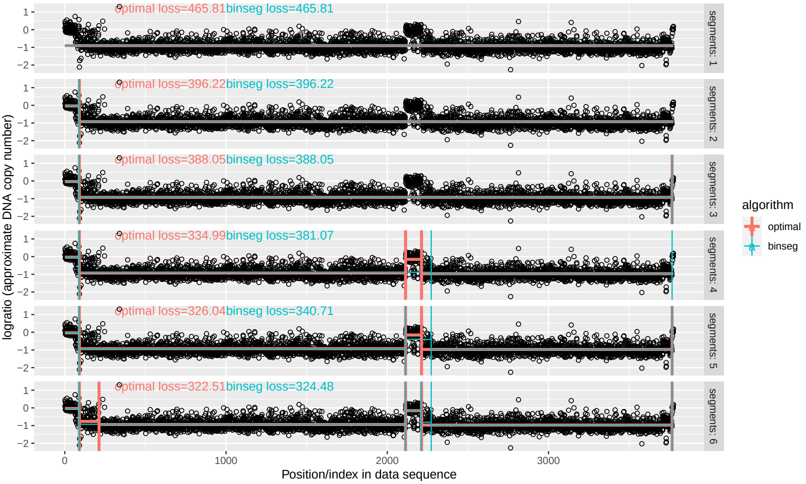
Optimal segmentation

Toby Dylan Hocking

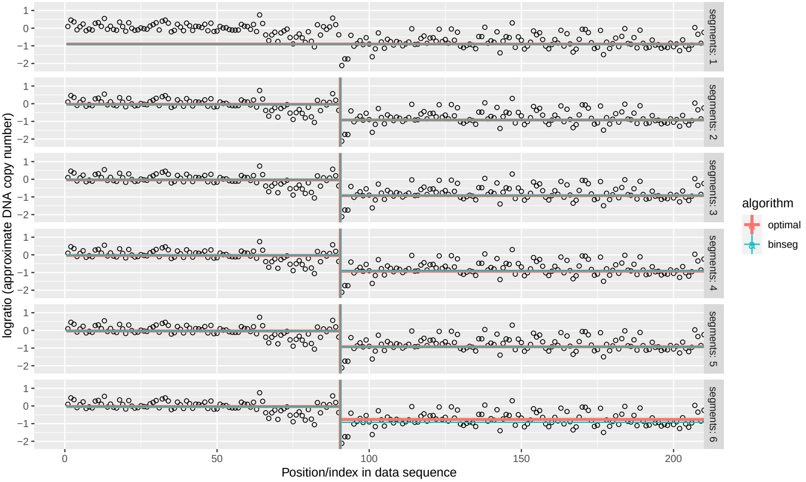
Motivation for optimal detection

- 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.
- Is it desirable to compute the optimal changepoints and segments? Yes, see examples in next slides.

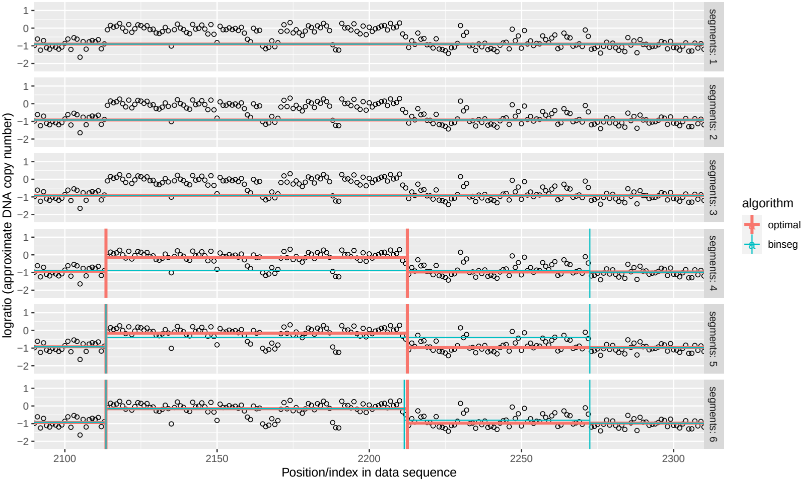
Example 1: stuck with sub-optimal change



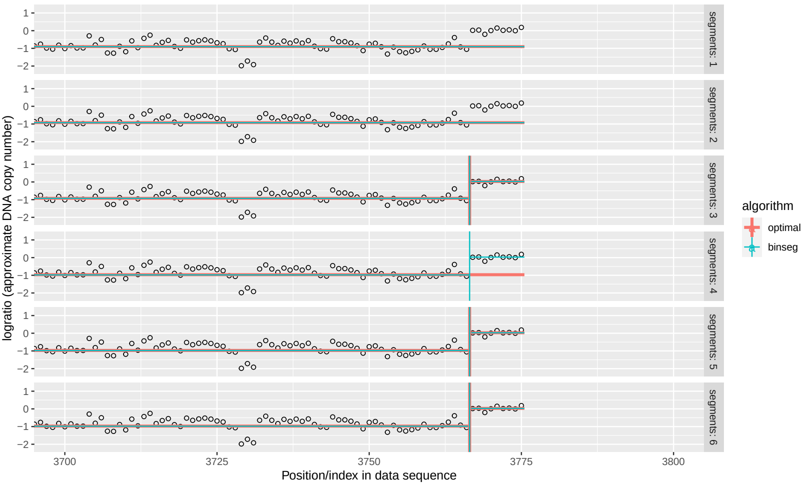
Zoom to start



Zoom to center

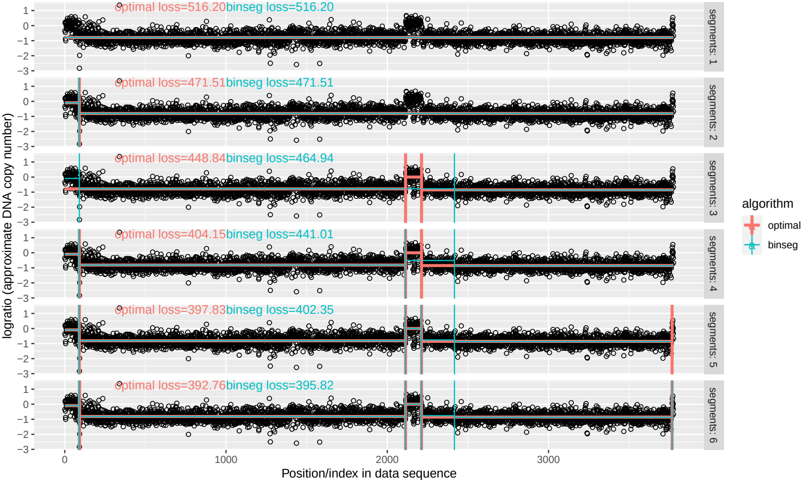


Zoom to end

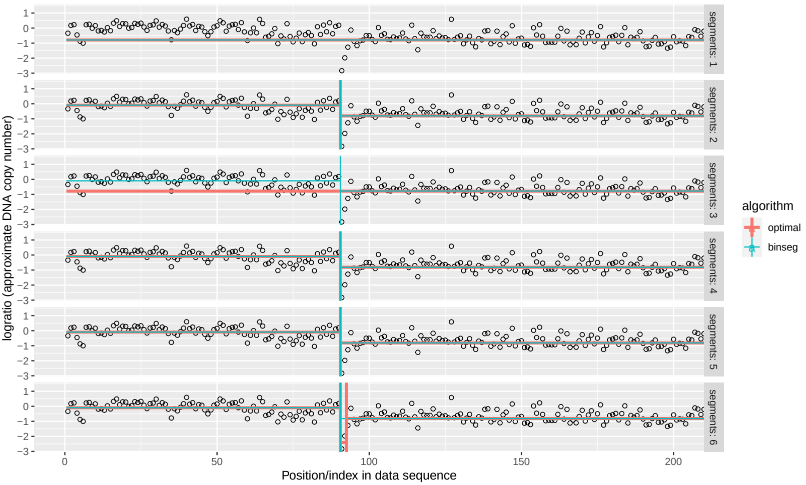


1
0
−1
−2
segments: 1
segments: 2
segments: 3
segments: 4
segments: 5
segments: 6
logratio (approximate DNA copy number)
3700
3725
3750
3775
3800
Position/index in data sequence
algorithm
optimal
binseg

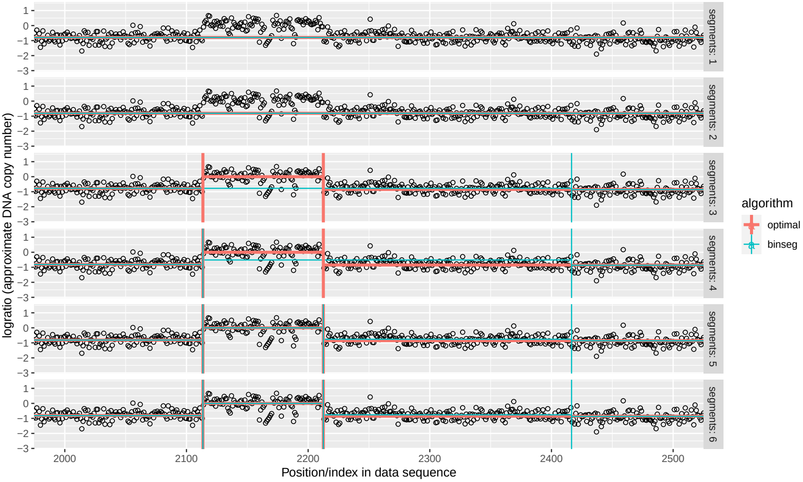
Example 2: missing a small change down



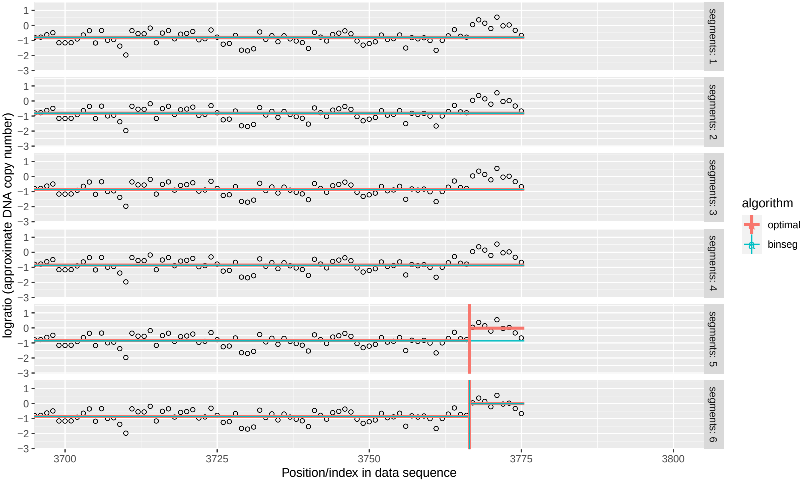
Zoom to start



Zoom to center

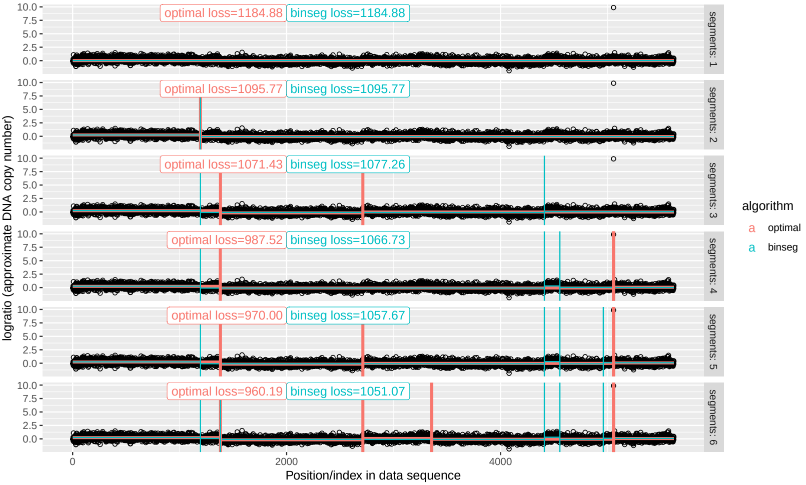


Zoom to end

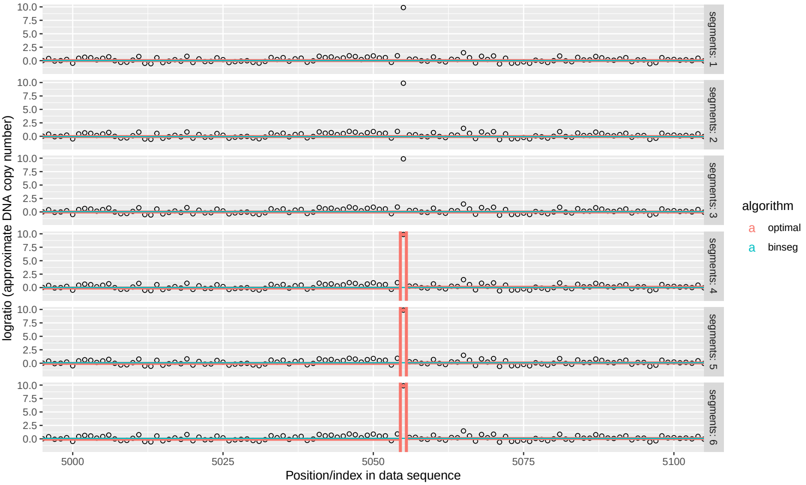


logratio (approximate DNA copy number)
Position/index in data sequence
3700
3725
3750
3775
3800
1
0
−1
−2
−3
segments: 1
segments: 2
segments: 3
segments: 4
segments: 5
segments: 6
algorithm
optimal
binseg

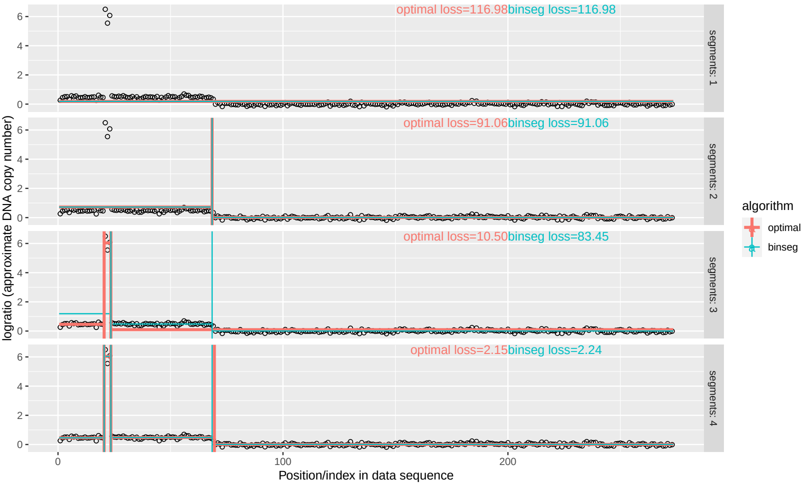
Example 3: difficult to detect an outlier



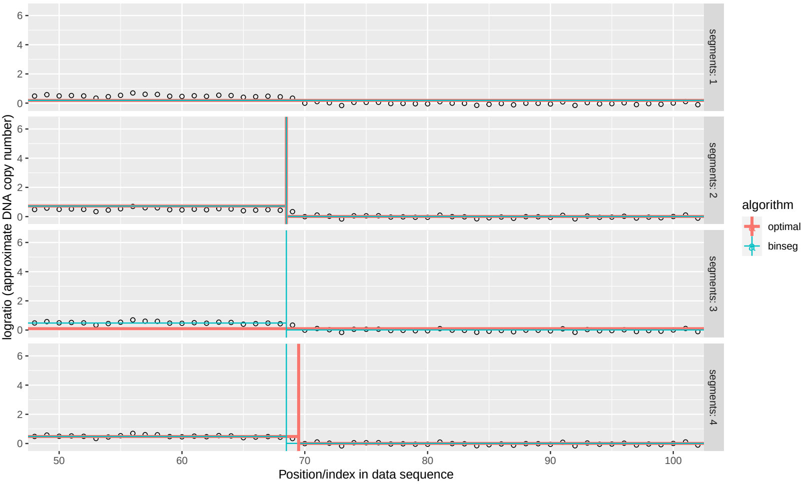
Zoom X axis to outlier



Example 4: stuck with sub-optimal change

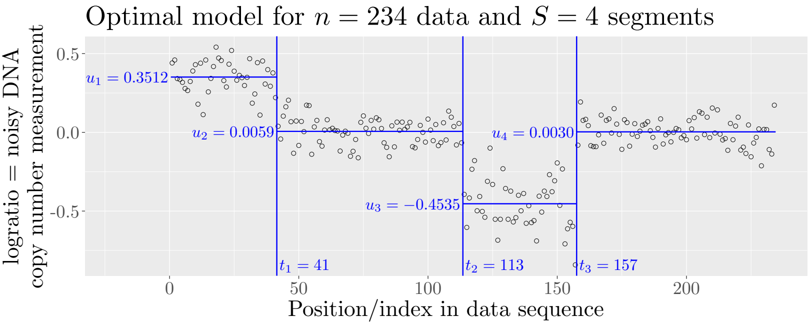


Zoom to changepoint



Dynamic programming for optimal changepoint detection

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



Maximum likelihood inference for S segments and n data

The best loss for S segments and n data is

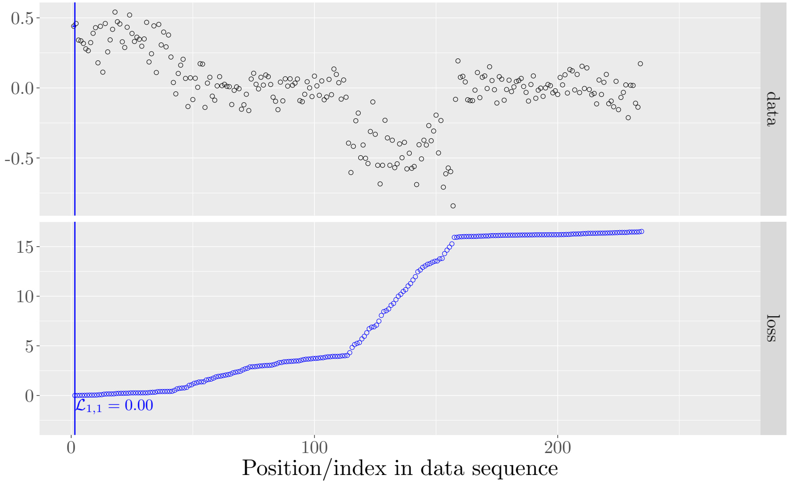
$$
\begin{aligned}
\mathcal{L}_{S,n} &= \min_{\substack{\mathbf{u}\in\mathbb{R}^S \\ 0=t_0<t_1<\cdots<t_{S-1}<t_S=n}} \sum_{s=1}^{S} \sum_{i=t_{s-1}+1}^{t_s} \ell(u_s, z_i) \\
&= \min_{t_{S-1}} \underbrace{\min_{\substack{u_1,\dots,u_{S-1} \\ t_1<\cdots<t_{S-2}}} \sum_{s=1}^{S-1} \sum_{i=t_{s-1}+1}^{t_s}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

- 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).

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

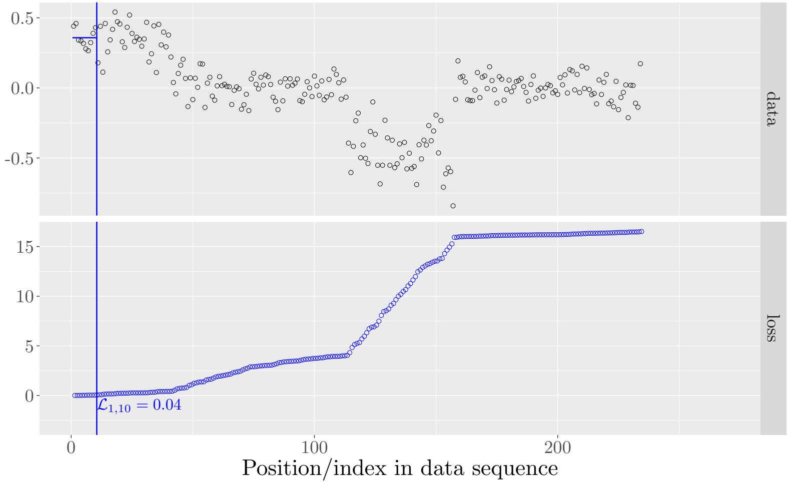
First step of dynamic programming

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



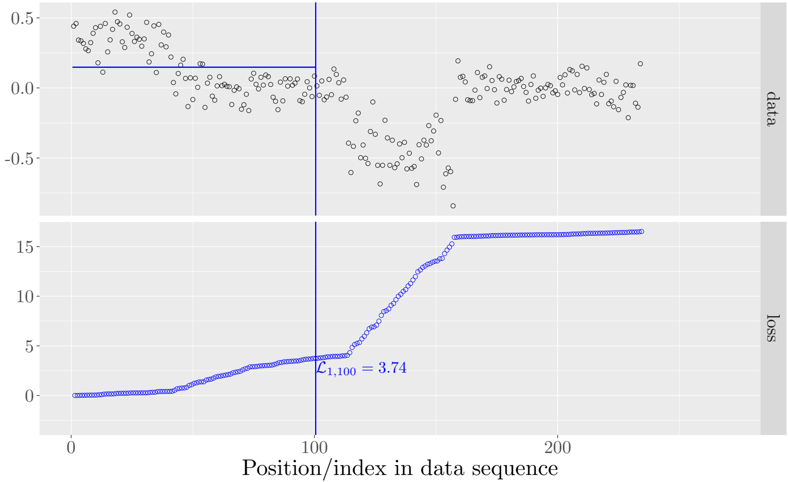
First step of dynamic programming

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



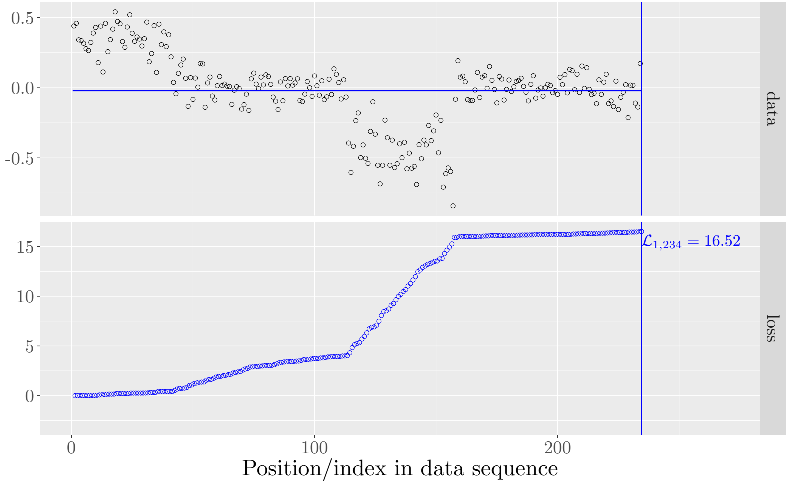
First step of dynamic programming

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



First step 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 can also be computed in linear time, $Q_t = \sum_{i=1}^{t} z_i^2$.

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

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