchip 1.13.0, VanillaICE 1.13.0
- Loaded via a namespace (and not attached): affyio 1.17.4, annotate 1.29.0, AnnotationDbi 1.13.0, Biostrings 2.19.0, DBI 0.2-5, ellipse 0.3-5, genefilter 1.33.0, mvtnorm 0.9-92, preprocessCore 1.13.0, RSQLite 0.9-2, splines 2.13.0, survival 2.35-8, xtable 1.5-6