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

Moiz Bootwalla and Rob Scharpf

November 4, 2011

Abstract

In studies involving case-parent trios assayed on high-throughput genotyping arrays, a common goal is to identify regions of de novo copy number alterations. 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 technical sources of variation, such as genomic waves that can effect parents and offspring differentially, often 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 studies.

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. Section 2 describes the data structures required for using this package. Section 3 outlines the steps for computing the minimum distance, segmenting the minimum distance, and computing maximum a posteriori probabilities to infer regions of de novo copy number alterations. We provide convenient tools for visualizing the low-level statistical summaries in the context of the segmented (*ranged* data) based on the lattice graphics. The workflow is illustrated using publicly available HapMap

trios assayed on the Illumina 610quad array. Where possible, we have adopted standard data structures for encapsulating the low-level data (`eSet` extensions) and the segmented data (`RangedData` extensions).

2 Required data structures

We begin by describing how to instantiate the data structures required for inferring changes in the de novo copy number. A key design consideration is that the 'unit' of our analysis is a trio, the sample size is the number of trios, and a common query will be to access the low level statistical summaries (log R ratios and B allele frequencies) for a set of markers for all members in a trio. For this particular use-case, it is much more convenient to store statistical summaries in a `feature x trios x family member` array rather than a `feature x samples` matrix. As the number of trios in a typical genome-wide association study may exceed 1000 (corresponding to several thousand samples), it may be impractical to store the low-level summaries in a single array. To make the size of the arrays more manageable, we organize the low-level summaries by chromosome in an object termed a `TrioSet`. We define a formal S4 class for a list of `TrioSet` objects as a `TrioSetList`. In addition to containing a `TrioSet` for each chromosome, the `TrioSetList` class has a slot for pedigree information defining the family relationships and a slot for the `sampleSheet` containing metadata on the experiment. We begin by constructing object of class `Pedigree` and `SampleSheet` that contain this information.

A constructor for the `SampleSheet` class is the `SampleSheet` function. The `SampleSheet` has 3 arguments (filename, id, and plate), but additional metadata on the samples can be supplied. For example, below we also include information in the well of the chemistry plate, the gender, and the SentrixPosition for the Illumina array. The id can either be the same as the filename, or a more convenient sample identifier. For example, we use the NA identifiers instead of the lengthy sample names in the example below.

```
R> library(MinimumDistance)
R> library(human610quadv1bCrlmm)
R> path <- system.file("extdata", package="MinimumDistance")
R> load(file.path(path, "sample.sheet.rda"))
R> samplesheetExample <- SampleSheet(filename=sample.sheet$Sample.Name,
                                     id=paste("NA",substr(sample.sheet$Sample.Name, 6, 10),sep=""),
                                     plate=sample.sheet$Plate,
                                     well=sample.sheet$Well,
                                     gender=sample.sheet$Gender,
                                     SentrixPosition=sample.sheet$SentrixPosition)
```

An object of class *Pedigree* is constructed in the following codechunk:

```
R> load(file.path(path, "pedigreeInfo.rda"))
R> indId <- unlist(pedigreeInfo)
R> trioFactor <- matrix(seq_len(nrow(pedigreeInfo)), nrow(pedigreeInfo), 3, byrow=FALSE)
R> memberId <- substr(names(indId), 1, 1)
R> ## which row in the pedigree file
R> trio.index <- as.integer(trioFactor) ## note this gives the index in the pedigree
R> pedigreeIndex <- data.frame(individualId=indId,
                               memberId=memberId,
                               index.in.pedigree=trio.index,
                               stringsAsFactors=FALSE)
R> rownames(pedigreeIndex) <- NULL
R> pedigreeExample <- new("Pedigree", trios=pedigreeInfo,
                          trioIndex=pedigreeIndex)
R> allNames(pedigreeExample)

[1] "NA06993" "NA11881" "NA06985" "NA11882" "NA06991" "NA10859"

R> show(pedigreeExample)
```

```
trios:
        F       M       O
1 NA06993 NA06985 NA06991
2 NA11881 NA11882 NA10859

pedigreeIndex:
  individualId memberId index.in.pedigree
1      NA06993        F                1
2      NA11881        F                2
3      NA06985        M                1
4      NA11882        M                2
5      NA06991        O                1
6      NA10859        O                2
```

Next, we load matrices of the $\log_2$ R ratios and B allele frequencies extracted from the BeadStudio output. These matrices contain 25 markers for each chromosome as the intent is to illustrate the steps required for initializing a data structure of the `TrioSetList` class. (In order to illustrate some of the smoothing functions downstream, we will use the `trioSetListExample` data provided with the package. The `trioSetListExample` contains many more markers, but only for two autosomes). A *TrioSetList* object can be created using the helper function `TrioSetList`.

```
R> load(file.path(path, "logRratio.rda"))
R> load(file.path(path, "baf.rda"))
R> trioSetList <- TrioSetList(lrr=logRratio,
                              baf=baf,
                              pedigree=pedigreeExample,
                              sampleSheet=samplesheetExample,
                              cdfname="human610quadv1bCrlmm")
```

Note that the log R ratio and B allele frequency matrices have been transformed into arrays using the information stored in the `pedigree` object.

```
R> trioSetList

TrioSetList of length 22

R> dims(trioSetList)

         chr 1 chr 2 chr 3 chr 4 chr 5 chr 6 chr 7 chr 8 chr 9
Features    25    25    25    25    25    25    25    25    25
Trios        2     2     2     2     2     2     2     2     2
F, M, O      3     3     3     3     3     3     3     3     3
         chr 10 chr 11 chr 12 chr 13 chr 14 chr 15 chr 16 chr 17
Features     25     25     25     25     25     25     25     25
Trios         2      2      2      2      2      2      2      2
F, M, O       3      3      3      3      3      3      3      3
         chr 18 chr 19 chr 20 chr 21 chr 22
Features     25     25     25     25     25
Trios         2      2      2      2      2
F, M, O       3      3      3      3      3

R> sampleSheet(trioSetList)

               filename      id         plate well gender
1 CIDR_06993@0072958140 NA06993 WG1001552-DNA  C12   Male
4 CIDR_11881@0072958128 NA11881 WG1001557-DNA  E06   Male
```

```
2 CIDR_06985@0072958138 NA06985 WG1001574-DNA  F12 Female
5 CIDR_11882@0072958127 NA11882 WG1001558-DNA  G09 Female
3 CIDR_06991@1007845680 NA06991 WG1001289-DNA  D10 Female
6 CIDR_10859@0072958146 NA10859 WG1001558-DNA  E03 Female
    SentrixPosition
1 4256206347_R01C02
4 4256206437_R01C01
2 4256206089_R02C01
5 4256206428_R01C02
3 4256193742_R01C02
6 4256193768_R01C01
```

```
R> pedigree(trioSetList)
```

```
trios:
        F       M       O
1 NA06993 NA06985 NA06991
2 NA11881 NA11882 NA10859

pedigreeIndex:
  individualId memberId index.in.pedigree
1      NA06993        F                 1
2      NA11881        F                 2
3      NA06985        M                 1
4      NA11882        M                 2
5      NA06991        O                 1
6      NA10859        O                 2
```

```
R> trios(trioSetList)
```

```
        F       M       O
1 NA06993 NA06985 NA06991
2 NA11881 NA11882 NA10859
```

```
R> trioSetList[1:5]
```

```
TrioSetList of length 5
```

```
R> trioSetList[[1]]
```

```
TrioSet (storageMode: lockedEnvironment)
assayData: 25 features, 2 samples
  element names: BAF, logRRatio
```

As the offspring name uniquely identifies a trio, `sampleNames` and `offspringNames` are interchangeable.

```
R> ## Note
R> identical(sampleNames(trioSetList), offspringNames(trioSetList))
```

```
[1] TRUE
```

Feature annotation such as chromosome and position for each SNP is added by the function `featureDataFrom` defined in the oligoClasses package. Alternatively, users may supply their own feature annotation through the optional `featureData` argument to the R function `TrioSetList`. Platforms supported by the function `featureDataFrom` are listed below.

```
R> annotationPackages()[grep("Crlmm", annotationPackages())]
```

```
 [1] "genomewidesnp6Crlmm"    "genomewidesnp5Crlmm"
 [3] "human370v1cCrlmm"       "human370quadv3cCrlmm"
 [5] "human550v3bCrlmm"       "human650v3aCrlmm"
 [7] "human610quadv1bCrlmm"   "human660quadv1aCrlmm"
 [9] "human1mduov3bCrlmm"     "humanomni1quadv1bCrlmm"
```

3 Detection of de novo copy number

In order to make the size of this package manageable, the `trioSetList` object created in the previous section contained only 25 markers for each of the 22 autosomes. While useful for illustrating the steps for creating the required data structures, it is not useful for detetecting de novo copy number alterations. To illustrate the smoothing steps, we load an object that contains all the markers for 2 chromosomes (7 and 22) for 2 HapMap trios. A few summary statistics for this object are displayed below.

```
R> data(trioSetListExample)
R> show(trioSetList)

TrioSetList of length 2

R> dims(trioSetList)

         chr 7 chr 22
Features 34080   9284
Trios        2      2
F, M, O      3      3

R> sampleNames(trioSetList)

[1] "NA12878" "NA12864"

R> allNames(trioSetList)

[1] "NA12891" "NA12872" "NA12892" "NA12873" "NA12878" "NA12864"

R> annotation(trioSetList)

[1] "human610quadv1bCrlmm"
```

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

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

The above calculation is vectorized in R. The following codechunk calculates the minimum distance for each chromosome in the `trioSetList` object, returning a list of matrices for each chromosome. We store the minimum distances in the *TrioSet* objects using the replacment method `mindist`.

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

  |
  |                                                      |   0%
  |
  |============================                          |  50%
  |
  |============================================================| 100%
```

```
  |
  |                                                              |   0%
  |
  |============================                                  |  50%
  |
  |==============================================================| 100%
```

```
R> mindist(trioSetList) <- md
```

We calculate the median absolute deviation (MAD) of the minimum distance and the log R ratios. For the log R ratios, we calculate the MAD across markers and across independent offspring.

```
R> ## calculate the mad of the minimum distance
R> mads.md <- mad2(mindist(trioSetList), byrow=FALSE)
R> mad.mindist(trioSetList) <- mads.md
R> mads.lrr.sample <- mad2(lrr(trioSetList), byrow=FALSE)
R> mads.lrr.marker <- mad2(lrr(trioSetList), byrow=TRUE)
R> mad.sample(trioSetList) <- mads.lrr.sample
R> mad(trioSetList)
```

```
                F         M         O
NA12878 0.1291345 0.1125293 0.1083781
NA12864 0.1377335 0.2068227 0.1654582
```

```
R> mad.marker(trioSetList) <- mads.lrr.marker
R> fvarLabels(trioSetList[[1]])
```

```
[1] "chromosome"        "position"          "isSnp"
[4] "marker.mad"        "marker.mad.father" "marker.mad.mother"
```

The circular binary segmentation algorithm implemented in the R package DNAcopy is used to smooth the minimum distance estimates. The method `segment2` is a wrapper to the `CNA.smooth` and `segment` functions defined in DNAcopy. 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=mindist(trioSetList),
                       pos=position(trioSetList),
                       chrom=chromosome(trioSetList), verbose=1)
```

The object returned by the `segment2` method is an object of class `RangedDataCBS`. Currently, I do not use `space` in `RangedData` objects.

```
R> md.segs
```

```
RangedDataCBS with 33 rows and 6 value columns across 1 space
       space               ranges   |     chrom  num.mark
    <factor>            <IRanges>   | <integer> <numeric>
1          1 [   37124,  8197116]   |         7      1936
2          1 [ 8200167,  8200467]   |         7         2
3          1 [ 8200896, 49923859]   |         7     11315
4          1 [   37124, 49923859]   |         7     13212
5          1 [50003183, 57941963]   |         7      1636
6          1 [50003183, 57941963]   |         7      1632
7          1 [61060840, 62120420]   |         7       101
8          1 [62128107, 72534486]   |         7      1829
9          1 [61060840, 61668154]   |         7        79
```

```
...      ...                   ... ...       ...       ...
25         1 [14456412, 17074487]   |        22       515
26         1 [17092563, 17118296]   |        22         2
27         1 [17135059, 20780261]   |        22       825
28         1 [20784580, 21554058]   |        22       290
29         1 [21579040, 39642296]   |        22      4621
30         1 [39719455, 44690948]   |        22      1439
31         1 [39719455, 44690948]   |        22      1432
32         1 [44801602, 49562479]   |        22      1559
33         1 [44801602, 49562479]   |        22      1552
             id start.index end.index  seg.mean
    <character>   <integer> <integer> <numeric>
1       NA12878           1      1936    0.0003
2       NA12878        1937      1938   -0.6592
3       NA12878        1939     13254   -0.0028
4       NA12864           1     13254   -0.0159
5       NA12878       13255     14890   -0.0092
6       NA12864       13255     14890   -0.0274
7       NA12878       14891     14991    0.0784
8       NA12878       14992     16820   -0.0080
9       NA12864       14891     14969   -0.0421
...         ...         ...       ...       ...
25      NA12864           1       516   -0.0199
26      NA12864         517       518   -0.7901
27      NA12864         519      1348   -0.0250
28      NA12864        1349      1640   -0.3484
29      NA12864        1641      6286   -0.0180
30      NA12878        6287      7725   -0.0029
31      NA12864        6287      7725   -0.0194
32      NA12878        7726      9284   -0.0026
33      NA12864        7726      9284   -0.0131
```

Accessors for objects of this class include

```
R> sampleNames(md.segs)

 [1] "NA12878" "NA12878" "NA12878" "NA12864" "NA12878" "NA12864"
 [7] "NA12878" "NA12878" "NA12864" "NA12864" "NA12864" "NA12878"
[13] "NA12864" "NA12878" "NA12864" "NA12878" "NA12864" "NA12878"
[19] "NA12878" "NA12878" "NA12878" "NA12878" "NA12878" "NA12878"
[25] "NA12864" "NA12864" "NA12864" "NA12864" "NA12864" "NA12878"
[31] "NA12864" "NA12878" "NA12864"

R> coverage2(md.segs)

 [1]  1936     2 11315 13212  1636  1632   101  1829    79    64
[11]  1782  3746  3730 12176 12143  1334  1330   445     2  1181
[21]    10   604    70  3974   515     2   825   290  4621  1439
[31]  1432  1559  1552

R> chromosome(md.segs)

 [1]  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7  7 22 22 22 22
[22] 22 22 22 22 22 22 22 22 22 22 22 22

R> mean(md.segs)
```

```
 [1]  0.0003 -0.6592 -0.0028 -0.0159 -0.0092 -0.0274  0.0784 -0.0080
 [9] -0.0421  0.2125 -0.0190 -0.0028 -0.0215 -0.0059 -0.0190 -0.0009
[17] -0.0215  0.0063  0.5924  0.0017 -0.3244 -0.0080 -0.2317 -0.0064
[25] -0.0199 -0.7901 -0.0250 -0.3484 -0.0180 -0.0029 -0.0194 -0.0026
[33] -0.0131
```

Having partitioned the genome by the segmentation of the minimum distance, we can call the trio copy number state for each genomic interval. Before we do so, consider that minimum distance estimates near zero are consistent with normal copy number in all members of the trio and a copy number alteration in the offspring inherited from either parent. Or that minimum distance estimates near -1 are consistent with a denovo hemizygous deletion in which both parents are normal and a denovo homozygous deletion in the offspring when both parents are hemizygous. While we can distinguish between these possibilities by examining the distribution of log R ratios and B allele frequencies in the family members, an automatic means to correctly classify the trio copy number state becomes more difficult if a single segment contains a mixture of these states. We therefore edit minimum distance ranges that are nonzero, where nonzero is defined by the number of MADs the mean is from zero. The editing is acheived by segmenting the log R ratios of the offspring. If a breakpoint is identified within a nonzero minimum distance segment, the breakpoints of the minimum distance segment are edited. The following diagram using vertical dashes (|) to denonte breakpoints illustrates the basic principle.

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

->

 --|-|--------|---|--     ## after editing
```

In the following codechunk, we segment the log R ratios for all chromosomes in the `trioSetList` object for the father, mother, and offspring. While only the segmentation of the offspring log R ratios are required to edit the minimum distance breakpoints, the segmenation of the parental log R ratios are useful for the visualization methods discussed in Section 3.1.

```
R> lrr.segs <- segment2(object=lrr(trioSetList),
                        pos=position(trioSetList),
                        chrom=chromosome(trioSetList),
                        id=trios(trioSetList), ## id is required
                        verbose=TRUE)
```

Note that the object `lrr.segs` is also a `RangedDataCBS` object. Minimum distance segments with a mean absolute value greater than 0.9 MADs from zero are edited using the `narrow`:

```
R> md.segs2 <- narrow(md.segs, lrr.segs, 0.9, mad.minimumdistance=mads.md)
```

Note that the number of segments after editing is typically greater than the original number of segments:

```
R> nrow(md.segs2) > nrow(md.segs)

[1] FALSE
```

Finally, we use the ranged data in the `md.segs2` object to call the trio copy number state for each interval. Conditional on the underlying trio copy number state, we calculate the probability of the observed log R ratios and the probability of the observed B allele frequencies. Details regarding the mixture model used to estimate the emission probabilities are discussed elsewhere. The copy number state with the maximum posterior probability is assigned to each genomic range. We assign to each range both the maximum posterior probability and the posterior probability of the normal range. The ratio of posterior probabilities provides a useful statistic to rank the genomic ranges by the evidence of de novo copy number alteration. Consecutive

genomic ranges in a sample that are assigned the same copy number state are collapsed into a single range. As a consequence, the number of ranges in the object returned by `computeBayesFactor` is typically smaller than the number of ranges passed to the `ranges` argument. We continue the previous example to illustrate how ranges are subject to pruning following the posterior classification of the copy number states:

```
 --|-|--------|---|--     ## after editing
 --|--------------|--     ## result 1: all segments have the same trio state call
 --|----------|---|--     ## result 2: the first two segments have the same call
```

Note that 'result 1' recovers the original segmentation of the minimum distance. Pragmatically, the above steps are implmented in the function `computeBayesFactor`:

```
R> map.segs <- computeBayesFactor(object=trioSetList, ranges=md.segs2)
R> map.segs <- map.segs[order(sampleNames(map.segs), chromosome(map.segs), start(map.segs)), ]

R> map.segs

RangedDataCBS with 16 rows and 11 value columns across 1 space
      space               ranges |          id     chrom  num.mark
   <factor>            <IRanges> | <character> <integer> <integer>
1         1 [   37124,  61668154] |     NA12864         7     14969
2         1 [61673005,  62336389] |     NA12864         7        64
3         1 [62342453, 158812247] |     NA12864         7     19047
4         1 [14456412,  20780261] |     NA12864        22      1348
5         1 [20784580,  21554058] |     NA12864        22       292
6         1 [21579040,  49562479] |     NA12864        22      7644
7         1 [   37124,   8197116] |     NA12878         7      1936
8         1 [ 8200167,   8200467] |     NA12878         7         2
9         1 [ 8200896, 158812247] |     NA12878         7     32142
10        1 [14456412,  16871770] |     NA12878        22       445
11        1 [16872032,  16872749] |     NA12878        22         2
12        1 [16873980,  21419339] |     NA12878        22      1181
13        1 [21441199,  21538519] |     NA12878        22        10
14        1 [21540389,  23970628] |     NA12878        22       604
15        1 [23980406,  24244593] |     NA12878        22        70
16        1 [24254444,  49562479] |     NA12878        22      6972
       seg.mean start.index end.index mindist.mad     lik.state
      <numeric>   <numeric> <numeric>   <numeric>     <numeric>
1  -0.017295137           1     14969  0.11682888  4.657932e+04
2   0.212500000       14970     15033  0.11682888  1.001422e+02
3  -0.019666772       15034     34080  0.11682888  6.021705e+04
4  -0.024182938           1      1348  0.11682888  4.531105e+03
5  -0.348400000        1349      1640  0.11682888  7.875919e+02
6  -0.017264194        1641      9284  0.11682888  2.689665e+04
7   0.000300000           1      1936  0.08880774  7.132457e+03
8  -0.659200000        1937      1938  0.08880774 -9.202714e-01
9  -0.004262367        1939     34080  0.08880774  1.163936e+05
10  0.006300000           1       445  0.08880774  1.571798e+03
11  0.592400000         446       447  0.08880774 -4.211252e+00
12  0.001700000         448      1628  0.08880774  4.166802e+03
13 -0.324400000        1629      1638  0.08880774  1.974743e+01
14 -0.008000000        1639      2242  0.08880774  2.178886e+03
15 -0.231700000        2243      2312  0.08880774  2.068662e+02
16 -0.004827897        2313      9284  0.08880774  2.660287e+04
       lik.norm    argmax       state
      <numeric> <integer> <character>
```

copy number	symbol for copy number state
0	1
1	2
2	3
3	4
4	5

Table 1: We use the notation adopted by PennCNV for representing the copy number states. The trio copy number state is indicated by the triplet xyz where x is the symbol for the copy number state of the father, y is the symbol for the copy number state of the mother, and z is the symbol for the offspring copy number state. For example, the trio copy number state '332' indicates a de novo hemizygous deletion in the offspring as both parents have *normal* copy number (copy number 2) and the offspring has a single copy.

```
1  41730.970277        61         333
2     45.951226        85         334
3   5691.972685        61         333
4   1670.035038        61         333
5    470.318982        37         332
6  16416.834538        61         333
7   7132.456873        61         333
8     -1.013983        37         332
9  40615.749758        61         333
10  1571.798135        61         333
11   -11.899091        55         223
12  4166.802470        61         333
13    16.348665        37         332
14  2178.885858        61         333
15   119.096697        37         332
16 15420.296813        61         333
```

The integer code for the copy number states are provided in the Table 1 and can be accessed by the function `state`. For example, here we tabulate the frequency of the trio copy number states by chromosome from the `map.segs` object.

```
R> table(state(map.segs), chromosome(map.segs))
```

```
     7 22
  223 0  1
  332 1  3
  333 4  6
  334 1  0
```

3.1 Visualizations

The R package VanillaICE contains methods for visualizing *RangedData* objects that build on the infrastructure in the lattice package. In the following codechunk, we construct a trellis object for the log R ratios, minimum distance, and B allele frequencies.

```
R> trioSet <- stack(trioSetList)
R> rd.denovoDel <- map.segs[state(map.segs) == 332, ]
R> library("VanillaICE")
R> ##md.segs$seg.mean <- -1*md.segs$seg.mean
R> lrrfig <- xyplot(lrr ~ x | id,
                    data=trioSet,
```

```
                    range=rd.denovoDel[1, ],
                    lrr.segs=lrr.segs,
                    md.segs=md.segs,
                    frame=500e3,
                    panel=xypanelMD,
                    xlab="physical position (Mb)",
                    cex=0.2,
                    pch=21,
                    border="orange",
                    scales=list(cex=0.5),
                    par.strip.text=list(lines=0.8, cex=0.6),
                    col.np="royalblue", fill.np="grey",
                    index.cond=list(4:1),
                    return.data.frame=FALSE,
                    cex.state=0.5)
R> baffig <- xyplot(baf ~ x | id,
                    data=trioSet,
                    range=rd.denovoDel[1, ],
                    xlab="",
                    lrr.segs=lrr.segs,
                    md.segs=md.segs,
                    frame=500e3,
                    panel=VanillaICE::xypanel,
                    cex=0.2,
                    pch=21,
                    border="orange",
                    scales=list(cex=0.6),
                    par.strip.text=list(lines=0.8, cex=0.6),
                    col.np="royalblue", fill.np="grey",
                    index.cond=list(3:1), cex.state=0.5)
```

The final graphic is displayed in Figure 1.

```
R> print(VanillaICE:::arrangeSideBySide(lrrfig, baffig))
```

4 Session information

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

- R Under development (unstable) (2011-11-03 r57560), `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=en_US.iso885915`, `LC_MESSAGES=en_US.iso885915`, `LC_PAPER=C`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.iso885915`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, grid, methods, stats, utils
- Other packages: Biobase 2.15.0, BiocInstaller 1.3.1, DNAcopy 1.29.0, human610quadv1bCrlmm 1.0.2, IRanges 1.13.1, MinimumDistance 0.1.47, oligoClasses 1.17.1, VanillaICE 1.17.2
- Loaded via a namespace (and not attached): affyio 1.23.0, Biostrings 2.23.0, bit 1.1-7, ff 2.2-3, lattice 0.20-0, msm 1.1, mvtnorm 0.9-9991, SNPchip 1.19.0, splines 2.15.0, survival 2.36-10, tools 2.15.0, zlibbioc 1.1.0

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.

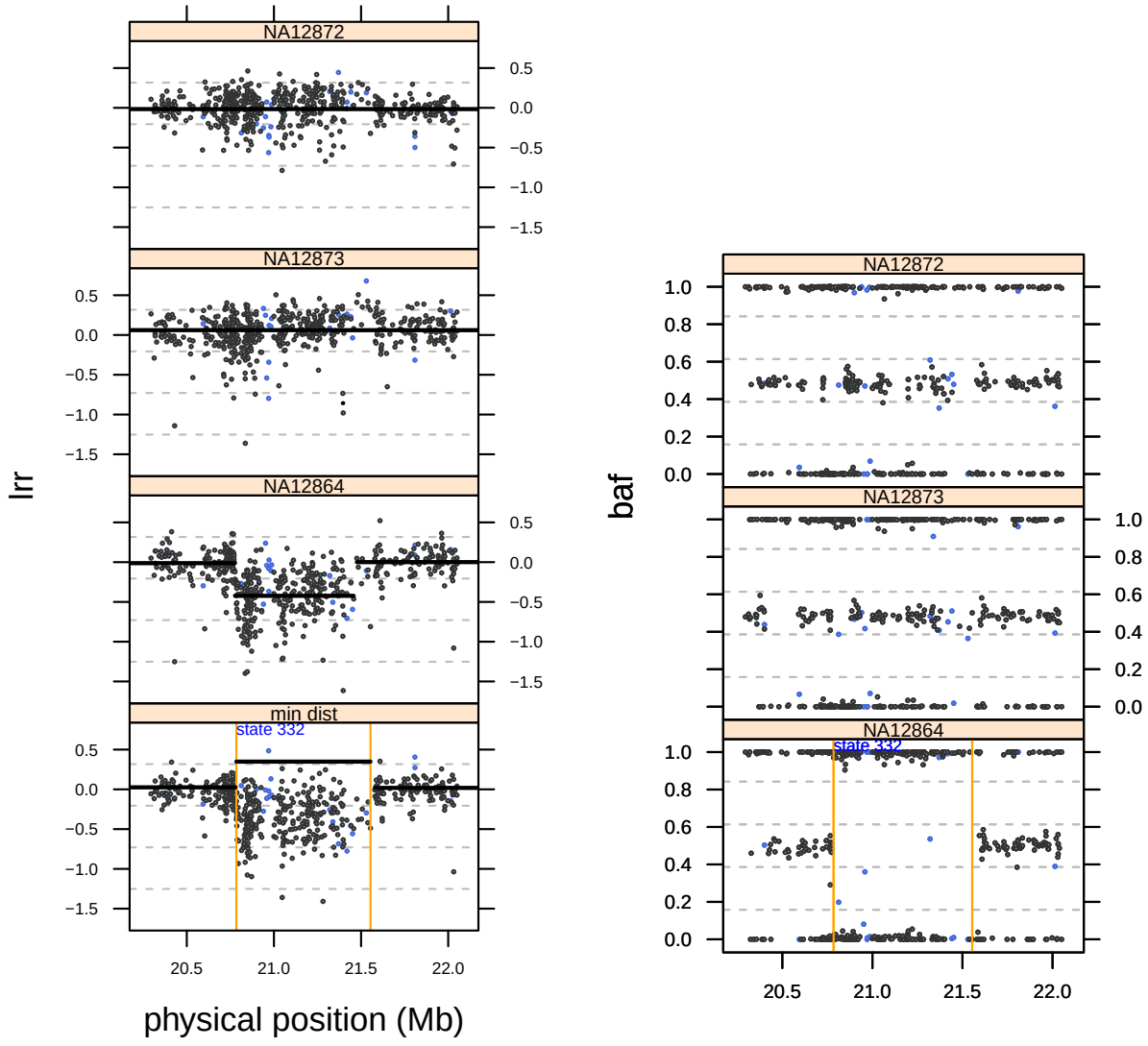

Figure 1: The left panels display the log R ratios for the father (top left), mother, and offspring, as well as the minimum distance (bottom left). The right panels display the B allele frequencies for the father, mother, and offspring. The orange vertical lines indicate the start and stop positions of an inferred de novo hemizygous deletion in the offspring (state '332').