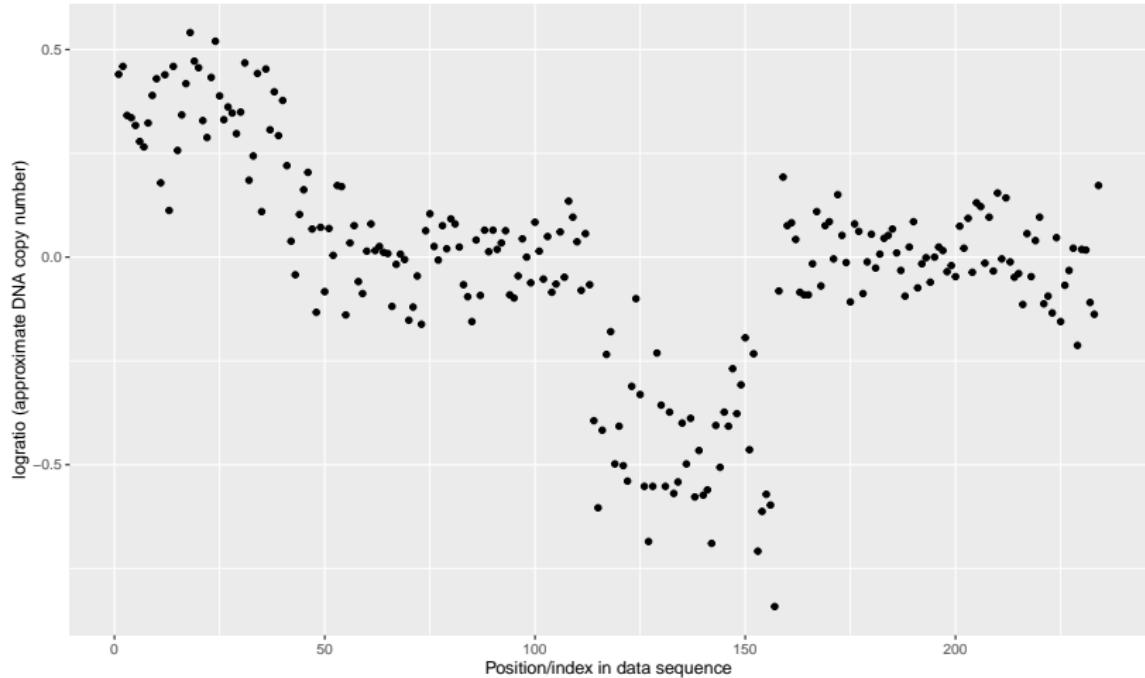


Optimal segmentation

Toby Dylan Hocking

Background: detecting abrupt changes is important

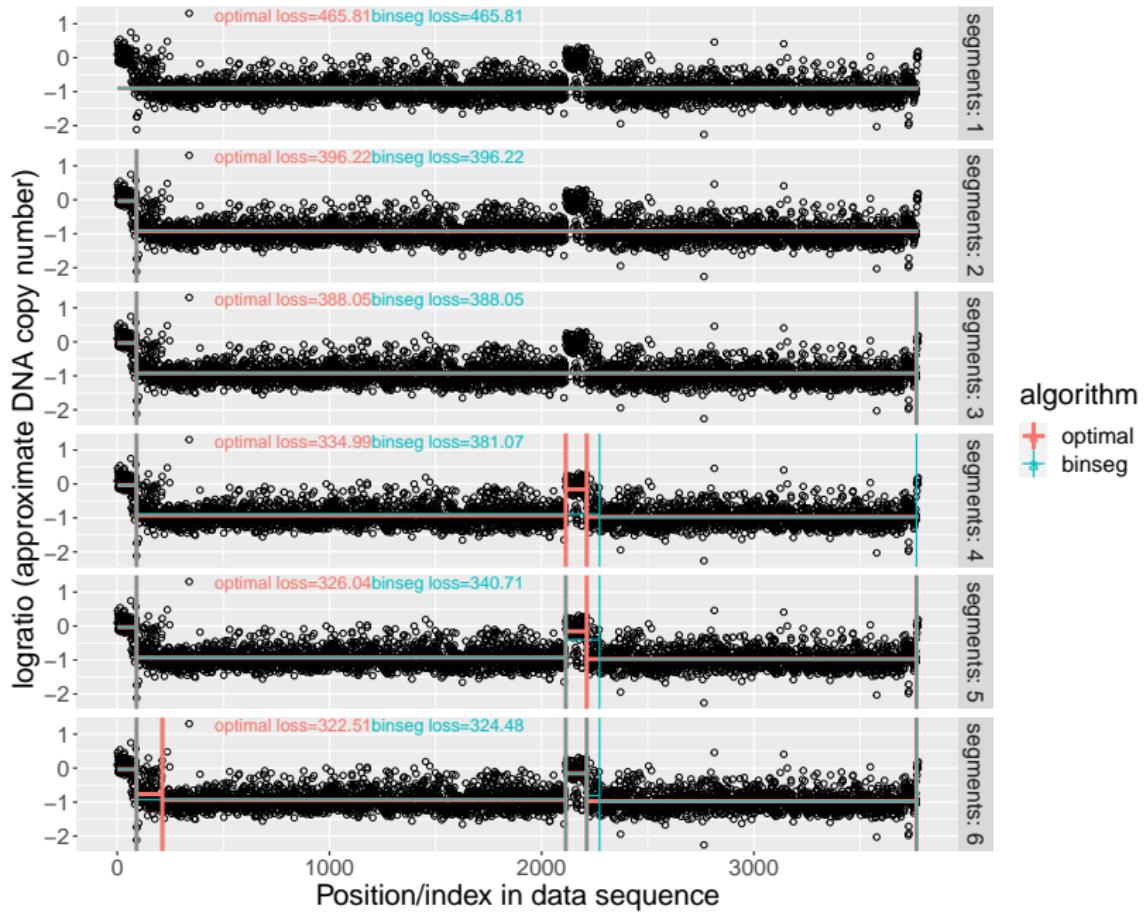
Example from cancer diagnosis: breakpoints are associated with aggressive disease in neuroblastoma.



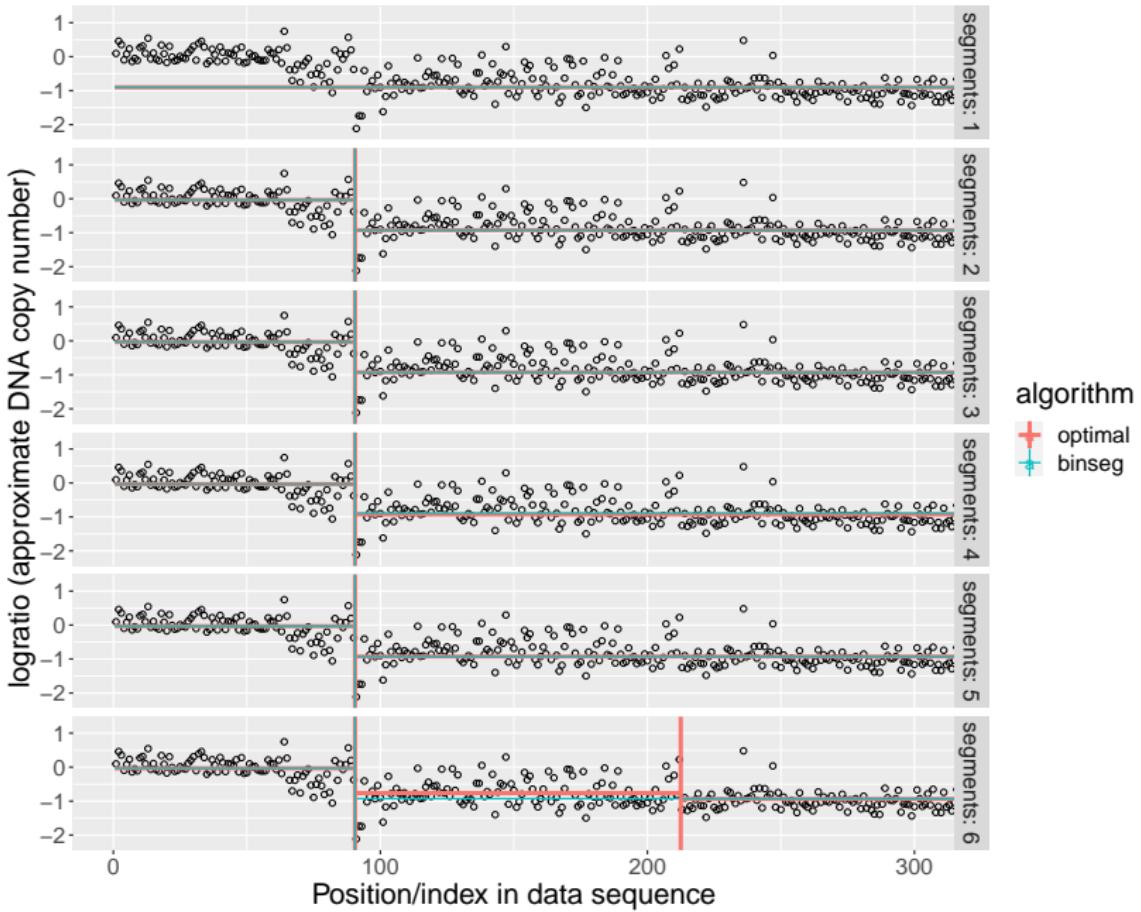
Motivation for optimal detection

- ▶ Binary segmentation is a greedy algorithm, so sometimes it chooses a changepoint which is sub-optimal for a larger model size.
- ▶ Is it possible to compute the changepoints and segments which are optimal for each model size? Yes, with dynamic programming.
- ▶ Dynamic programming can be applied to any computational problem which involves optimization, and can be broken down into independent sub-problems. In this context we use it to compute the changepoints and segments which are optimal for a given loss function.
- ▶ Is it desirable to compute the optimal changepoints and segments? Yes, see examples in next slides.

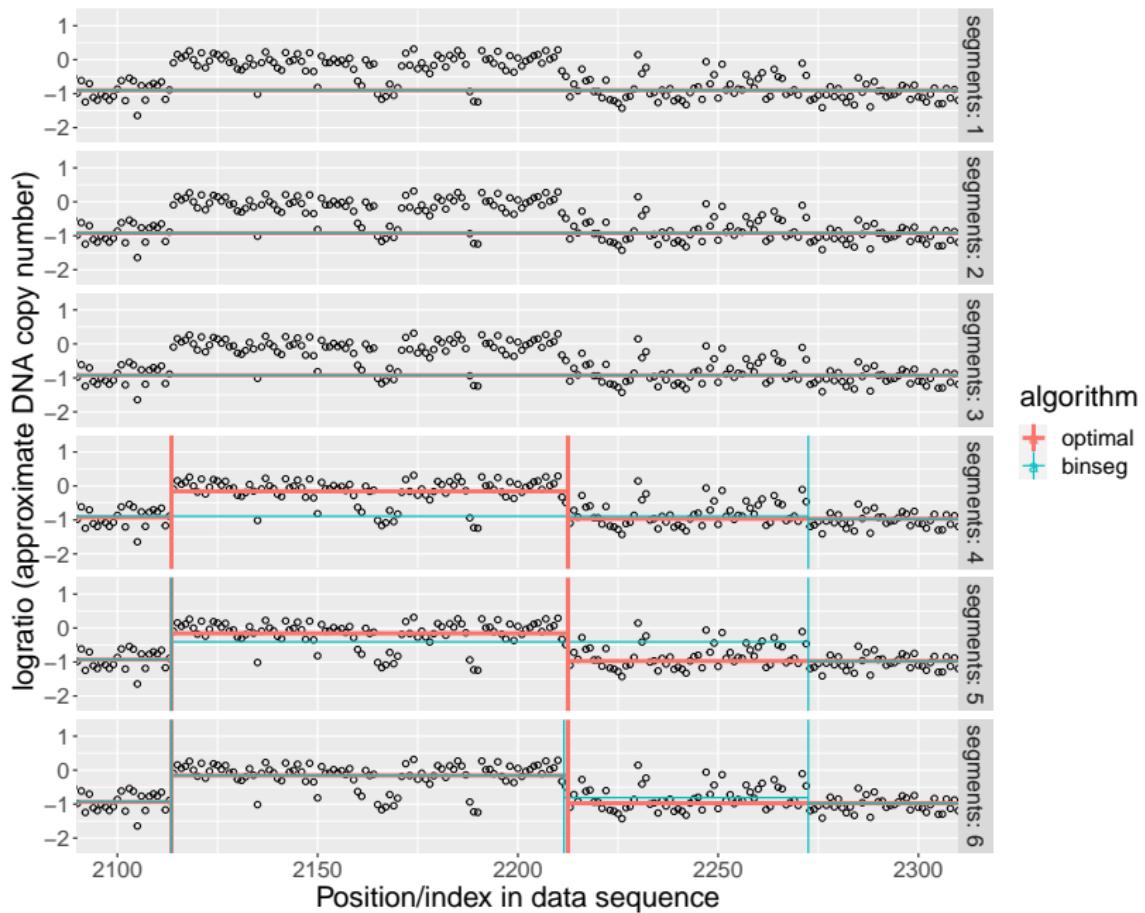
Example 1: stuck with sub-optimal change



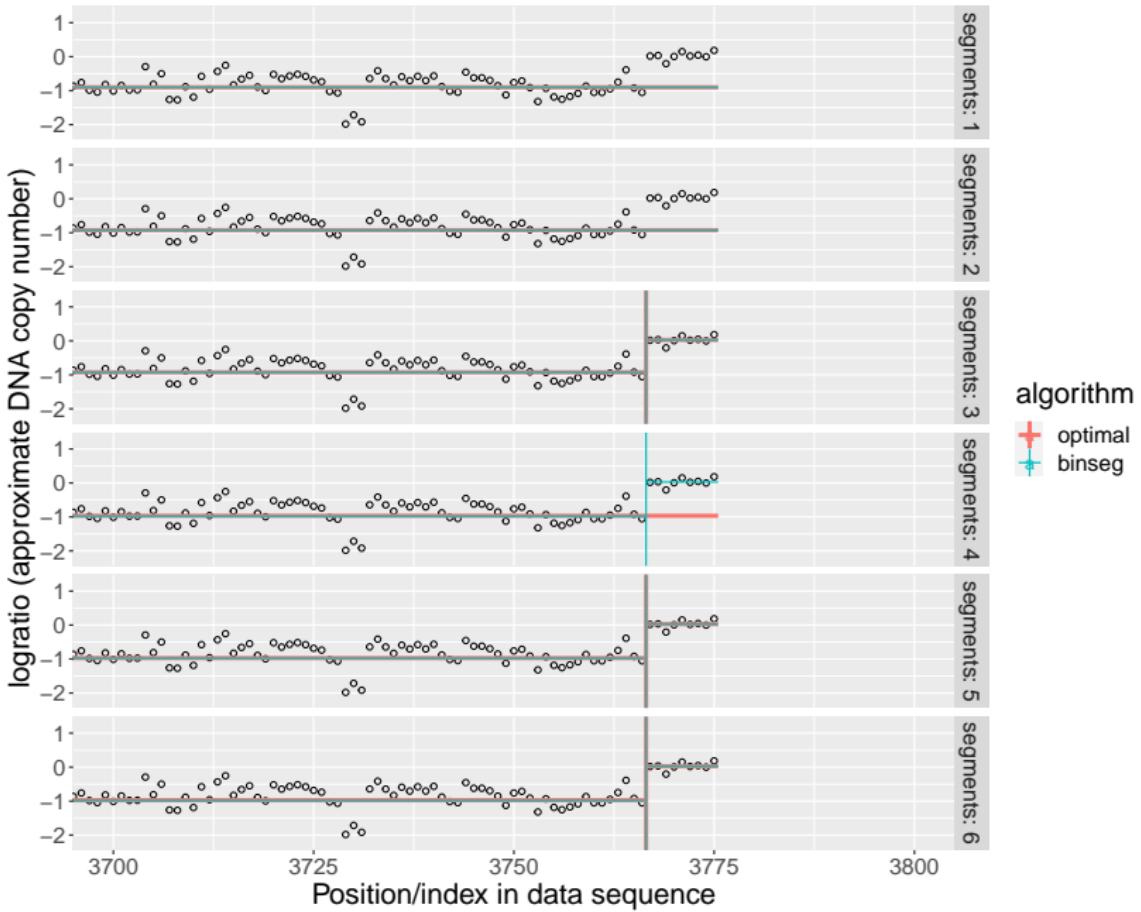
Zoom to start



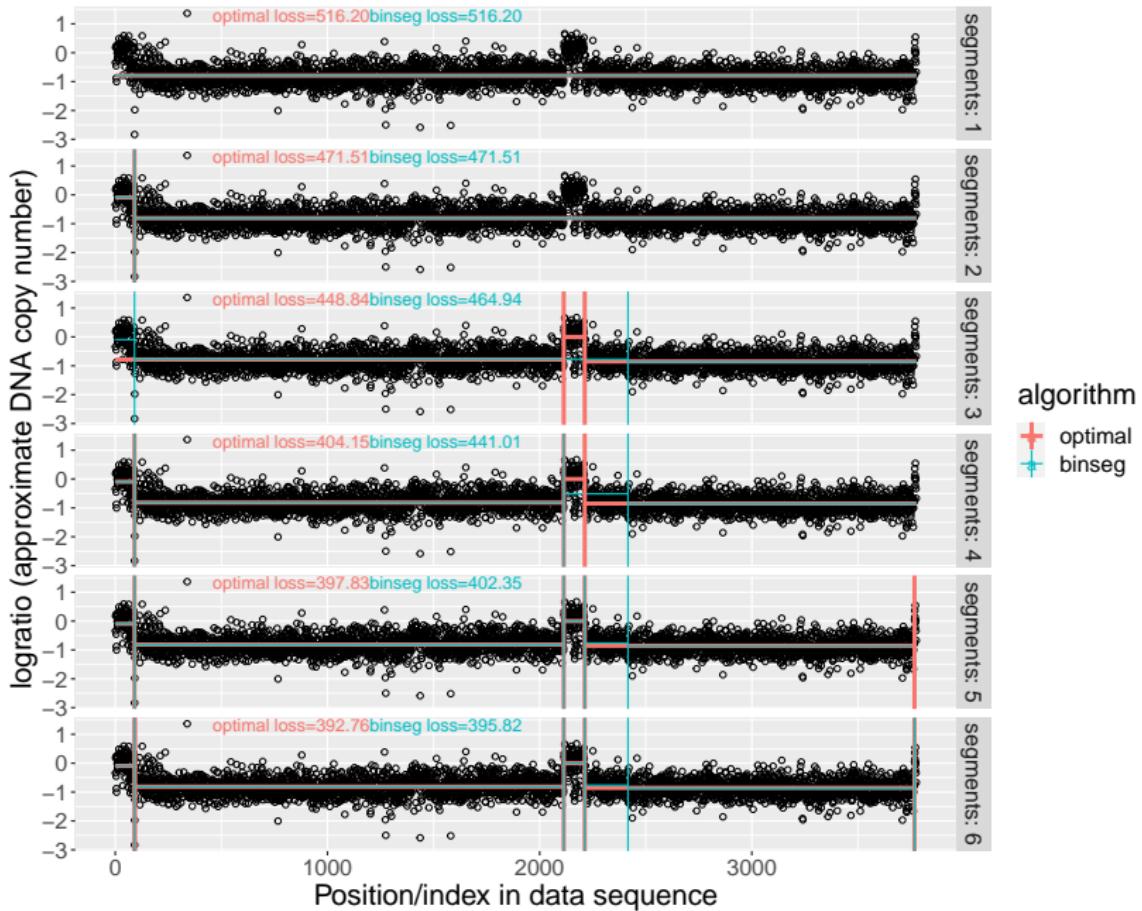
Zoom to center



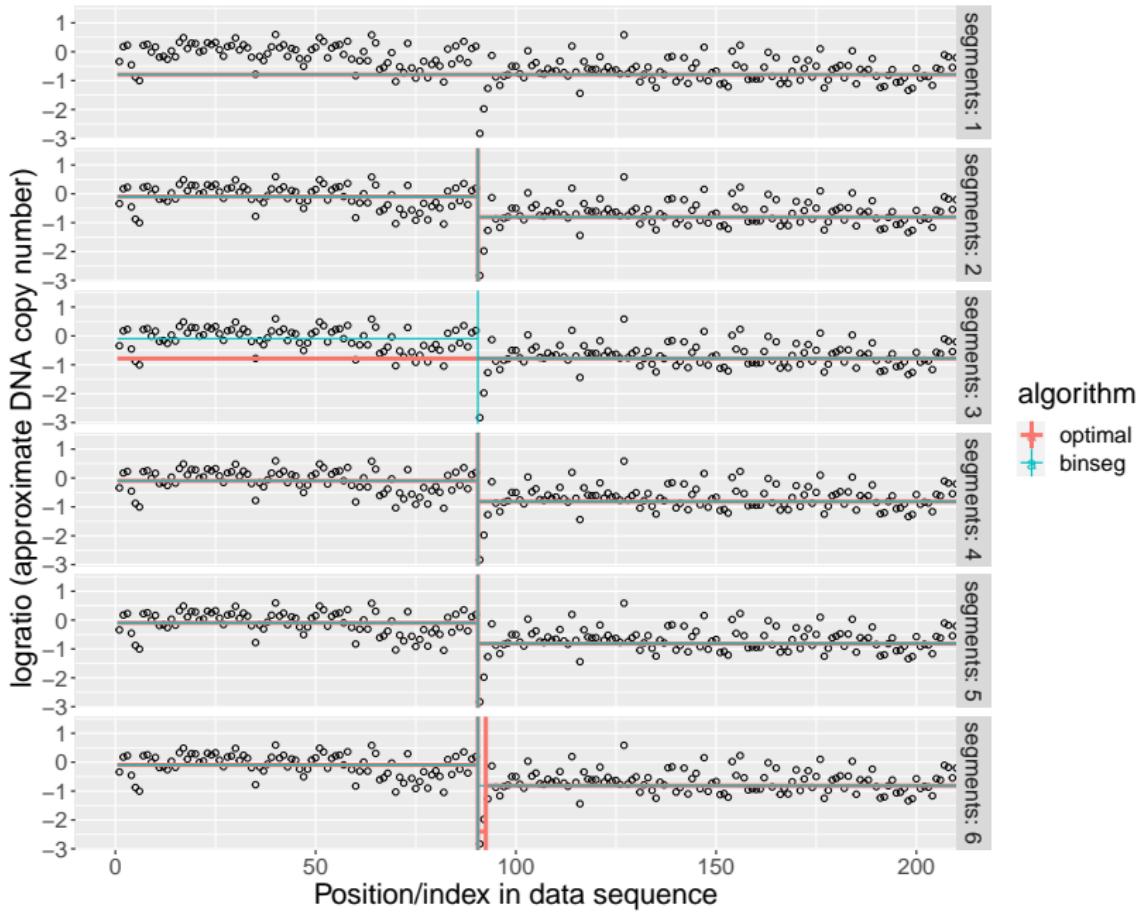
Zoom to end



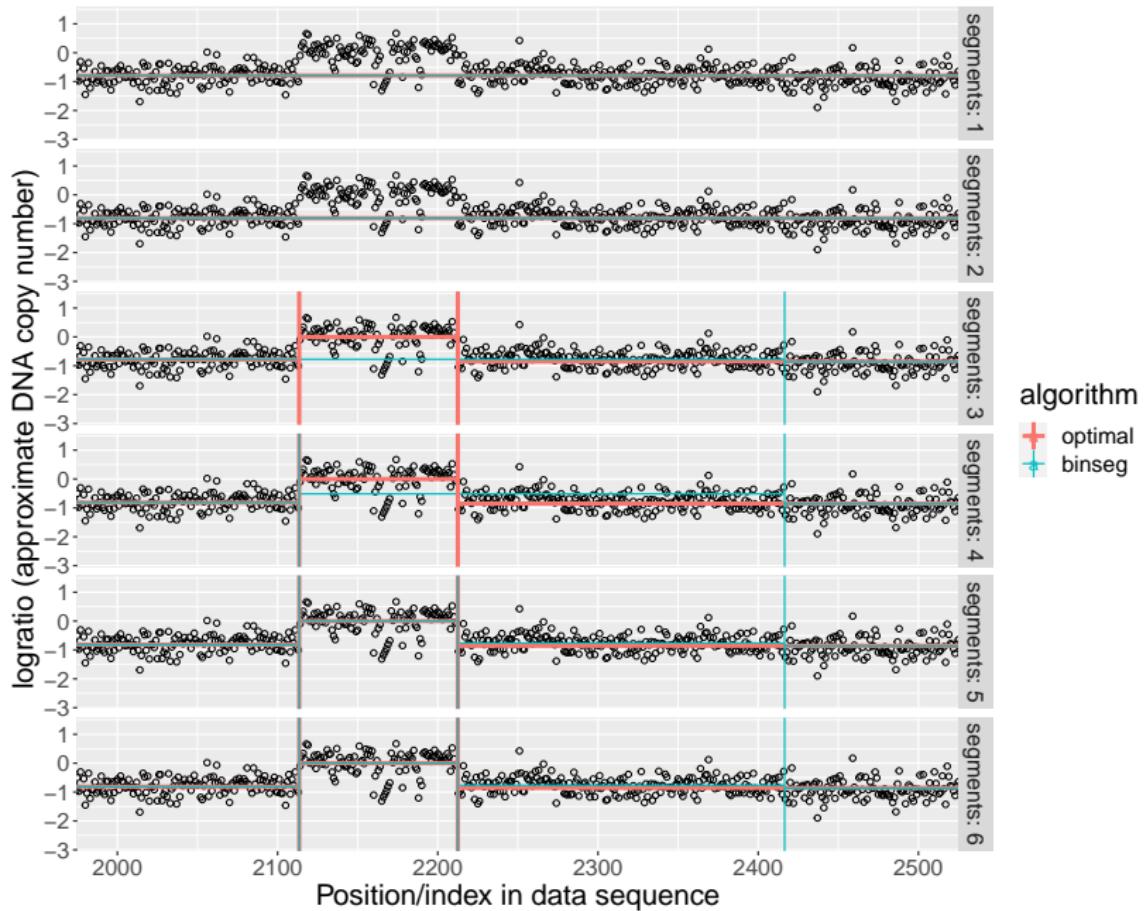
Example 2: missing a small change down



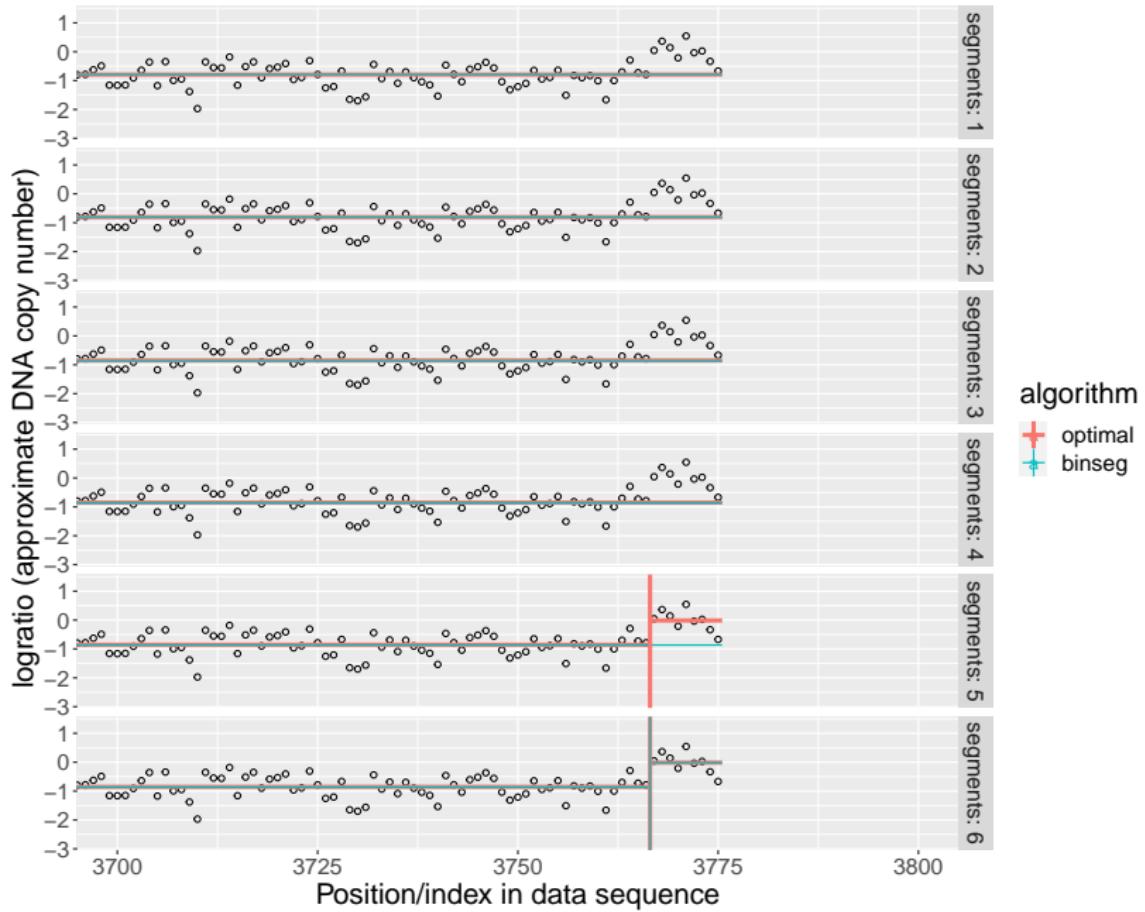
Zoom to start



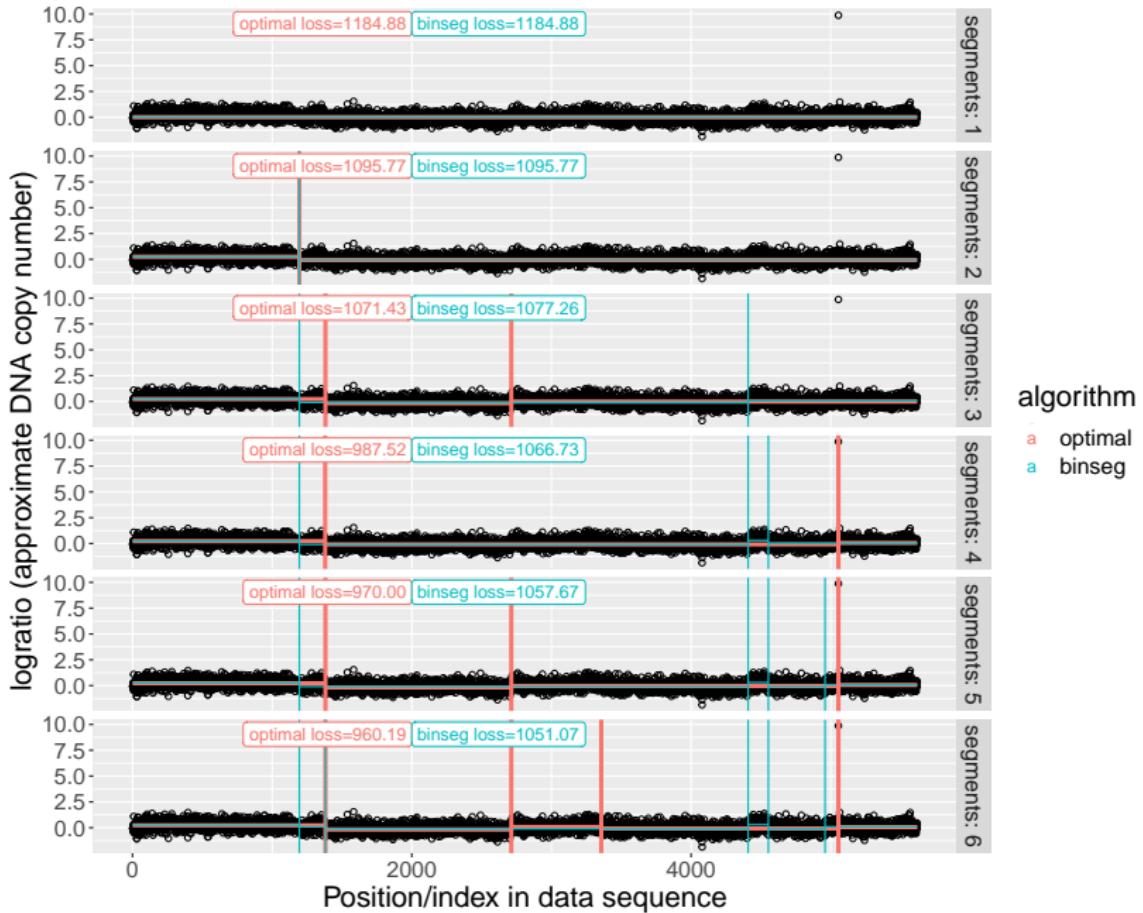
Zoom to center



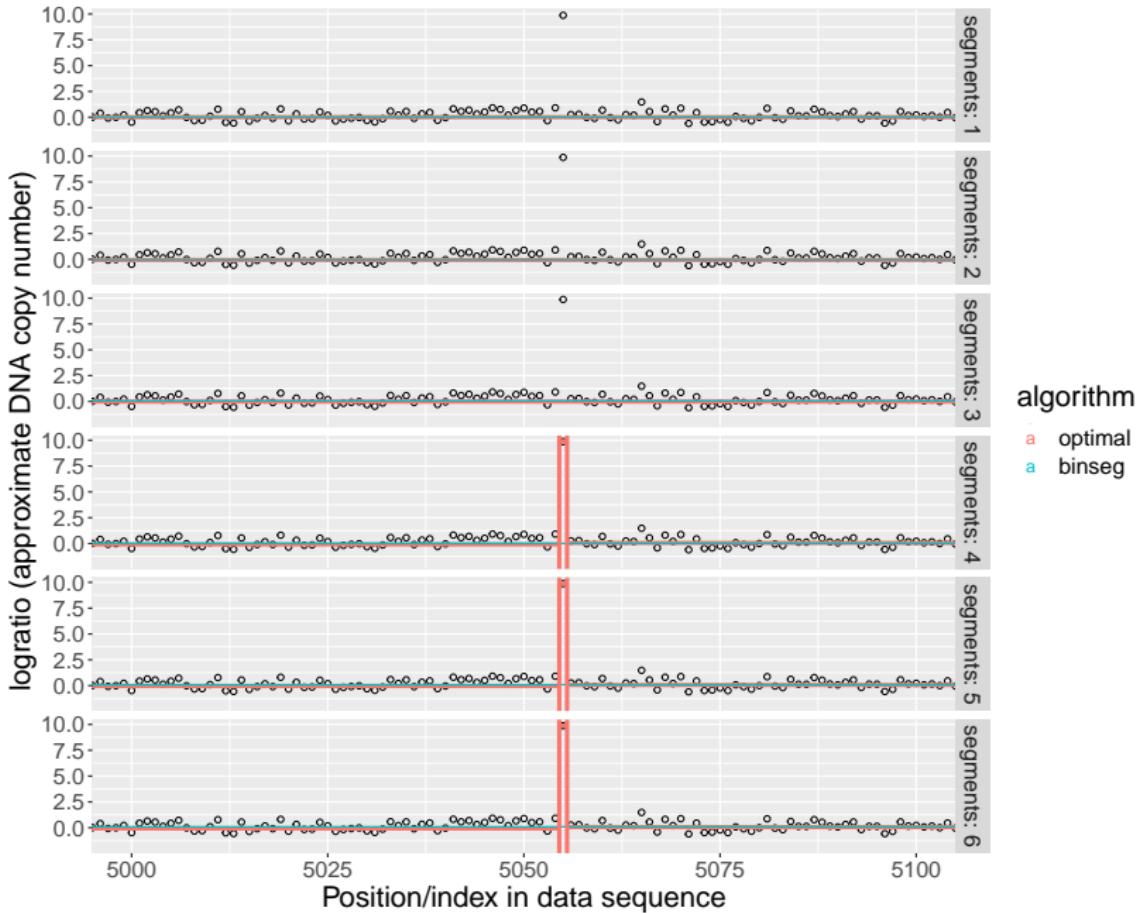
Zoom to end



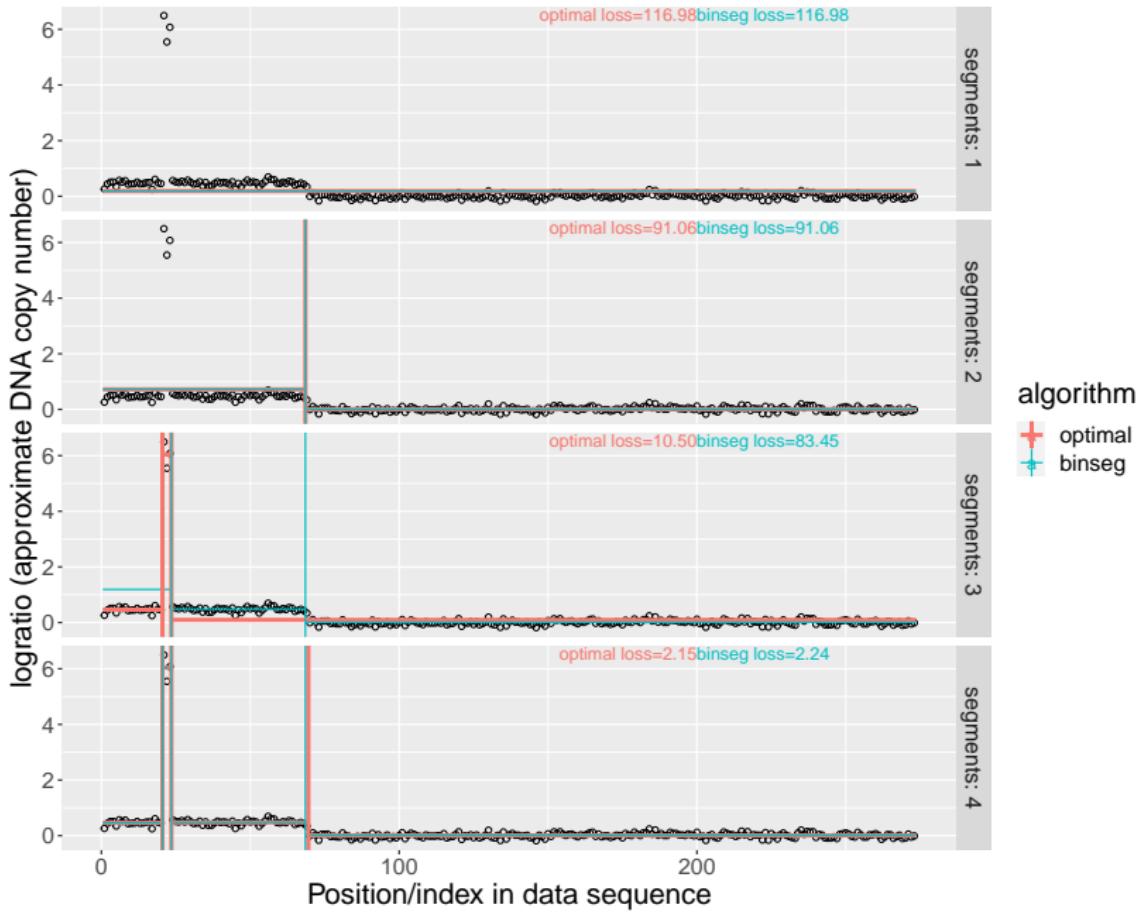
Example 3: difficult to detect an outlier



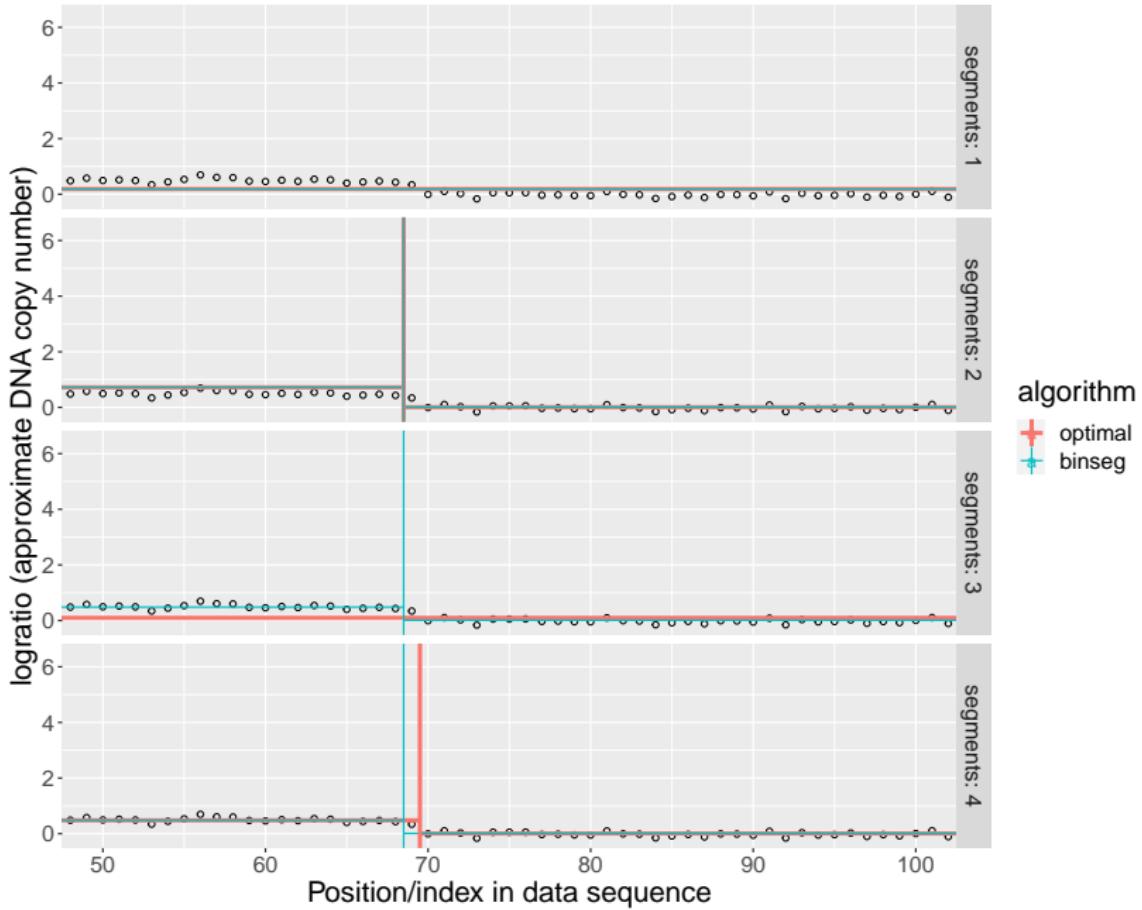
Zoom X axis to outlier



Example 4: stuck with sub-optimal change



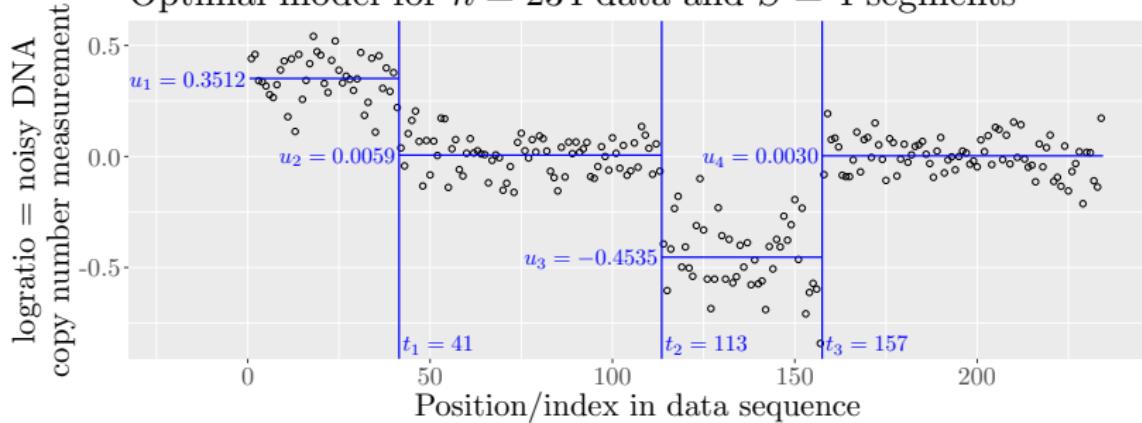
Zoom to changepoint



Dynamic programming for optimal changepoint detection

- ▶ We have n data x_1, \dots, x_n .
- ▶ Fix the number of segments $S \in \{1, 2, \dots, n\}$.
- ▶ Optimization variables: $S - 1$ changepoints $t_1 < \dots < t_{S-1}$ and S segment means $u_1, \dots, u_S \in \mathbb{R}$ ($t_0 = 0$, $t_S = n$).
- ▶ Statistical model: for every segment $s \in \{1, \dots, S\}$,
 $x_i \stackrel{\text{iid}}{\sim} N(u_s, \sigma^2)$ for every data point $i \in (t_{s-1}, t_s]$ implies square loss function $\ell(u_s, x_i) = (u_s - x_i)^2$ to minimize.

Optimal model for $n = 234$ data and $S = 4$ segments



Maximum likelihood inference for S segments and n data

The best loss for S segments and n data is

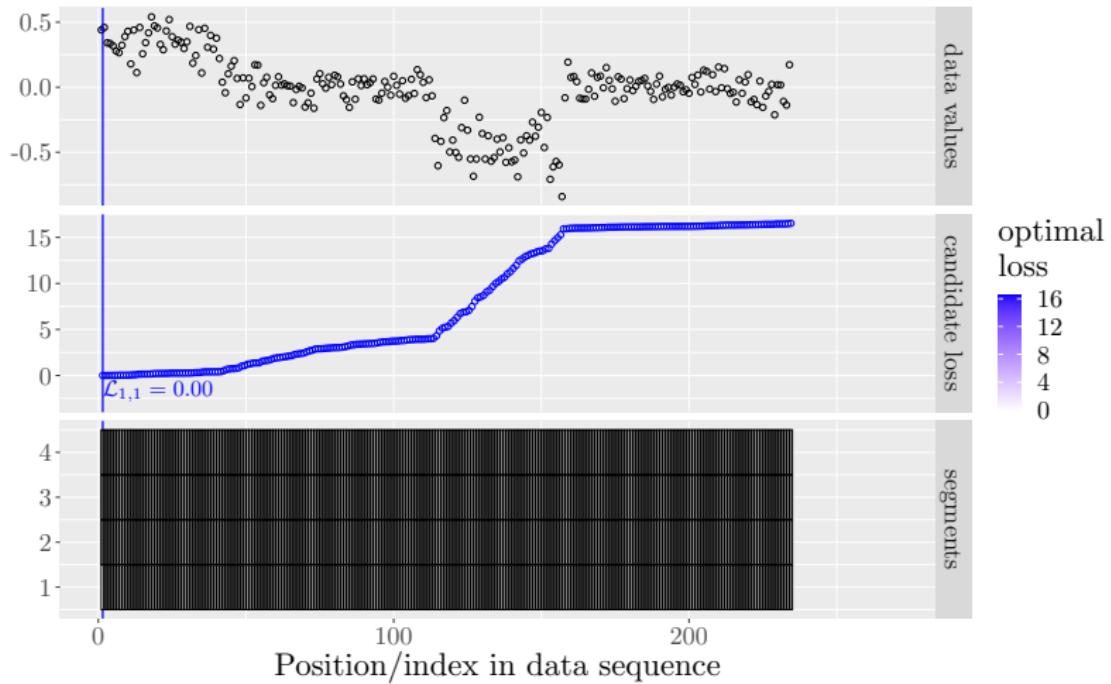
$$\begin{aligned}\mathcal{L}_{S,n} &= \min_{\substack{\mathbf{u} \in \mathbb{R}^S \\ 0 = t_0 < t_1 < \dots < t_{S-1} < t_S = n}} \sum_{s=1}^S \sum_{i=t_{s-1}+1}^{t_s} \ell(u_s, x_i) \\ &= \underbrace{\min_{t_{S-1}} \min_{\substack{u_1, \dots, u_{S-1} \\ t_1 < \dots < t_{S-2}}} \sum_{s=1}^{S-1} \sum_{i=t_{s-1}+1}^{t_s} \ell(u_s, x_i)}_{\mathcal{L}_{S-1, t_{S-1}}} + \underbrace{\min_{u_S} \sum_{i=t_{S-1}+1}^{t_S=n} \ell(u_S, x_i)}_{c_{(t_{S-1}, t_S=n]}}\end{aligned}$$

- ▶ Hard optimization problem because of integer-valued changepoint t_s variables, naively $O(n^S)$ time.
- ▶ Auger and Lawrence (1989): $O(Sn^2)$ time classical dynamic programming algorithm (best loss computed recursively).

$$\mathcal{L}_{s,t} = \min_{t' < t} \mathcal{L}_{s-1, t'} + c_{(t', t]}$$

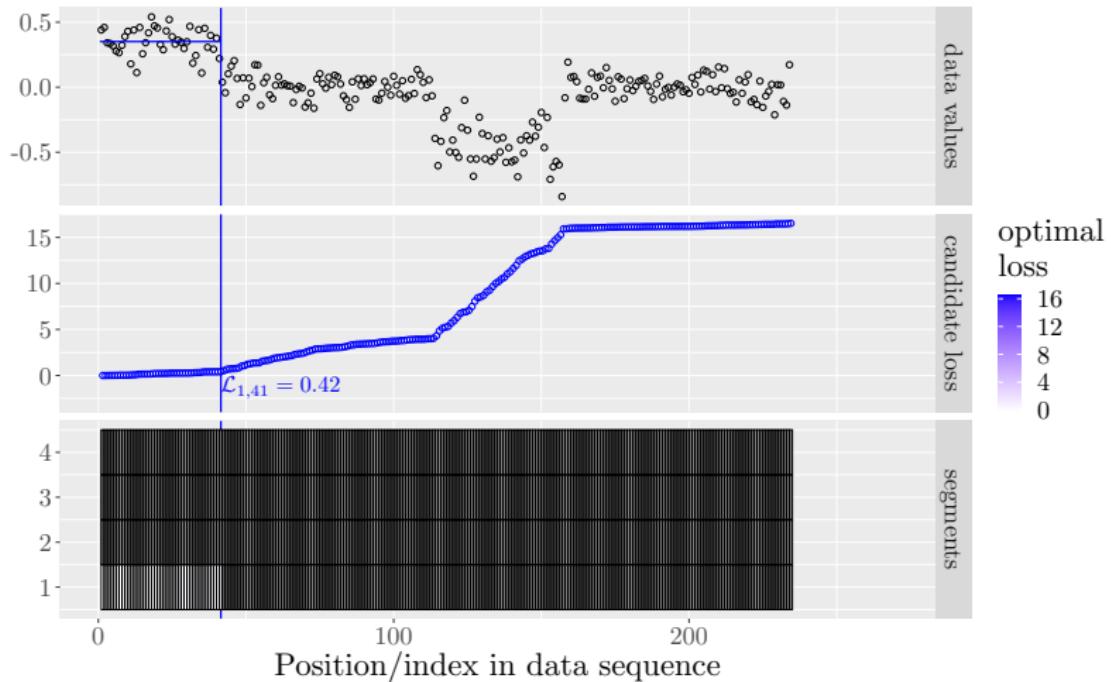
Initialization of dynamic programming

$$\mathcal{L}_{1,t} = c_{(0,t]}$$



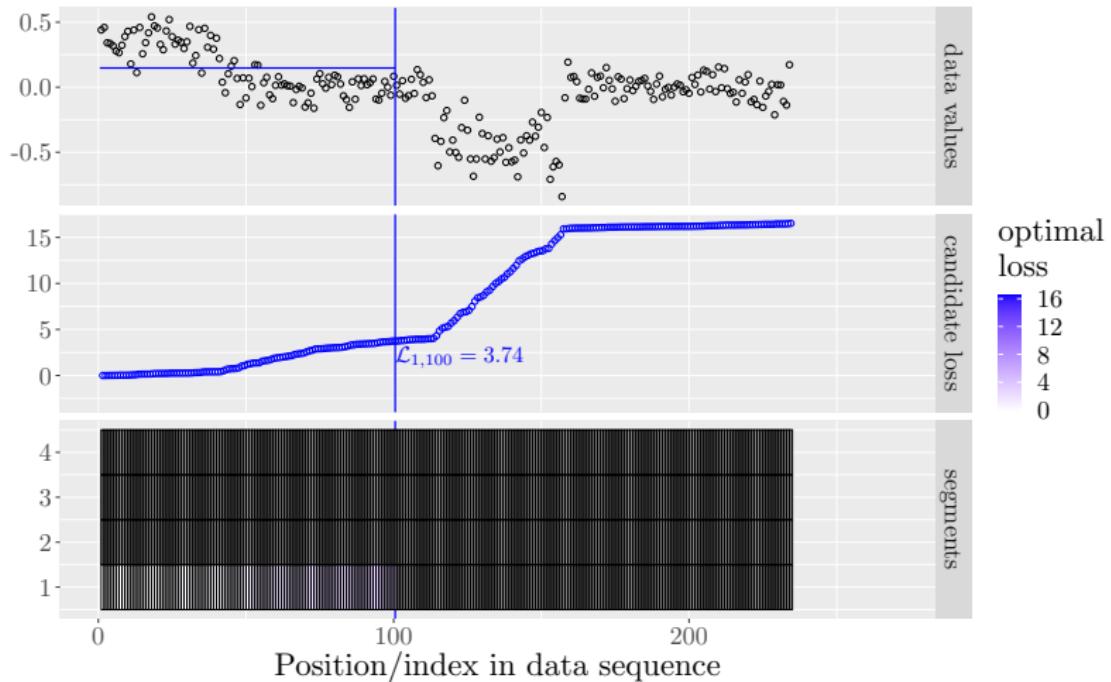
Initialization of dynamic programming

$$\mathcal{L}_{1,t} = c_{(0,t]}$$



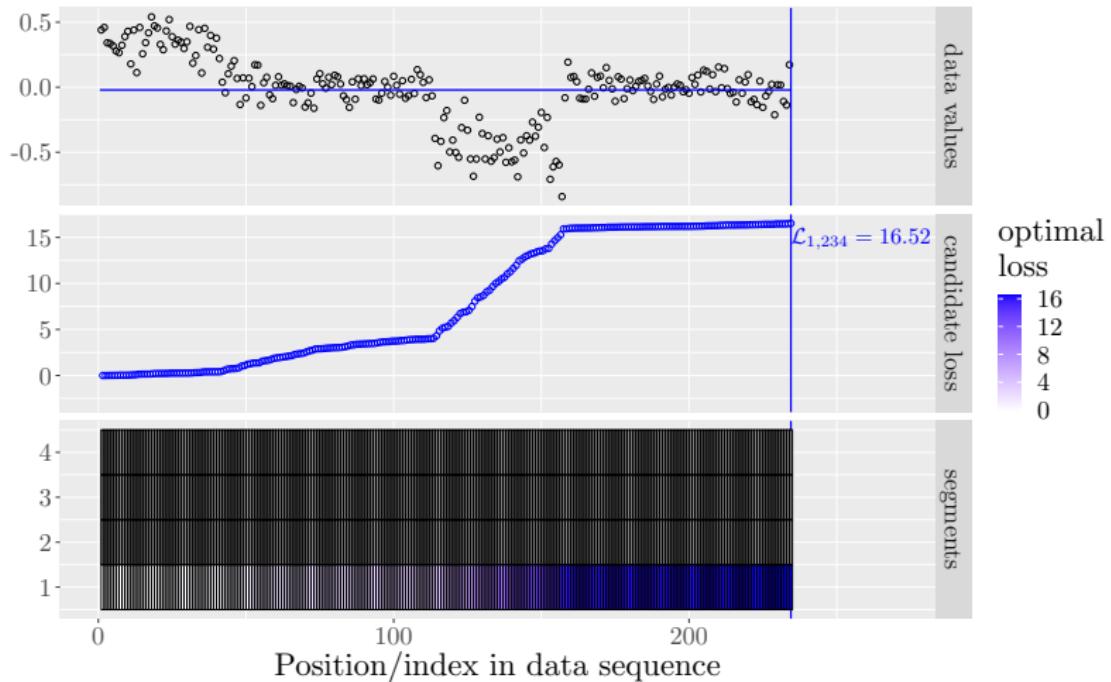
Initialization of dynamic programming

$$\mathcal{L}_{1,t} = c_{(0,t]}$$



Initialization of dynamic programming

$$\mathcal{L}_{1,t} = c_{(0,t]}$$



Efficient computation of c values

The best square loss up to t is

$$c_{(0,t]} = \min_u \sum_{i=1}^t (x_i - u)^2 = \sum_{i=1}^t x_i^2 - 2\hat{u}_t \sum_{i=1}^t x_i + t\hat{u}_t^2.$$

We can use cumulative sum to compute y_t for all t in linear $O(t)$ time, $y_t = \sum_{i=1}^t x_i$.

And best means up to t , $\hat{u}_t = y_t/t$.

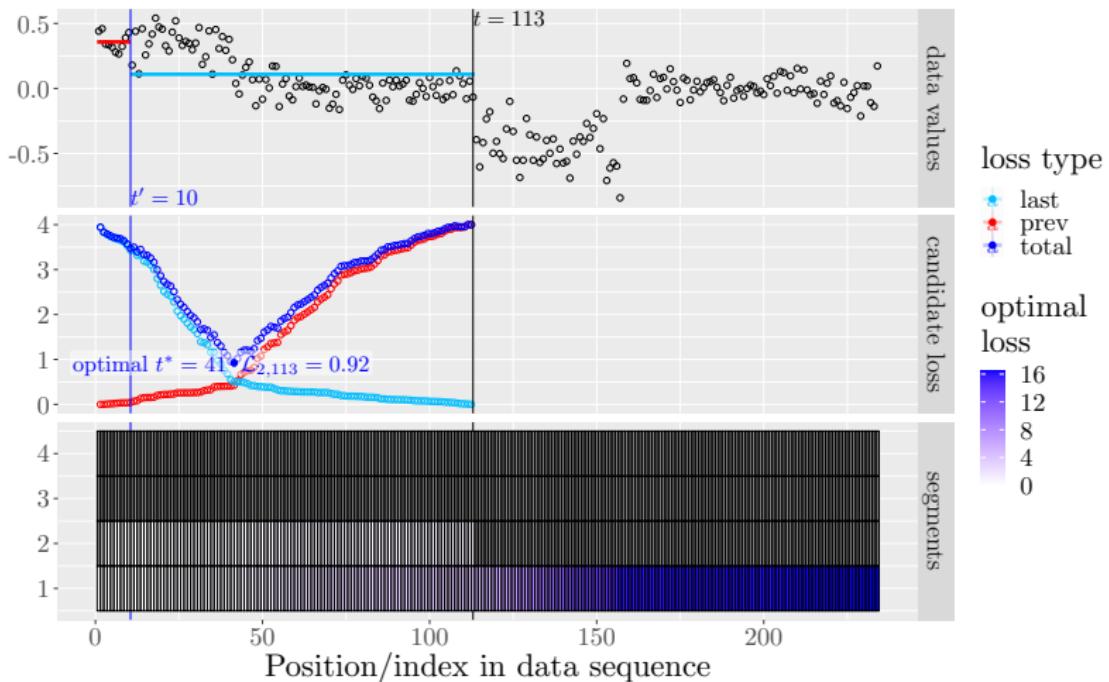
And cumulative sum of squares, $z_t = \sum_{i=1}^t x_i^2$.

And best loss up to t , $c_{(0,t]} =$

$$\sum_{i=1}^t x_i^2 - 2\hat{u}_t \sum_{i=1}^t x_i + t\hat{u}_t^2 = z_t - 2(y_t/t)(y_t) + t(y_t/t)^2 = z_t - y_t^2/t.$$

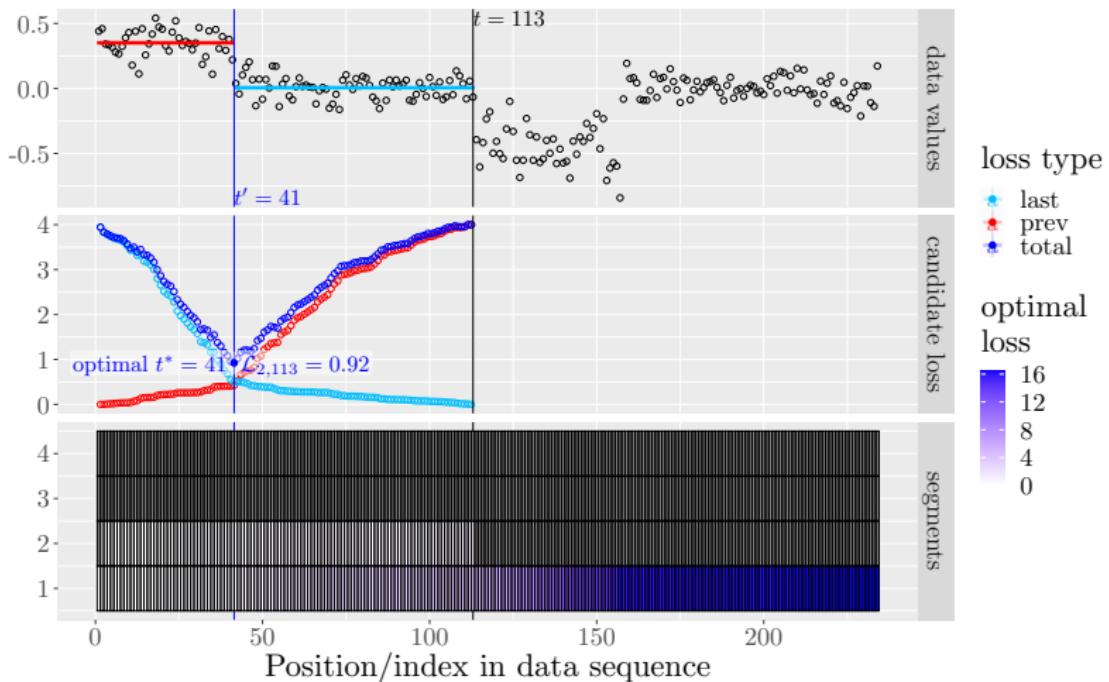
First iteration/changepoint of dynamic programming

$$\mathcal{L}_{2,t} = \min_{t' < t} \mathcal{L}_{1,t'} + c_{(t',t]}$$



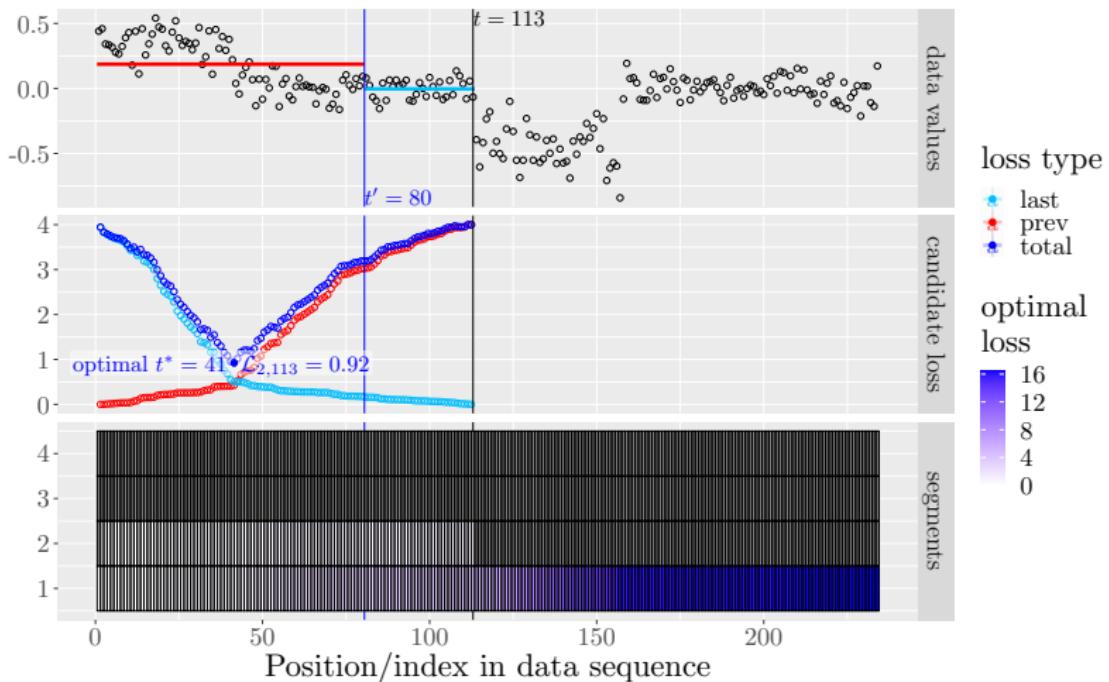
First iteration/changepoint of dynamic programming

$$\mathcal{L}_{2,t} = \min_{t' < t} \mathcal{L}_{1,t'} + c_{(t',t]}$$



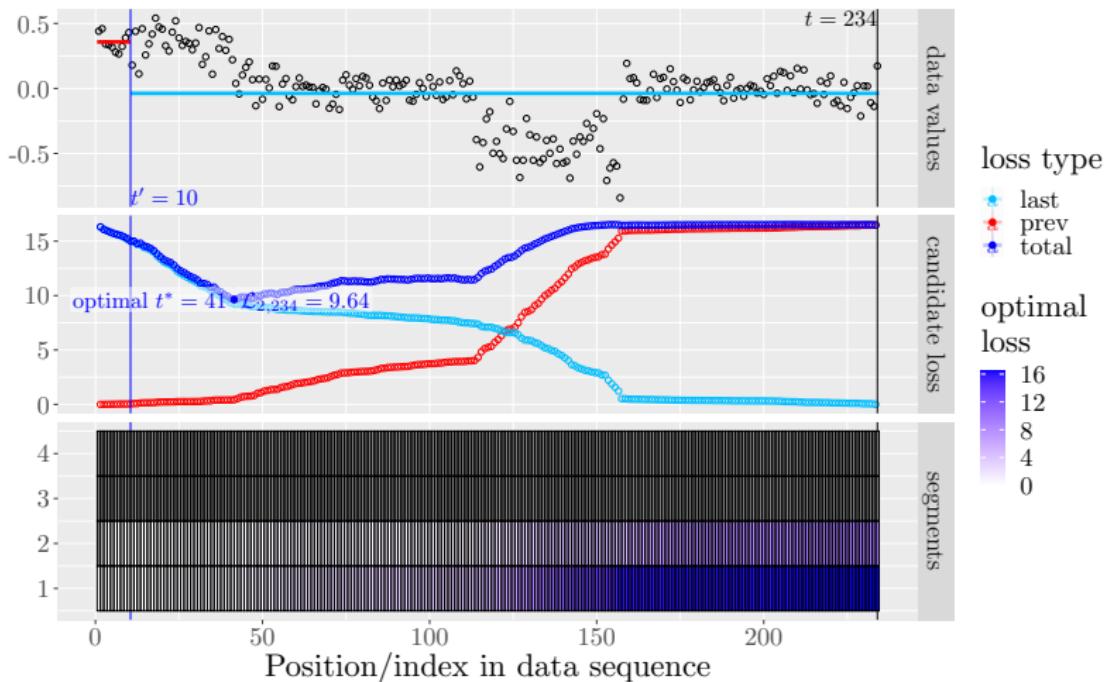
First iteration/changepoint of dynamic programming

$$\mathcal{L}_{2,t} = \min_{t' < t} \mathcal{L}_{1,t'} + c_{(t',t]}$$



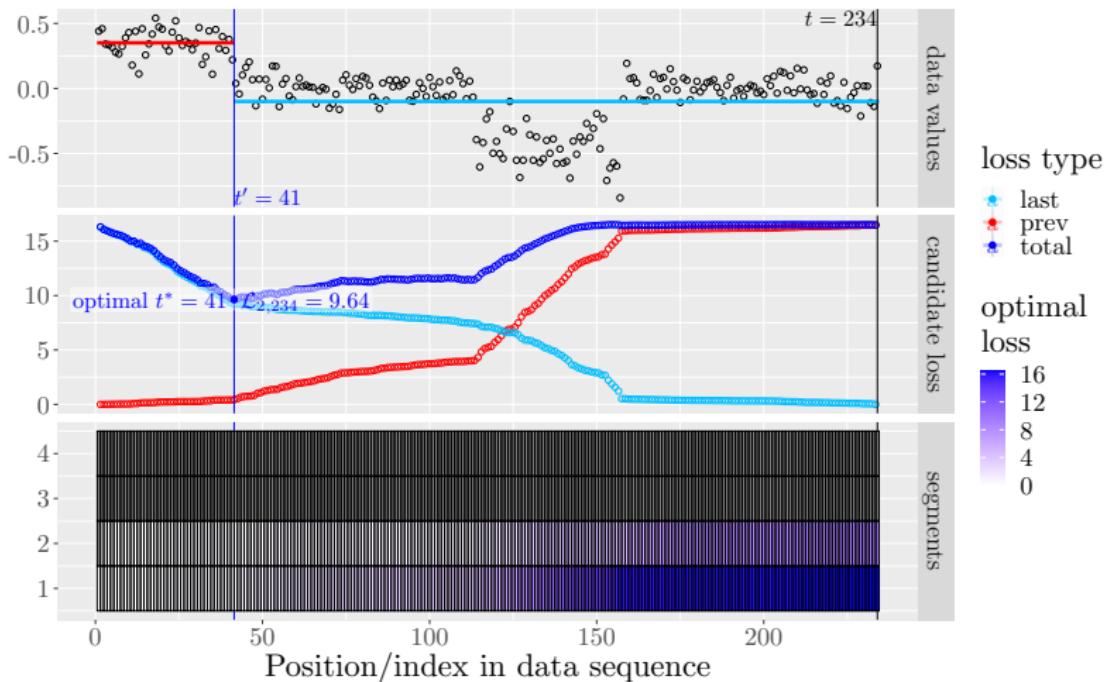
First iteration/changepoint of dynamic programming

$$\mathcal{L}_{2,t} = \min_{t' < t} \mathcal{L}_{1,t'} + c_{(t',t]}$$



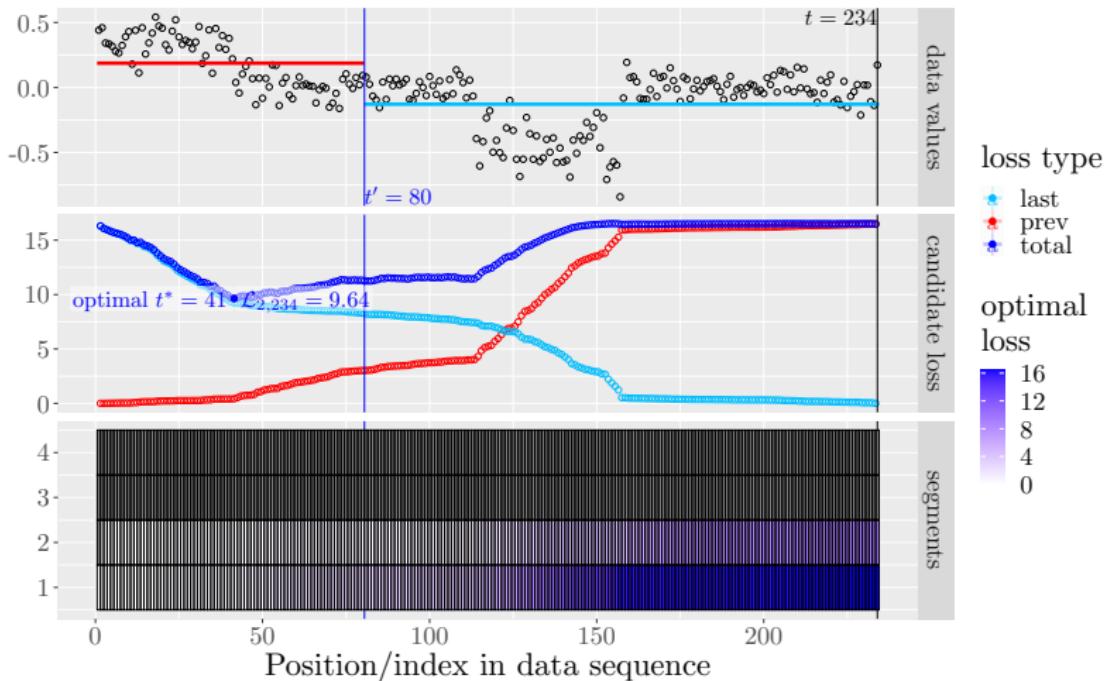
First iteration/changepoint of dynamic programming

$$\mathcal{L}_{2,t} = \min_{t' < t} \mathcal{L}_{1,t'} + c_{(t',t]}$$



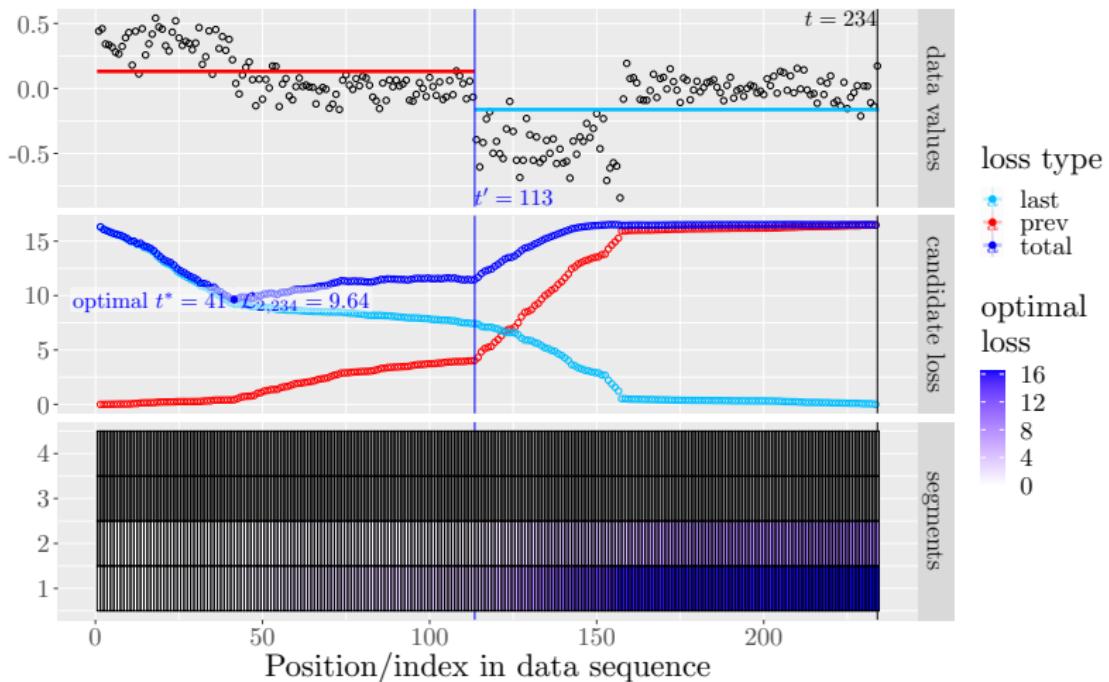
First iteration/changepoint of dynamic programming

$$\mathcal{L}_{2,t} = \min_{t' < t} \mathcal{L}_{1,t'} + c_{(t',t]}$$



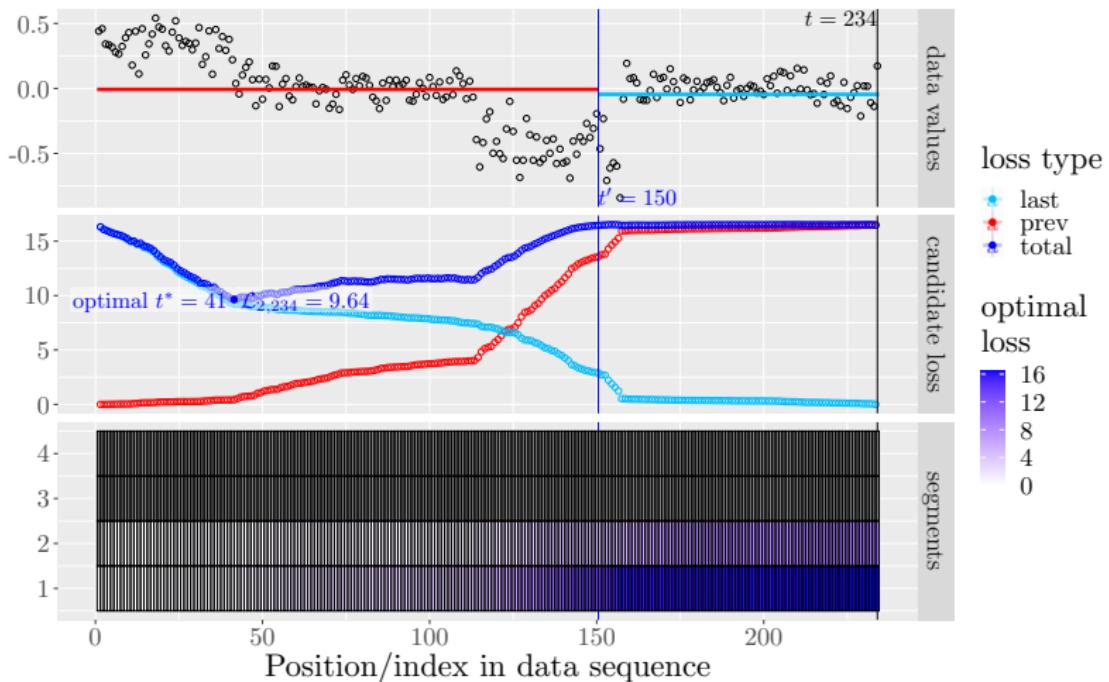
First iteration/changepoint of dynamic programming

$$\mathcal{L}_{2,t} = \min_{t' < t} \mathcal{L}_{1,t'} + c_{(t',t]}$$



First iteration/changepoint of dynamic programming

$$\mathcal{L}_{2,t} = \min_{t' < t} \mathcal{L}_{1,t'} + c_{(t',t]}$$



Time complexity for S segments and n data

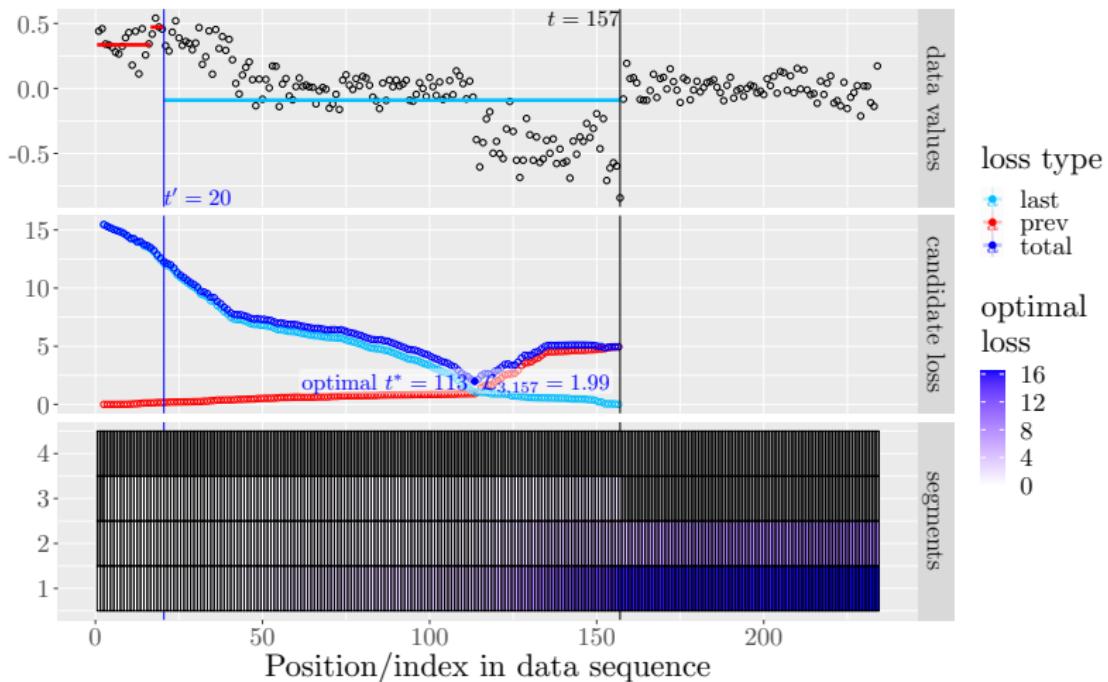
changes	naive	DP(classic)	DP(pruned)	BinSeg
1	$O(n)$	$O(n)$	$O(n)-O(n^2)$	$O(n)$
2	$O(n^2)$	$O(n^2)$	$O(n)-O(n^2)$	$O(\log n)-O(n)$
3	$O(n^3)$	$O(n^2)$	$O(n)-O(n^2)$	$O(\log n)-O(n)$
4	$O(n^4)$	$O(n^2)$	$O(n)-O(n^2)$	$O(\log n)-O(n)$
:	:	:	:	

Dynamic programming with either functional or inequality pruning leads to asymptotic speedups, see Maidstone R, Hocking TD, Rigaill G, Fearnhead P. On optimal multiple changepoint algorithms for large data. *Statistics and Computing* (2016).

Space: DP stores $O(nS)$ loss $\mathcal{L}_{s,t}$ values (always), BinSeg stores $O(S)$ split candidates (worst case).

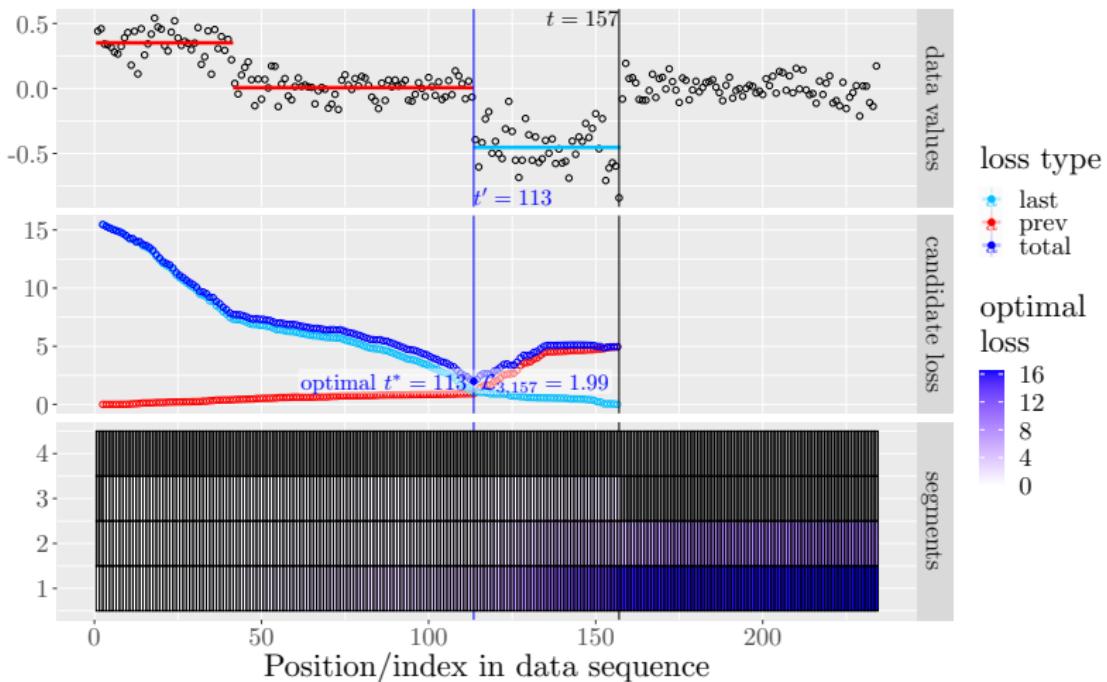
Second iteration/changepoint of dynamic programming

$$\mathcal{L}_{3,t} = \min_{t' < t} \mathcal{L}_{2,t'} + c_{(t',t]}$$



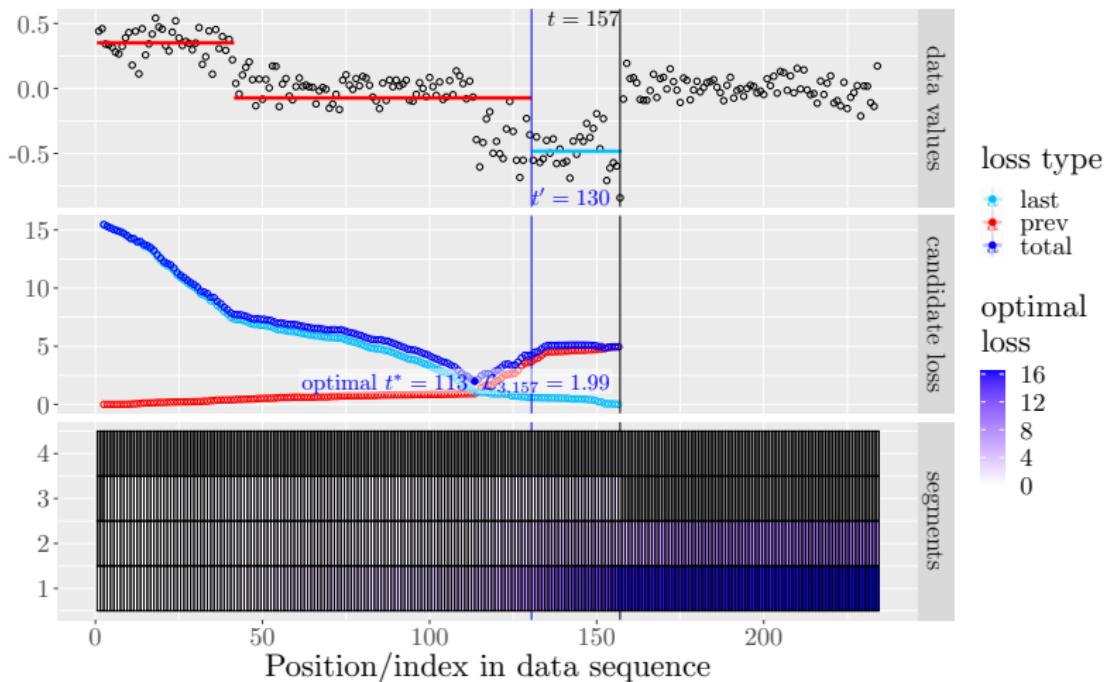
Second iteration/changepoint of dynamic programming

$$\mathcal{L}_{3,t} = \min_{t' < t} \mathcal{L}_{2,t'} + c_{(t',t]}$$



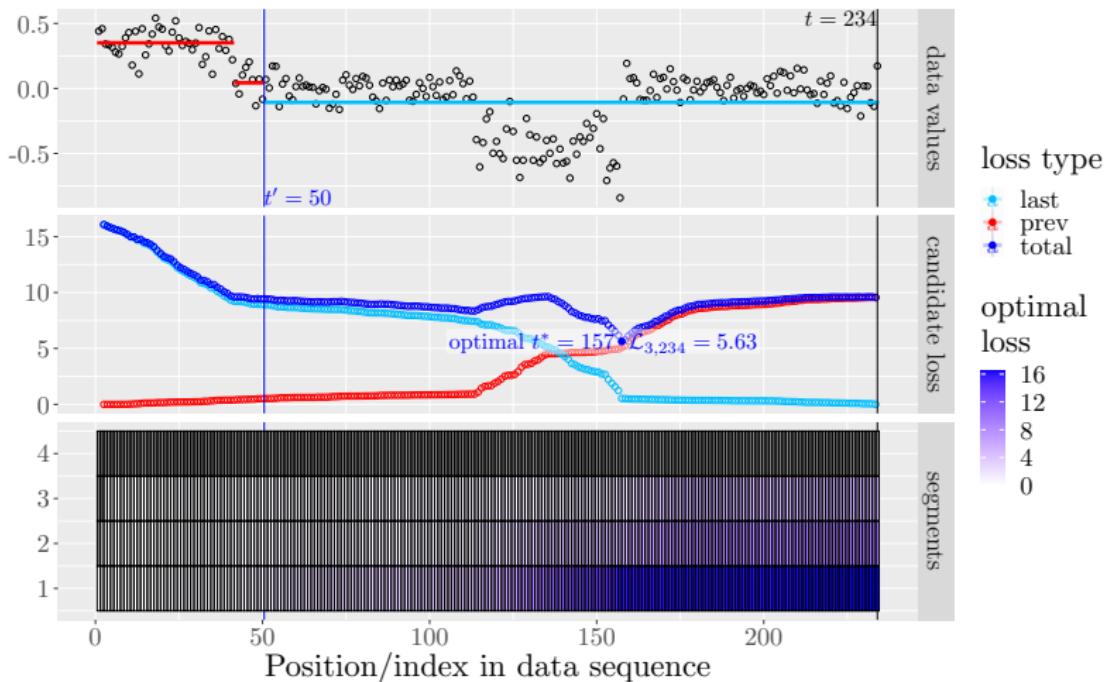
Second iteration/changepoint of dynamic programming

$$\mathcal{L}_{3,t} = \min_{t' < t} \mathcal{L}_{2,t'} + c_{(t',t]}$$



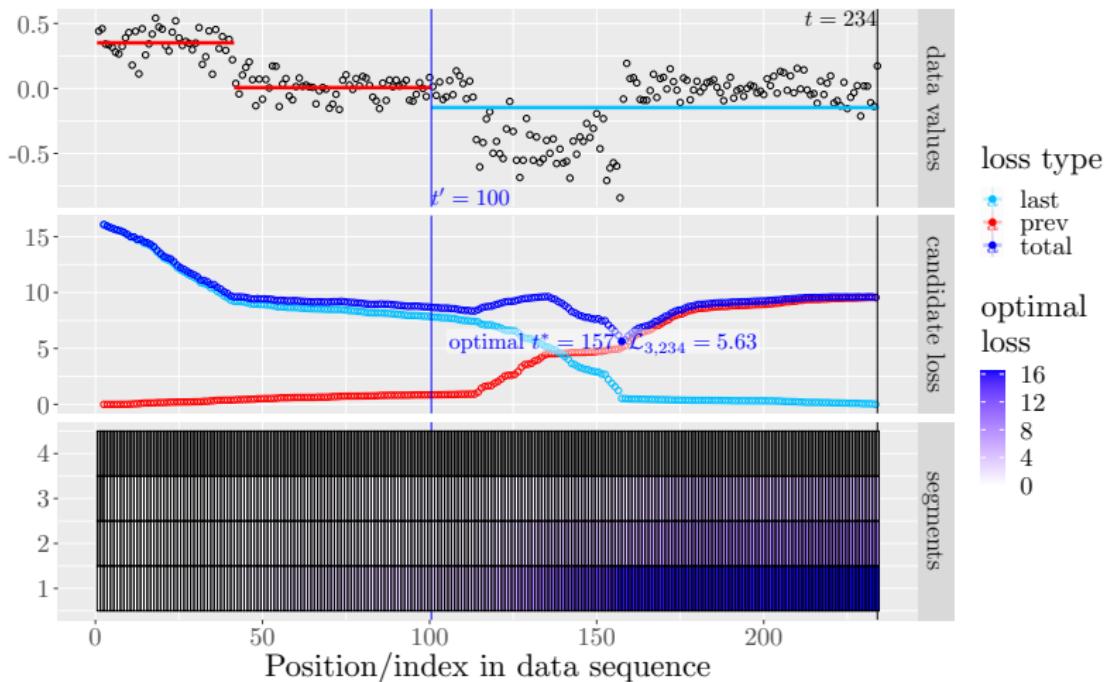
Second iteration/changepoint of dynamic programming

$$\mathcal{L}_{3,t} = \min_{t' < t} \mathcal{L}_{2,t'} + c_{(t',t]}$$



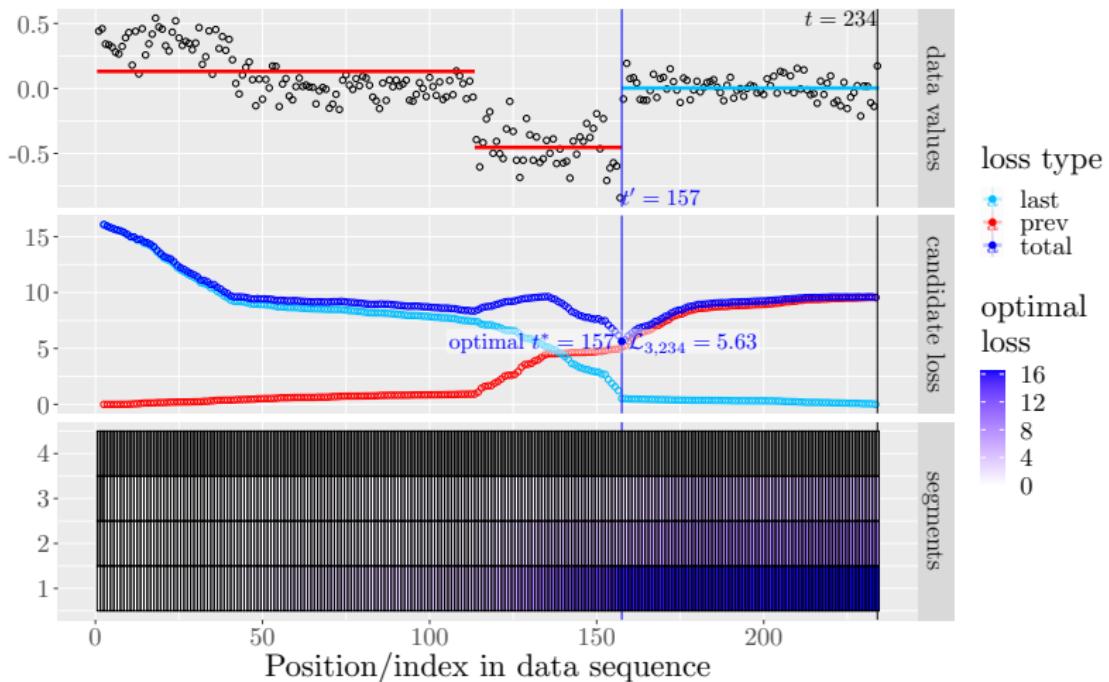
Second iteration/changepoint of dynamic programming

$$\mathcal{L}_{3,t} = \min_{t' < t} \mathcal{L}_{2,t'} + c_{(t',t]}$$

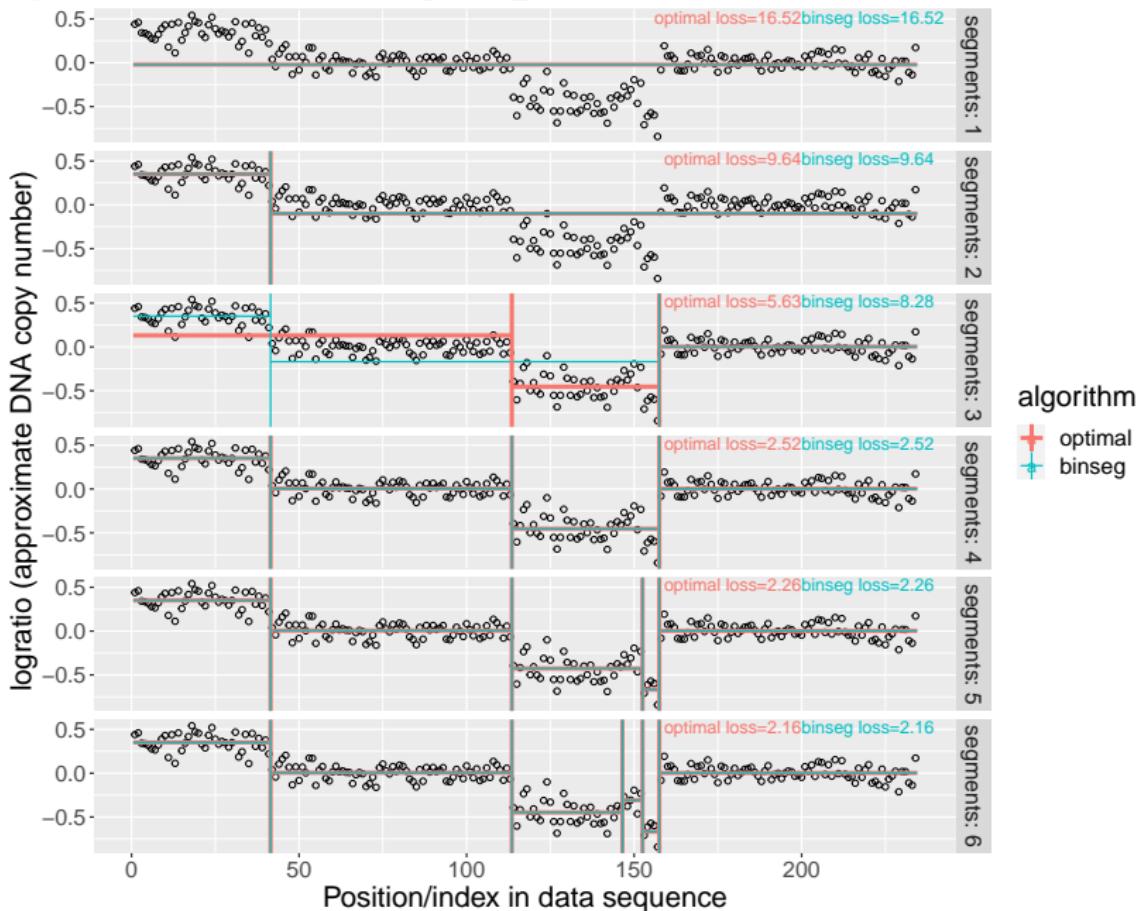


Second iteration/changepoint of dynamic programming

$$\mathcal{L}_{3,t} = \min_{t' < t} \mathcal{L}_{2,t'} + c_{(t',t]}$$

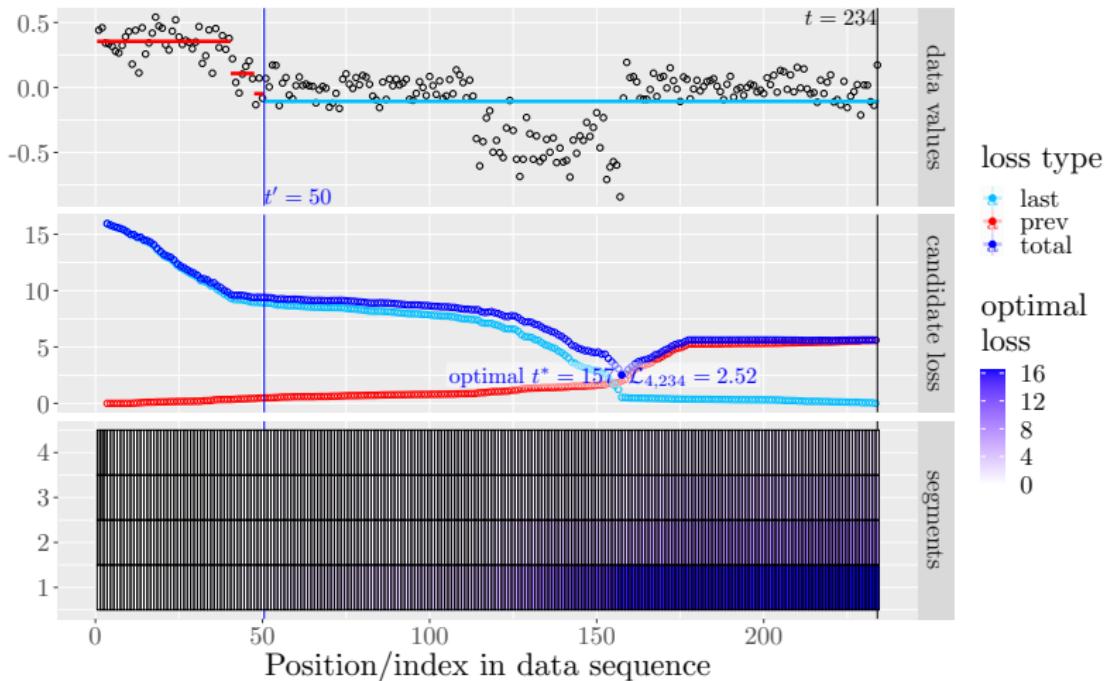


Comparison with binary segmentation



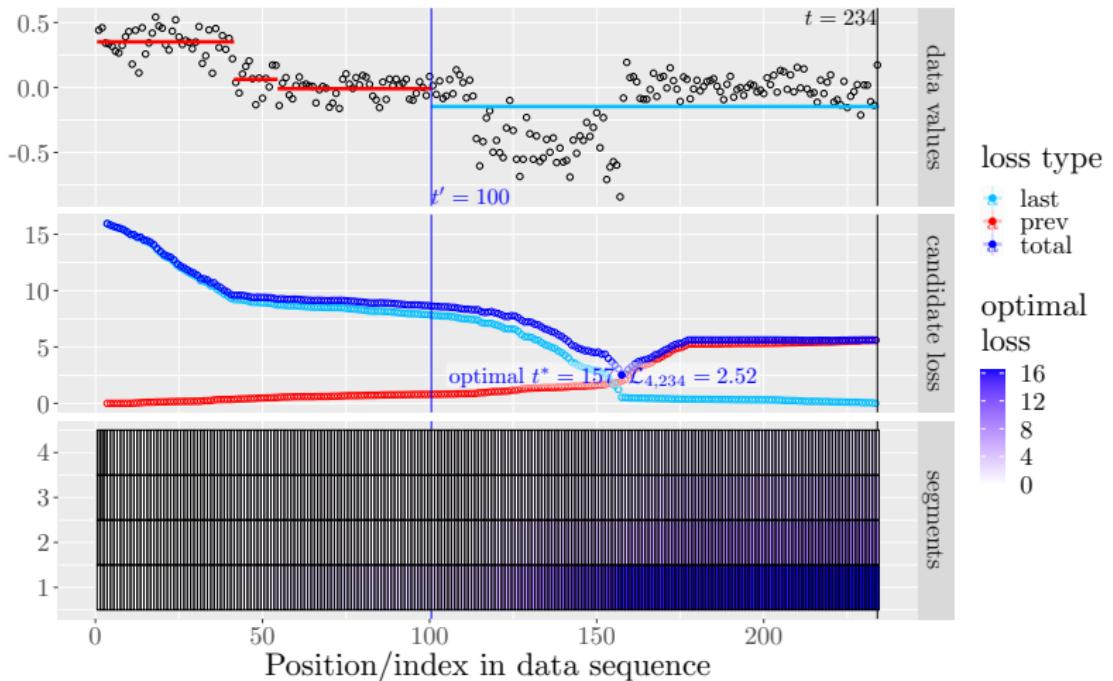
Third iteration/changepoint of dynamic programming

$$\mathcal{L}_{4,t} = \min_{t' < t} \mathcal{L}_{3,t'} + c_{(t',t]}$$



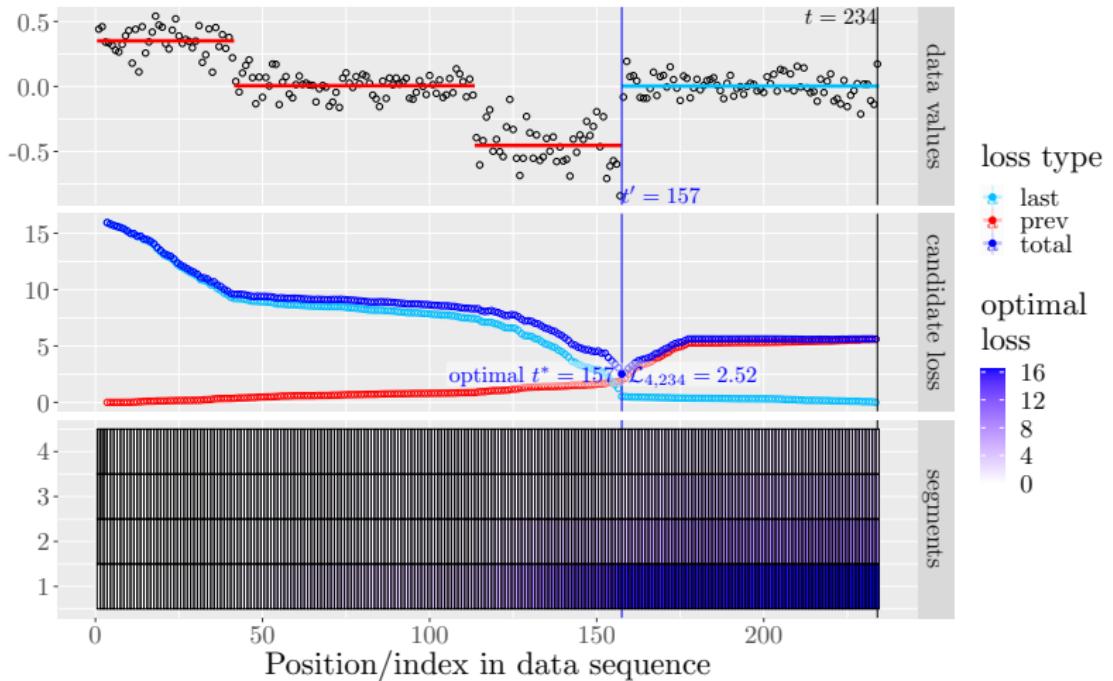
Third iteration/changepoint of dynamic programming

$$\mathcal{L}_{4,t} = \min_{t' < t} \mathcal{L}_{3,t'} + c_{(t',t]}$$



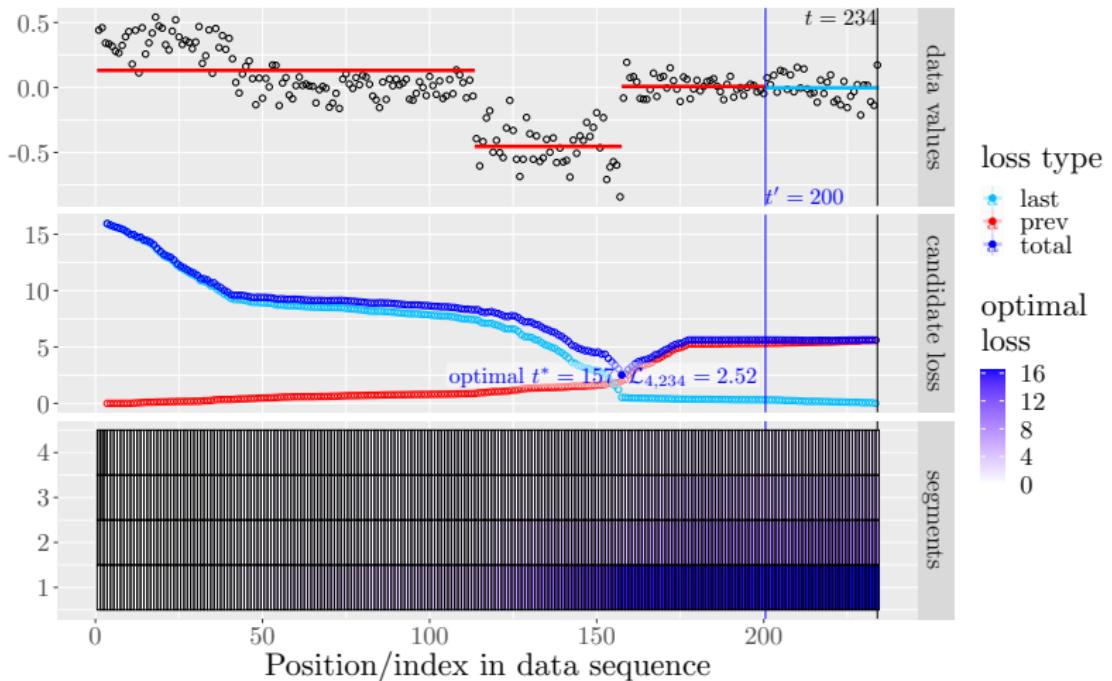
Third iteration/changepoint of dynamic programming

$$\mathcal{L}_{4,t} = \min_{t' < t} \mathcal{L}_{3,t'} + c_{(t',t]}$$



Third iteration/changepoint of dynamic programming

$$\mathcal{L}_{4,t} = \min_{t' < t} \mathcal{L}_{3,t'} + c_{(t',t]}$$



Comparison with previous approximate/heuristic algorithms

Every other algorithm we have seen so far has been approximate/heuristic, because it is not guaranteed to compute a global optimum of the objective function.

- ▶ K-means: local minimum squared error.
- ▶ EM algo for Gaussian mixtures: local maximum likelihood.
- ▶ Binary segmentation: local minimum squared error (maximum Gaussian likelihood).

In contrast the dynamic programming algorithm is guaranteed to compute a solution (segment means and changepoints) which is globally optimal (maximum Gaussian likelihood, minimum squared error).

Dynamic programming is $O(n^2S)$ for S segments and n data, whereas binary segmentation is $O(n \log S)$ best case and $O(nS)$ worst case. (optimal algorithms are typically slower than approximate/heuristic algorithms)

Other optimal algorithms

So far we have only discussed the *constrained problem*:
hyper-parameter is number of segments $S \in \{1, \dots, n\}$.

An alternative formulation involves specifying a non-negative penalty hyper-parameter, $\lambda \in \mathbb{R}^+$, and then solving the *penalized problem*:

$$\min_{m_1, \dots, m_n \in \mathbb{R}} \underbrace{\sum_{i=1}^n \ell(m_i, x_i)}_{\text{loss}} + \lambda \underbrace{\sum_{i=1}^{n-1} I[m_i \neq m_{i+1}]}_{\text{number of changes}}.$$

The resulting model is the solution to a constrained problem (for some number of segments S).

Dynamic programming can also be used to solve this problem, using $O(n)$ space, best case $O(n)$ time (PELT algorithm), worst case $O(n^2)$ time. Optimal Detection of Changepoints With a Linear Computational Cost Killick R, Fearnhead P and Eckley IA. Journal of the American Statistical Association (2012).

Possible exam questions

- ▶ For S segments how does the binary segmentation loss compare to the dynamic programming loss? Answer for each $S \in \{1, 2, 3, 4\}$ one or more of: less than, equal, and/or greater than.
- ▶ How many values for the last changepoint variable are considered in the computation of $\mathcal{L}_{3,100}$?
- ▶ Is $\{\text{binary segmentation, dynamic programming}\}$ an $\{\text{optimal, approximate}\}$ algorithm? Why?