SegAnnot: an R package for fast segmentation of annotated piecewise constant signals

Toby Dylan Hocking
Guillem Rigaill

March 19, 2013

Abstract

We describe and propose an implementation of a dynamic programming algorithm for the segmentation of annotated piecewise constant signals. The algorithm is exact in the sense that it recovers the best possible segmentation w.r.t. the quadratic loss that agrees with the annotations.

Contents

1 Introduction: annotations motivate dedicated segmentation algorithms

In bioinformatics, there are many noisy assays that attempt to measure chromosomal copy number of a sample of cells. For example, one can recover a piecewise constant signal of copy number from array CGH or SNP microarrays.

Many algorithms have been proposed to analyze these data, and in practice an expert biologist will examine scatterplots to judge if the model segmentation is a good fit to the data. Hocking et˜al. [2012] make this visual criterion for model selection concrete by defining annotated regions that can quantify the accuracy of a segmentation. That study showed that the maximum likelihood segmentation given by the pruned dynamic programming (DP) algorithm of Rigaill [2010] or PELT [Killick et˜al., 2011] resulted in the best breakpoint learning (see extended results of Hocking et˜al. [2012]). However, there are two drawbacks to that approach:

- **(Speed)** The pruned DP or PELT algorithms were not the fastest examined in that comparison.
- **(Fitting)** We may have more than one breakpoint annotation per chromosome. In that case, there may be no maximum likelihood segmentation that agrees with all the annotations.

Furthermore, no optimization algorithms have been specifically designed to exploit the breakpoint annotations. The goal of this work is to characterize a new optimization problem and segmentation algorithm, **SegAnnot**. It exploits the structure of the annotated region data to increase both **Speed** and **Fitting**.

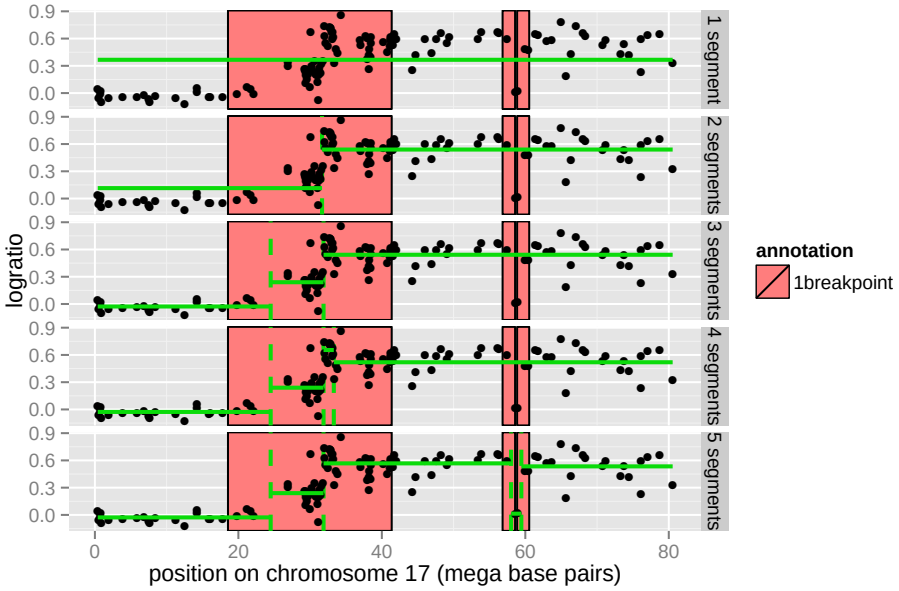
2 Motivating examples

We begin by showing two annotated copy number profiles from low and high density microarrays of neuroblastoma tumors. In both cases, there is no maximum likelihood segmentation that agrees with the breakpoint annotations.

Let us first give a few notations. We analyze a chromosome with D base pairs $\mathcal{X} = \{1, \ldots, D\}$, so the set of all possible breakpoints is $\mathbb{B} = \{1, \ldots, D-1\}$. The microarray gives us a vector of logratio observations $y \in \mathbb{R}^d$ at positions $p \in \mathcal{X}^d$, sorted in increasing order $p_1 < \cdots < p_d$. We define the estimated signal with k segments as

$$\begin{aligned} \hat{y}^k = \arg\min_{x \in \mathbb{R}^d} & \quad ||y - x||_2^2 \\ \text{subject to} & \quad k - 1 = \sum_{j=1}^{d-1} 1_{x_j \neq x_{j+1}}. \end{aligned} \tag{1}$$

Note that we can quickly calculate $\hat{y}^k$ for $k \in \{1, \ldots, k_{\max}\}$ using pruned DP [Rigaill, 2010]. In the figure below, we plot the noisy observations y for one chromosome on a low-density array as $d = 123$ black points.

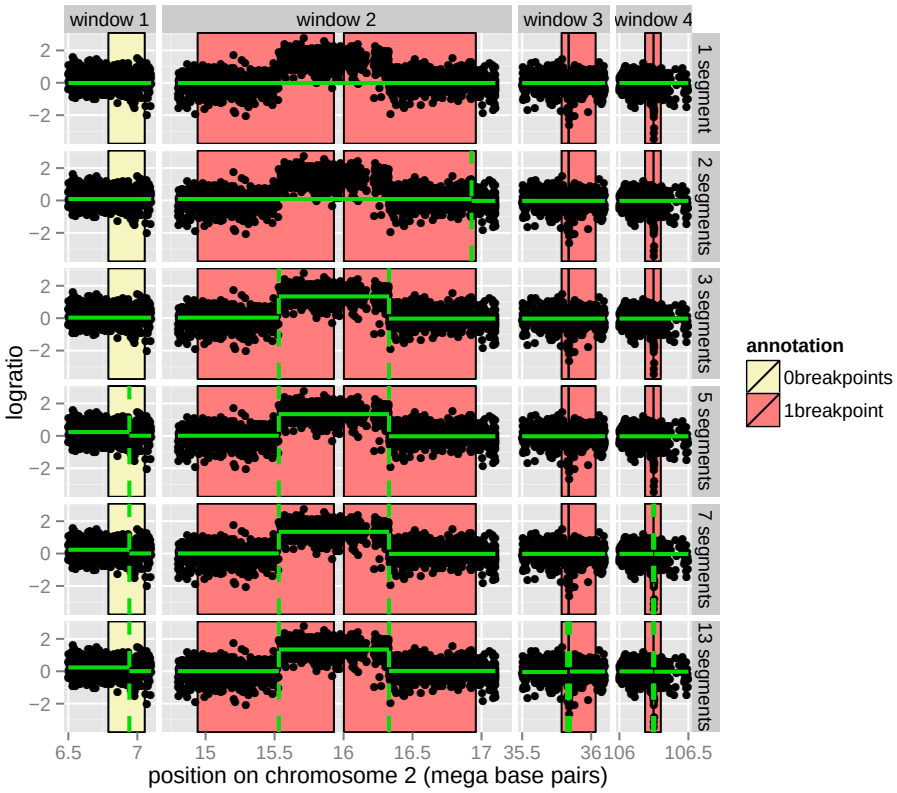


In the figure above, the estimated signals $\hat{y}^k$ are drawn as green lines for $k = 1$ to 5 segments. The cghseg model tells us the points after which a break occurs, not the precise bases. So we define the estimated breakpoint locations shown as vertical green dashed lines using the mean

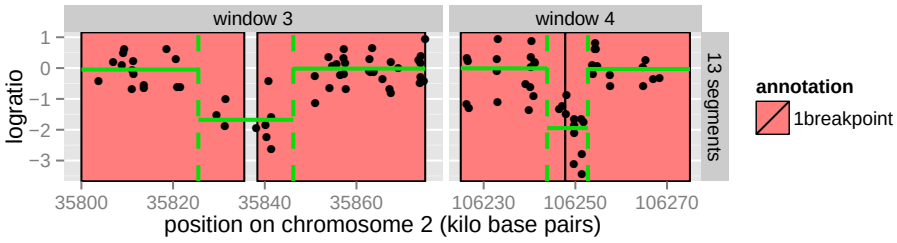
$$\phi(\hat{y}^k, p) = \left\{ \lfloor (p_j + p_{j+1})/2 \rfloor \text{ for all } j \in \{1, \ldots, d-1\} \text{ such that } \hat{y}^k_j \neq \hat{y}^k_{j+1} \right\}. \tag{2}$$

Note that this is a function $\phi : \mathbb{R}^d \times \mathcal{X}^d \to 2^{\mathbb{B}}$ that gives the positions after which there is a break in the estimated signal $\hat{y}^k$. Clearly, none of the models $\hat{y}^1, \ldots, \hat{y}^5$ agree with all 3 of the breakpoint annotations.

The second example we show is a high-density array with 153,663 probes on chromosome 2. We zoom into 4 windows of interest in the plot below, showing only 3,068 of the probes.



Models with $k \in \{4, 6, 8, \ldots, 12\}$ segments are not shown since each is identical to one of the shown models on these four windows. Clearly, no model agrees with all of the breakpoint annotations. Indeed, it is only for $k = 13$ segments that all breakpoint regions (red) are correctly identified, but a breakpoint is identified in a flat region (yellow) as soon as $k \geq 5$. To show detail of windows 3 and 4, we zoom in the plot below to clearly show their small $\approx$ 10kb losses.



The fact that the maximum-likelihood segmentation sometimes cannot match the expert's annotations is unsatisfying. In those cases, one would like to perform constrained segmentation, and recover the best segmentation that satisfies the expert's annotations. We explore this idea in the rest of this paper and propose an exact algorithm to solve it.

3 Segmentation via annotation-aware optimization

First, we will give a precise definition of a segmentation and the breakpoint annotations. Then, we will define what it means for a segmentation to be consistent with the annotations. Finally, we will use this definition to formulate an optimization problem.

3.1 Definition of a segmentation

We represent our signal to segment as a vector $y \in \mathbb{R}^d$, and we want to find a **segmentation** $m \in \mathcal{M}_{K,d}$, the set of all possible segmentations with K segments. We define a **segment** $r_k(m) = [\![\tau_k, \tau_{k+1}[\![$ in terms of its limiting indices $\tau_k, \tau_{k+1} \in [\![1, d]\!]$. For $K > 1$, we define $\tau_2, \ldots, \tau_K$ to be the **breakpoints**. We can write any segmentation $m \in \mathcal{M}_{K,d}$ as a set of K segments, in this form:

$$\begin{array}{rcccccccccc} m & = & \{ & r_1(m) & , & r_2(m) & , & \ldots & , & r_{K-1}(m) & , & r_K(m) & \} \\ & = & \{ & [\![\tau_1 = 1, \tau_2[\![& , & [\![\tau_2, \tau_3[\![& , & \ldots & , & [\![\tau_{K-1}, \tau_K[\![& , & [\![\tau_K, \tau_{K+1} = d+1[\![& \} \end{array} \tag{3}$$

How to choose among the $|\mathcal{M}_{K,d}| = \binom{d-1}{K-1}$ possible segmentations? We would like

1. the reconstruction error to be as low as possible, and
2. the breakpoint annotations to be respected.

3.2 Definition of the breakpoint annotations

The breakpoint annotations represent prior knowledge about where breaks should occur in any valid model. For example, they can represent an expert's desired segmentation on inspection of a scatterplot of observations y against position p.

Assume the expert annotates J regions $A_1, \ldots, A_J$ and that each region is defined as $A_j = [\![\underline{\alpha}_j, \overline{\alpha}_j[\![$ Note that the smallest regions on the left and right designate the gaps between the last 2 points, i.e. $[\![1, 2[\![$ and $[\![d-1, d[\![$. And the largest possible annotated region is $[\![1, d[\![$. Furthermore, the maximal number of breakpoints per region is $\overline{\alpha}_j - \underline{\alpha}_j$.

For each region A_j the annotator associates a set of acceptable breakpoint counts a_j, typically defined as one of the following:

in words	abbreviation	breakpoint counts a_j
no breakpoints, flat	0	$\{0\}$
exactly 1 breakpoint	1	$\{1\}$
exactly b breakpoints	b	$\{b\}$
1 or more breakpoints	1+	$\{1, \ldots, \underline{\alpha}_j - \overline{\alpha}_j\}$
no information	0+	$\{0, 1, \ldots, \underline{\alpha}_j - \overline{\alpha}_j\}$

3.3 Definition of a consistent segmentation

For any segmentation m we can write $1 = \tau_1 < \tau_2 < \cdots < \tau_K < \tau_{K+1} = d + 1$. For ̃$2 \leq i \leq K$ the following function indicates when breakpoint ̃τ_i is inside annotation ̃A_j:

$$b(\tau_i, A_j) = \begin{cases} 1 & \underline{\alpha}_j < \tau_i \leq \overline{\alpha}_j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and we can count the number of breakpoints inside annotation ̃A_j using

$$B(m, A_j) = \sum_{i=2}^{K} b(\tau_i, A_j) \tag{5}$$

Definition 1. *We say that segmentation ̃m is **consistent** with the annotations ̃A if for all annotated regions ̃j, we have $B(m, A_j) \in a_j$.*

3.4 The annotation-aware segmentation problem

Let us define the set $\mathcal{M}_{K,d,A} \subseteq \mathcal{M}_{K,d}$ as the set of all segmentations of d points with K segments which are **consistent** with the annotations A. One way to formalize the annotation-constrained optimization problem is to just replace the set $\mathcal{M}_{K,d}$ with $\mathcal{M}_{K,d,A}$:

$$\min_{m \in \mathcal{M}_{K,d,A}} \sum_{r \in m} \min_{\mu \in \mathbb{R}} \sum_{i \in r} (Y_i - \mu)^2 \tag{6}$$

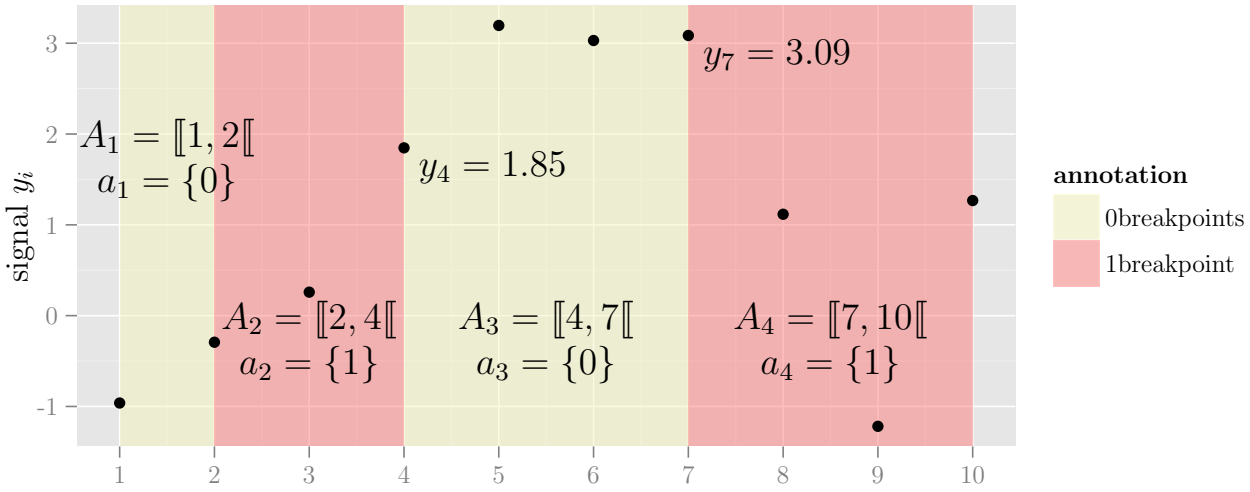
Proposition 1. *The minimum number of segments k such that $|\mathcal{M}_{k,d,A}| > 0$ is*

$$k_{\min} = 1 + \sum_{j=1}^{J} \min a_j,$$

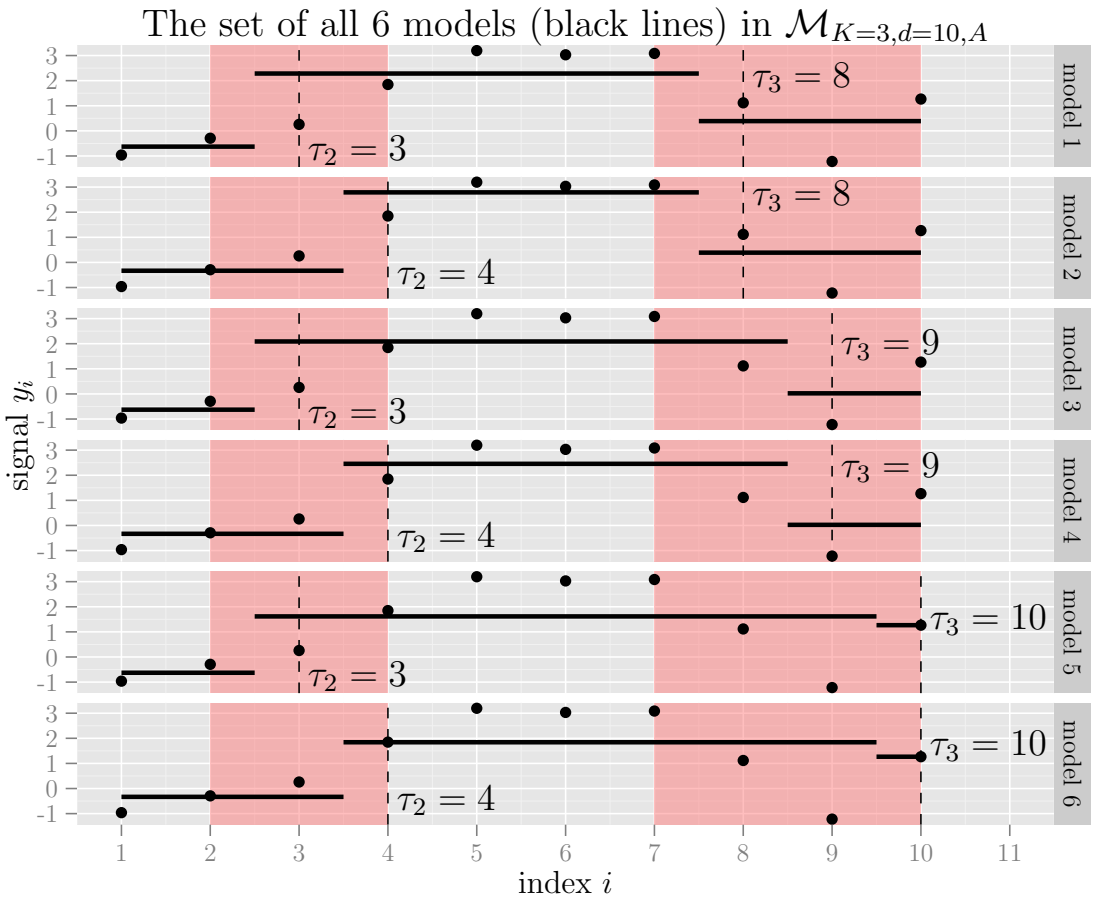
so the optimization problem in Equation ̃6 is well-defined only when $K \geq k_{\min}$.

4 Only 0 and 1 annotations

In this section, we will explore the annotation-aware segmentation problem for the special case where there are only 0 and 1 annotations over the entire signal. For example, consider the signal $y \in \mathbb{R}^{10}$ and breakpoint annotations in the figure below.



For the data and annotations above, the only consistent models are shown below.



The optimization problem is: which of these 6 segmentations is the most likely?

4.1 Number of changes and number of possible segmentations

In the 0/1 case there is only one possible number of segments K and the number of possible segmentations is $|\mathcal{M}_{K,d,A}| = \prod_{k=1}^{K}(\overline{\alpha}_k^1 - \underline{\alpha}_k^1)$, where $\overline{\alpha}_k^1$ and $\underline{\alpha}_k^1$ are the borders of the k-th 1-annotated region. If we define $l = \sum_{k=1}^{K}(\overline{\alpha}_k^1 - \underline{\alpha}_k^1) \leq d$ we get an upper bound on $|\mathcal{M}_{K,d,A}|$ by taking all $\overline{\alpha}_k^1 - \underline{\alpha}_k^1$ equal to l/K:

$$|\mathcal{M}_{K,d,A}| = \prod_{k=1}^{K}(\overline{\alpha}_k^1 - \underline{\alpha}_k^1) \leq (\frac{l}{K})^K \leq (\frac{d}{K})^K \leq |\mathcal{M}_{K,d}| = \binom{d-1}{K}.$$

When d goes to infinity and K remains fixed we have $\binom{d-1}{K} \sim \frac{(d-K)^{k+1}}{dk!}$ (Stirling). Thus if the length of the 1-annotated regions are fixed when d goes to infinity, we obviously get that $\lim_{d\to+\infty} \frac{|\mathcal{M}_{K,d,A}|}{|\mathcal{M}_{K,d}|} = 0$. On the other hand, if the size of all these 1-annotated regions is d/K we get that $\lim_{d\to+\infty} \frac{|\mathcal{M}_{K,d,A}|}{|\mathcal{M}_{K,d}|} = K!/K^K > 0$.

4.2 Algorithm

Suppose that we have a complete 0/1 annotation. We want to recover the consistent segmentation which is best in terms of the square loss.

We define $C_{k,t,A}$ as the best cost in k segments up to and including t under our set of annotations A. By definition a segmentation that is not consistent with the annotations A has an infinite cost.

Equation˜4 tells us that the k-th breakpoint τ_{k+1} must verify $\underline{\alpha}_k^1 < \tau_{k+1} \leq \overline{\alpha}_k^1$ in the 0/1 case. In other words, $C_{k,t,A}$ is finite only for $\underline{\alpha}_k^1 < t \leq \overline{\alpha}_k^1$ or $t \in]\!]\underline{\alpha}_k^1, \overline{\alpha}_k^1]\!]$.

Update rule Let us fix $k < K$ and assume that we know $C_{k,t,A}$ for all t. Then for all t in $]\!]\underline{\alpha}_{k+1}^1, \overline{\alpha}_{k+1}^1]\!]$ we have the update formula:

$$C_{k+1,t,A} = \min_{\tau \in]\!]\underline{\alpha}_k^1, \overline{\alpha}_k^1]\!]} \{C_{k,\tau,A} + \min_{\mu} \{ \sum_{i=\tau+1}^{t} (Y_i - \mu)^2 \}\} \tag{7}$$

In the case where we have overlapping regions the update rule is slightly different and becomes:

$$C_{k+1,t,A} = \min_{\tau \in]\!]\underline{\alpha}_k^1, \min(\overline{\alpha}_k^1, t-1)]\!]} \{C_{k,\tau,A} + \min_{\mu} \{ \sum_{i=\tau+1}^{t} (Y_i - \mu)^2 \}\} \tag{8}$$

Initialization and last step The initialization is straightforward. We just need to compute for t in $]\!]\underline{\alpha}_1^1, \overline{\alpha}_1^1]\!]$:

$$C_{1,t,A} = \min_{\mu} \{ \sum_{i=1}^{t} (Y_i - \mu)^2 \} \tag{9}$$

The last step of the algorithm is done using the update rule for $k = K$ and $t = d$.

We implemented these rules in a dynamic programming algorithm available in the **SegAnnot** package.

Complexity To compute $C_{k+1,t,A}$ we need to perform $(\overline{\alpha}_k^1 - \underline{\alpha}_k^1)$ operations. To get all $C_{k+1,t,A}$ we thus need $(\overline{\alpha}_k^1 - \underline{\alpha}_k^1)(\overline{\alpha}_{k+1}^1 - \underline{\alpha}_{k+1}^1)$ operations. Overall the number of operations is

$$\sum_{k=1}^{K-1} (\overline{\alpha}_k^1 - \underline{\alpha}_k^1)(\overline{\alpha}_{k+1}^1 - \underline{\alpha}_{k+1}^1). \tag{10}$$

If the annotations are not overlapping, then the number of operations is always smaller than d^2. Indeed we have

$$\sum_{k=1}^{K-1} (\overline{\alpha}_k^1 - \underline{\alpha}_k^1)(\overline{\alpha}_{k+1}^1 - \underline{\alpha}_{k+1}^1) \leq \left(\sum_{k=1}^{K} (\overline{\alpha}_k^1 - \underline{\alpha}_k^1) \right)^2 \leq d^2, \tag{11}$$

and we get a complexity of $O(d^2)$. This worst case scenario is reached when we have two successive regions of size $(d-K)/2$. If the regions are of size d/K we get a complexity of $O(d^2/K)$. However, if the annotations are overlapping the number of operations is $O(Kd^2)$.

In our experience, annotations define relatively small and non-overlapping regions of the profile. So in practice $\sum_{k=1}^{K-1} (\overline{\alpha}_k^1 - \underline{\alpha}_k^1)(\overline{\alpha}_{k+1}^1 - \underline{\alpha}_{k+1}^1)$ is usually much smaller than d^2. This often results in fast run times in practice on real data.

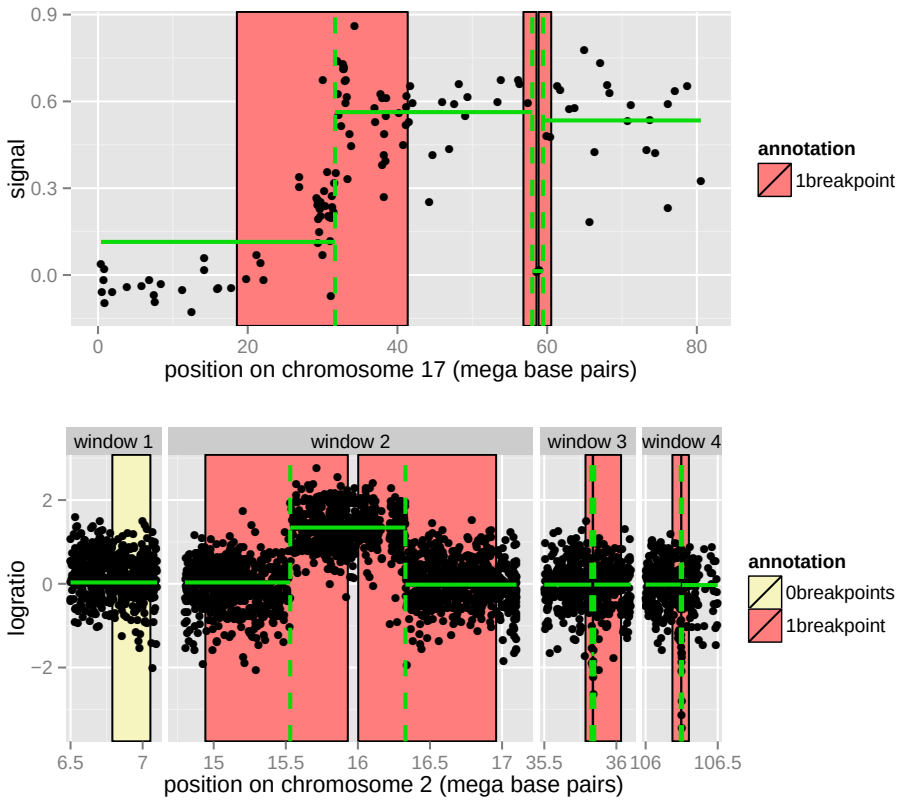
5 Only 0, 1 and b annotations

Now let us consider the case were the number of breaks per annotated regions is unique. Obviously for a region of size n we cannot have more than $n-1$ breaks. In fact, we can easily convert this problem to a problem with only 0/1 annotations. First, assume that we expect $a_j = b$ breaks in the j-th region $A_j = [\overline{\alpha}_j, \underline{\alpha}_j]$. The m-th break of this region is between $[\overline{\alpha}_j + m - 1, \underline{\alpha}_j - b + m]$. Thus, we can replace the region A_j with b regions annotated to have exactly 1 breakpoint.

6 Results

6.1 Fitting 0/1 annotations perfectly

In this section, we show that the **SegAnnot** algorithm can find a consistent segmentation for the motivating examples that we saw in Section˜2. In the two figures below, we show the segmentation recovered by the **SegAnnot** algorithm for these two examples.



Clearly, the **SegAnnot** algorithm is able to find a segmentation consistent with the expert's annotations, which was not true of the maximum-likelihood segmentations given by the pruned DP algorithm.

7 0, 0+, 1 and 1+ annotations

Although not yet implemented in the **SegAnnot** package, we could build on these ideas to define a segmentation for more complicated signals with 0+ and 1+ annotations.

Here are the key ideas of the 0/1 algorithm:

- the number of changes before a possible change at data-point t in the $(k+1)$-th 1-region is necesarily k.
- the position of the previous change before a possible change at t in the $(k+1)$-th 1-region is restricted to the k-th 1-region.

The algorithm in the 0/0+/1/1+ case is conceptually the same, the main difference is that

- the number of changes before a possible change at data-point t is not unique (but it can be restricted).
- the set possible changes before a change at t includes but is not necesarily restricted to the previous 1-region.

Some more notation To properly define the algorithm in the general case we need to introduce some more notation. We define I_l as the annotation status of the l-th annotation region (0, 0+, 1 or 1+). We define $\check{k}_l$ as the minimum number of breaks in region (l: $M_l = 1$ if $I_l = 1$ or 1+ and $M_l = 0$ otherwise). The function l map any data-point t to its given annotation region: $l(t) = \{l | t \in [\overline{\alpha}_l - \underline{\alpha}_l]\}$. $K_{\max}$ is the maximum number of changes authorised by the user. We assume that $K_{\max} \geq \overleftarrow{k}_{l(n)}$

Then we get that

- the minimum number of changes before region l is $\overleftarrow{k}_l = \sum_{j<k} M_j$
- the minimum number of changes after region l is $\overrightarrow{k}_l = \sum_{j>k} M_j$
- the minimum number of changes before t is $\overleftarrow{k}_{l(t)}$
- the maximum number of changes before t is $K_{\max} - \overrightarrow{k}_{l(t)}$.

For region l we can define the closest 1 or 1+ region to the left:

$$\overleftarrow{1}_l = \arg\max_{j<l} \{M_j = 1\}.$$

Let $V_l = \cup_{\overleftarrow{1}_l \leq j < l | I_j \neq 0} [\![\underline{\alpha}_j, \overline{\alpha}_j^1]\!]$.

The set of possible changes just before a change at t is called $V(t)$ and is:

- $V(t) = V_{l(t)}$ if $I_l(t) = 1$
- $V(t) = V_{l(t)} \cup [\![\underline{\alpha}_j, t-1]\!]$ if $I_l(t) = 1+$ or $I_l(t) = 0+$
- $V(t) = \emptyset$ if $I_l(t) = 0$

Update rule For any t such that $I_{l(t)} \neq 0$ and $k+1$ such that $\overleftarrow{k}_l \leq k+1 \leq K_{\max} - \overrightarrow{k}_l$ the update rule is

$$C_{k+1,t,A} = \min_{\tau \in V(t)} \{C_{k,\tau,A} + \min_{\mu} \{\sum_{i=\tau}^{t} (Y_i - \mu)^2\}\}.$$

Initialization For the initialization we define the set of possible position for the first change

$$V' = \cup_{l \leq \arg\min_l \{M_l = 1\}} [\overline{\alpha}_l - \underline{\alpha}_l].$$

Then for any t in V' we use

$$C_{1,t,A} = \min_{\mu} \{\sum_{i=1}^{t} (Y_i - \mu)^2\}.$$

Last step For the last step we need to recover the minimum and maximum number of changes at $t = n$ and apply the update rule for all k in this range. This minimum and maximum are respectively $\overleftarrow{k}_{l(n)}$ and $K_{\max} - \overrightarrow{k}_{l(n)}$.

8 Conclusions and future work

The **SegAnnot** package provides a new segmentation model for noisy observations from annotated piecewise constant signals. It takes as input the observations and a set of 0/1 breakpoint annotations, and returns the precise locations of the most likely breakpoints. We showed 2 cases where pruned DP yields no consistent models, so **SegAnnot** can be used for **Fitting** the annotations in these cases.

For the future, it may be interesting to implement the 0/1/0+/1+ algorithm.

References

T.˜D. Hocking, G.˜Schleiermacher, I.˜Janoueix-Lerosey, O.˜Delattre, F.˜Bach, and J.-P. Vert. Learning smoothing models using breakpoint annotations. HAL-00663790, 2012. URL `http://hal.inria.fr/hal-00663790`.

R.˜Killick, P.˜Fearnhead, and I.˜A. Eckley. Optimal detection of changepoints with a linear computational cost. arXiv:1101.1438, Jan. 2011. URL `http://arxiv.org/abs/1101.1438`.

G.˜Rigaill. Pruned dynamic programming for optimal multiple change-point detection. arXiv:1004.0887, 2010. URL `http://arxiv.org/abs/1004.0887`.