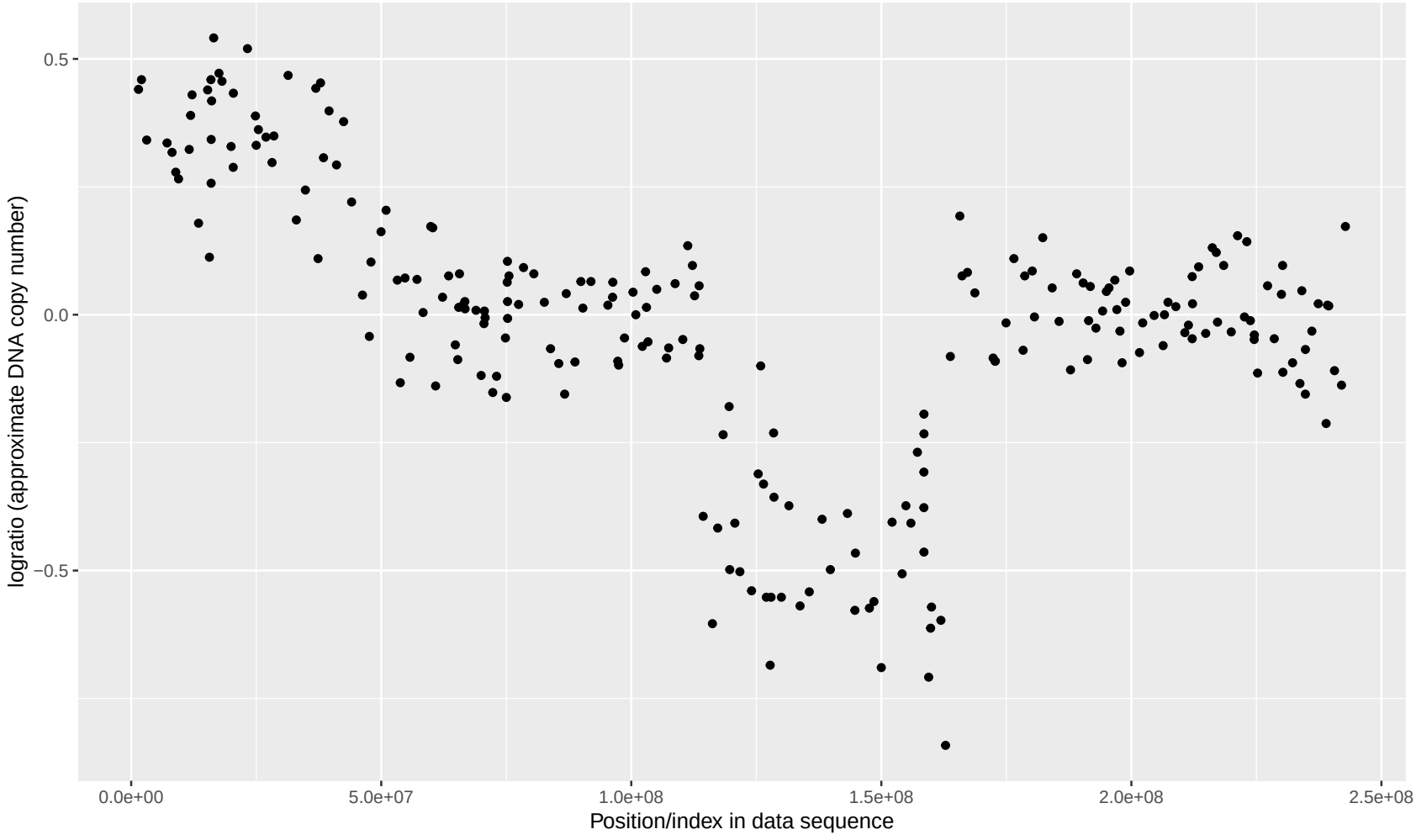
Optimal segmentation

Toby Dylan Hocking

Background: detecting abrupt changes is important

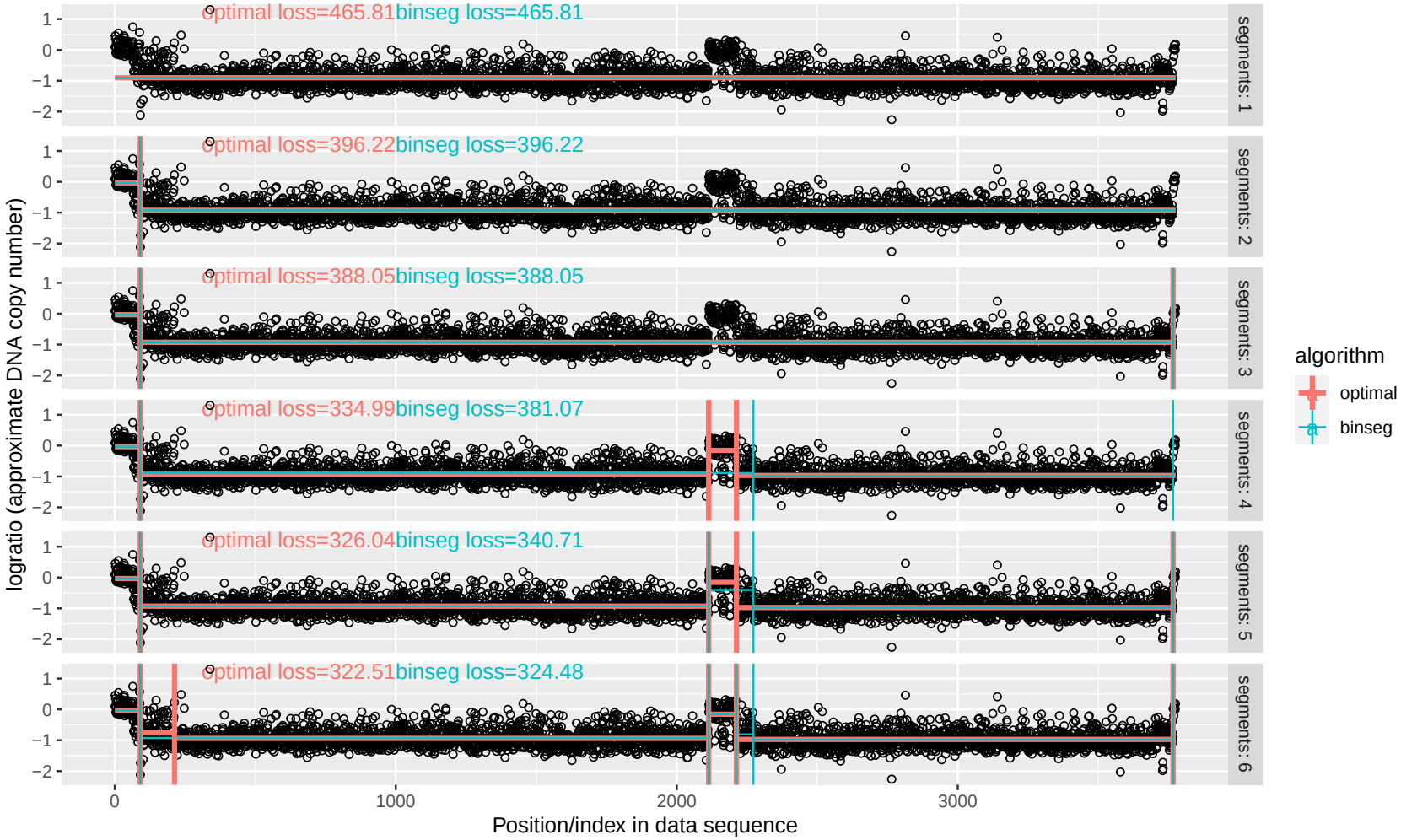
Example from cancer diagnosis: breakpoints are associated with aggressive disease in neuroblastoma.



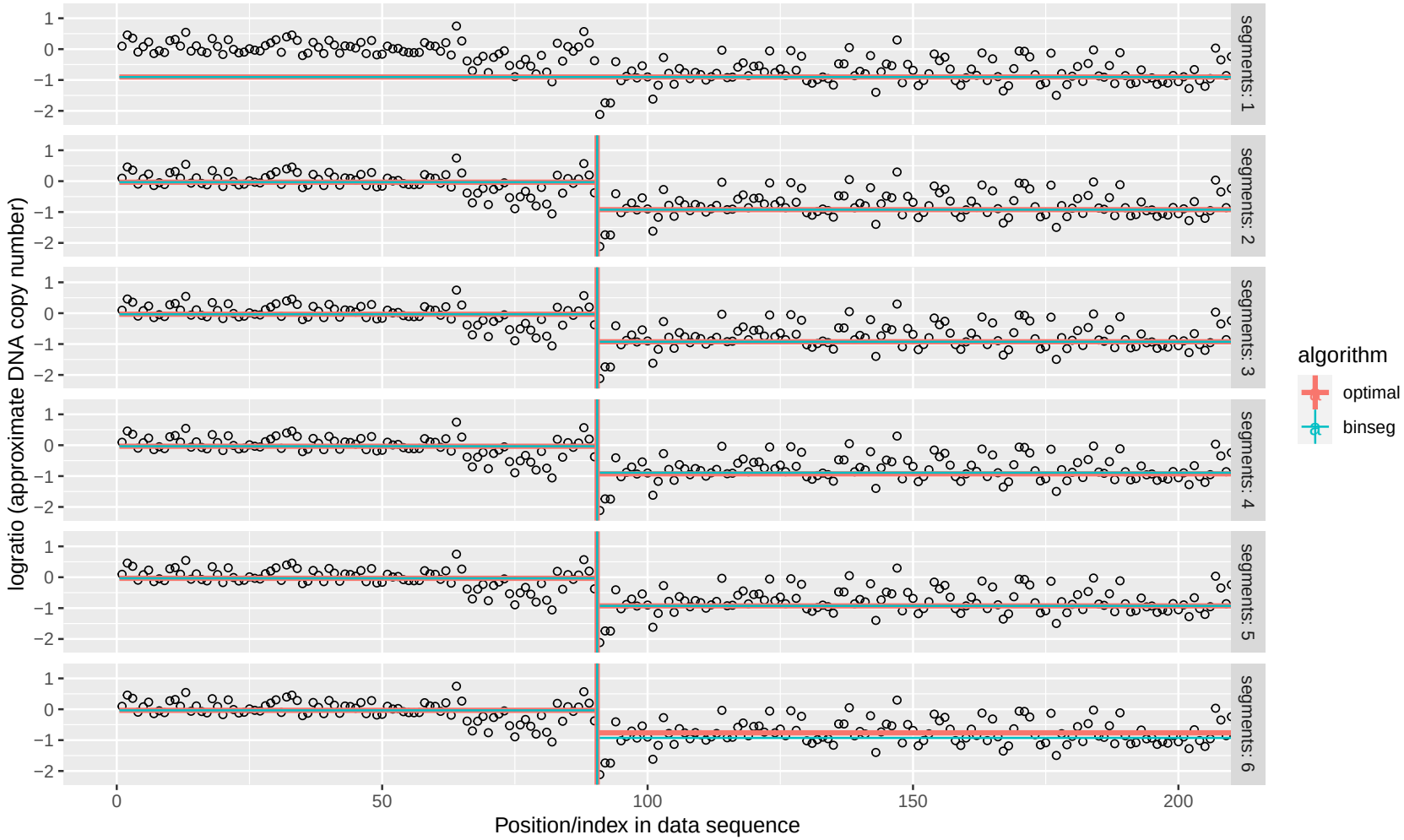
Motivation for optimal detection

- 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.

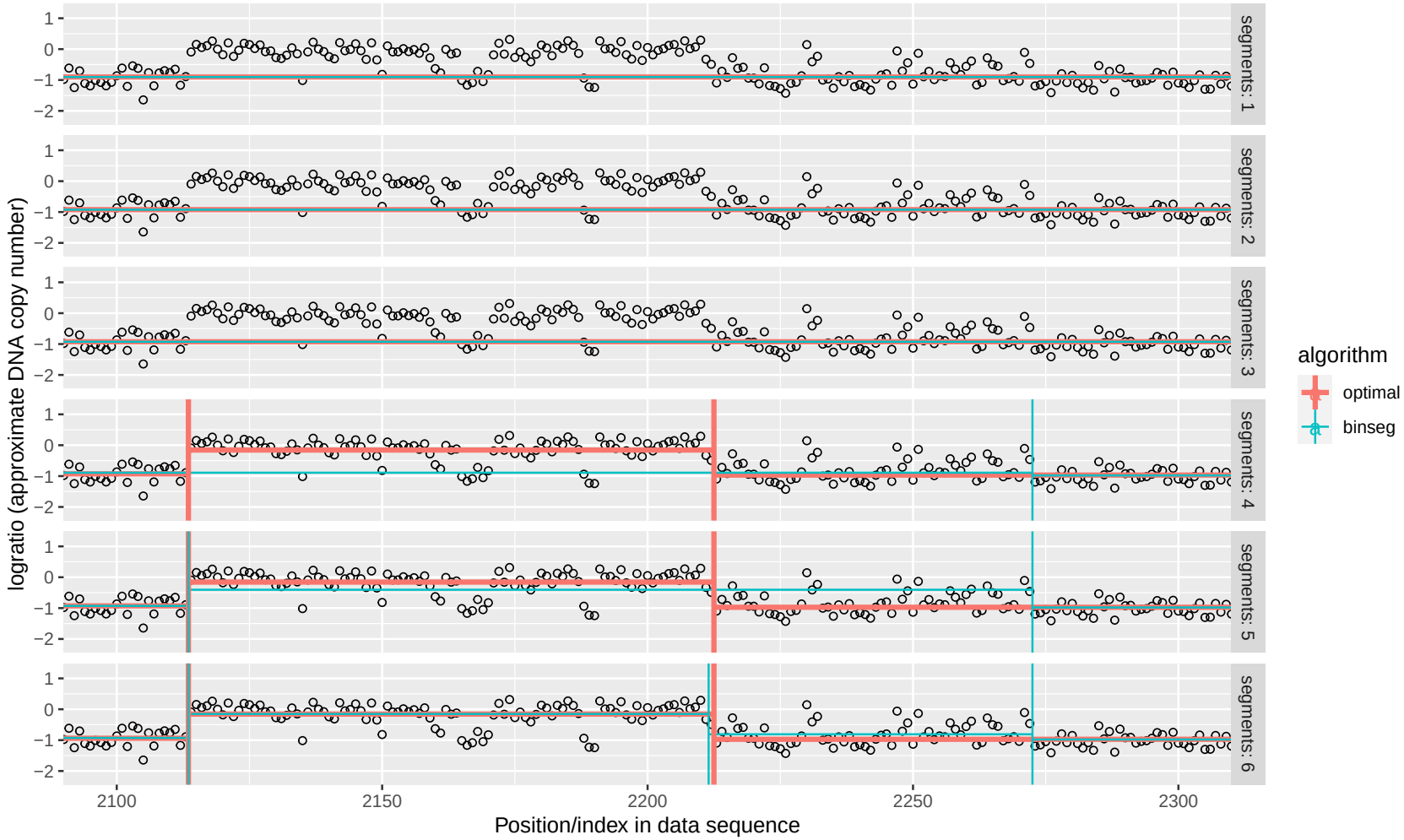
Example 1: stuck with sub-optimal change



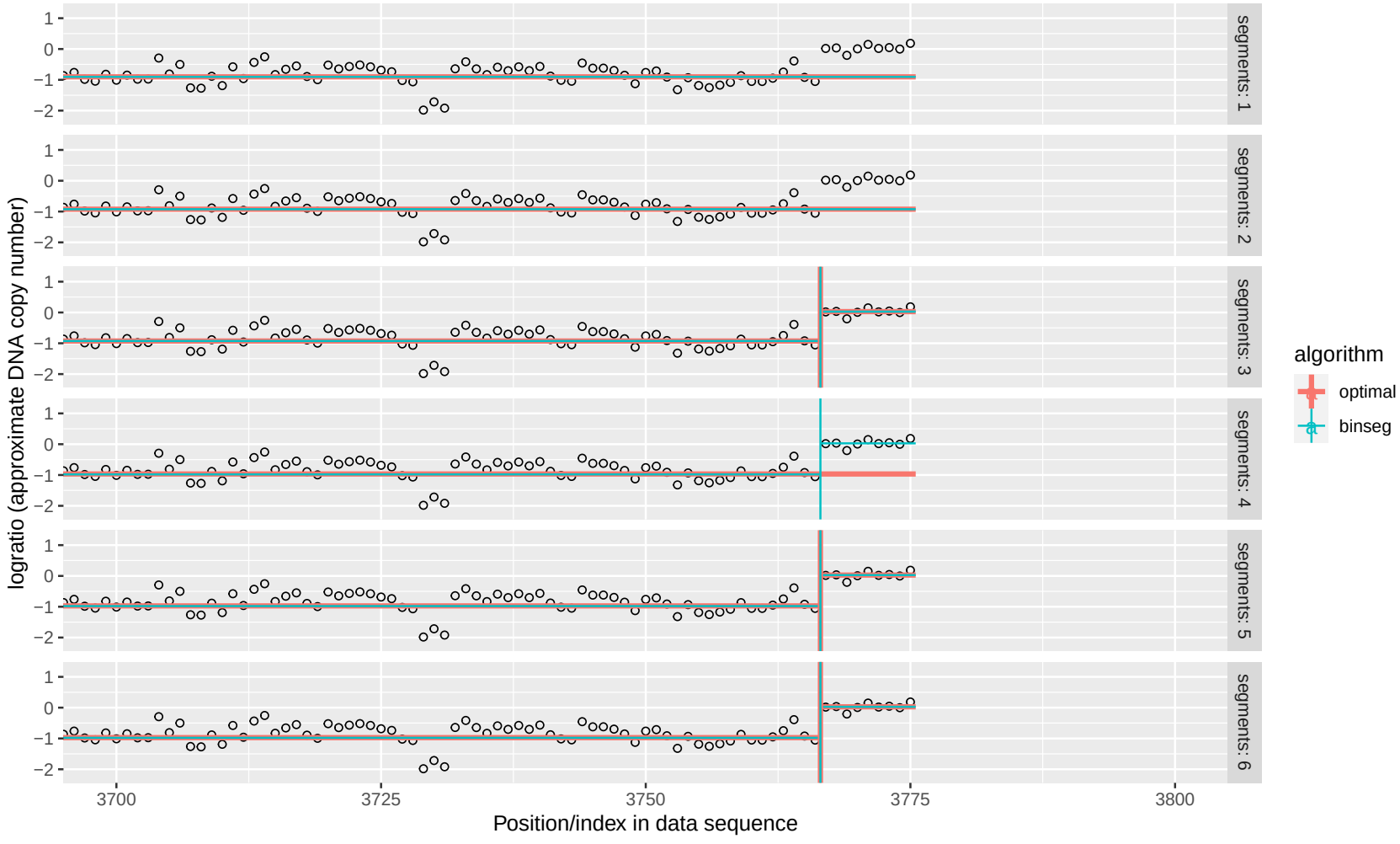
Zoom to start



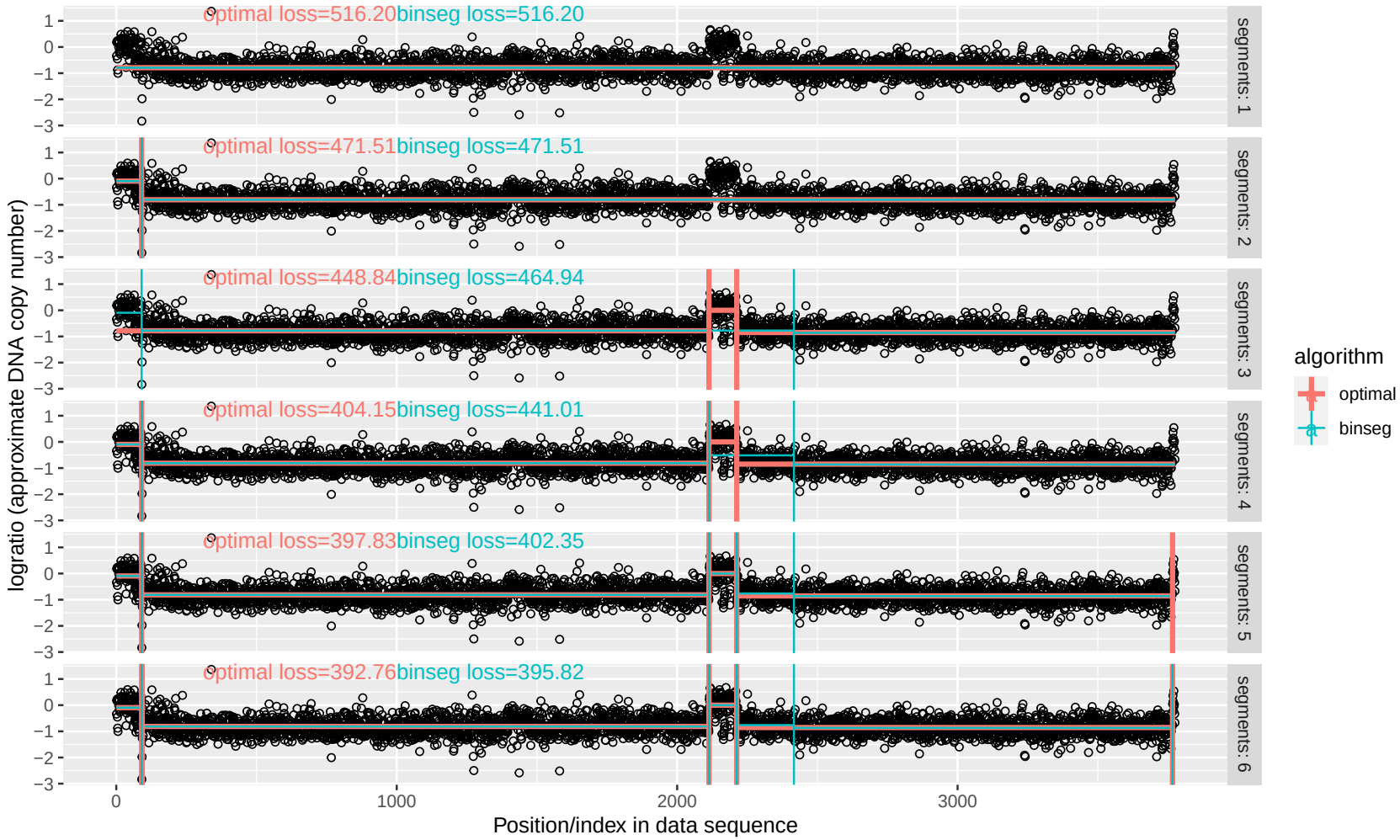
Zoom to center



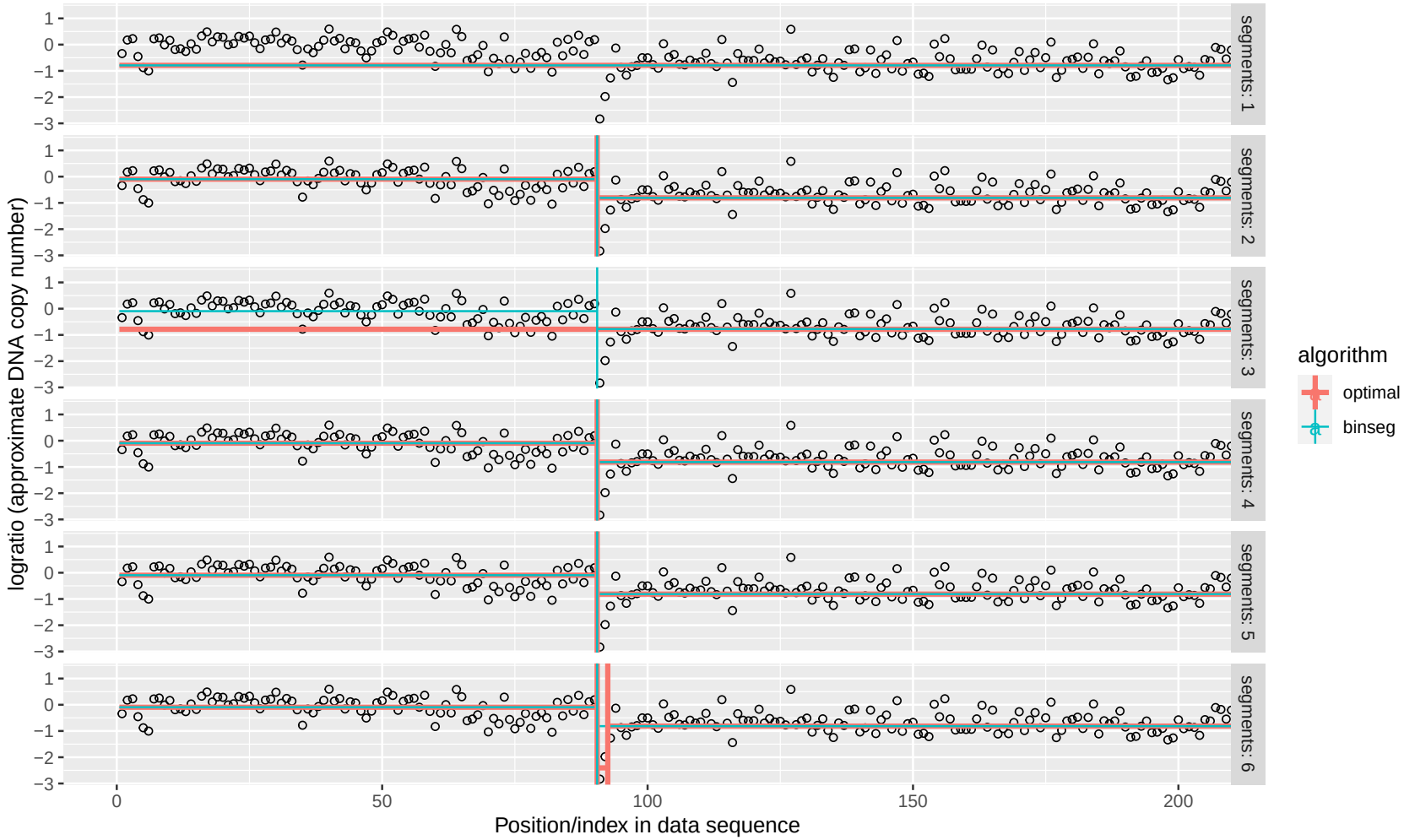
Zoom to end


logratio (approximate DNA copy number)
Position/index in data sequence
segments: 1
segments: 2
segments: 3
segments: 4
segments: 5
segments: 6
algorithm
optimal
binseg

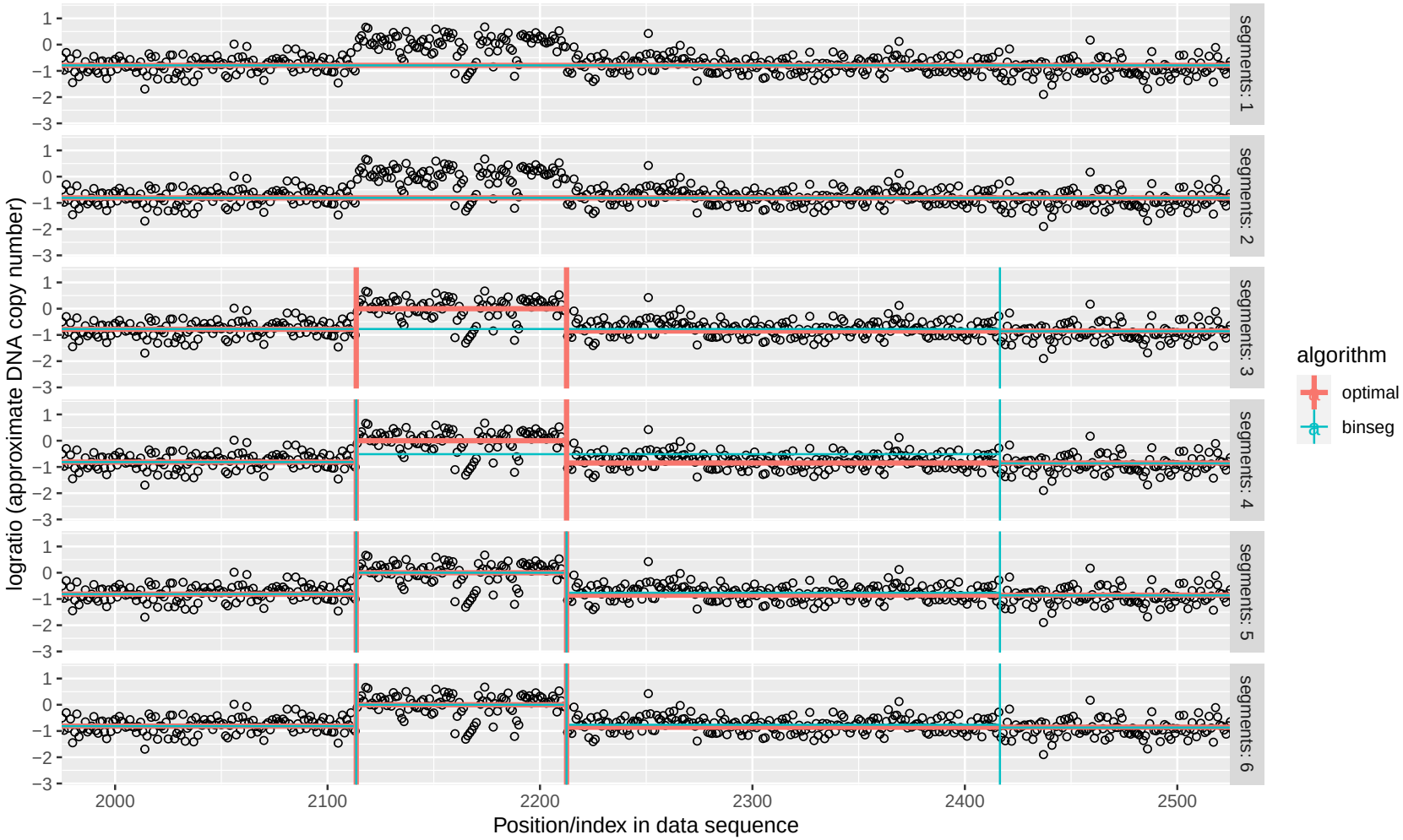
Example 2: missing a small change down



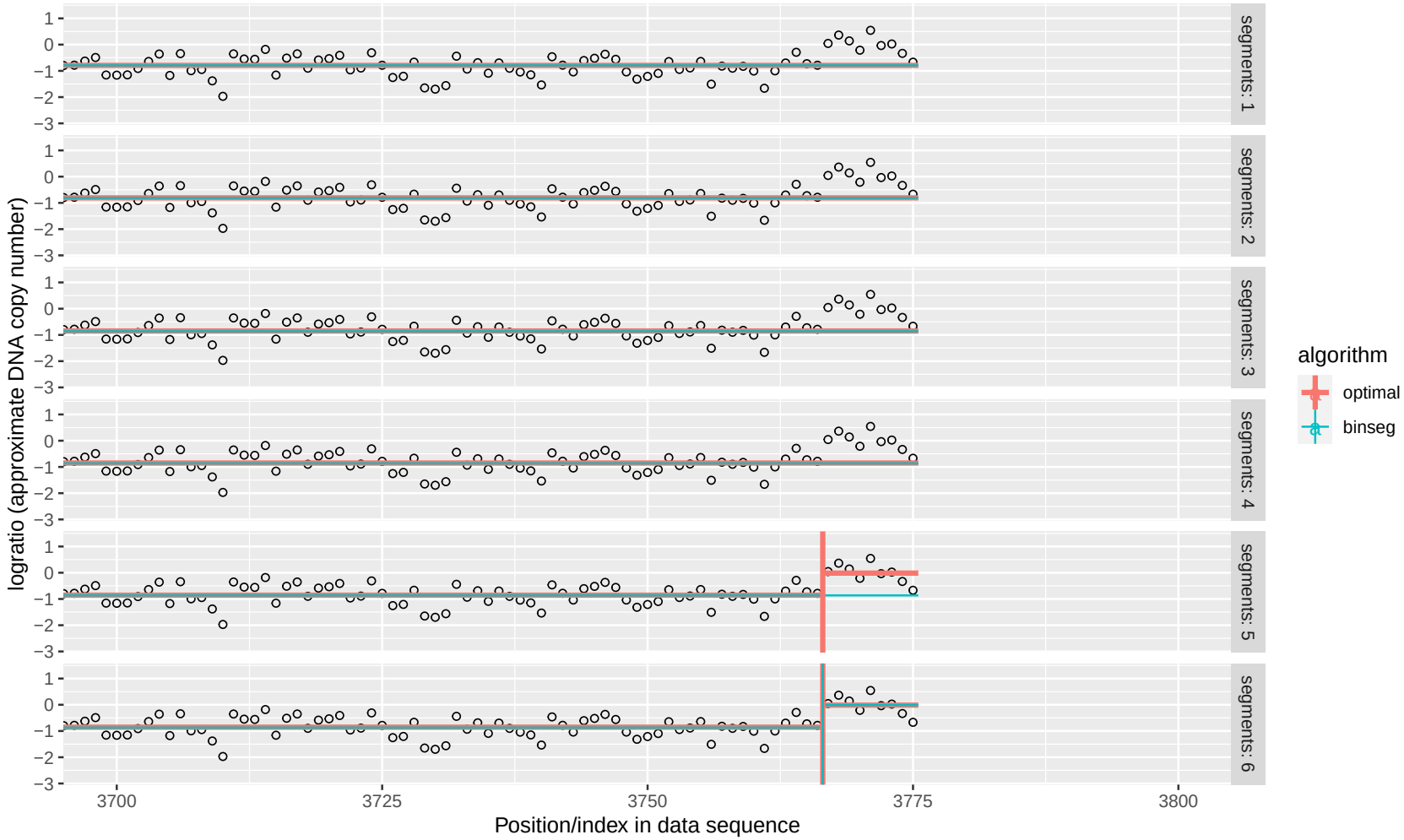
Zoom to start



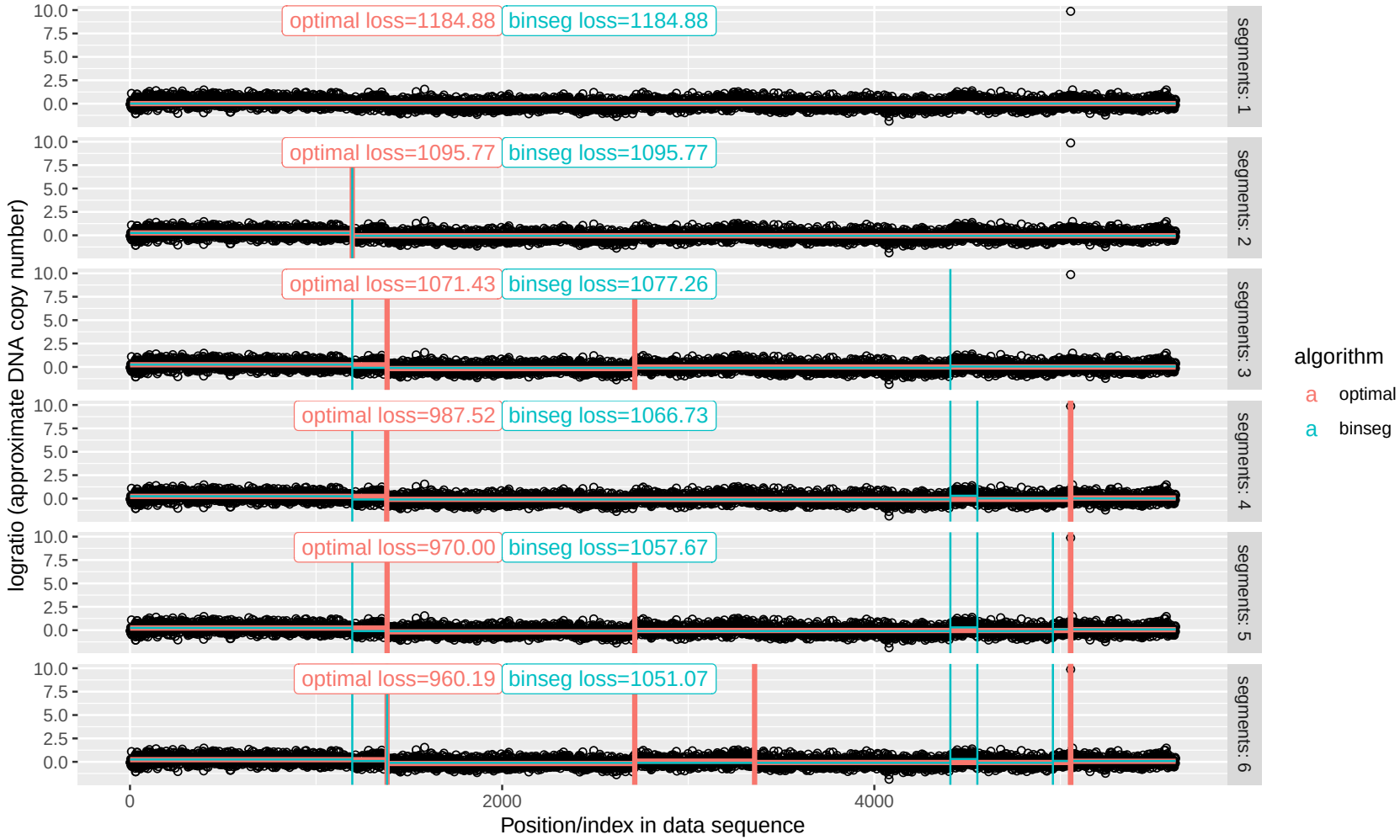
Zoom to center



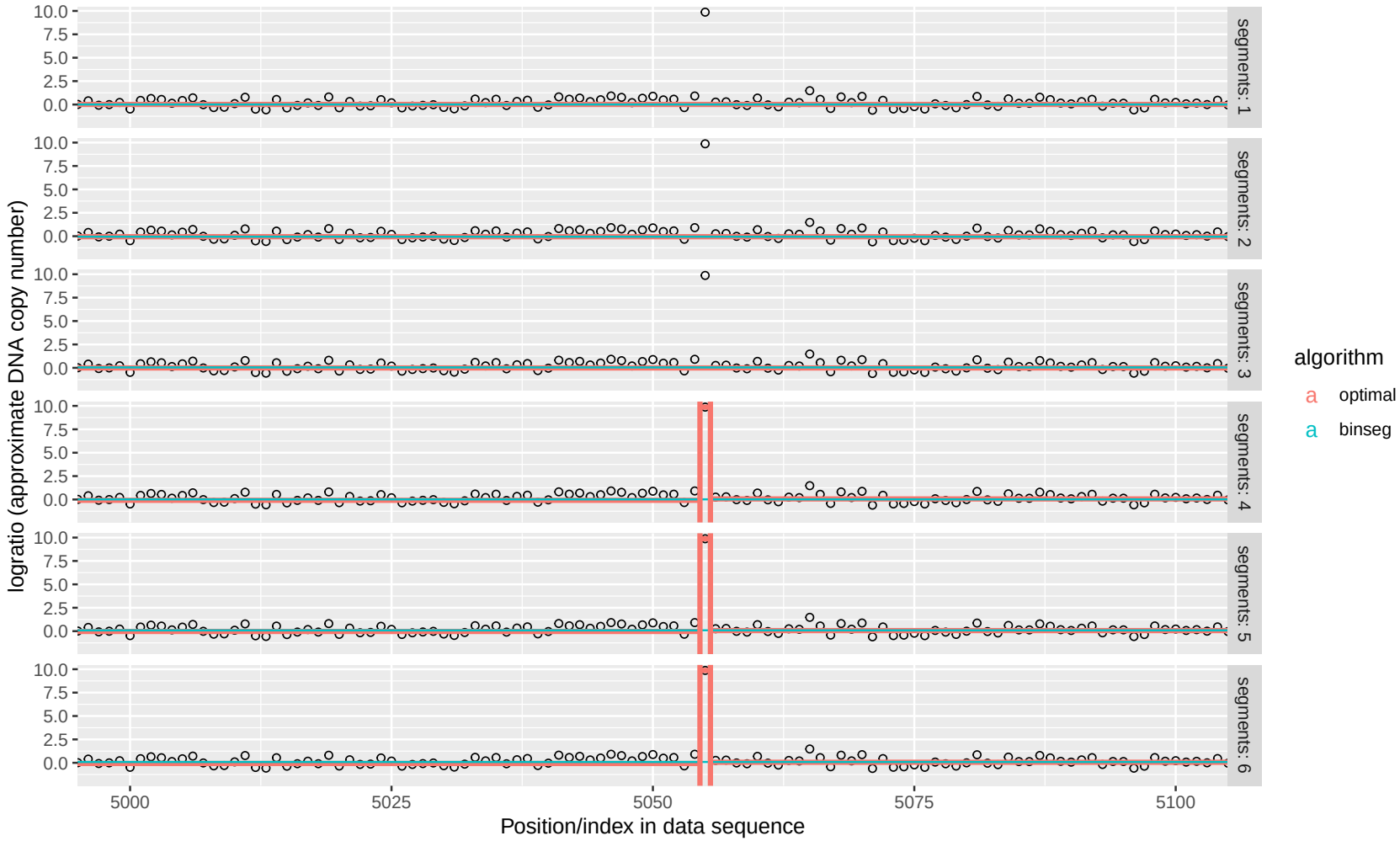
Zoom to end


logratio (approximate DNA copy number)
Position/index in data sequence
segments: 1
segments: 2
segments: 3
segments: 4
segments: 5
segments: 6
algorithm
optimal
binseg

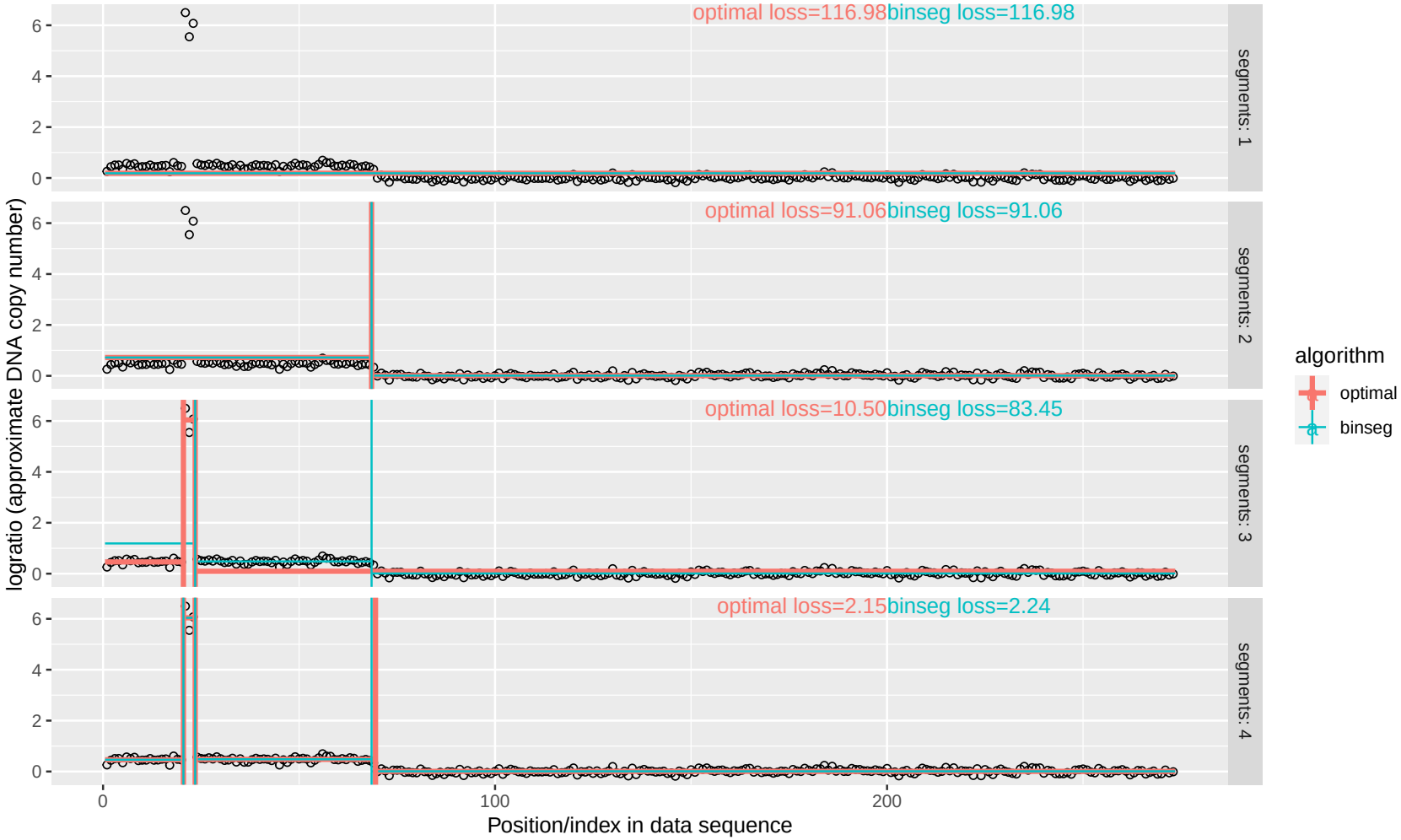
Example 3: difficult to detect an outlier



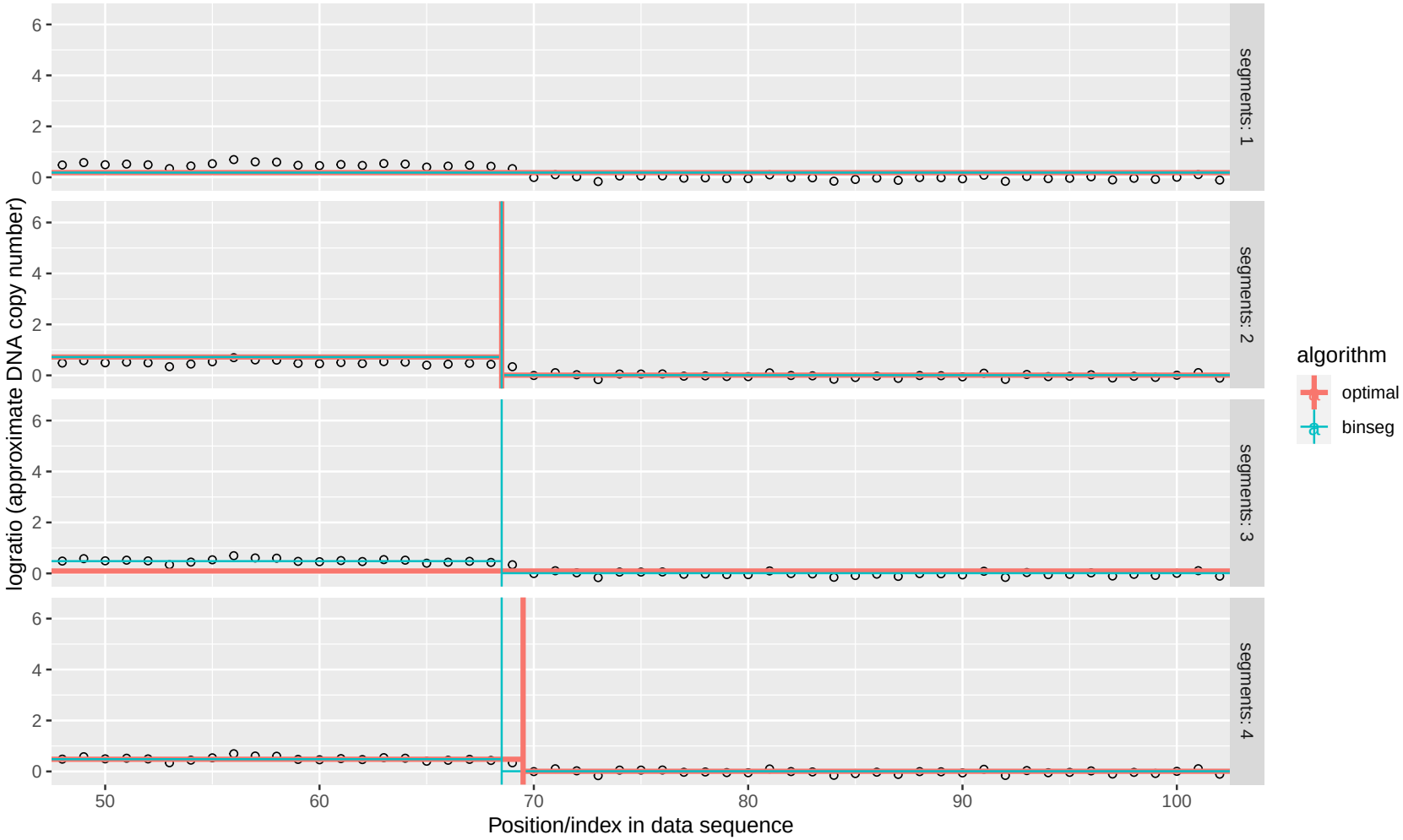
Zoom X axis to outlier



Example 4: stuck with sub-optimal change

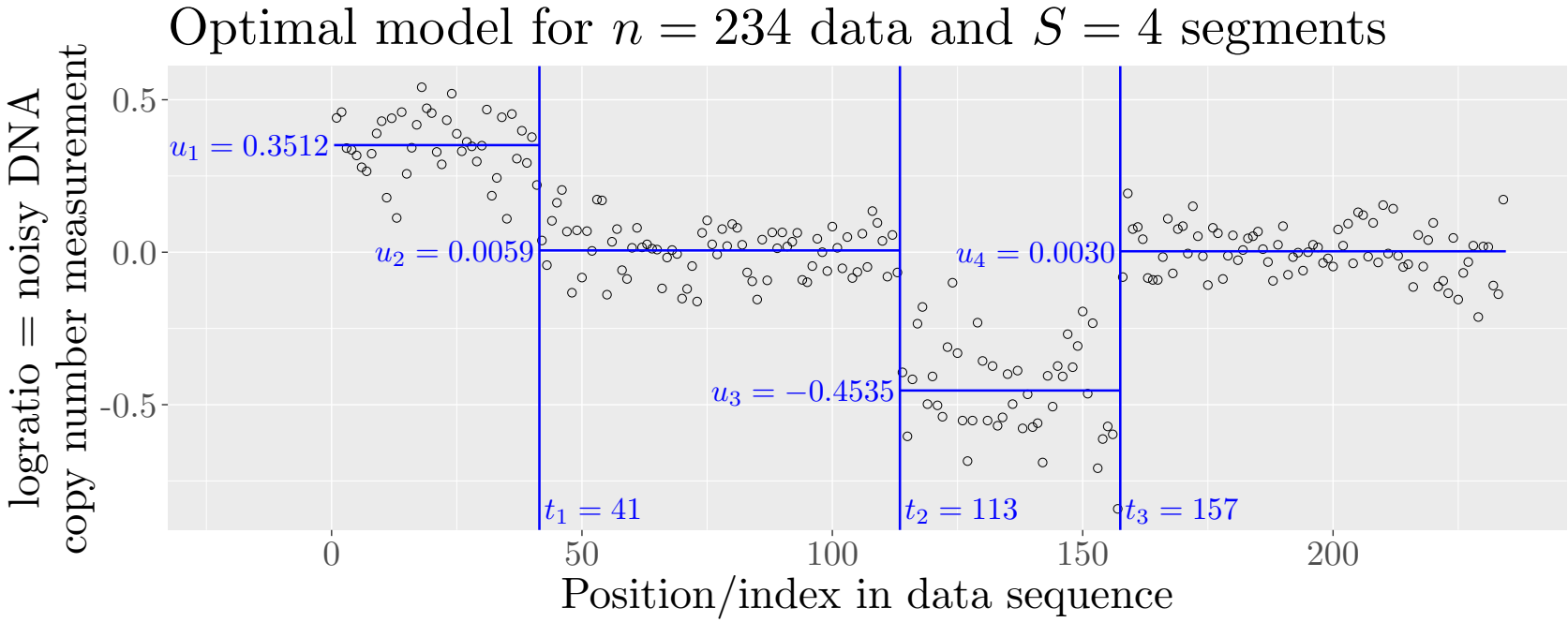


Zoom to changepoint



Dynamic programming for optimal changepoint detection

- We have n data $z_1, \ldots, z_n$.
- Fix the number of segments $S \in \{1, 2, \ldots, n\}$.
- Optimization variables: $S-1$ changepoints $t_1 < \cdots < t_{S-1}$ and S segment means $u_1, \ldots, u_S \in \mathbb{R}$ ($t_0 = 0$, $t_S = n$).
- Statistical model: for every segment $s \in \{1, \ldots, S\}$, $z_i \overset{\text{iid}}{\sim} N(u_s, \sigma^2)$ for every data point $i \in (t_{s-1}, t_s]$ implies square loss function $\ell(u_s, z_i) = (u_s - z_i)^2$ to minimize.



Maximum likelihood inference for S segments and n data

The best loss for S segments and n data is

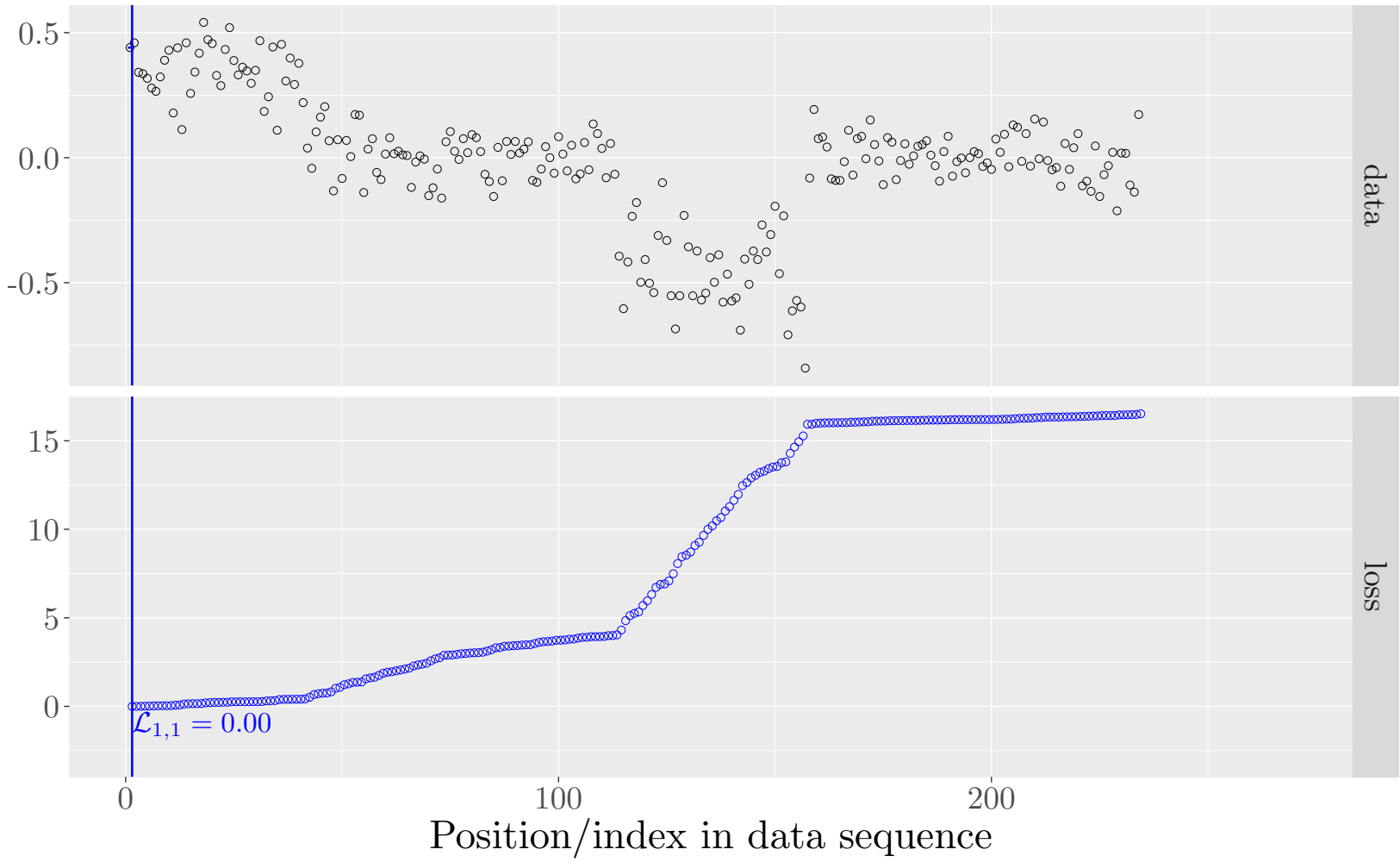
$$\begin{aligned}
\mathcal{L}_{S,n} &= \min_{\substack{\mathbf{u}\in\mathbb{R}^S \\ 0=t_0<t_1<\cdots<t_{S-1}<t_S=n}} \sum_{s=1}^{S} \sum_{i=t_{s-1}+1}^{t_s} \ell(u_s, z_i) \\
&= \min_{t_{S-1}} \underbrace{\min_{\substack{u_1,\ldots,u_{S-1} \\ t_1<\cdots<t_{S-2}}} \sum_{s=1}^{S-1} \sum_{i=t_{s-1}+1}^{t_s}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

- 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).

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

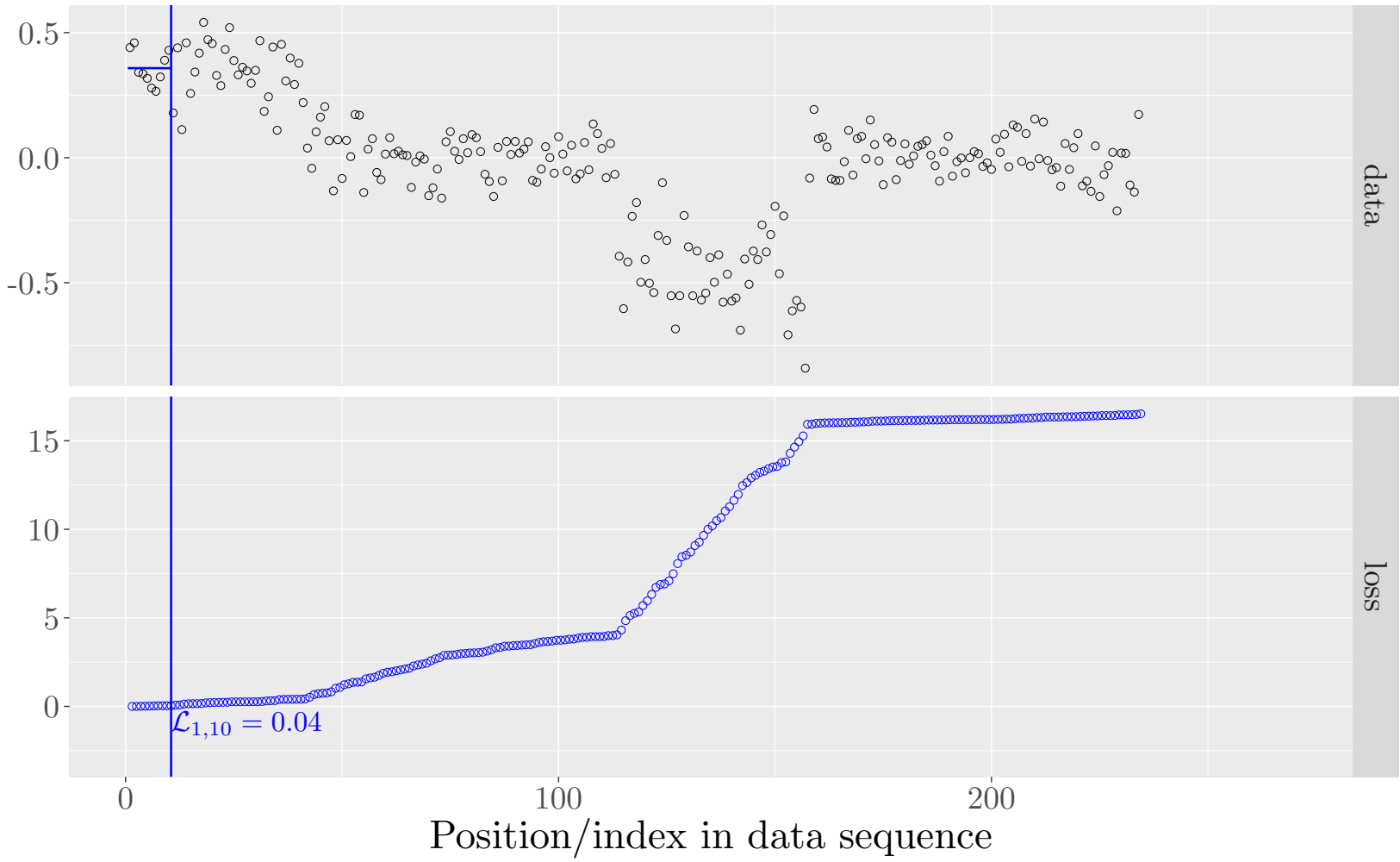
Initialization of dynamic programming

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



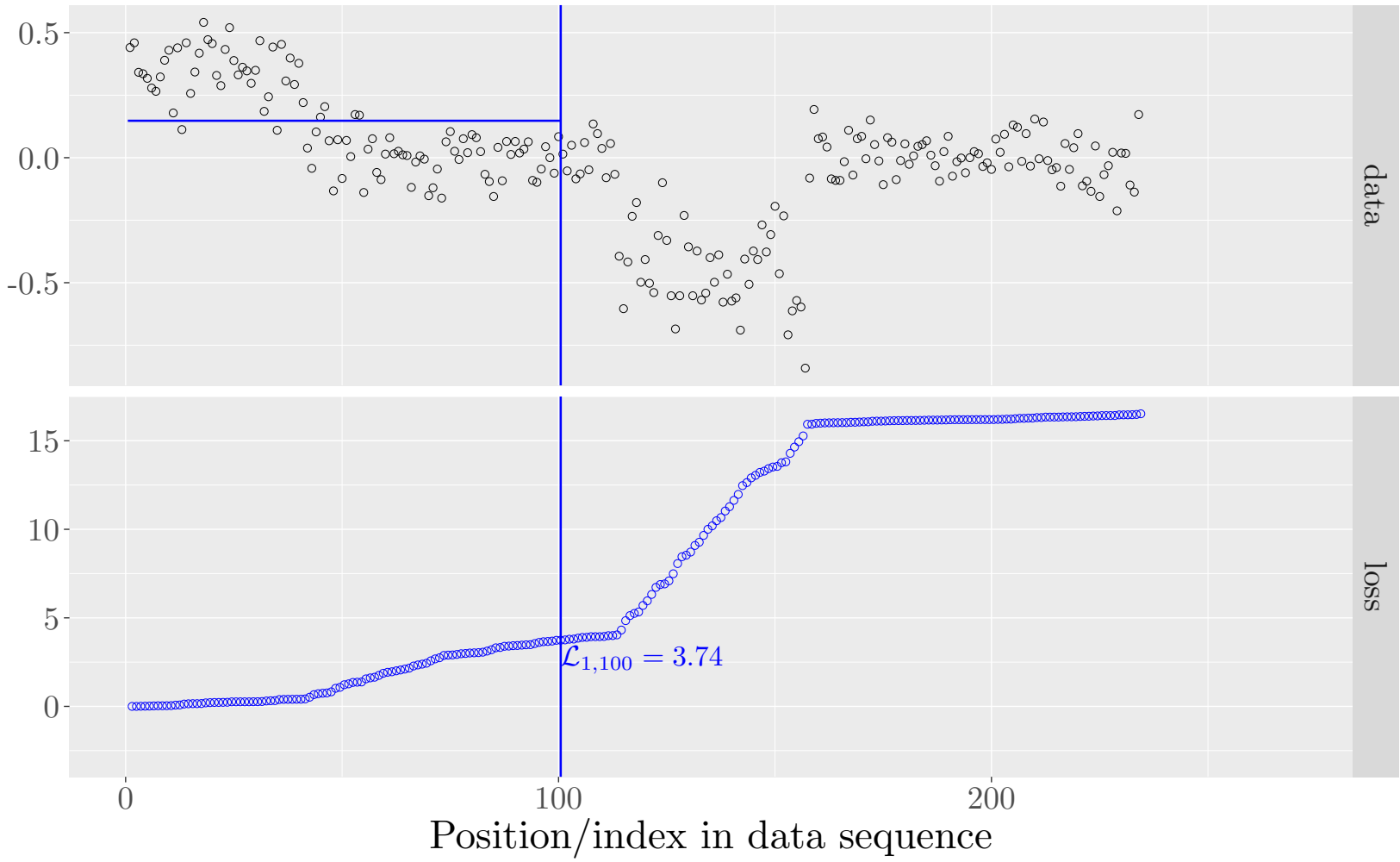
Initialization of dynamic programming

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



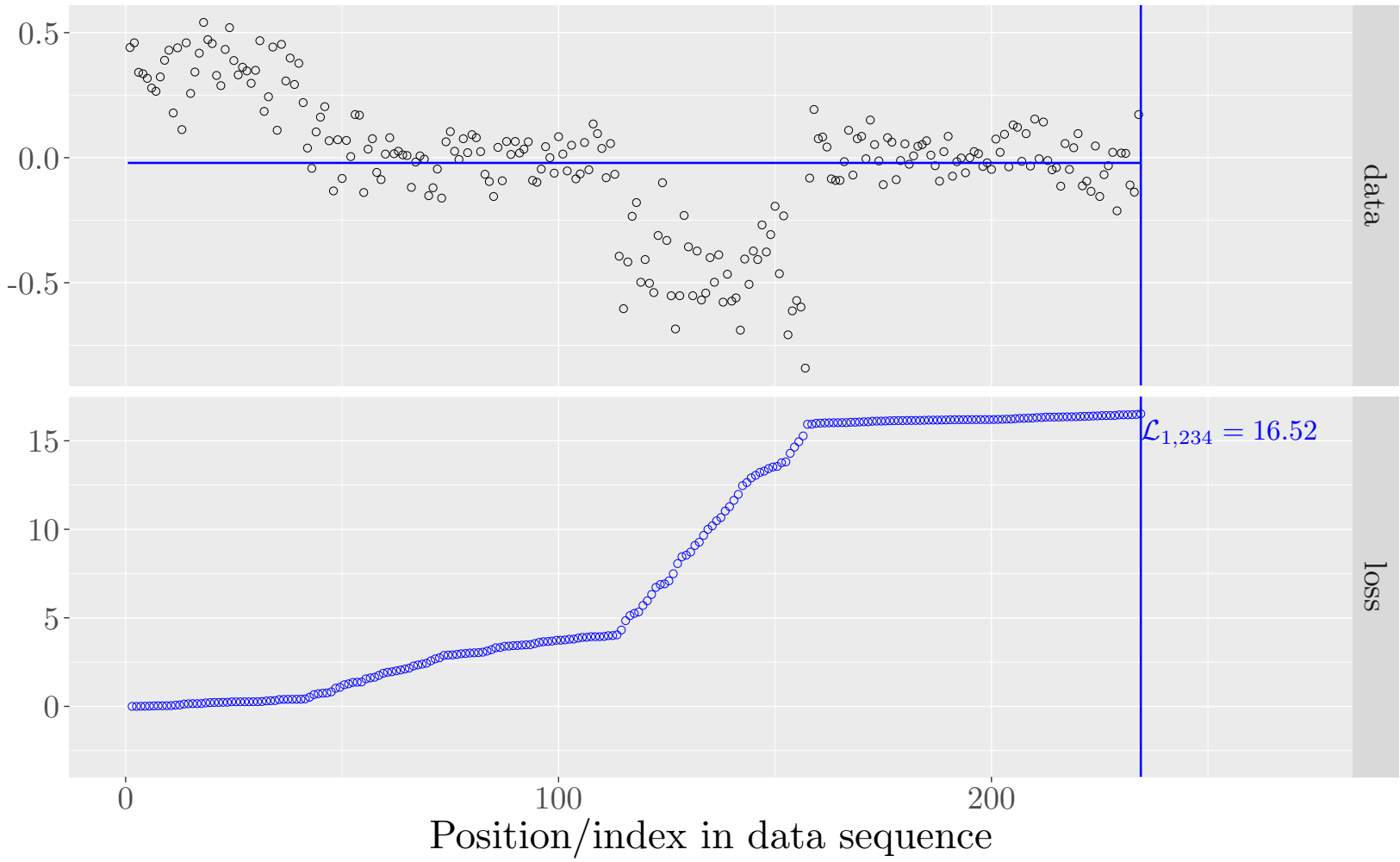
Initialization of dynamic programming

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



Initialization of dynamic programming

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



Efficient computation of c values

The best square loss up to t is

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

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

As can all best means up to t, $\hat{u}_t = S_t/t$.

As can cumulative sum of squares, $Q_t = \sum_{i=1}^{t} z_i^2$.

All best loss values up to t can also be computed in linear time,

$$t\hat{u}_t^2 - 2\hat{u}_t \sum_{i=1}^{t} z_i + \sum_{i=1}^{t} z_i^2 = t(S_t/t)^2 - 2(S_t/t)(S_t) + Q_t = Q_t - S_t^2/t.$$

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

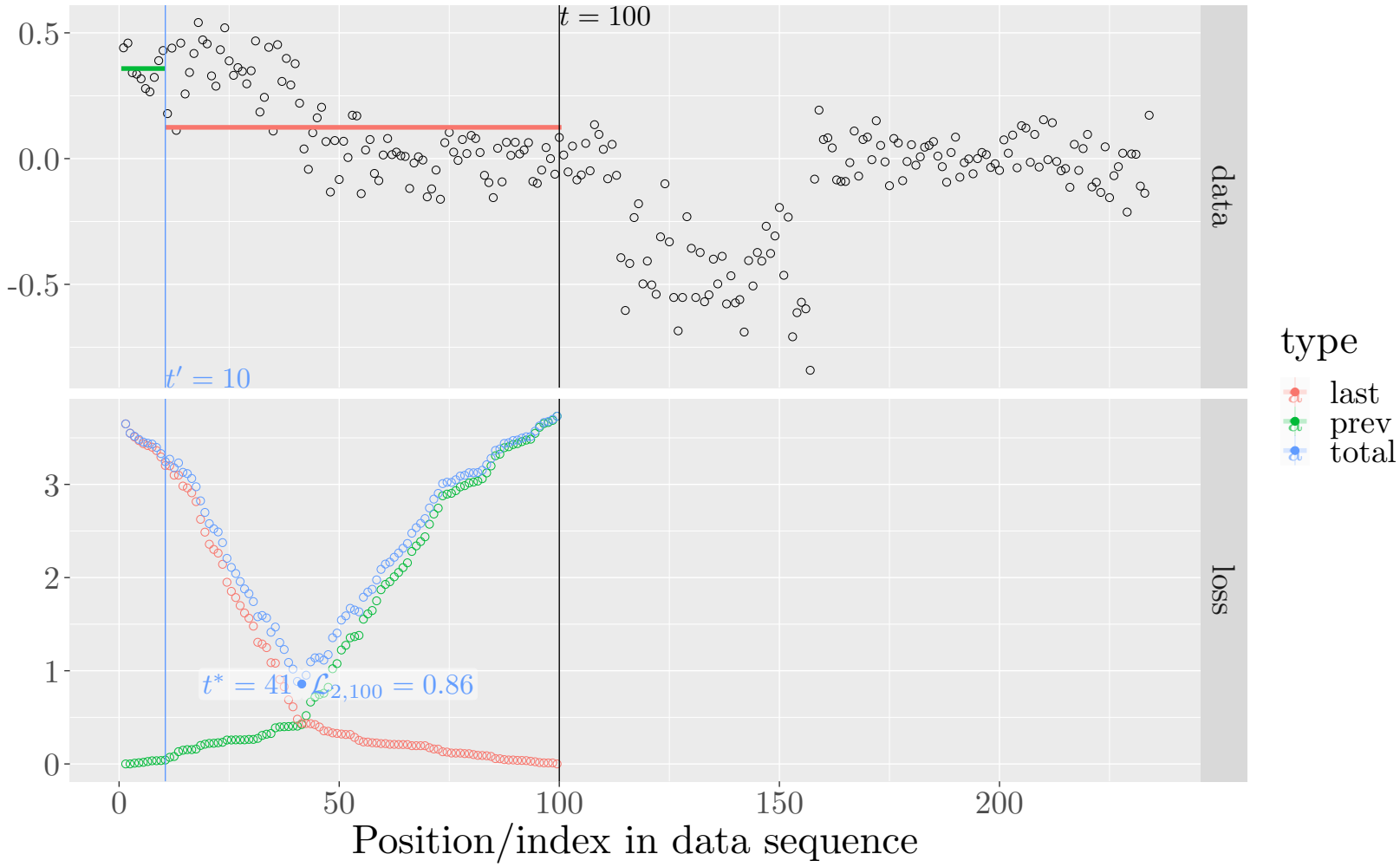
$$\mathcal{L}_{2,t} = \min_{t'<t} \mathcal{L}_{1,t'} + c_{(t',t]}$$

First iteration/changepoint of dynamic programming

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

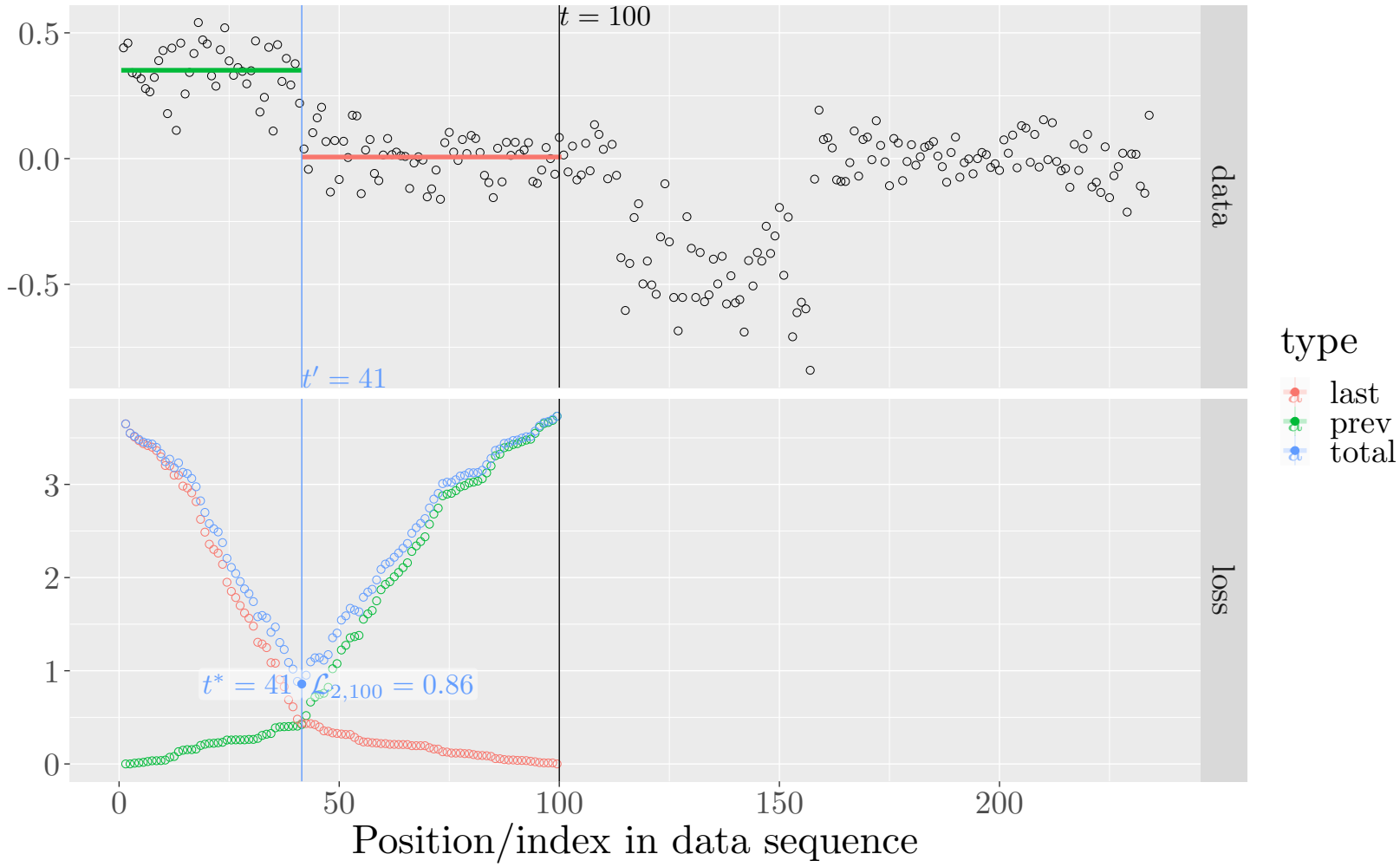
First iteration/changepoint of dynamic programming

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



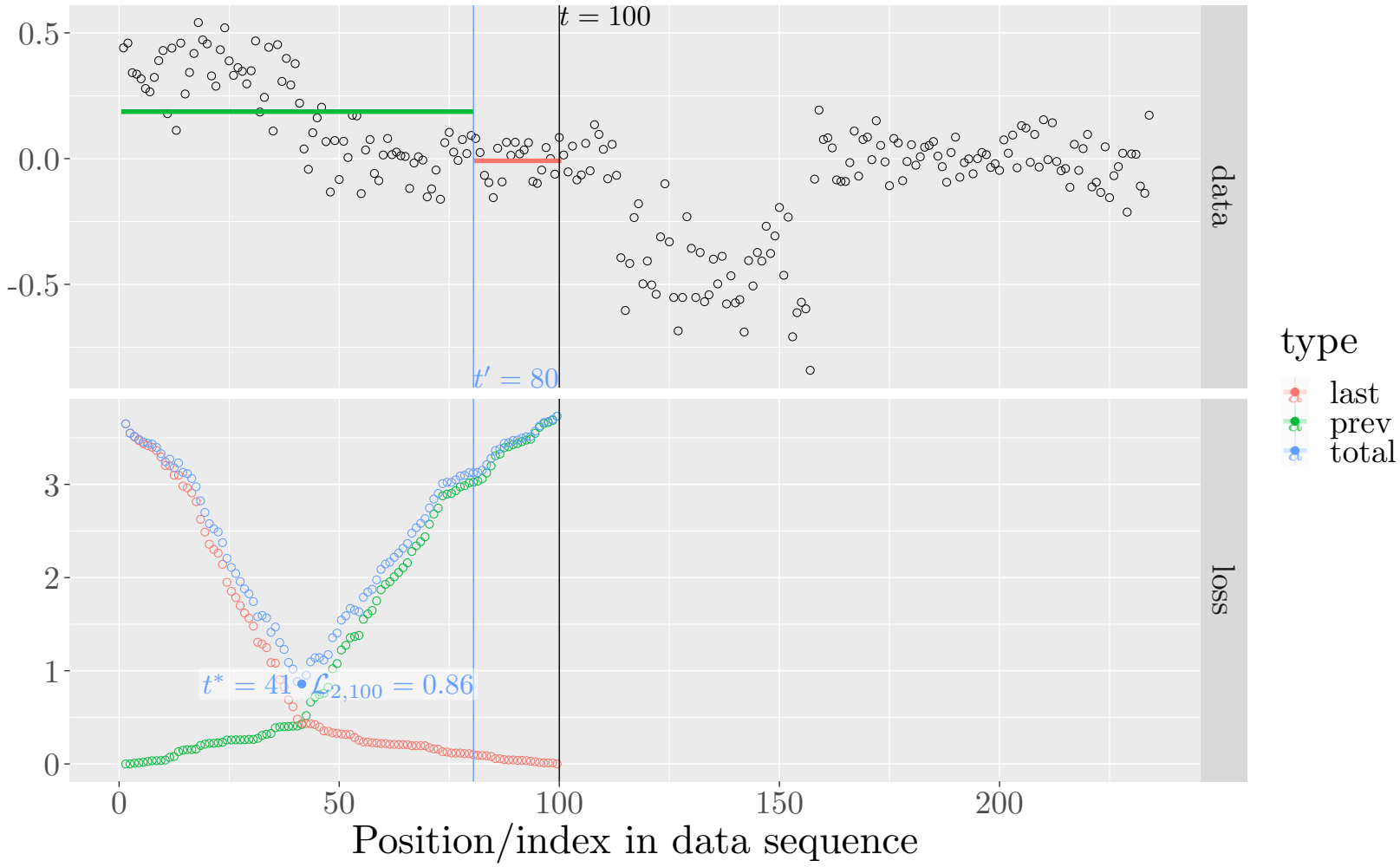
First iteration/changepoint of dynamic programming

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



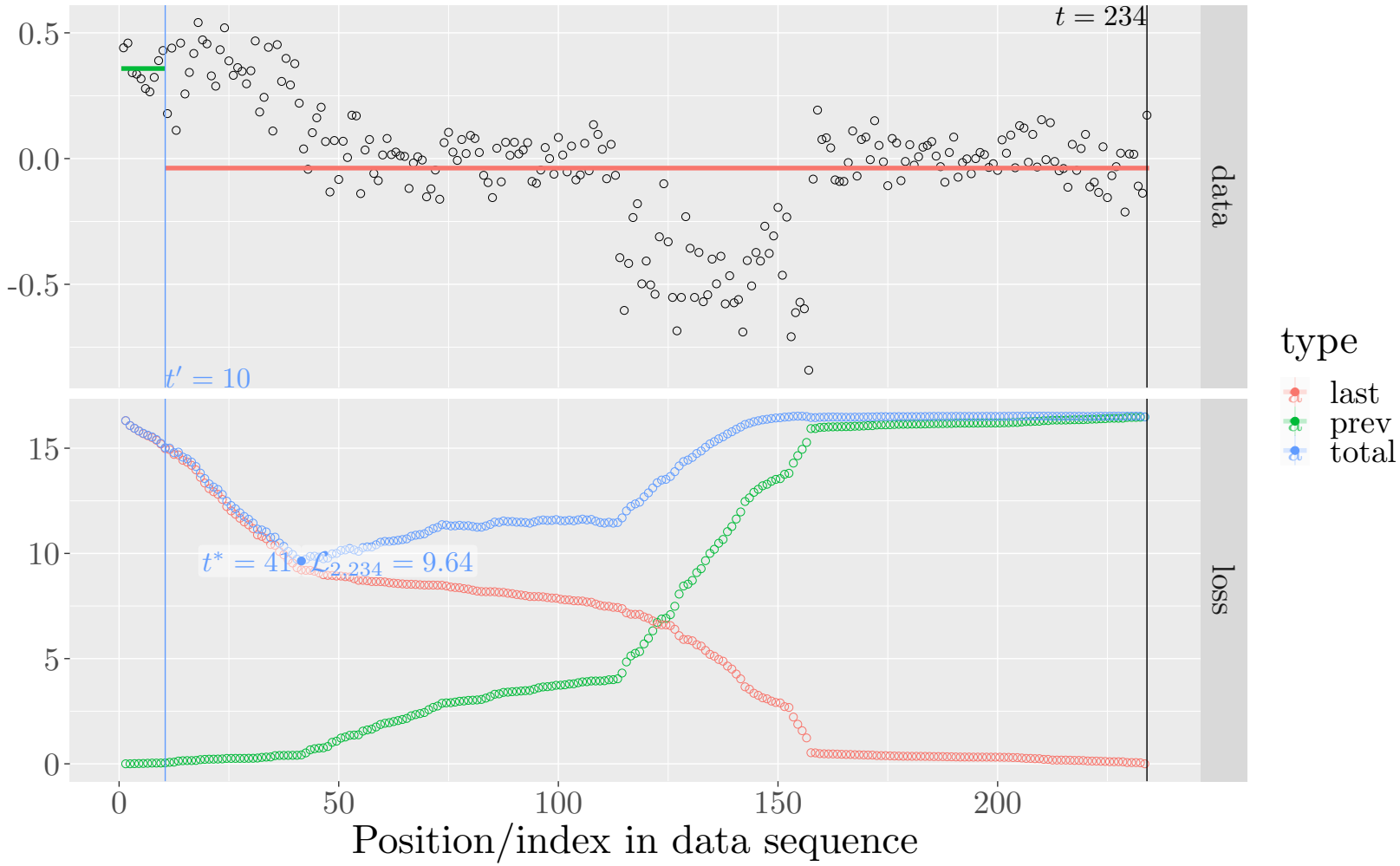
First iteration/changepoint of dynamic programming

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



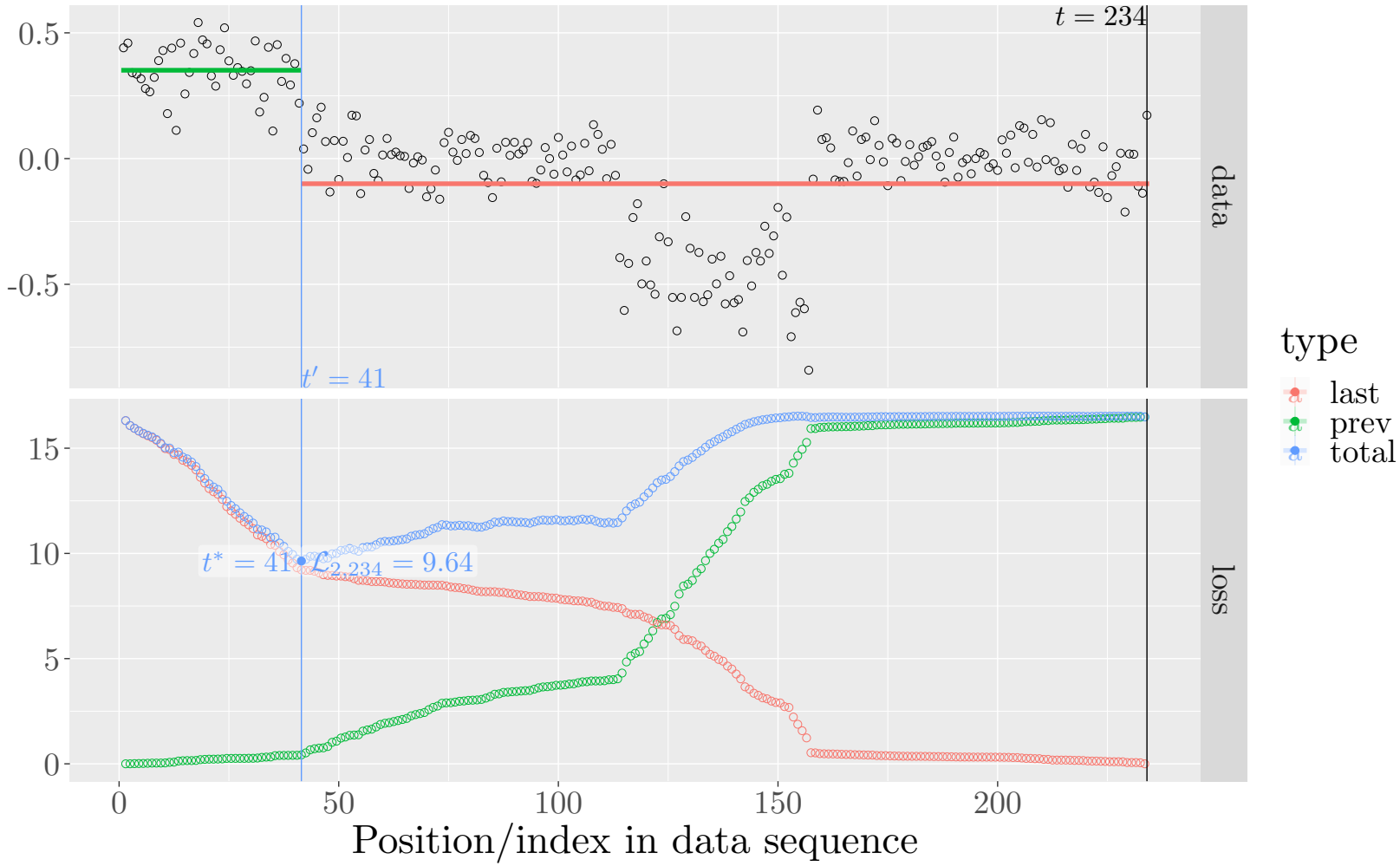
First iteration/changepoint of dynamic programming

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



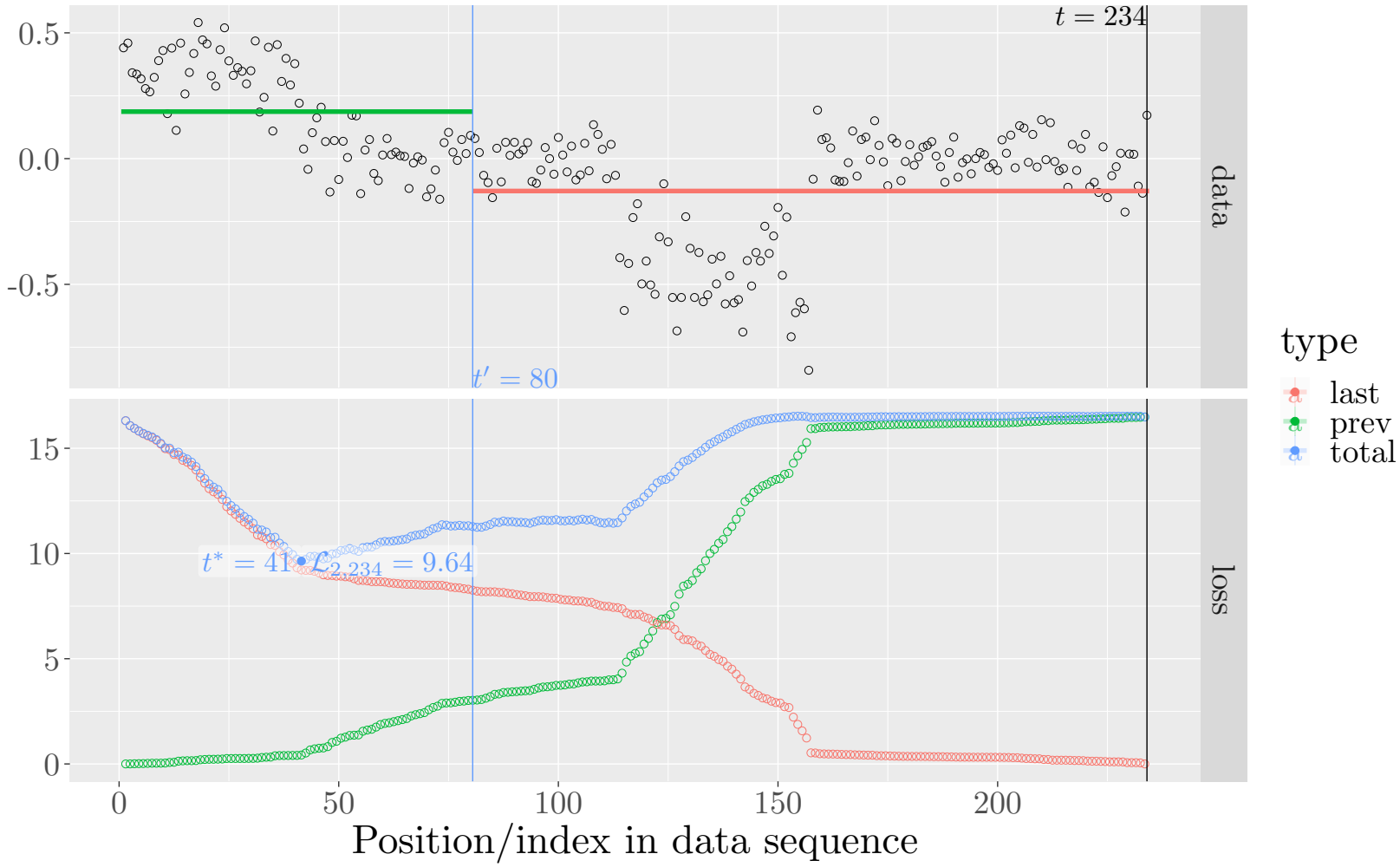
First iteration/changepoint of dynamic programming

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



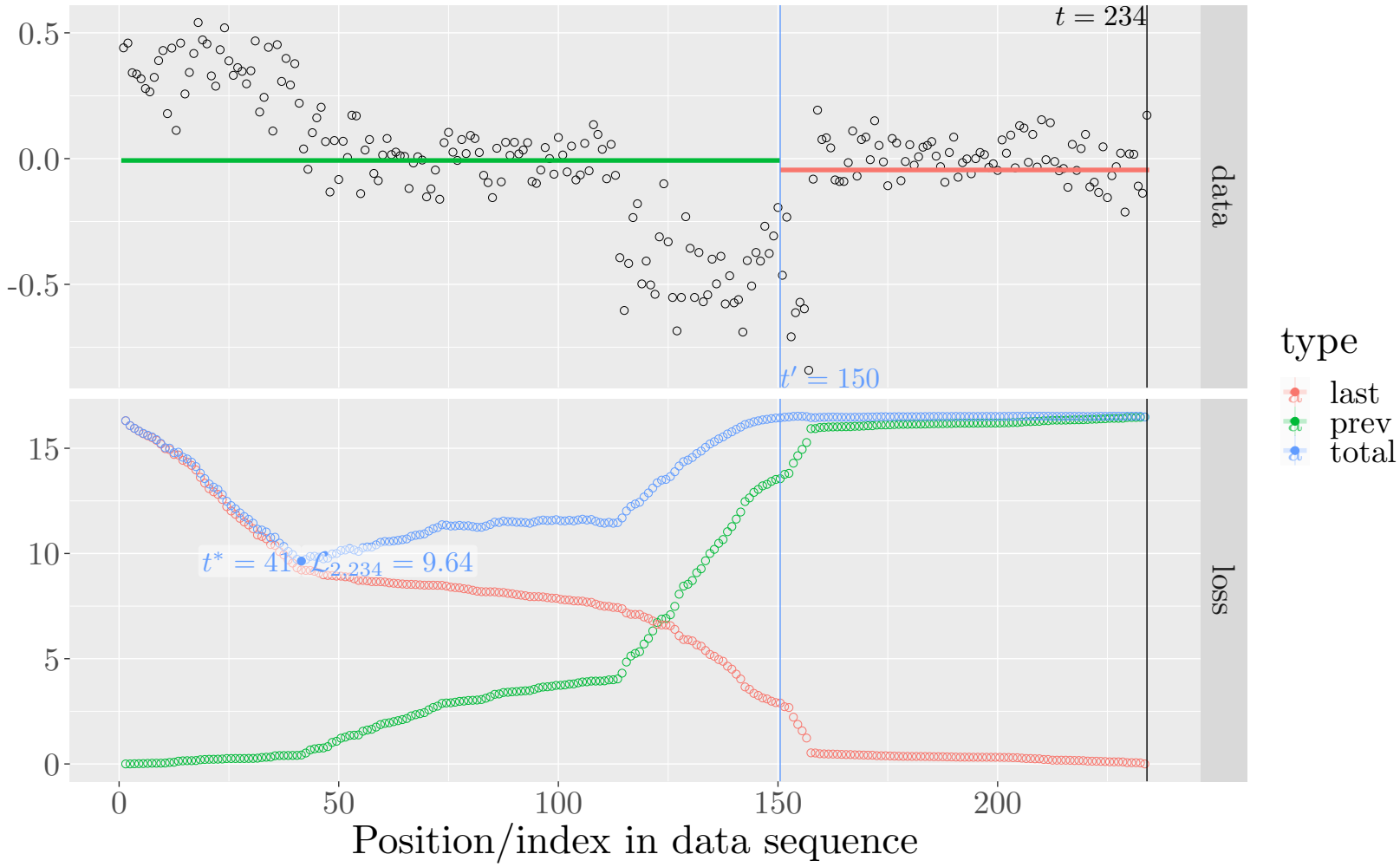
First iteration/changepoint of dynamic programming

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



First iteration/changepoint of dynamic programming

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



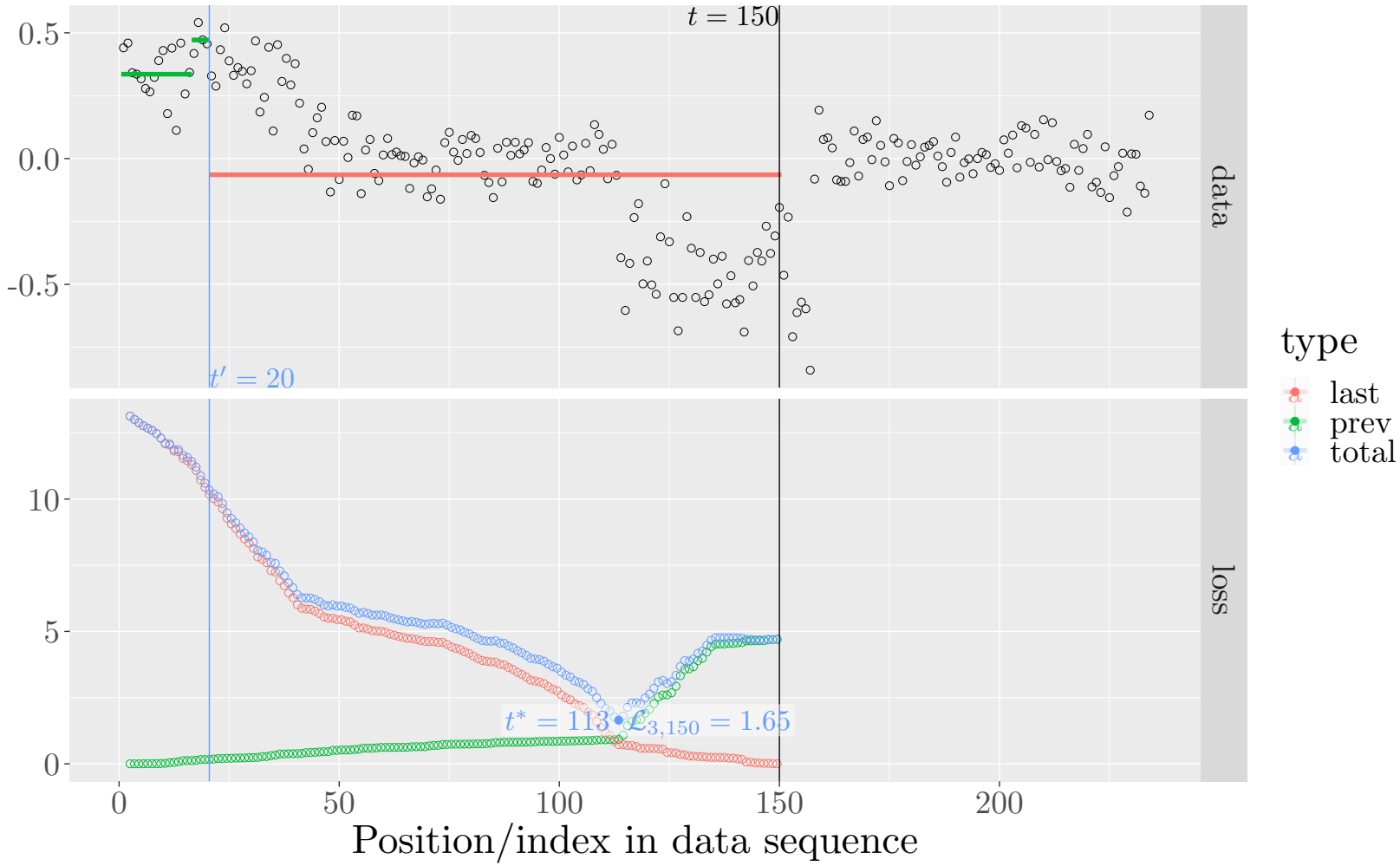
Computational complexity analysis

segments	grid search	dynamic programming	binary segmentation
1	$O(n)$	$O(n)$	$O(n)$
2	$O(n^2)$	$O(n^2)$	$O(\log n)$–$O(n)$
3	$O(n^3)$	$O(n^2)$	$O(\log n)$–$O(n)$
4	$O(n^4)$	$O(n^2)$	$O(\log n)$–$O(n)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

For example with $n = 234$ we have $\log n = 5.4553211$,
$n^2 = 5.4756 \times 10^4$ and $n^3 = 1.2812904 \times 10^7$.

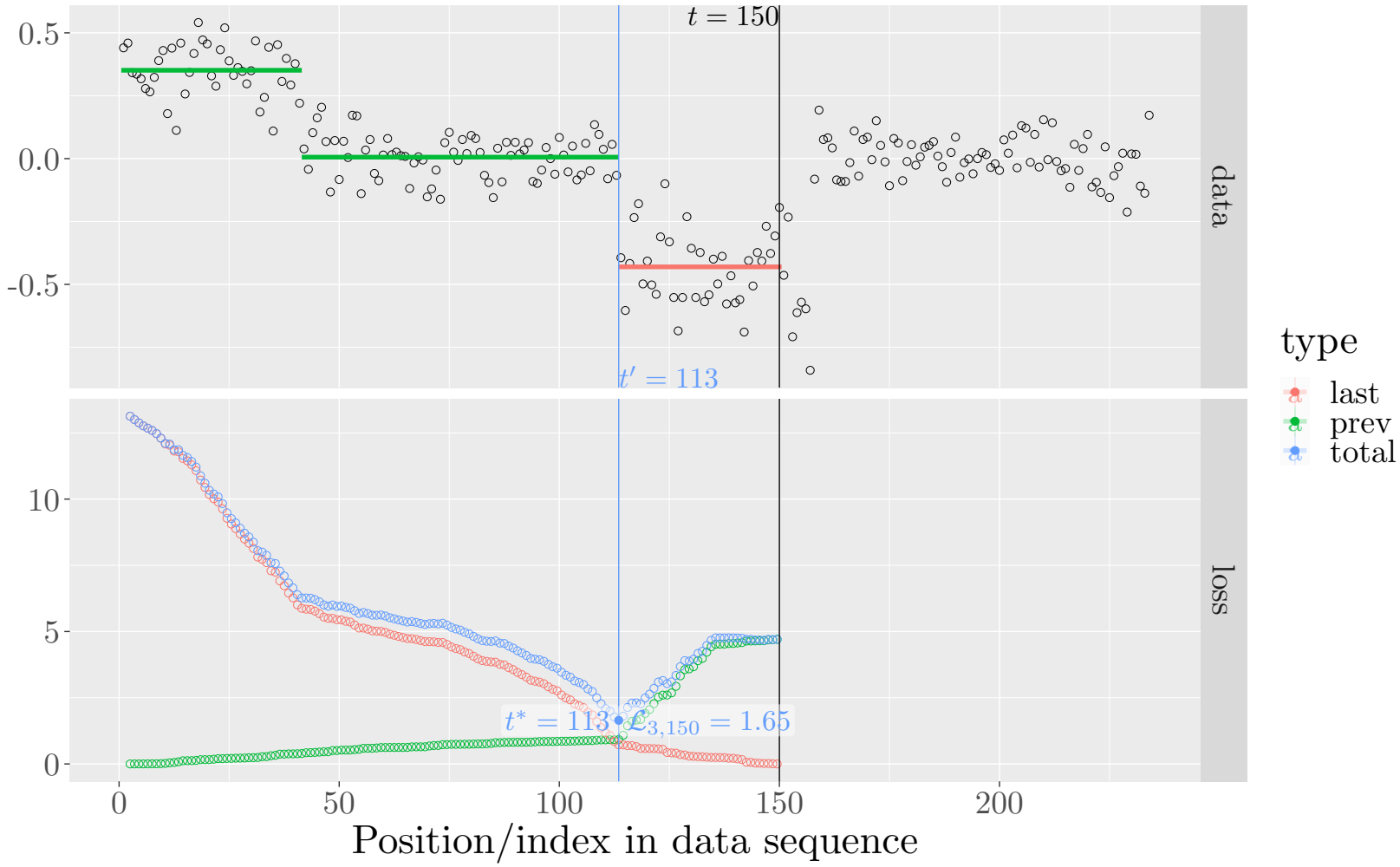
Second iteration/changepoint of dynamic programming

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



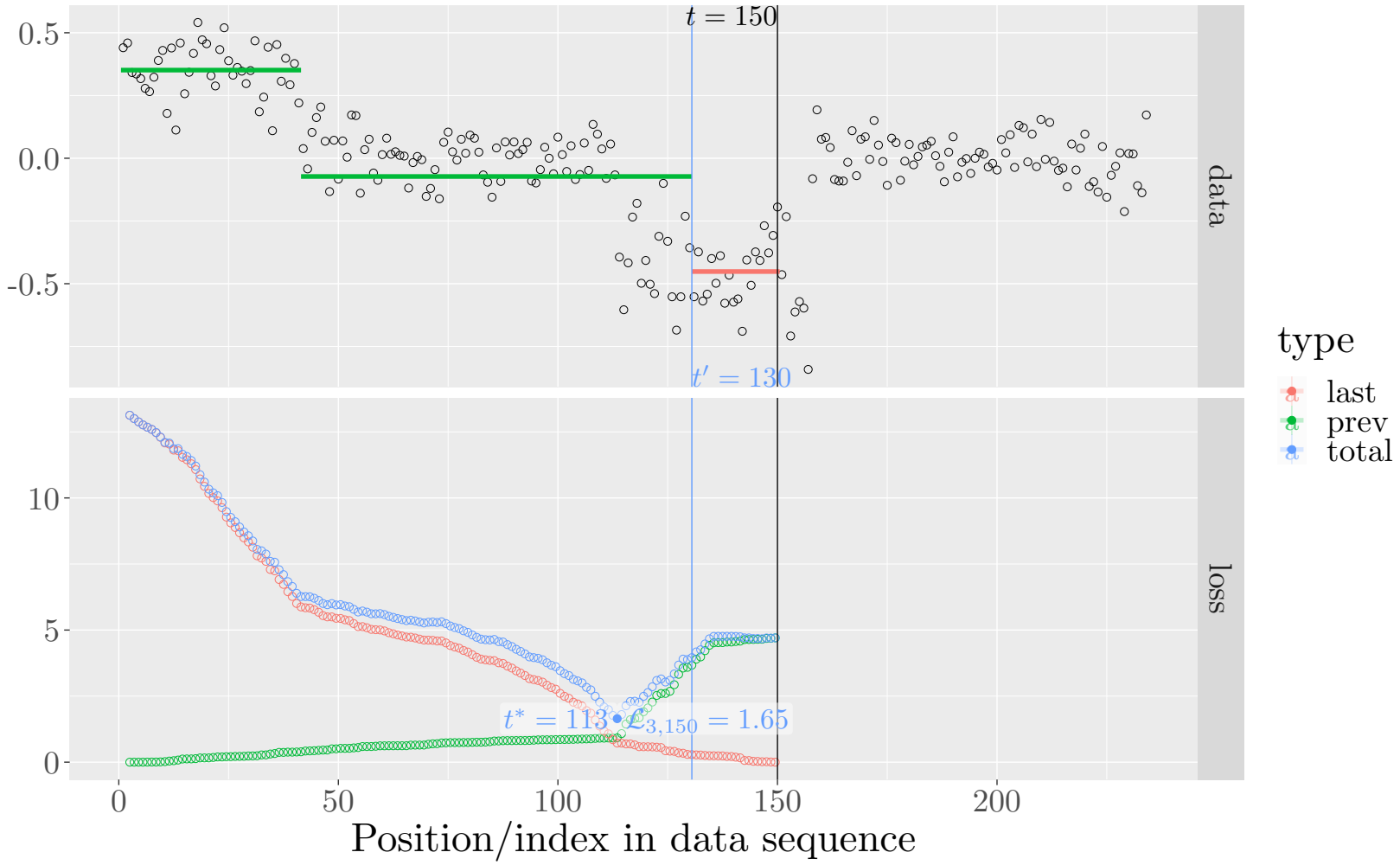
Second iteration/changepoint of dynamic programming

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



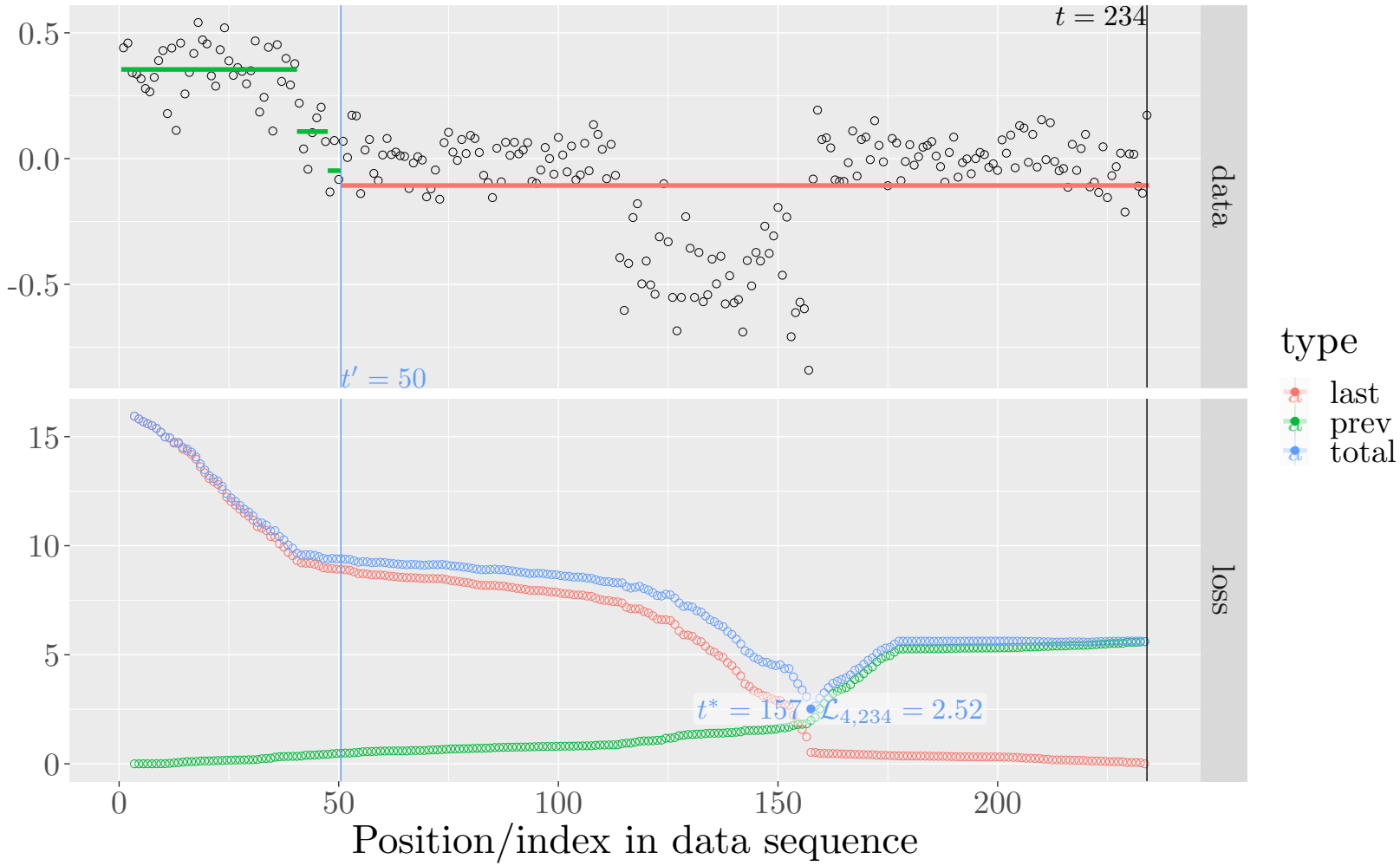
Second iteration/changepoint of dynamic programming

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



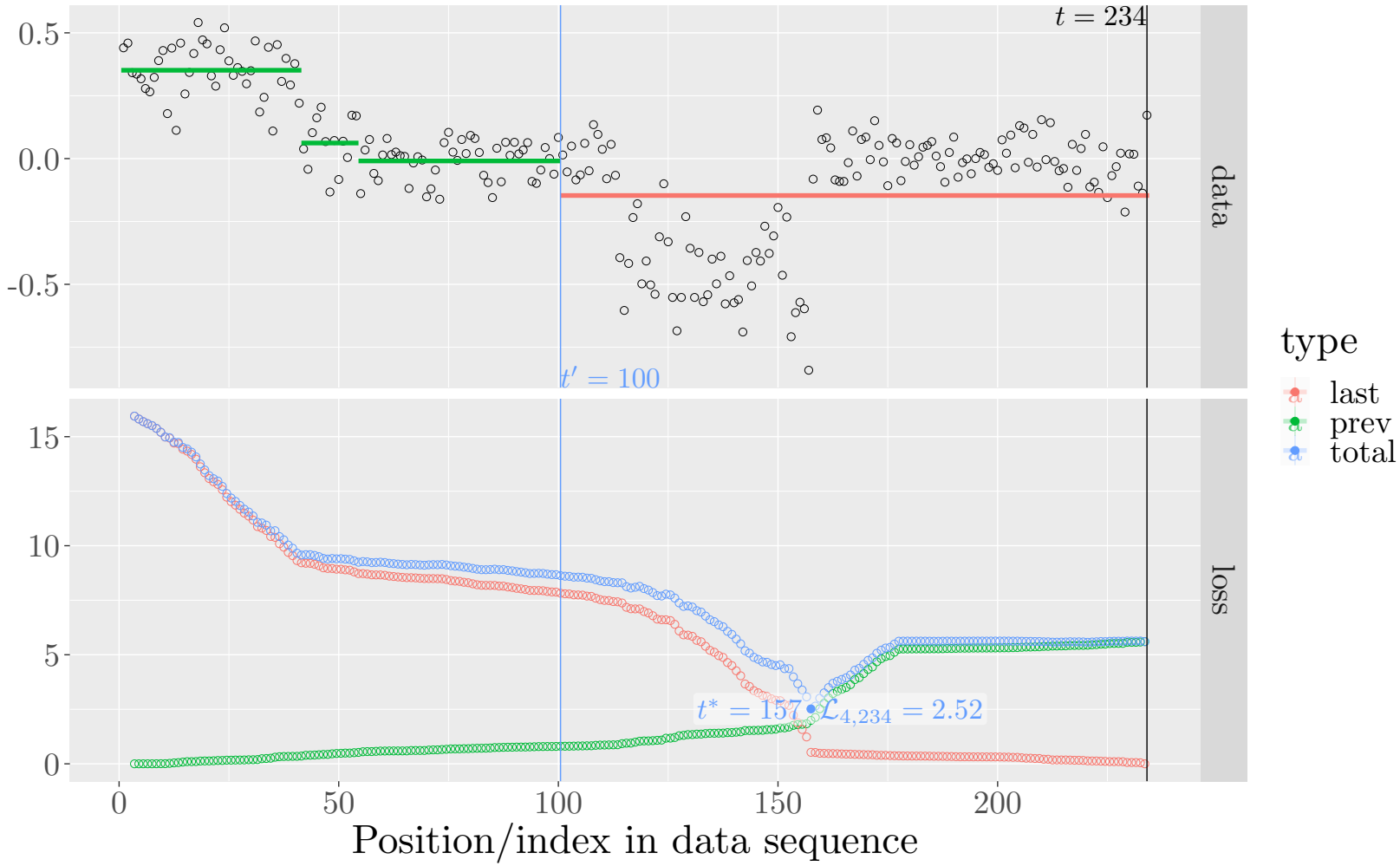
Third iteration/changepoint of dynamic programming

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



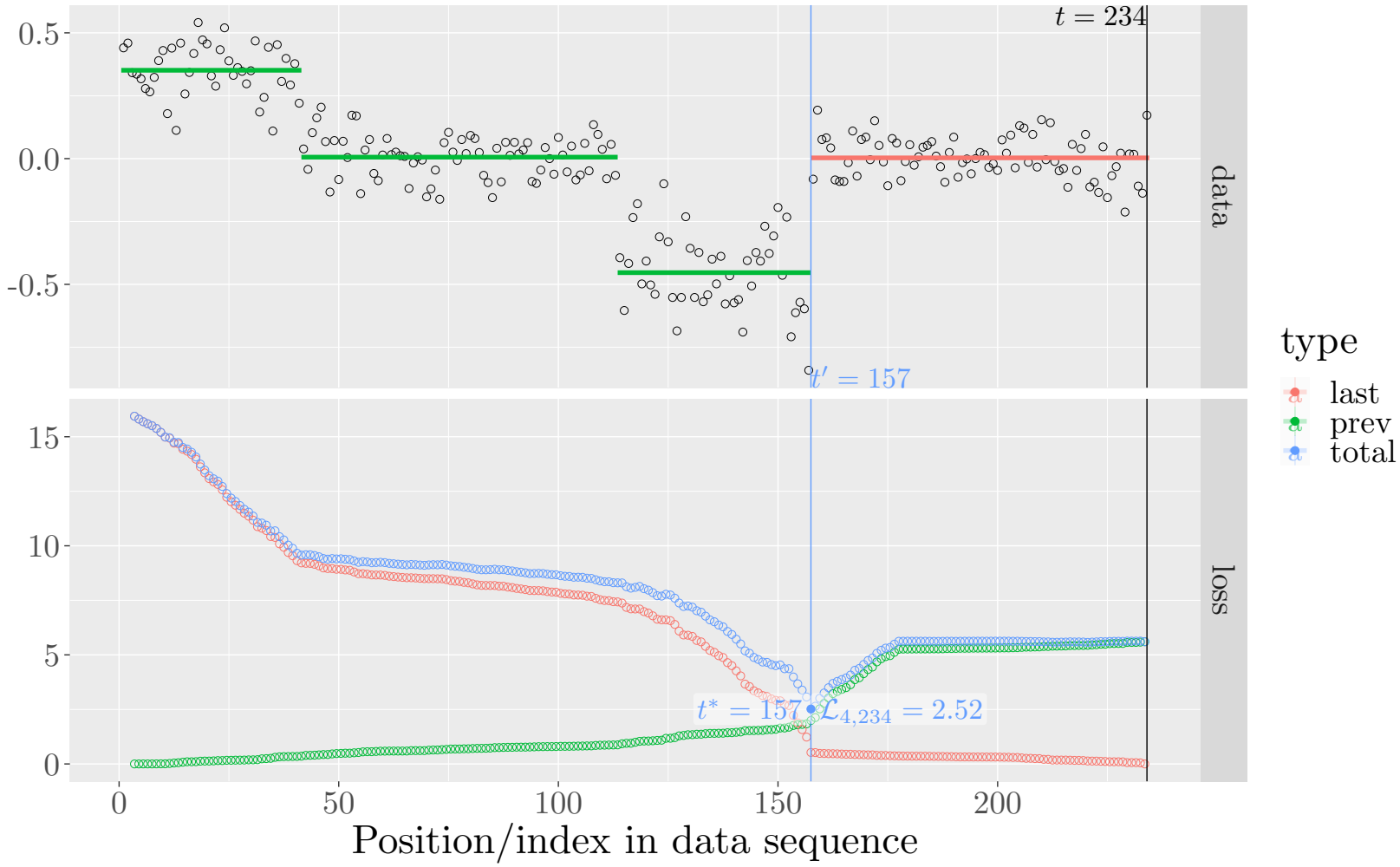
Third iteration/changepoint of dynamic programming

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



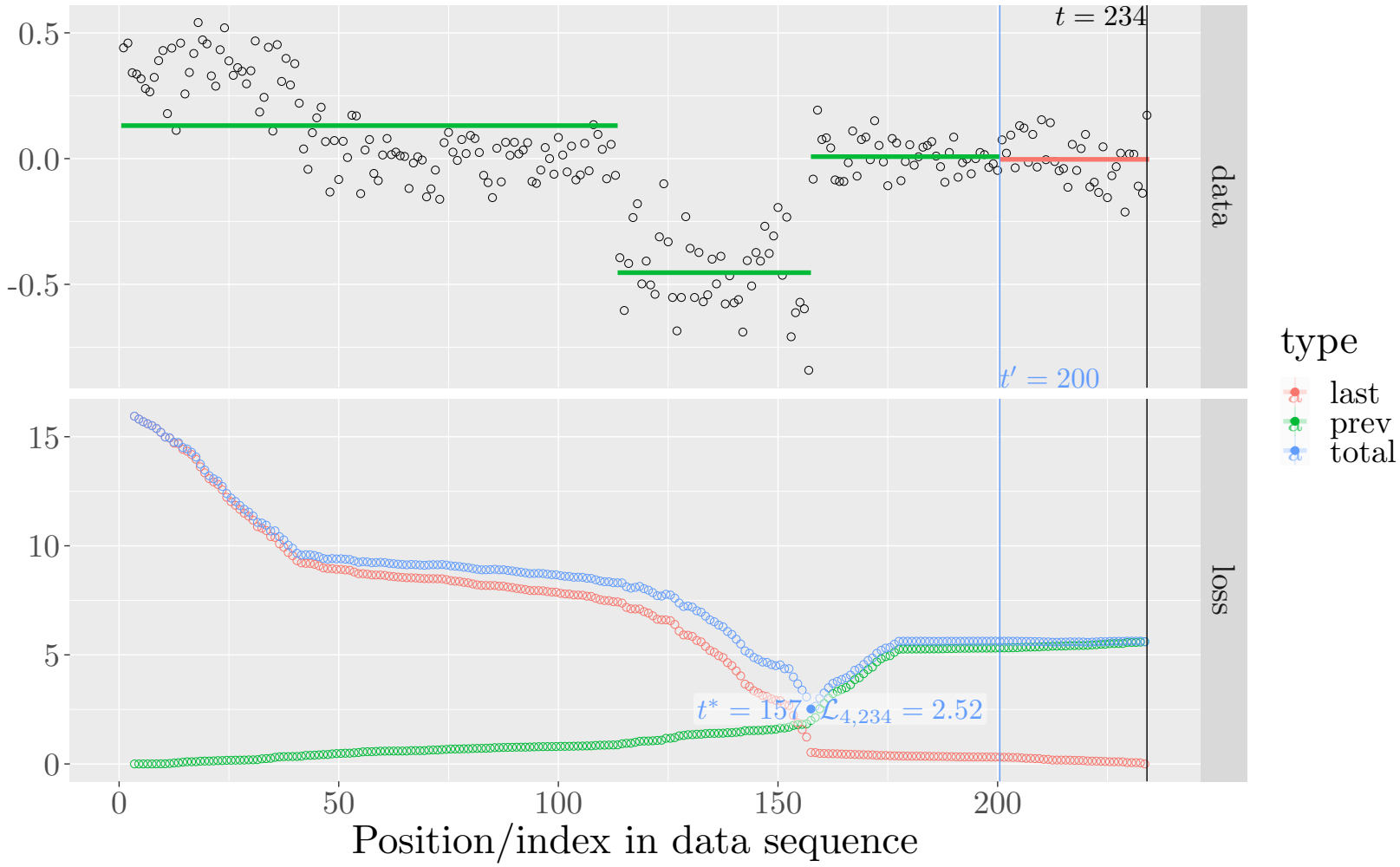
Third iteration/changepoint of dynamic programming

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



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 likelihoood, minimum squared error).

Dynamic programming is $O(n^2 S)$ 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)

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 $\{$binary segmentation, dynamic programming$\}$ an $\{$optimal, approximate$\}$ algorithm? Why?