Detection of de novo copy number alterations in case-parent trios using the R package MinimumDistance

Moiz Bootwalla and Rob Scharpf

September 30, 2014

Abstract

For the analysis of case-parent trio genotyping arrays, copy number variants (CNV) appearing in the offspring that differ from the parental copy numbers are often of interest (de novo CNV). This package defines a statistic, referred to as the minimum distance, that can be useful for identifying de novo copy number alterations in the offspring. We smooth the minimum distance using the circular binary segmentation algorithm implemented in the Bioconductor package DNAcopy. Trio copy number states are inferred from the maximum a posteriori probability of the segmented data, incorporating information from the log R ratios and B allele frequencies. As both log R ratios and B allele frequencies can be estimated from Illumina and Affymetrix arrays, this package supports de novo copy number inference in both platforms.

1 Introduction

There are numerous R packages available from Bioconductor for smoothing copy number alterations. For example, the biocview `CopyNumberVariants` in the 2.9 release of Bioconductor lists 27 packages. For the analysis of germline diseases, hidden Markov models have emerged as a popular tool as inference regarding the latent copy number state incorporates information from both the estimates of copy number (or relative copy number) and the allelic frequencies, such as the B allele frequency [4] or genotype calls [1, 7, 5]. For the analysis of somatic cell diseases such as cancer, algorithms that segment the genome into regions of constant copy number (referred to here as segmentation algorithms) may be more preferable for the detection of copy number aberrations (CNA) as a mixture of cells with different copy numbers give rise to mosaic (non-integer) copy numbers. Examples include circular binary segmentation implemented in the R package DNAcopy and the GLAD, both of which were orginally developed for array CGH platforms [3, 2, 6]. One disadvantage of segmentation algorithms is that inference regarding duplication and deletions is not directly available.

More recently, HMMs and segmentation algorithms have been developed for inferring common regions of copy number alterations in multiple samples. However, relatively few algorithms are available for inferring copy number alterations, especially de novo copy number alterations, in family-based study designs involving case-parent trios. Instead, a common strategy has been a two-step approach of identifying copy number alterations in the individual samples and then comparing the results across samples to infer whether an alteration observed in the offspring is de novo. A disadvantage of the two-step approach is that unadjusted sources of technical variation, such as waves, can contribute to false positives. To our knowledge, the joint HMM implemented in the PennCNV software is the only software to date that provides direct inference regarding de novo alterations in case parent study designs.

This package develops an alternative framework for inferring regions of de novo copy number alterations in case-parent trios. Like PennCNV, inference regarding de novo alterations integrates information from both the log R ratios and B allele frequencies. Differences in the two approaches are most apparent in non-HapMap experimental datasets in which technical and experimental sources of variation appear to contribute to a large number of false positives.

This vignette describes the analysis pipeline from preprocessed and normalized estimates of copy number and allele frequencies to inference regarding de novo copy number alterations. The workflow is illustrated using publicly available HapMap trios assayed on the Illumina 610quad array. Section 3 outlines two approaches for processing the data: Section 3.1 describes a pipeline in which a list of files are provided as input and the output is a RangedDataCBS object of genomic intervals and the inference for the trio copy number state; Section 3 provides a more detailed workflow in which intermediate data structures are created to facilitate visualization of the low-level copy number summaries along with inference from the MinimumDistance algorithm. Details regarding the intermediate data structures and visualization are described in Sections 3 and 3.2, respectively. Where possible, we have adopted standard data structures for encapsulating the low-level data (eSet extensions) and the segmented data (RangedData extensions).

2 Large data applications and parallel processing

A separate vignette for using MinimumDistance with large datasets is forthcoming. For now, we remark that the discussion regarding the infrastructure for large data support and parallelization in the R package oligoClasses is similar. In particular, to use a cluster we check for three requirements: 1) ’ff’ is loaded; 2) ’snow’ is loaded; and 3) the ’cluster’ option is set. For example, in the following unevaluated code chunk would enable large data support and parallelization with 22 worker nodes. Finally, a check that the three requirements are satisfied is obtained with the function parStatus in the oligoClasses package.

```
R> library(snow)
R> library(doSNOW)
R> library(oligoClasses)
R> cl <- makeCluster(22, type="SOCK")
R> registerDoSNOW(cl)
R> parStatus()
```

Had we executed the preceding code chunk, the output from this vignette would be more or less the same. However, the containers for storing the low level copy number summaries and allele frequencies would contain ff-objects and many of the chromosome-specific tasks that can easily be parallelized such as segmentation and the posterior calling steps would be split to multiple workers.

When parallelization is not enabled, one may want to ’register’ the backend so that annoying warning messages do not appear during processing. This is accomplished using the function registerDoSEQ in the foreach package.

```
R> library(SNPchip)
R> library(GenomicRanges)
R> library(oligoClasses)
R> library(MinimumDistance)
R> foreach::registerDoSEQ()
```

3 Detection of de novo copy number

There are two options for processing. Section provides a simple pipeline for reading a list of files containing log R ratios and B allele frequencies and returning the segmented data with the trio copy number calls. The second option in Section provides a more detailed workflow of de novo copy number analysis with intermediate data structures useful for storing the log R ratios and B allele frequencies in a manner than can be easily accessed for subsequent plotting. The latter can be particularly useful for visual inspection of the trio copy number calls.

3.1 Simple Usage: generating ranged data with posterior calls

Here, we illustrate a simple pipeline for reading in BeadStudio files and returning a list of genomic ranges. There are 3 BeadStudio files provided with the MinimumDistance package – one for the father (“F.txt”), mother (“M.txt”), and offspring (“O.txt”). In order to keep the size of the MinimumDistance package small, these files contain data for only 1000 markers.

```
R> path <- system.file("extdata", package="MinimumDistance")
R> fnames <- list.files(path, pattern=".txt")
R> fnames

[1] "F.txt"  "M.txt"  "O.txt"  "O1.txt"
```

Before reading the data, we create a `data.frame` indicating the filenames for the father, mother, and offspring. In the following codechunk, we have a single file for the father, mother, and offspring (one trio). For processing multiple trios, the arguments to `data.frame` can be vectors where the ith element in each vector are the files for one trio.

```
R> pedigreeInfo <- data.frame(F="F.txt", M="M.txt", O="O.txt")
```

Note that the sample identifiers for father and mother can be duplicated, but the offspring sample identifiers must be unique. We encapsulate this information more formally in an S4 class called *Pedigree* and use the `Pedigree` constructor to generate an instance of the class:

```
R> ped <- Pedigree(pedigreeInfo)
R> ped

      F     M     O
1 F.txt M.txt O.txt

Slot "pedigreeIndex":
  individualId memberId index.in.pedigree
1        F.txt        F                 1
2        M.txt        M                 1
```

Instantiating a `TrioSetList` object .

A key design consideration for data structures that capture information on the low-level copy number and allele frequencies is that the 'unit' of our analysis is a trio. A common query is to access the low level statistical summaries for a set of markers for all members in a trio. By organizing statistical summaries in arrays with dimension `feature x trios x family member` such queries are straightforward. As the number of trios in a typical genome-wide association study may exceed 1000, it may be impractical to store the low-level summaries in a single array. To make the size of the arrays more manageable and to facilitate parallelization, we organize the low-level summaries by chromosome using the *TrioSetList* class. The *TrioSetList* class has a slot for storing an object of class `Pedigree` (discussed previously), as well as slots for storing sample-level covariates on the offspring (`phenoData`), mother (`motherPhenoData`), and the father (`fatherPhenoData`). Often technical, experimental, and phenotypic covariates for the samples in an experiment are stored in a single tab-delimited file in which columns are covariates and rows are samples. Such a spreadsheed can be read into R as a `data.frame`. Supplying such a `data.frame` to the argument `sample.sheet` in the `TrioSetList` constructor function will build `AnnotatedDataFrame` objects for father, mother, and offspring, extracting the appropriate rows (samples) from the `data.frame`. For example, in the following codechunk we load a `data.frame` called 'sample.sheet' and pass the `sample.sheet` object to the `TrioSetList` constructor. In addition, we load a matrix of log R ratios and a matrix of B allele frequencies. The matrices for the log R ratios and B allele frequences contain only 25 markers for each chromosome as the intent is to illustrate the steps required for initializing a data structure of the `TrioSetList` class. Using the information stored in the `Pedigree` class, the matrices are transformed to 3-dimensional arrays and stored in the `assayDataList` slot of the *TrioSetList* class. It is important that the column identifiers for the log R ratio and B allele frequency matrices correspond to sample identifiers in the `Pedigree` object. Accessors `lrr` and `baf` are available for extracting the list of arrays for the log R ratios and B allele frequencies.

```
R> library(human610quadv1bCrlmm)
R> path <- system.file("extdata", package="MinimumDistance")
R> load(file.path(path, "pedigreeInfo.rda"))
```

```
R> load(file.path(path, "sample.sheet.rda"))
R> load(file.path(path, "logRratio.rda"))
R> load(file.path(path, "baf.rda"))
R> stopifnot(colnames(logRratio) %in% allNames(Pedigree(pedigreeInfo)))
R> nms <- paste("NA",substr(sample.sheet$Sample.Name, 6, 10),sep="")
R> trioSetList <- TrioSetList(lrr=logRratio, ## must provide row.names
                              baf=baf,
                              pedigree=Pedigree(pedigreeInfo),
                              sample.sheet=sample.sheet,
                              row.names=nms,
                              cdfname="human610quadv1bCrlmm",
                              genome="hg18")
R> show(trioSetList)

TrioSetList of length 22
genome: hg18
```

Segmentation and posterior calls. To keep the size of the MinimumDistance package small, the `trioSetList` object created in the previous section contained only 25 markers for each of the 22 autosomes. While useful for illustrating construction of the *TrioSetList* class, there are too few markers included in the example to illustrate the smoothing and posterior calling algorithm for detecting de novo copy number events. To demonstrate these steps, we load a `TrioSetList` object containing several thousand markers for chromosomes 7 and 22 for 2 HapMap trios:

```
R> data(trioSetListExample)
```

For a given trio, the signed minimum absolute difference of the offspring and parental $\log_2$ R ratios ($\boldsymbol{r}$) is defined as

$$\boldsymbol{d} \quad \equiv \quad (\boldsymbol{r}_{\mathrm{O}} - \boldsymbol{r}_{\mathrm{M}}) \times \mathbb{I}_{[|\boldsymbol{r}_{\mathrm{O}}-\boldsymbol{r}_{\mathrm{F}}|>|\boldsymbol{r}_{\mathrm{O}}-\boldsymbol{r}_{\mathrm{M}}|]} + (\boldsymbol{r}_{\mathrm{O}} - \boldsymbol{r}_{\mathrm{F}}) \times \mathbb{I}_{[|\boldsymbol{r}_{\mathrm{O}}-\boldsymbol{r}_{\mathrm{F}}|\leq|\boldsymbol{r}_{\mathrm{O}}-\boldsymbol{r}_{\mathrm{M}}|]} \,. \qquad (1)$$

Computation of $\boldsymbol{d}$ by the function `calculateMindist` is vectorized in R:

```
R> md <- calculateMindist(lrr(trioSetList))
```

The circular binary segmentation algorithm implemented in the R package DNAcopy can be used to smooth the minimum distance estimates. The method `segment2` is a wrapper to the `CNA.smooth` and `segment` functions defined in DNAcopy. Additional arguments to the `segment` function can be passed through the `...` argument in `segment2`, allowing users to modify the segmentation from the default values in DNAcopy.

```
R> md.segs <- segment2(object=trioSetList, md=md)
```

The object returned by the `segment2` method is an object of class `RangedDataCBS`.

```
R> head(md.segs)

GRanges with 6 ranges and 3 metadata columns:
      seqnames               ranges strand |      sample
         <Rle>            <IRanges>  <Rle> | <character>
  [1]     chr7 [   37124,  8197116]      * |     NA12878
  [2]     chr7 [ 8200167,  8200467]      * |     NA12878
  [3]     chr7 [ 8200896, 49923859]      * |     NA12878
  [4]     chr7 [   37124, 49923859]      * |     NA12864
  [5]     chr7 [50003183, 57941963]      * |     NA12878
  [6]     chr7 [50003183, 57941963]      * |     NA12864
```

```
      numberProbes  seg.mean
         <numeric> <numeric>
  [1]         1936    0.0004
  [2]            2   -0.6600
  [3]        11315   -0.0030
  [4]        13212   -0.0156
  [5]         1636   -0.0099
  [6]         1632   -0.0273
  ---
  seqlengths:
        chr7     chr22
   159138663  51304566
```

If the offspring copy number changes within a minimum distance segment (as determined by the segmentation of the offspring copy number), the start and stop position of the minimum distance segments may be edited. The approach currently implemented is to define a new start and stop position if a breakpoint for the offspring segmentation occurs in the minimum distance interval. To illustrate, the following diagram uses vertical dashes (|) to denote breakpoints:

```
1 ...--|--------------|--...     ## minimum distance segment (before editing)
2 ...----|--------|------...     ## segmenation of log R ratios for offspring

->

3 ...--|-|--------|---|--...     ## after editing
```

In the above illustration, posterior calls are provided for the 3 segments indicated in line 3 instead of the single segment in line 1. Two additional steps are therefore required: (1) segmentation of the offspring log R ratios and (2) editing of the minimum distance breakpoints when appropriate. The following codechunk, segments the log R ratios for all chromosomes in the `trioSetList` object. For purposes of visualization, we also segment the parental log R ratios:

```
R> lrr.segs <- segment2(trioSetList, segmentParents=TRUE)
```

Minimum distance segments with a mean absolute value greater than 1 median absolute deviation from zero are edited using the `narrowRanges`:

```
R> mads.md <- mad2(md, byrow=FALSE)
R> md.segs2 <- narrowRanges(md.segs, lrr.segs, thr=0.75,
                            mad.minimumdistance=mads.md,
                            fD=featureData(trioSetList))
```

To each range (segment) the maximum posterior probability and the posterior probability of the normal range are computed using the function `MAP`.

```
R> gr <- MAP(object=trioSetList, ranges=md.segs2, nupdates=5)
R> show(gr)
```

Note the `MAP` can be applied to a single interval. For example,

```
R> MAP(trioSetList, md.segs2[1, ])

GRanges with 1 range and 9 metadata columns:
      seqnames               ranges strand |      sample
         <Rle>            <IRanges>  <Rle> | <character>
  [1]    chr22 [14456412, 17074487]      * |     NA12864
```

```
      numberProbes  seg.mean mindist.mad exceeds.md.thr
         <integer> <numeric>   <numeric>      <logical>
  [1]          515   -0.0193    0.118608          FALSE
            state    argmax lik.state  lik.norm
      <character> <integer> <numeric> <numeric>
  [1]         222        61 -5283.989 -5283.989
  ---
  seqlengths:
        chr7     chr22
   159138663  51304566
```

The ratio of posterior probabilities can be used to rank genomic ranges by the evidence for a de novo copy number alteration. Note that physically adjacent genomic ranges in a sample that are assigned the same copy number state will be collapsed into a single range by the `computeBayesFactor` function.

3.2 Visualizations

To plot the log R ratios and B allele frequencies for one of the de novo deletions, we extract these low-level summaries from the `TrioSetList` object into a *data.frame* that can be used by the `xyplot` function in the lattice package.

```
R> denovo.range <- GRanges("chr22", IRanges(20.8e6, 21.4e6))
R> i <- subjectHits(findOverlaps(denovo.range, gr))
R> gr.denovo <- gr[i, ]
R> gr.denovo <- gr.denovo[state(gr.denovo)=="221", ]
R> library(lattice)
R> library(foreach)
R> trioSet <- trioSetList[[2]][, match(sampleNames(gr.denovo), sampleNames(trioSetList))]
R> mindist(trioSet) <- md[[2]][, match(sampleNames(gr.denovo), sampleNames(trioSetList)), drop=FALSE]
R> fig <- xyplotTrio(gr.denovo, trioSet, frame=500e3,
             ylab="log R ratio and BAFs",
             xlab="physical position (Mb)",
             panel=xypanelTrio,
             scales=list(x="same",
             y=list(alternating=1,
             at=c(-1, 0, log2(3/2), log2(4/2)),
             labels=expression(-1, 0, log[2](3/2), log[2](4/2)))),
             lrr.segments=lrr.segs,
             md.segments=md.segs,
             layout=c(1, 4),
             col.hom="grey50",
             col.het="grey50",
             col.np="grey20",
             state.cex=0.8,
             cex=0.3,
             ylim=c(-3, 1.5),
             par.strip.text=list(lines=0.8, cex=0.6),
             key=list(text=list(c(expression(log[2]("R ratios")), expression("B allele freqencies")),
                      col=c("black", "blue")), columns=2))
```

The final graphic is displayed in Figure 1.

4 Session information

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

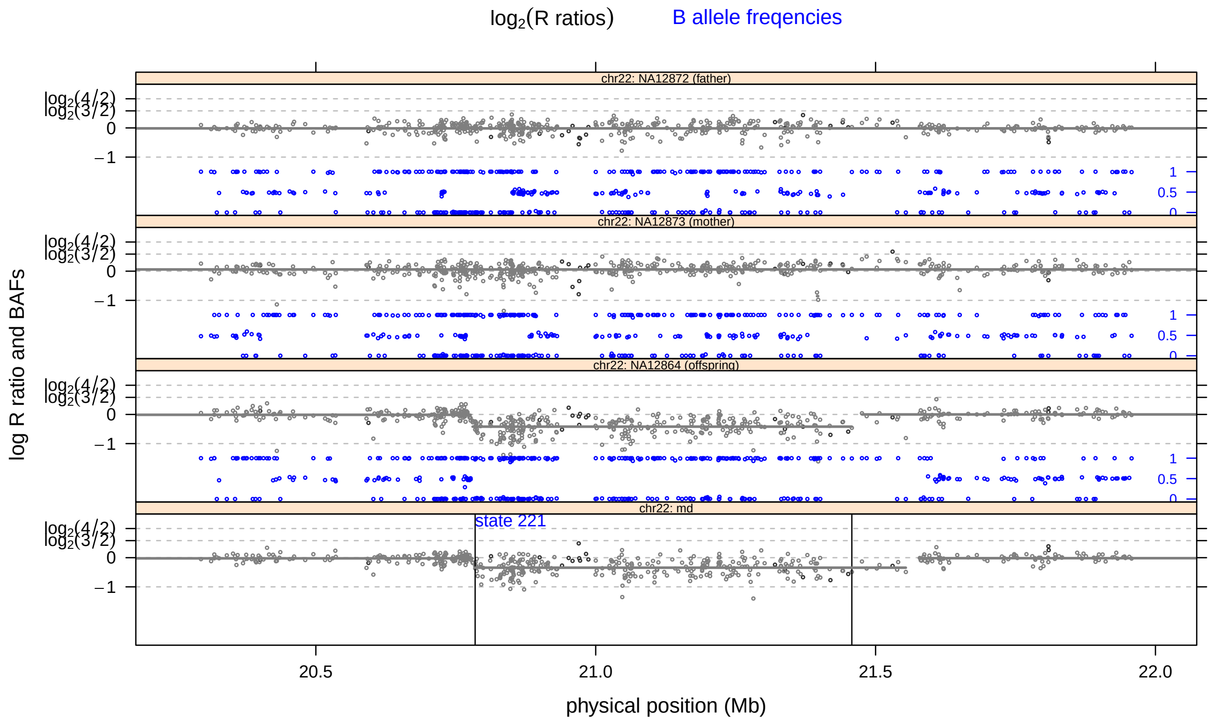

Figure 1: The log R ratios (grey) and B allele frequencies (blue) for the parent-offspring trio, as well as the minimum distance (bottom panel). The solid vertical lines in the bottom panel indicate the start and stop positions of an inferred de novo hemizygous deletion in the offspring (state '221').

- R version 3.1.1 (2014-07-10), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: Biobase 2.24.0, BiocGenerics 0.10.0, DNAcopy 1.38.1, GenomeInfoDb 1.0.2, GenomicRanges 1.16.4, IRanges 1.22.10, MinimumDistance 1.8.1, SNPchip 2.10.0, VanillaICE 1.26.1, XVector 0.4.0, foreach 1.4.2, human610quadv1bCrlmm 1.0.3, lattice 0.20-29, oligoClasses 1.24.0
- Loaded via a namespace (and not attached): BiocInstaller 1.14.2, Biostrings 2.32.1, Matrix 1.1-4, R.methodsS3 1.6.1, affyio 1.32.0, bit 1.1-12, codetools 0.2-9, compiler 3.1.1, expm 0.99-1.1, ff 2.2-13, grid 3.1.1, iterators 1.0.7, matrixStats 0.10.0, msm 1.4, mvtnorm 1.0-0, splines 3.1.1, stats4 3.1.1, survival 2.37-7, tools 3.1.1, zlibbioc 1.10.0

References

[1] Stefano Colella, Christopher Yau, Jennifer M Taylor, Ghazala Mirza, Helen Butler, Penny Clouston, Anne S Bassett, Anneke Seller, Christopher C Holmes, and Jiannis Ragoussis. QuantiSNP: an Objective Bayes Hidden-Markov Model to detect and accurately map copy number variation using SNP genotyping data. *Nucleic Acids Res*, 35(6):2013–2025, 2007.

[2] Philippe Hupe, Nicolas Stransky, Jean-Paul Thiery, Francois Radvanyi, and Emmanuel Barillot. Analysis of array cgh data: from signal ratio to gain and loss of dna regions. *Bioinformatics*, 20(18):3413–3422, Dec 2004.

[3] Adam B Olshen, E. S. Venkatraman, Robert Lucito, and Michael Wigler. Circular binary segmentation for the analysis of array-based DNA copy number data. *Biostatistics*, 5(4):557–72, Oct 2004.

[4] Daniel A Peiffer, Jennie M Le, Frank J Steemers, Weihua Chang, Tony Jenniges, Francisco Garcia, Kirt Haden, Jiangzhen Li, Chad A Shaw, John Belmont, Sau Wai Cheung, Richard M Shen, David L Barker, and Kevin L Gunderson. High-resolution genomic profiling of chromosomal aberrations using infinium whole-genome genotyping. *Genome Res*, 16(9):1136–1148, Sep 2006.

[5] Robert B Scharpf, Giovanni Parmigiani, Jonathan Pevsner, and Ingo Ruczinski. Hidden Markov models for the assessment of chromosomal alterations using high-throughput SNP arrays. *Annals of Applied Statistics*, 2(2):687–713, 2008.

[6] E. S. Venkatraman and Adam B Olshen. A faster circular binary segmentation algorithm for the analysis of array cgh data. *Bioinformatics*, 23(6):657–663, Mar 2007.

[7] Kai Wang, Zhen Chen, Mahlet G Tadesse, Joseph Glessner, Struan F A Grant, Hakon Hakonarson, Maja Bucan, and Mingyao Li. Modeling genetic inheritance of copy number variations. *Nucleic Acids Res*, 36(21):e138, Dec 2008.