VanillaICE: Hidden Markov Models for the Assessment of Chromosomal Alterations using High-throughput SNP Arrays

Robert Scharpf

November 2, 2011

Abstract

This package provides an implementation of a hidden Markov Model for high throughput SNP arrays. Users of this package should already have available locus-level estimates of copy number. Copy number estimates can be relative or absolute.

1 Overview

This vignette requires that you have

- an absolute estimate of the *total* copy number organized such that rows correspond to loci and columns correspond to samples

  and / or

- a matrix of genotype calls (1=AA, 2 = AB, 3= BB): rows correspond to loci and columns correspond to samples

Additional options that can improve the HMM predictions include

- a CRLMM confidence score of the genotype call
- standard errors of the copy number estimates

Other HMM implementations are available for the joint analysis of copy number and genotype, including QuantiSNP [1] and PennCNV [4].

Data considerations. The HMM implemented in this package is most relevant for heritable diseases for which integer copy numbers are expected. For somatic cell diseases such as cancer, we suggest circular binary segmentation, as implemented in the R package DNAcopy [2].

Citing this software. 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.

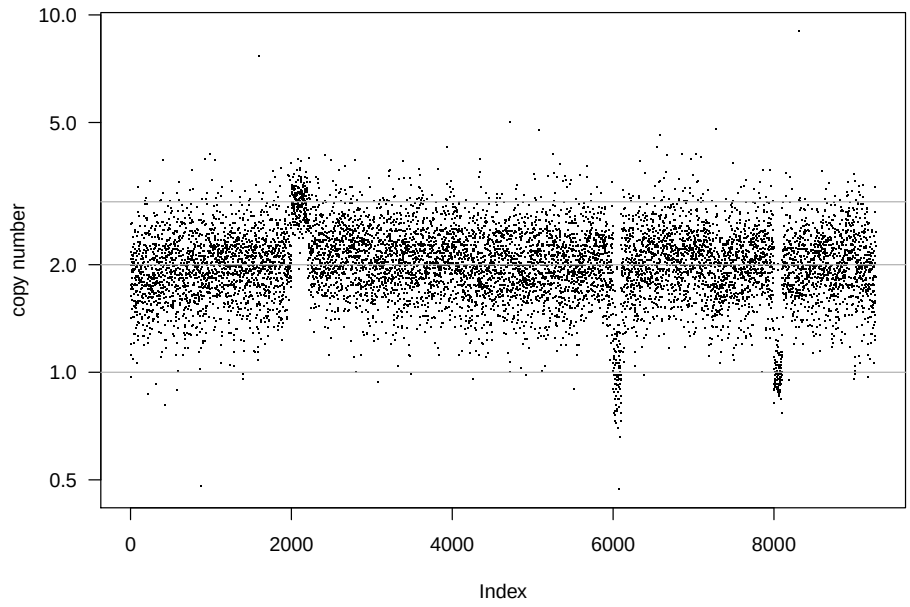
2 Organizing the locus-level data

This package includes simulated genotype and copy number data for 9165 SNPs.

```
> library(VanillaICE)
> data(locusLevelData)
```

The copy number estimates in the locusLevelData object were multiplied by 100 and saved as an integer. Verify that it is reasonable to assume integer copy number for the HMM by plotting the locus-level estimates as a function of the physical position.

```
> par(las=1)
> plot(locusLevelData[["copynumber"]][, 1]/100, pch=".", ylab="copy number", log="y")
> abline(h=1:3, col="grey70")
```



Next, create an object of class `oligoSnpSet` from the simulated data:

```
> oligoSet <- new("oligoSnpSet",
+                 copyNumber=log2(locusLevelData[["copynumber"]]/100),
+                 call=locusLevelData[["genotypes"]],
+                 callProbability=locusLevelData[["crlmmConfidence"]],
+                 annotation=locusLevelData[["platform"]])
> oligoSet <- oligoSet[!is.na(chromosome(oligoSet)), ]
> oligoSet <- oligoSet[chromosome(oligoSet) < 3, ]
```

If confidence scores or inverse standard errors for the copy number estimates are available, these should be supplied to the `cnConfidence` slot in the `assayData`. For illustration, in the following code chunk we transform the copy number estimates to the log scale and calculate a robust estimate of the standard deviation across autosomes. If uncertainty estimates are not available for copy number, the HMM will calculate the median absolute deviation (MAD). See the the function `robustSds`.

```
> sds <- sd(oligoSet)
```

The inverse of the `sds` object can be assigned to the `cnConfidence` slot.

```
> cnConfidence(oligoSet) <- 1/sds
```

3 Fitting the HMM

3.1 Vanilla HMM

When jointly modeling the copy number and genotype data, we assume that the genotype estimates and copy number estimates are independent conditional on the underlying hidden state. The emission probabilities for the genotypes are then calculated using either (i) assumptions of the probability of observing a homozygous genotype call given the underlying state. Next we order the markers by chromosome and physical position. The `hmm` method will order the `oligoSnpSet` object automatically, so the following step is not required.

```
> oligoSet <- order(oligoSet)
> hmmOpts <- hmm.setup(oligoSet, is.log=TRUE)
```

The viterbi algorithm is used to obtain the most likely sequence of hidden states given the observed data. For efficiency, we return an object of class `RangedDataHMM` with genomic coordinates of the normal and altered regions. We also return the log-likelihood ratio (LLR) of the predicted sequence in an interval versus the null of normal copy number. For intervals with typical copy number (2) and percent heterozygosity (the 3rd state in the above codechunk), the LLR is zero.

```
> ## first 2 chromosomes
> oligoSet <- oligoSet[chromosome(oligoSet) <= 2, ]
> fit.van <- hmm(oligoSet, hmmOpts)

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

Next we plot the data for chromosome 1 and overlay the predictions from the hidden markov model. See Figure 1.

To find which markers are included in each genomic interval returned by the `hmm` method, one can use the `findOverlaps` method in the *oligoClasses*. This method returns the 'match matrix'. The first column in the match matrix are indices of the genomic intervals (rows) of the `RangedDataHMM` object; the second column in the match matrix is a vector of indices for the markers in the `oligoSet` object. For example, the second interval in the `RangedDataHMM` object `fit.van` contains 102 markers.

```
> fit.van[2, ]
```

```
RangedDataHMM with 1 row and 7 value columns across 1 space
     space               ranges |     chrom  num.mark          id
  <factor>            <IRanges> | <integer> <integer> <character>
1        1 [49825223, 54637167] |         1       102     NA06993
  start.index end.index     state       LLR
    <integer> <integer> <integer> <numeric>
1           0         0         4  14.69012
```

To find the names of the 102 markers that are included in this genomic interval, one could do the following

```
> mm <- matchMatrix(findOverlaps(fit.van, oligoSet))
> markersInRange <- featureNames(oligoSet)[mm[,1]==2]
```

Multipanel displays can be useful for visualizing the low-level data for copy number alterations. We extend the `xyplot` method in the R package lattice for two common use-cases: by-locus and by-sample.

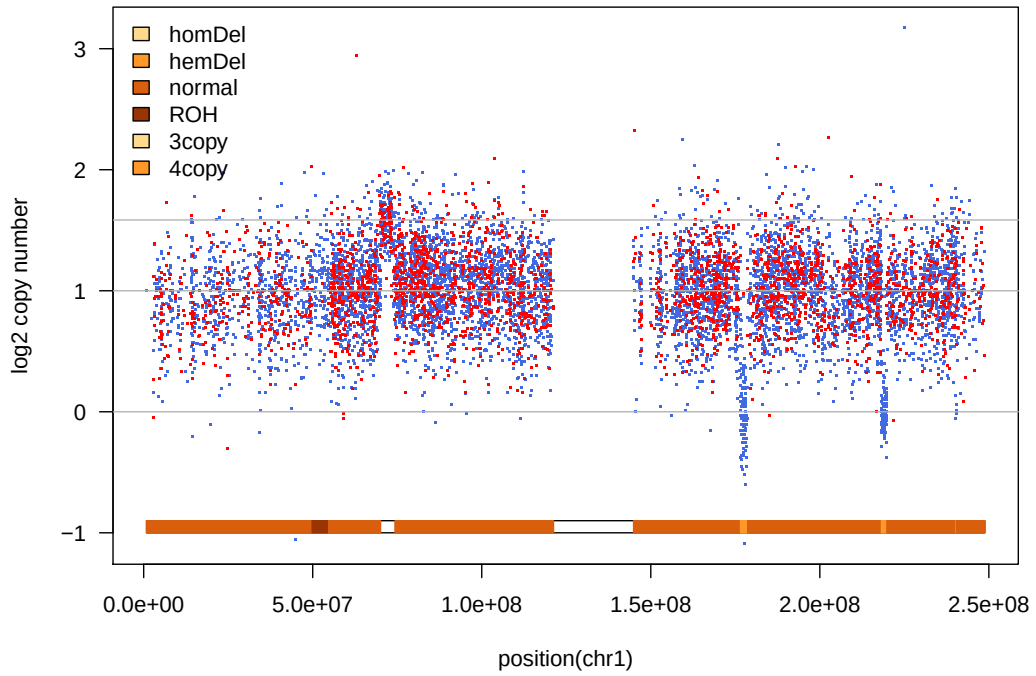


Figure 1: Plot of artificial data for one chromosome.

By locus. To plot the genomic data for a set of ranges at a given locus, we provide an unevaluated code chunk below (our example contains only a single sample):

```
> xyplot(cn ~ x | id, oligoSet, range=RangedDataObject, ylim=c(-0.5,4), panel=xypanel)
```

Note that the default in the above command is to use a common x- and y-scale for each panel. To allow the x-axes to change for each panel, one could set the x-scales to 'free':

```
> xyplot(cn ~ x | id, oligoSet, range=RangedDataObject, ylim=c(-0.5,4), scales=list(x="free"),
+        panel=xypanel)
```

The function `xypanel` provides default colors for annotating the plotting symbols by genotype and by whether the markers are polymorphic. The `RangedDataHMM` must be passed by the name *range* to the `xyplot` method.

By sample. To plot the low-level data for multiple alterations occurring in a single sample, one can again pass a `RangedDataHMM` object with name *range* to the `xyplot`. For example, see Figure 2. The code for producing Figure 2 is in the following code chunk. Note that the formula in the example below conditions on *range* instead of *id*. The conditioning variable for displaying multiple panels of a single sample must be called 'range'. We plot a 2 Mb window surrounding each of the alterations in the simulated data by specifying `frame=2e6`.

```
> ranges.altered <- fit.van[state(fit.van) != 4, ]
> xy.example <- xyplot(cn ~ x | range, oligoSet, range=ranges.altered, frame=2e6,
```

```
+                       scales=list(x="free"), ylim=c(-0.5,4),
+                       panel=xypanel, cex=0.4, pch=21, border="grey",
+                       ylab=expression(log[2]("copy number")))
```

See `?xyplot` for additional details.

Note also that `scales` must be set to *free* in the above call to `xyplot`.

3.2 ICE HMM

To compute emission probabilities that incorporate the *crlmm* genotype confidence scores, (i) set `ICE` to `TRUE` in the `hmm.setup` function and (ii) indicate which of the states are expected to be largely homozygous (`rohStates`). Note that this option is limited too a few platforms and the Affy 100k platforms is not one of them.

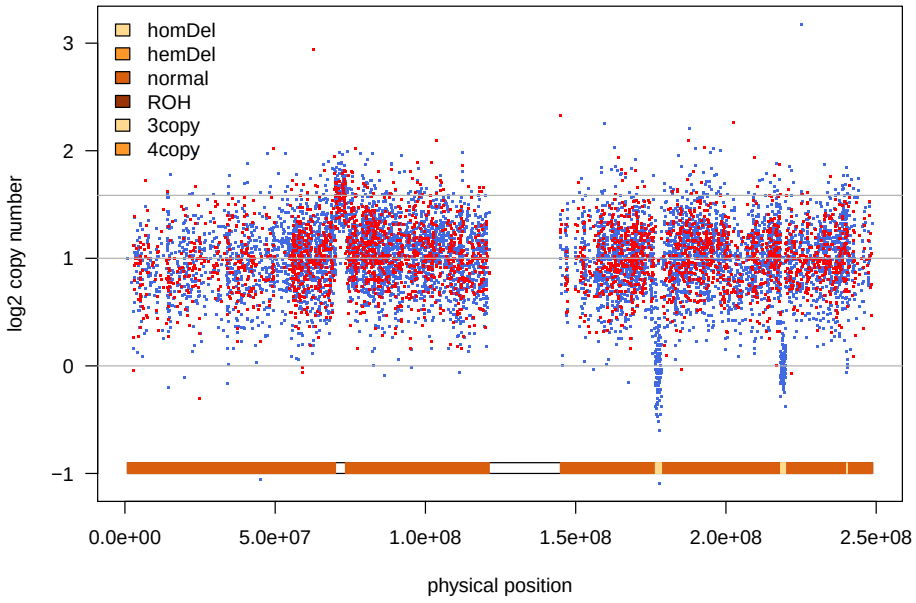
```
> hmmOpts <- hmm.setup(oligoSet, ICE=TRUE, rohStates=c(FALSE, TRUE, TRUE, FALSE, FALSE))
> res <- tryCatch(hmm(oligoSet, hmmOpts),
+                 error=function(e) "platform not supported")
> print(res)

[1] "platform not supported"

> ## supported platforms
> VanillaICE:::icePlatforms()

[1] "pd.genomewidesnp.6"
[2] "genomewidesnp6"
[3] "pd.mapping250k.nsp"
[4] "pd.mapping250k.sty"
[5] "pd.mapping250k.nsp, pd.mapping250k.sty"
```

For the purpose of illustration, we assign 'genomewidesnp6' to the annotation slot since this platform is supported.

```
> ann <- annotation(oligoSet)
> annotation(oligoSet) <- "genomewidesnp6"
> fit.ice <- hmm(oligoSet, hmmOpts)

  |
  |                                                                   |   0%
  |
  |===================================================================| 100%

> fit.ice

RangedDataHMM with 12 rows and 7 value columns across 1 space
      space                 ranges |     chrom  num.mark          id
   <factor>              <IRanges> | <integer> <integer> <character>
1         1 [   846864,  70065480] |         1      2000     NA06993
2         1 [ 70081878,  73381312] |         1       202     NA06993
3         1 [ 73401482, 121226982] |         1      2499     NA06993
4         1 [144908589, 176547669] |         1      1294     NA06993
5         1 [176548473, 178534221] |         1       101     NA06993
6         1 [178659933, 218173294] |         1      1898     NA06993
7         1 [218219379, 219806187] |         1        99     NA06993
```

```
8         1 [219825554, 240236045] |         1       900     NA06993
9         1 [240253320, 240351241] |         1         5     NA06993
10        1 [240363241, 248794371] |         1       160     NA06993
11        1 [   170616,  88467934] |         2        40     NA06993
12        1 [102998279, 240767758] |         2        60     NA06993
   start.index end.index     state        LLR
     <integer> <integer> <integer>  <numeric>
1             0         0         3   0.000000
2             0         0         5 306.031321
3             0         0         3   0.000000
4             0         0         3   0.000000
5             0         0         1 334.520760
6             0         0         3   0.000000
7             0         0         1 378.708245
8             0         0         3   0.000000
9             0         0         1   4.617305
10            0         0         3   0.000000
11            0         0         3   0.000000
12            0         0         3   0.000000
```

```
> annotation(oligoSet) <- ann
```



3.3 Other options

Copy number. A HMM for copy number only (e.g., if genotypes are ignored or are unavailable) can be fit as follows.

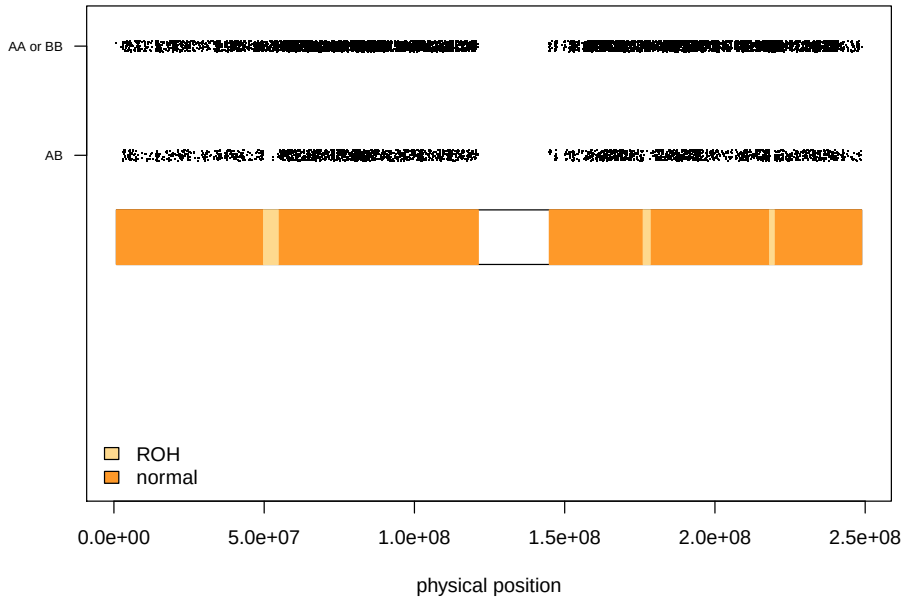
```
> cnSet <- new("CopyNumberSet",
+               copyNumber=log2(locusLevelData[["copynumber"]]/100),
```

```
+              annotation=locusLevelData[["platform"]])
> cnSet <- cnSet[chromosome(cnSet) <= 2 & !is.na(chromosome(cnSet)), ]
> hmmOpts <- hmm.setup(cnSet, is.log=TRUE)
> fit.cn <- hmm(cnSet, hmmOpts)

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

Regions of homozygosity. A HMM for genotype-only data can be used to find long stretches of homozygosity. Note that hemizygous deletions are also identified as 'ROH' when copy number is ignored (as the biallelic genotypte call in a hemizygous deletions tends to be all homozygous calls).

```
> snpSet <- new("SnpSet",
+               call=locusLevelData[["genotypes"]],
+               callProbability=locusLevelData[["crlmmConfidence"]],
+               annotation=locusLevelData[["platform"]])
> featureData(snpSet) <- addFeatureAnnotation(snpSet)
> snpSet <- snpSet[chromosome(snpSet) < 3, ]
> hmmOpts <- hmm.setup(snpSet)
> fit.gt <- hmm(snpSet, hmmOpts)

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

A suggested visualization:



4 Quality control

4.1 Outliers

Copy number outliers can cause the HMM to become too jumpy. One approach to reduce the influence of outliers is to some *light*-smoothing prior to fitting the HMM, as suggested in the R package `DNAcopy`. For instance, one could identify outliers by some criteria and then average the outliers using the estimates from neigboring probes. Here, we use the defaults in `smooth.CNA`.

```
> if(require("DNAcopy")){
+         ##create an outlier
+         copyNumber(cnSet)[50] <- 10
+         copyNumber(cnSet)[45:55]
+         cnaObj <- CNA(genomdat=copyNumber(cnSet),
+                       chrom=chromosome(cnSet),
+                       maploc=position(cnSet),
+                       data.type="logratio",
+                       sampleid=sampleNames(cnSet))
+         smoothed.cnaObj <- smooth.CNA(cnaObj)
+         copyNumber(cnSet) <- matrix(smoothed.cnaObj[, "NA06993"], nrow(cnSet), 1)
+         copyNumber(cnSet)[50]
+ }
```

```
[1] 1.281046
```

One could also increase the value of `TAUP` in the viterbi algorithm to encourage a fit with fewer jumps. Note that with improved estimates of copy number uncertainty, many of these *post-hoc* approaches for addressing outliers would be less critical.

4.2 Batch effects

VanillaICE can be used in conjunction with the *crlmm* package to reduce batch effects. See [3] for details regarding the *crlmm* package.

5 Session Information

The version number of R and packages loaded for generating the vignette were:

- R version 2.14.0 (2011-10-31), `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=C`, `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, stats, utils
- Other packages: Biobase 2.14.0, DBI 0.2-5, DNAcopy 1.28.0, IRanges 1.12.1, RColorBrewer 1.0-5, RSQLite 0.10.0, VanillaICE 1.16.1, oligo 1.18.0, oligoClasses 1.16.0, pd.mapping50k.hind240 1.4.0, pd.mapping50k.xba240 1.4.0
- Loaded via a namespace (and not attached): Biostrings 2.22.0, SNPchip 1.18.0, affxparser 1.26.1, affyio 1.22.0, bit 1.1-7, ff 2.2-3, grid 2.14.0, lattice 0.20-0, preprocessCore 1.16.0, splines 2.14.0, tools 2.14.0, zlibbioc 1.0.0

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] 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.

[3] Robert B Scharpf, Ingo Ruczinski, Benilton Carvalho, Betty Doan, Aravinda Chakravarti, and Rafael Irizarry. A multilevel model to address batch effects in copy number estimation using snp arrays. *Biostatistics*, 2010.

[4] Kai Wang, Mingyao Li, Dexter Hadley, Rui Liu, Joseph Glessner, Struan F A Grant, Hakon Hakonarson, and Maja Bucan. Penncnv: an integrated hidden markov model designed for high-resolution copy number variation detection in whole-genome snp genotyping data. *Genome Res*, 17(11):1665–1674, Nov 2007.

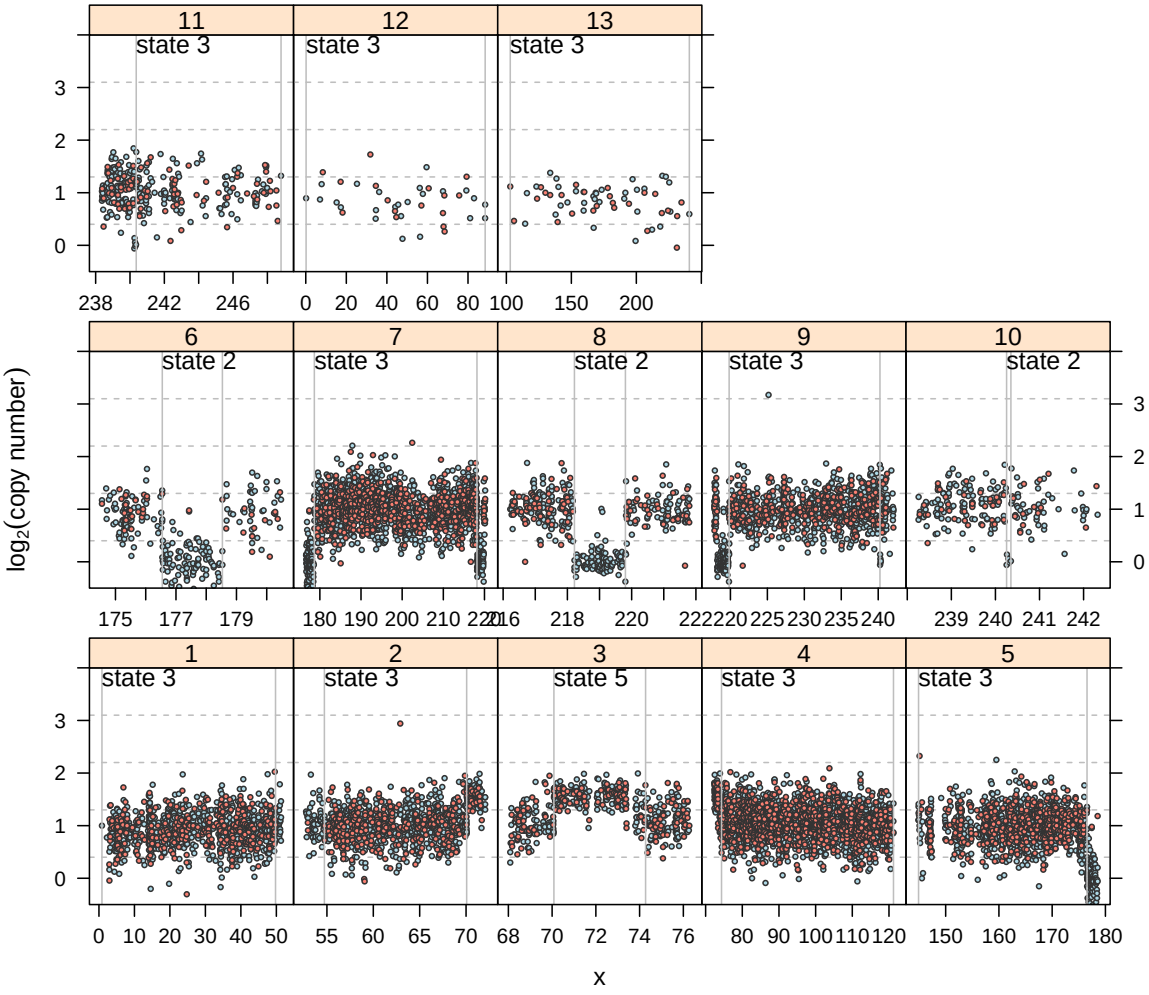


Figure 2: The method `xyplot` is used to create a multi-panel display of alterations in a single sample. Each panel displays a single copy number alteration detected by the HMM that is boxed by a rectangle. The alteration is framed by specifying the number of basepairs to plot upstream and downstream of the alteration. Here, we used a frame of 2 Mb. Homozygous SNPs with diallelic geotypes 'AA' and 'BB' are shaded blue; SNPs with diallelic genotype call 'AB' are shaded in red.