SMAP: A Segmental Maximum A Posteriori Approach to Array-CGH Copy Number Profiling

Robin Andersson, Jan Komorowski

April 30, 2008

The Linnaeus Centre for Bioinformatics,
Uppsala University, Uppsala, Sweden

robin.andersson@lcb.uu.se

Contents

1 Overview

This document describes classes and functions in the *SMAP* package for copy number profiling of array-CGH data. The data analyzed is glioblastoma multiforme sample *G24460* obtained from Teresita Diaz de Ståhl, Uppsala University, Sweden.

```
> library(SMAP)
```

2 Observations

The glioblastoma multiforme data is stored in a *data.frame* which needs to be converted to a *SMAPObservations* object prior to analysis. The required arguments for the `SMAPObservations` constructor function are:

value A numeric vector of intensity ratios for each clone on the array

chromosome A character vector of chromosomes annotated to the clones on the array

startPosition A numeric vector of start positions (bp) of the sequences corresponding to the clones on the array

endPosition A numeric vector of end positions (bp) of the sequences corresponding to the clones on the array

Optional data are:

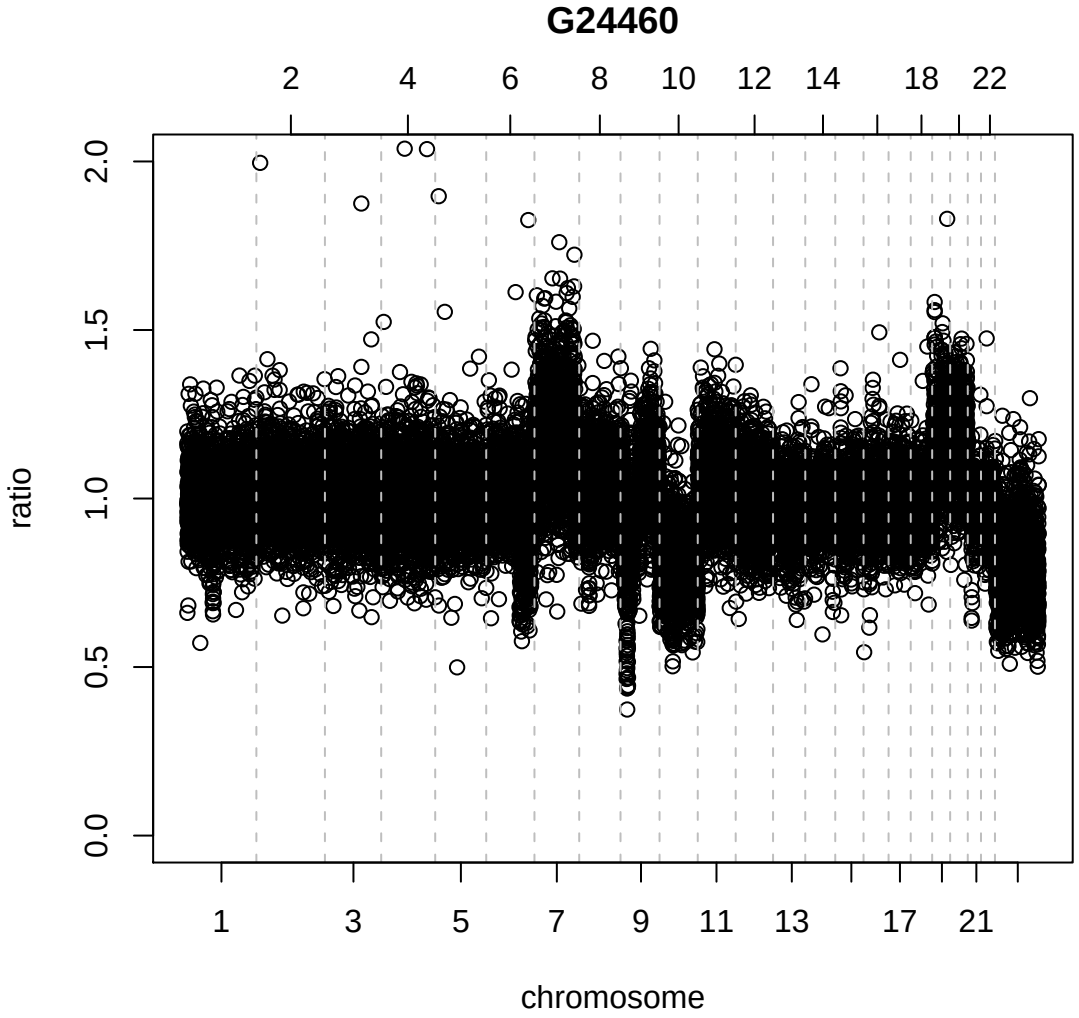
name The name (identifier) of the array

reporterId Identifiers of the clones on the array

```
> data(GBM)
> obs <- SMAPObservations(value = as.numeric(GBM[, 2]), chromosome = as.character(GBM[,
+     3]), startPosition = as.numeric(GBM[, 4]), endPosition = as.numeric(GBM[,
+     5]), name = "G24460", reporterId = as.character(GBM[, 1]))
```

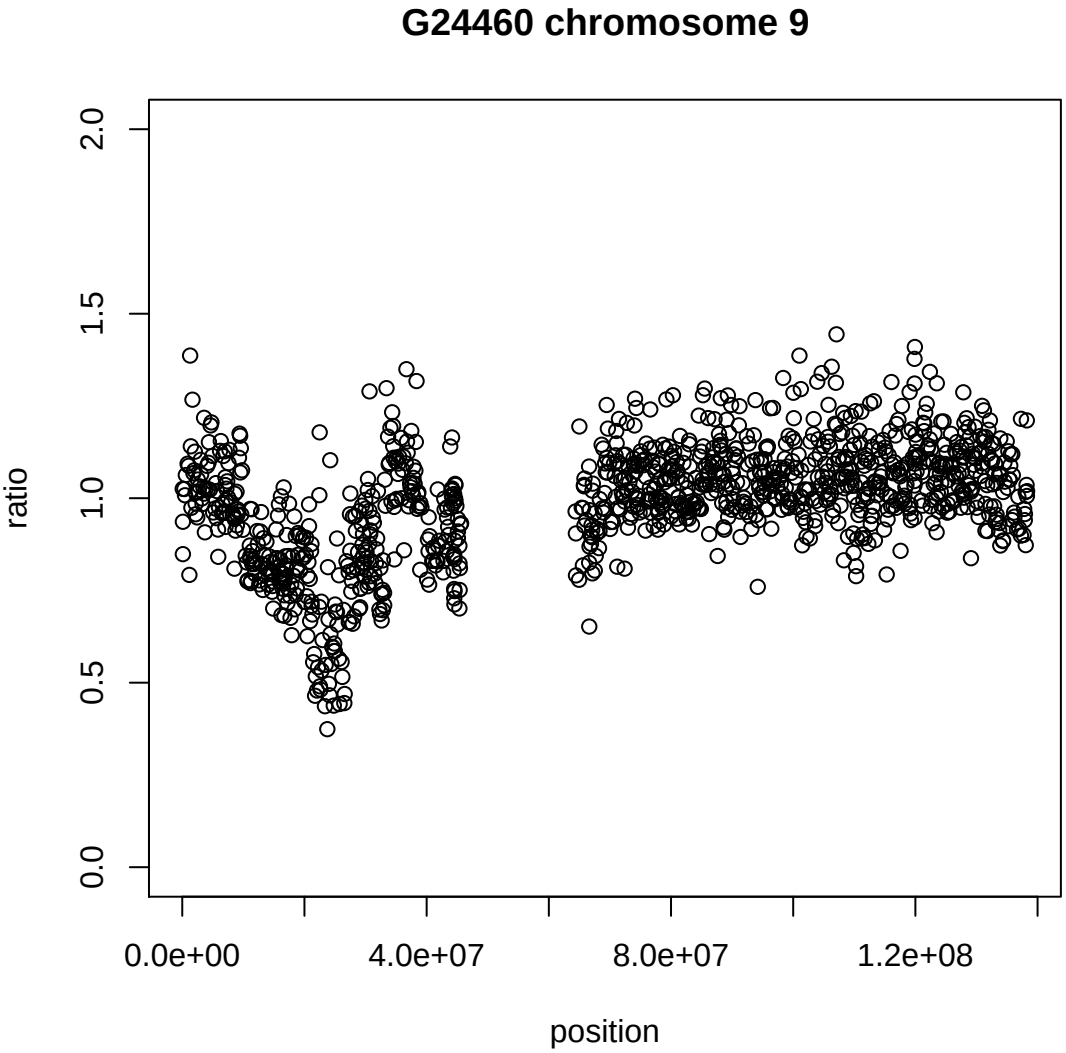
The observations can be visualized by using the generic `plot` function on the *SMAPObservations* object. If multiple chromosomes are present, the chromosomes are separated by vertical dashed lines and indexed on the horizontal axis.

```
> plot(obs, ylab = "ratio", ylim = c(0, 2))
```



Subsets of observations may also be plotted using general subscripts. For instance, chromosome 9 may be plotted in the following manner:

```
> ids <- which(chromosome(obs) == "9")
> plot(obs[ids], ylab = "ratio", ylim = c(0, 2), main = paste(name(obs),
+     "chromosome 9"))
```



The observations plotted in this example has been normalized using the `normalizeWithinArrays` function in the *limma* package.

3 A Hidden Markov Model for copy number assignments

SMAP uses a Hidden Markov Model (HMM) to model the copy number assignments. We recommend using a six state model describing states corresponding to homozygous and heterozygous deletions, normal, one copy gain, two copy gain, and amplification. A *SMAPHMM* class is used in the *SMAP* package to manage HMMs and initiated using the `SMAPHMM` function. The required arguments to `SMAPHMM` are:

noStates The number of hidden states in the HMM

Phi A *noStates* * 2 matrix of Gaussian distributions associated with each hidden state, the first

column described means and the second described standard deviations

Optional arguments to `SMAPHMM` are:

A A *noStates* * *noStates* transition probability matrix (probabilities of moving between states in the HMM)

Pi A numeric vector of initial probabilities (probabilities of starting in each state)

initTrans The probability of changing state in the HMM (used if *A* is *NULL*), defaults to $0.2/(noStates-1)$ which means the probability of staying in the same state is 0.8

Initiate a *SMAPHMM* Hidden Markov Model object with 6 states:

```
> init.means <- c(0.4, 0.7, 1, 1.3, 1.6, 3)
> init.sds <- rep(0.1, 6)
> phi <- cbind(init.means, init.sds)
> hmm <- SMAPHMM(noStates = 6, Phi = phi, initTrans = 0.02)
> hmm

An object of class "SMAPHMM"

Slot "A":
     1    2    3    4    5    6
1 0.90 0.02 0.02 0.02 0.02 0.02
2 0.02 0.90 0.02 0.02 0.02 0.02
3 0.02 0.02 0.90 0.02 0.02 0.02
4 0.02 0.02 0.02 0.90 0.02 0.02
5 0.02 0.02 0.02 0.02 0.90 0.02
6 0.02 0.02 0.02 0.02 0.02 0.90

Slot "Pi":
[1] 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667

Slot "Phi":
  mean  SD
1  0.4 0.1
2  0.7 0.1
3  1.0 0.1
4  1.3 0.1
5  1.6 0.1
6  3.0 0.1
```

4 Copy number profiling by segmental a posteriori maximization

Given a set of observations O and a HMM λ, the `smap` function finds the most probable state sequence Q (assignment of clones to HMM states) in the HMM by maximizing the joint posterior probability of Q and λ given O. This is done by, starting with an initial estimate of the HMM, alternating optimization of the joint posterior probability over Q and λ until no further improvements can be made or a maximum number of iterations has been reached. Optimization over Q and λ is done

using the Viterbi algorithm and a gradient descent scheme with individual learning rate adaptation, respectively.

The `smap` function requires the following arguments:

x A *SMAPHMM* object

Obs A *SMAPObservations* object

Other arguments (default values) are:

eta **(0.005)** Initial learning rate in the gradient descent optimization

overlap **(TRUE)** If *TRUE*, genomic overlap of clones is considered in the optimization

distance **(TRUE)** If *TRUE*, genomic distance between clones is considered in the optimization, in terms of distance based transition probabilities

chrom.wise **(FALSE)** If *TRUE*, the observations are analyzed chromosome-wise rather than genome-wise

verbose **(1)** Specifies the amount of output produced; 0 means no information and 3 a lot of information

L **(5000000)** A positive length parameter that controls the convergence of distance based transition probabilities towards 1 / *noStates(x)*

All arguments are described in detail in the man pages for `smap`.

The choice of parameters sent to the `smap` function as well as the initial HMM used may influence the results. A too high or too low value of *eta* may reduce the ability to fit the HMM to the data. The initial estimates of changing state in the HMM may also influence the results. A too high value may find too much variation in the data whereas a too small value may restrain the ability of finding true variations in the data. If *chrom.wise* is set to FALSE (recommended), one HMM is fit to all data which controls the adaptation of HMM parameters to local non-biological trends which may be present in some chromosome only. If set to TRUE, one HMM per chromosome is trained and the resulting state distributions may conflict between chromosomes.

The *overlap* argument specifies whether overlap should be taken into account during optimization. If set to TRUE, each observation is considered to be drawn from a mixture of distributions where the mixture proportions are determined in terms of relative overlap between clones.

Run `smap` on the *SMAPHMM* and *SMAPObservations* objects.

```
> profile <- smap(hmm, obs, verbose = 2)
```

```
Calculating overlaps
RUNNING SMAP ON 'G24460'
init P: -160886.218423
Iteration 1, P: -140380.311018
Iteration 2, P: -133481.49274
Iteration 3, P: -133481.49274
Optimal P: -133481.49274  found after 2 iterations
```

The result of the `smap` run may be retrieved by accessing the Q slot of the resulting *SMAPProfile* object.

```
> Q(profile)
```

The resulting (adapted) HMM may be examined by accessing the HMM slot of the *SMAPProfile*.

```
> Phi(HMM(profile))
```

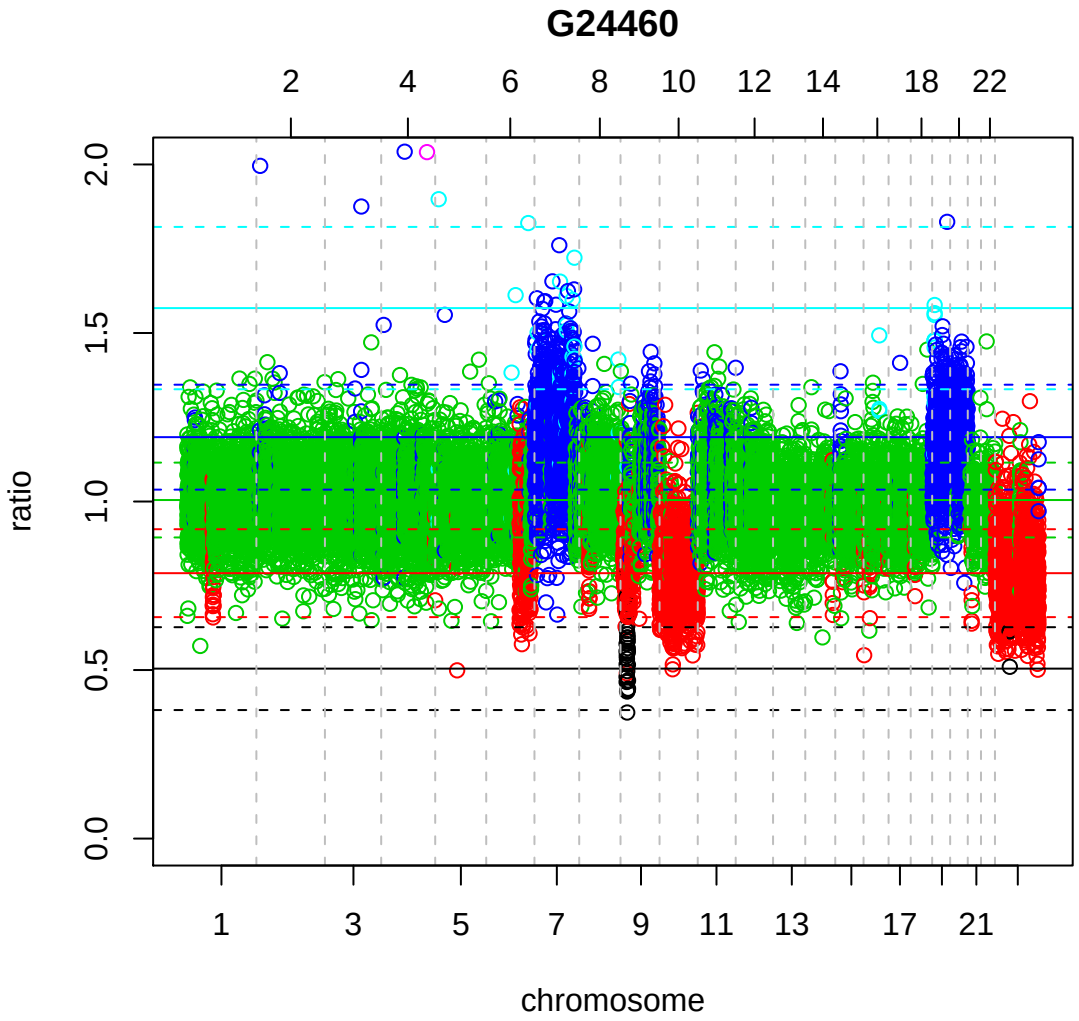
```
       mean        SD
1 0.5042062 0.1229025
2 0.7873844 0.1304443
3 1.0043969 0.1110023
4 1.1909501 0.1556679
5 1.5739008 0.2408134
6 2.9986192 0.7516224
```

5 Plotting results

The results of the `smap` run may be visualized using the generic `plot` function.

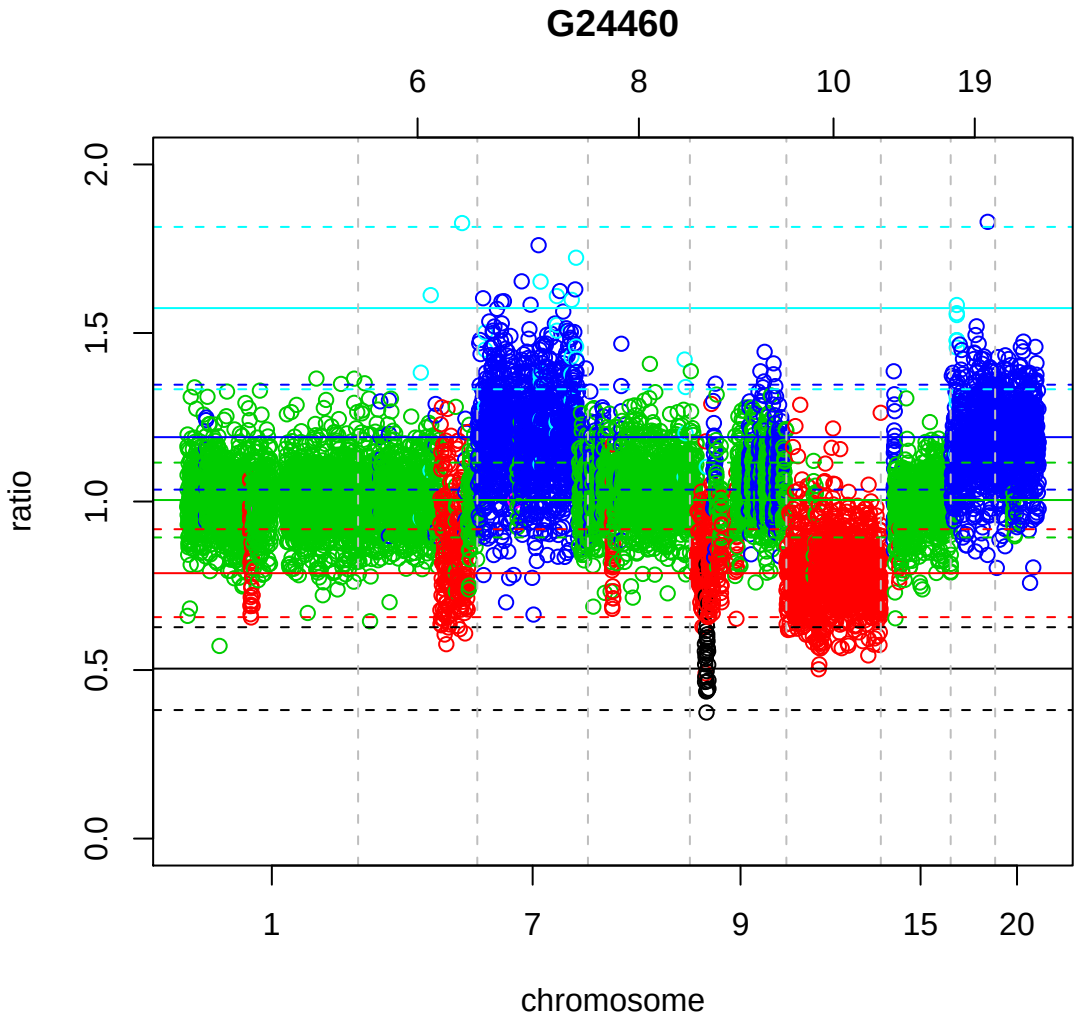
Plot results of all data:

```
> plot(profile, ylab = "ratio", ylim = c(0, 2))
```



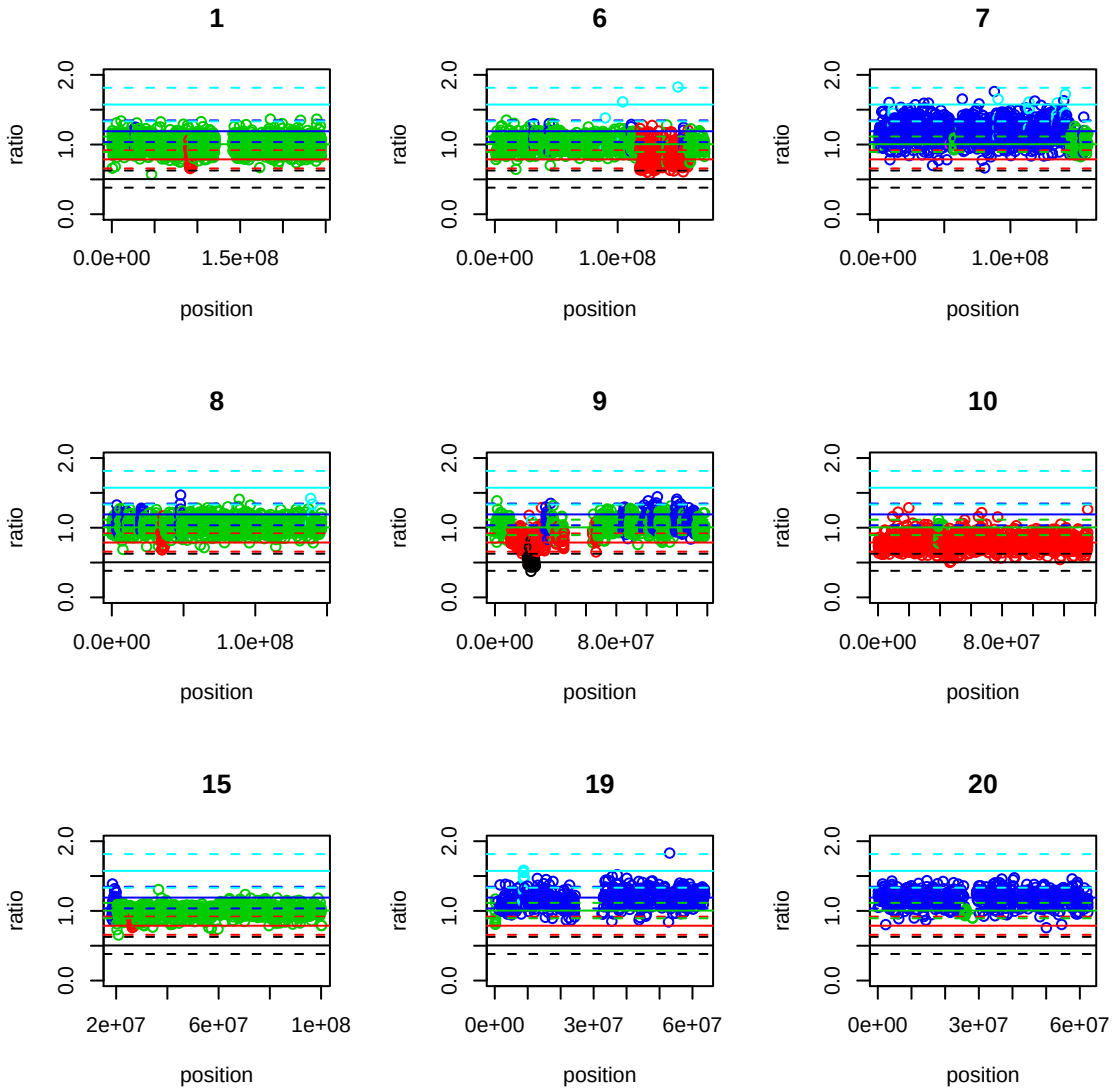
Plot chromosomes with aberrations

```
> chrom.selection <- as.character(c(1, 6, 7, 8, 9, 10, 15, 19,
+     20))
> selection <- which(chromosome(obs) %in% chrom.selection)
> plot(profile[selection], ylab = "ratio", ylim = c(0, 2))
```



Plot all chromosomes with aberrations separately:

```
> par(mfrow = c(3, 3))
> for (c in chrom.selection) {
+     ids <- which(chromosome(obs) == c)
+     plot(profile[ids], ylab = "ratio", ylim = c(0, 2), main = c)
+ }
```


1
6
7
8
9
10
15
19
20
ratio
position
0.0
1.0
2.0
0.0e+00
1.5e+08
1.0e+08
8.0e+07
2e+07
6e+07
1e+08
0e+00
3e+07
6e+07