PSSeg: Parent-Specifc copy number segmentation

M. Pierre-Jean, G. Rigaill, P. Neuvial

May 30, 2014

Abstract

This vignette describes how to use the `jointseg` package to partition bivariate DNA copy number signals from SNP array data into segments of constant parent-specific copy number. We demonstrate the use of the `PSSeg` function of this package for applying two different strategies. Both strategies consist in first identifying a list of candidate change points through a fast (greedy) segmentation method, and then to prune this list is using dynamic programming [1]. The first segmentation method is Recursive Binary Segmentation (RBS, [2]), and the second one is a group fused LARS approach (GFLARS, [5]), for which we ported the original `Matlab` implementation into `R`.

keywords: segmentation, change point model, binary segmentation, dynamic programming, DNA copy number, parent-specific copy number.

Contents


```
library(jointseg)
```

1 Preparing data to be segmented

`PSSeg` requires normalized copy number signals, in the form of total copy number estimates and allele B fractions for tumor, the (germline) genotype of SNP. Loci are assumed to come from a single chromosome and to be ordered by genomic position.

For illustration, we show of such a data set may be created from real data. We use data from a public SNP array data set, which is distributed in the `acnr` package (on which the `jointseg` package depends).

```
data <- loadCnRegionData(platform = "GSE29172", tumorFraction = 1)
str(data)
```

```
## 'data.frame': 192667 obs. of  4 variables:
##  $ c       : num  0.909 0.859 1.304 0.647 0.947 ...
##  $ b       : num  NaN NaN NaN NaN NaN NaN NaN -0.035 NaN NaN ...
##  $ genotype: num  NA NA NA NA NA NA NA 0 NA NA ...
##  $ region  : chr  "(0,1)" "(0,1)" "(0,1)" "(0,1)" ...
```

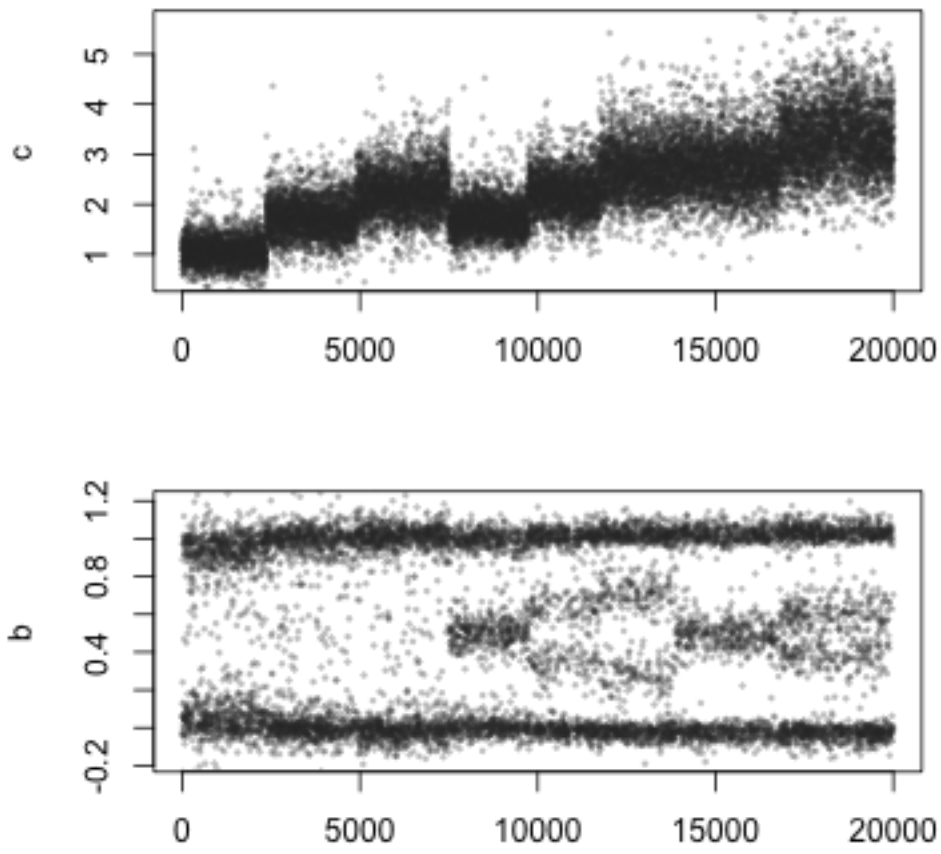
This data set consists of copy number signals from 8 types of genomic regions:

```
table(data[["region"]])

##
## (0,1) (0,2) (0,3) (1,1) (1,2) (1,3) (2,2) (2,3)
## 22615 24135 25405 21539 19048 20903 27924 31098
```

These regions are coded as (C_1, C_2), where C_1 denotes the minor copy number and C_2 denotes the major copy number, i.e. the smallest and the largest of the two parental copy numbers (see e.g. [4] for more detailed definitions). For example, $(1, 1)$ corresponds to a normal state, $(0, 1)$ to an hemizygous deletion, $(1, 2)$ to a single copy gain and $(0, 2)$ to a copy-neutral LOH (loss of heterowygosity).

```
idxs <- sort(sample(1:nrow(data), 20000))
plotSeg(data[idxs, ])
```



These real data can then be used to create a realistic DNA copy number profile of user-defined length, and harboring a user-defined number of breakpoints. This is done using the `getCopyNumberDataByResampling`

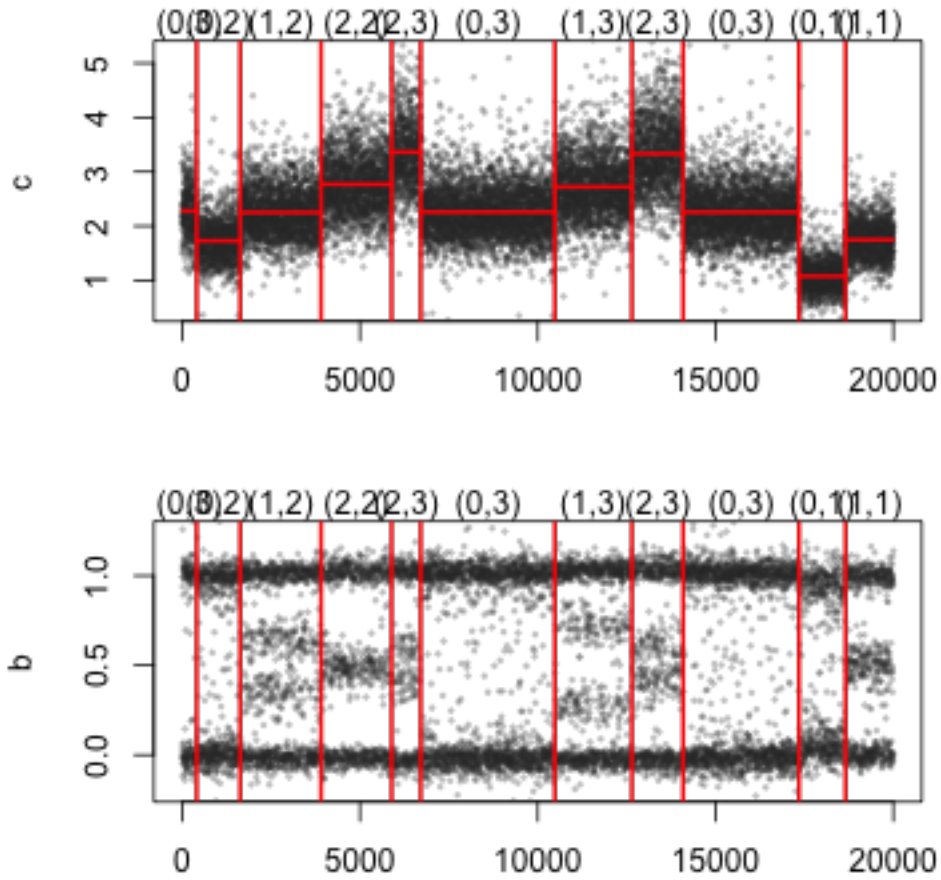
function. Breakpoint positions are drawn uniformly) among all possible loci. Between two breakpoints, the copy number state corresponds to one of the types of regions in `data`, and each data point is drawn with replacement from the corresponding true copy number signal from the region. More options are available from the documentation of `getCopyNumberDataByResampling`.

```
K <- 10
bkp <- c(408, 1632, 3905, 5890, 6709, 10481, 12647, 14089, 17345, 18657)
len <- 20000
sim <- getCopyNumberDataByResampling(len, bkp = bkp, minLength = 500, regData = data)
datS <- sim$profile
str(datS)

## 'data.frame': 20000 obs. of  4 variables:
##  $ c       : num  2.22 1.51 1.6 2.4 3.17 ...
##  $ b       : num  -0.013 NaN NaN NaN 0.06 NaN NaN NaN -0.118 0.008 ...
##  $ genotype: num  0 NA NA NA 0 NA NA NA 0.5 0 ...
##  $ region  : chr  "(0,3)" "(0,3)" "(0,3)" "(0,3)" ...
```

The resulting copy-number profile is plotted below.

```
plotSeg(datS, sim$bkp)
```



2 PSSeg segmentation using RBS

We can now use the `PSSeg` function to segment signals. The method consists in three steps:

1. run a fast (yet approximate) segmentation on these signals in order to obtain a set of (at most hundreds of) candidate change points. This is done using Recursive Binary Segmentation (RBS) [2];
2. prune the obtained set of change points usign dynamic programming [1]
3. select the best number of change points using a model selection criterion proposed by [3]

2.1 Initial segmentation and pruning

```
resRBS <- PSSeg(data = datS, K = 2 * K, method = "RBS", stat = c("c", "d"), profile = TRUE)
```

Note that this is fast:

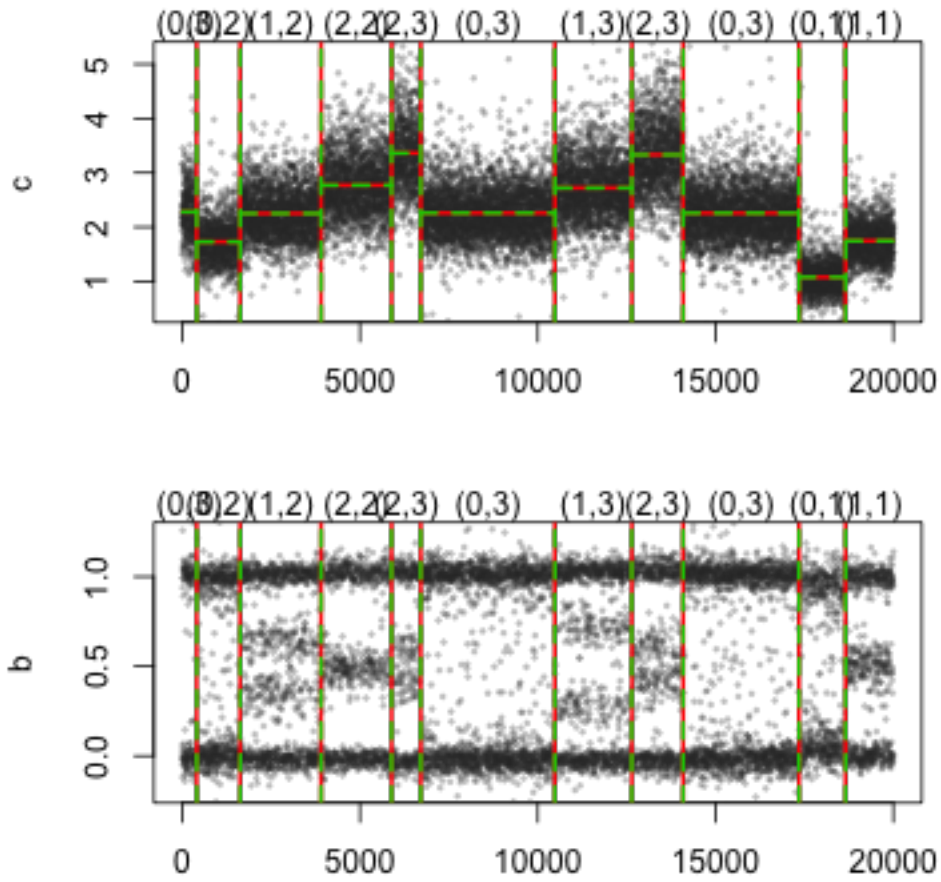
```
resRBS$prof[, "time"]

## segmentation        dpseg
##         0.24         0.00
```

2.2 Plot segmented profile

To plot the PSSeg segmentation results together with the true breakpoints, do :

```
plotSeg(datS, list(true = sim$bkp, est = resRBS$bestBkp))
```



2.3 Results evaluation

The `PSSeg` function returns the original segmentation (by `RBS`), the result of the pruning step, and the best model (among those selected by dynamic programming) according to the criterion proposed by [3].

The quality of the best segmentation can be assessed as follows. The number of true positives (TP) is the number of true change points for which there exists a candidate change point closer than a given tolerance `tol`. The number of false positives is defined as the number of true negatives (all those which are not change points) for which the candidate change points are out of tolerance area and those in tolerance area where there already exists a candidate change point. By construction, $TP \in \{0, 1, \cdots, K\}$ where K is the number of true change points.

```
print(getTpFp(resRBS$bestBkp, sim$bkp, tol = 5))

## TP FP
## 10  0
```

Obviously, this performance measure depends on the chosen tolerance:

```
perf <- sapply(0:10, FUN = function(tol) {
    getTpFp(resRBS$bestBkp, sim$bkp, tol = tol, relax = -1)
})
print(perf)

##     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]
```

```
## TP    4    7    8    9   10   10   10   10   10    10    10
## FP    6    3    2    1    0    0    0    0    0     0     0
```

3 PSSeg segmentation using GFLARS

We can now use the `PSSeg` function to segment signals with GFLARS method only on heterozygous SNP. The method consists in three steps:

1. run a fast (yet approximate) segmentation on these signals in order to obtain a set of (at most hundreds of) candidate change points. This is done using Group Fused Lars [5];
2. prune the obtained set of change points usign dynamic programming [1]
3. select the best number of change points using a model selection criterion proposed by [3]

3.1 Initial segmentation and pruning

```
resGFL <- PSSeg(data = datS, K = 5 * K, method = "GFLars", stat = c("c", "d"), profile = TRUE)

## Warning:  Missing values detected.  Smoothing will be performed.
```

Note that this is fast due to the low number in the data.

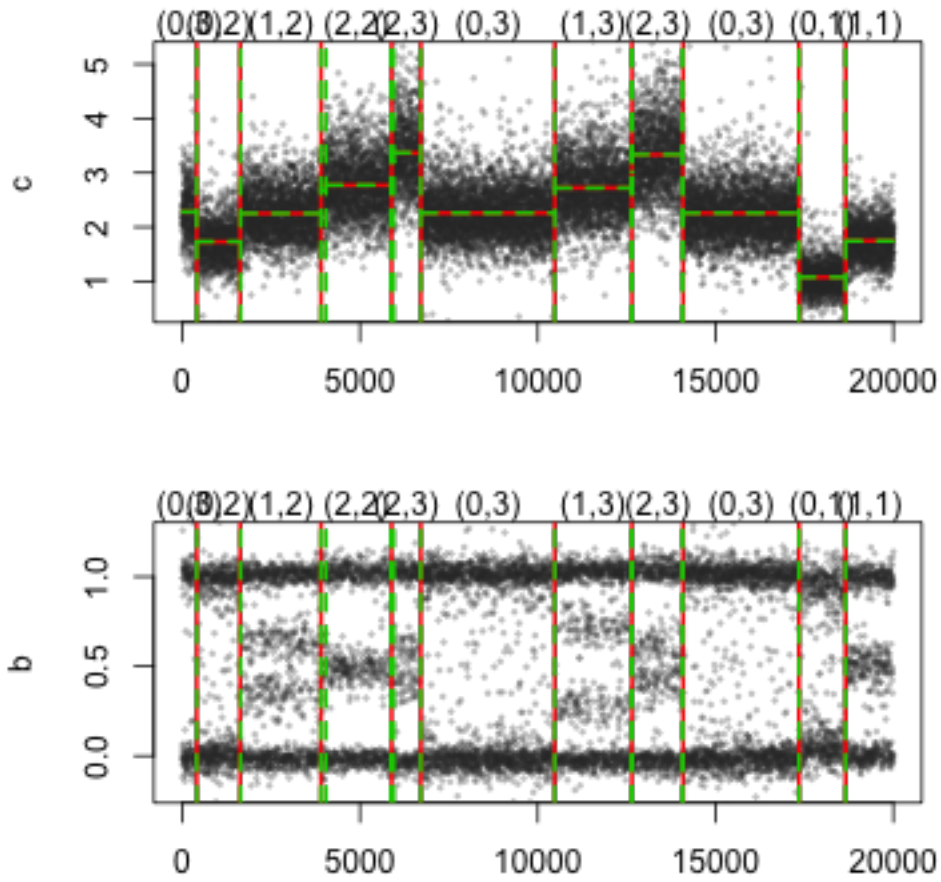
```
resGFL$prof[, "time"]

## segmentation        dpseg
##         0.34         0.00
```

3.2 Plot segmented profile

To plot the PSSeg segmentation results together with the true breakpoints, do :

```
plotSeg(datS, list(true = sim$bkp, est = resGFL$bestBkp))
```



3.3 Results evaluation

The `PSSeg` function returns the original segmentation (by `GFLARS`), the result of the pruning step, and the best model (among those selected by dynamic programming) according to the criterion proposed by [3].

```
print(getTpFp(resGFL$bestBkp, sim$bkp, tol = 15))

## TP FP
##  9  5
```

Obviously, this performance measure depends on the chosen tolerance:

```
perf <- sapply(0:20, FUN = function(tol) {
    getTpFp(resGFL$bestBkp, sim$bkp, tol = tol)
})
print(perf)

##    [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13] [,14] [,15] [,16]
## TP    1    2    4    5    6    7    7    7    7     8     9     9     9     9     9     9
## FP   13   12   10    9    8    7    7    7    7     6     5     5     5     5     5     5
##    [,17] [,18] [,19] [,20] [,21]
## TP     9     9     9     9    10
## FP     5     5     5     5     4
```

A Session information

```
sessionInfo()

## R version 3.0.3 (2014-03-06)
## Platform: x86_64-apple-darwin10.8.0 (64-bit)
##
## locale:
## [1] fr_FR.UTF-8/fr_FR.UTF-8/fr_FR.UTF-8/C/fr_FR.UTF-8/fr_FR.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  base
##
## other attached packages:
## [1] jointseg_0.5.0    acnr_0.1.7        R.utils_1.29.8    R.oo_1.18.0
## [5] R.methodsS3_1.6.1 matrixStats_0.9.0 codetools_0.2-8   knitr_1.5
##
## loaded via a namespace (and not attached):
## [1] evaluate_0.5.3 formatR_0.10   highr_0.3      methods_3.0.3  stringr_0.6.2
## [6] tools_3.0.3
```

References

[1] Richard Bellman. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM*, 4(6):284, 1961.

[2] S. Gey and E. Lebarbier. Using cart to detect multiple change points in the mean for large sample. Technical report, Statistics for Systems Biology research group, 2008.

[3] E. Lebarbier. Detecting multiple change-points in the mean of gaussian process by model selection. *Signal processing*, 85(4):717–736, 2005.

[4] Pierre Neuvial, Henrik Bengtsson, and Terence P Speed. Statistical analysis of single nucleotide polymorphism microarrays in cancer studies. In *Handbook of Statistical Bioinformatics*, Springer Handbooks of Computational Statistics. Springer, 1st edition, March 2011.

[5] J.-P. Vert and K. Bleakley. Fast detection of multiple change-points shared by many signals using group LARS. *Advances in Neural Information Processing Systems*, 23:2343–2351, 2010.