PSSeg: Parent-Specifc copy number segmentation

M. Pierre-Jean, P. Neuvial

January 31, 2013

Abstract

This vignette describes how to use the `PSSeg` function from the `jointSeg` package to partition bivariate DNA copy number signals into segments of constant parent-specific copy number. The proposed method [4] starts by identifying a list of candidate change points through a fast (greedy) recursive binary segmentation (RBS). This list is then pruned using dynamic programming [1]. The resulting approach is able to quickly and accurately identify true changes in DNA copy numbers.

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

Contents

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 microarray data set, which is distributed in the `acnr` package (from which the `jointSeg` package depends).

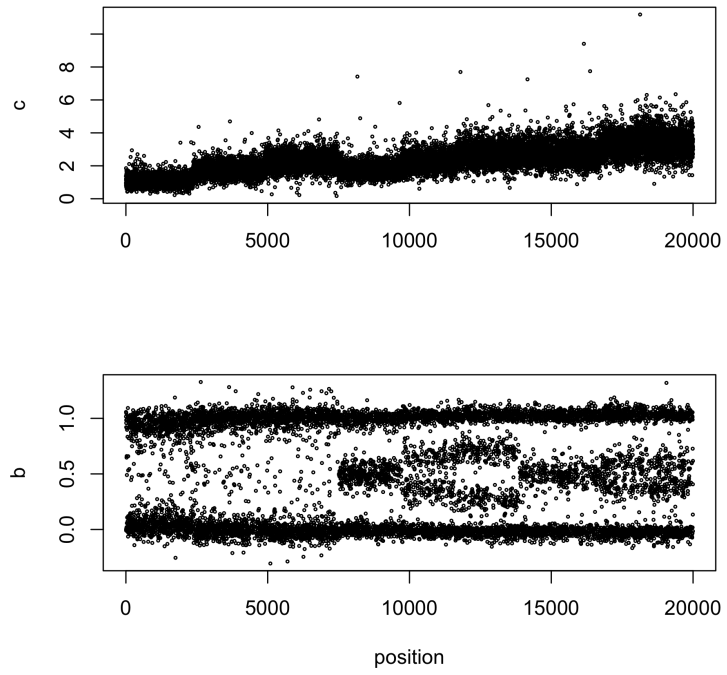
```
> library(jointSeg)
> ## load known real copy number regions
> data <- loadCnRegionData(platform="Affymetrix", tumorFraction=1)
> str(data)
```

```
'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:

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. [3] 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), 2e4))
> 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`.

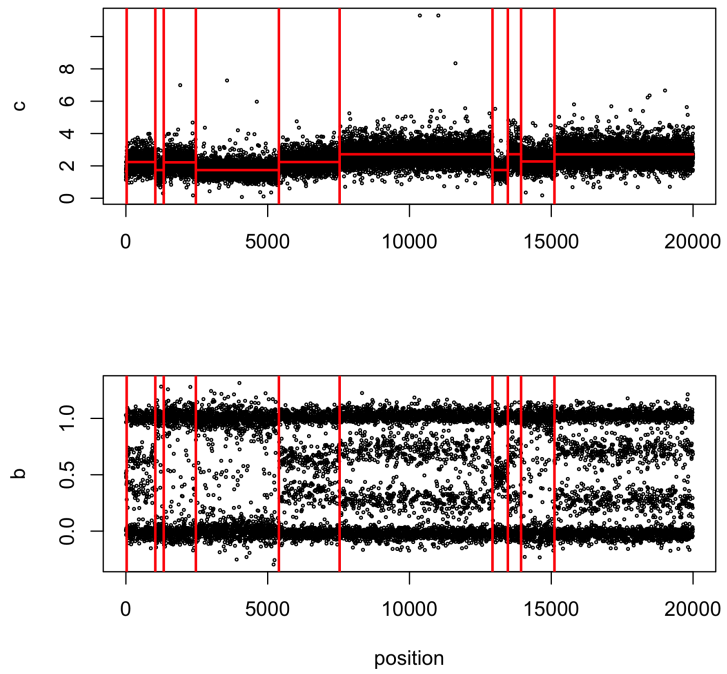
```
> K <- 10
> len <- 2e4
```

```
> sim <- getCopyNumberDataByResampling(len, K, minLength=100, regData=data)
> datS <- sim$profile
> str(datS)

'data.frame':        20000 obs. of  4 variables:
 $ c       : num  1.14 1.59 1.79 2.24 1.3 ...
 $ b       : num  1.029 0.514 0.004 0.01 0.966 ...
 $ genotype: num  1 0.5 0 0 1 0 0.5 0 NA 0.5 ...
 $ region  : chr  "(1,1)" "(1,1)" "(1,1)" "(1,1)" ...
```

The resulting copy-number profile is plotted below.

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



2 PSSeg segmentation

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) [4];
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 [2]

2.1 Initial segmentation and pruning

```
> resRBS <- PSSeg(data=datS, K=2*K, flavor="RBS", prof=TRUE)
```

Note that this is fast:

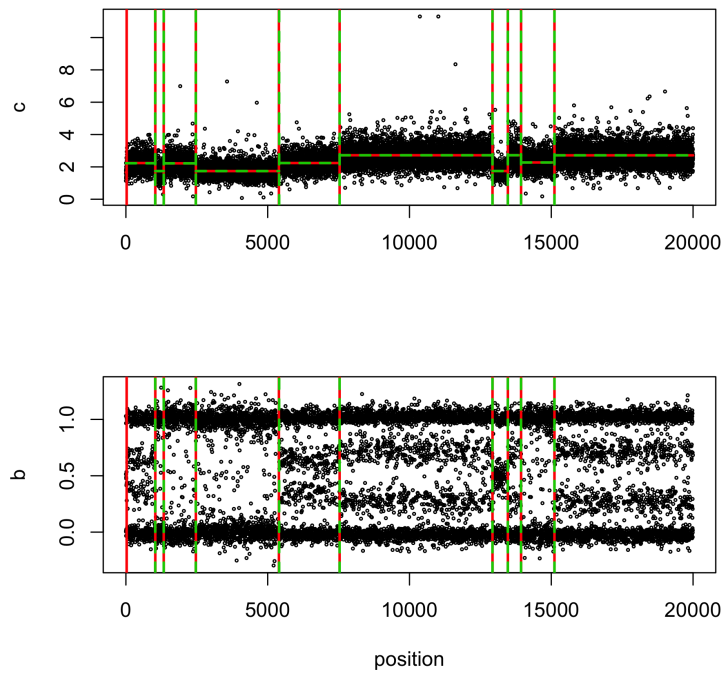
```
> resRBS$prof[, "time"]
```

```
segmentation         dpseg
           0             0
```

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 [2].

The quality of the best segmentation can be assessed by calculating the true positive rate and true negative rate. The true positive rate (TPR) is defined as the proportion of true change points for which there exists a candidate change point closer than a given tolerance `tol`. True negative are defined as the midpoints of intervals between true change points (augmented by

points 0 and $n+1$, where n is the number of loci. The true negative rate (TNR) is the proportion of true negatives for which there is no candidate change point closer than `tol`. By construction, $TPR \in \{0, 1/K, \cdots, 1-1/K, 1\}$, and $TNR \in \{0, 1/(K+1), \cdots, 1-1/(K+1), 1\}$ where K is the number of true change points.

```
> print(getTprTnr(resRBS$bestBkp, sim$bkp, nrow(datS), tol=5))
```

```
      TPR       TNR
0.8000000 0.9090909
```

Obviously, this performance measure depends on the chosen tolerance:

```
> perf <- sapply(0:10, FUN=function(tol) {
+   getTprTnr(resRBS$bestBkp, sim$bkp, nrow(datS), tol=tol)
+ })
> print(perf)
```

```
         [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
TPR 0.3000000 0.5000000 0.7000000 0.8000000 0.8000000 0.8000000 0.8000000
TNR 0.4545455 0.6363636 0.8181818 0.9090909 0.9090909 0.9090909 0.9090909
         [,8]      [,9] [,10] [,11]
TPR 0.8000000 0.8000000   0.9   0.9
TNR 0.9090909 0.9090909   1.0   1.0
```

A Session information

```
> sessionInfo()
```

```
R version 2.15.1 Patched (2012-09-18 r60759)
Platform: i386-apple-darwin9.8.0/i386 (32-bit)

locale:
[1] C/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  methods   base

other attached packages:
[1] jointSeg_0.3.0    acnr_0.1.0        matrixStats_0.6.2 R.utils_1.19.3
[5] R.oo_1.11.4       R.methodsS3_1.4.2

loaded via a namespace (and not attached):
[1] tools_2.15.1
```

References

[1] R. Bellman. Dynamic programming and lagrange multipliers. *Proceedings of the National Academy of Sciences*, 42(10):767, 1956.

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

[3] 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.

[4] Morgane Pierre-Jean and Pierre Neuvial. In preparation.