Unconstrained FPOP for Poisson Loss

GSoC 2026 — Medium Test

William Zhang

February 2026

github.com/williamzhang7792/gsoc2026-gfpop-william-zhang

Optimal Partitioning

Given count data $y_1, \ldots, y_n$ and penalty $\beta > 0$, find the segmentation minimizing

$$\sum_{k=1}^{K} \sum_{i \in S_k} \big(\mu_k - y_i \log \mu_k\big) \; + \; \beta \, (K - 1)$$

Each μ_k is set to the segment MLE $\bar{y}_{S_k}$.

(The constant $\log(y_i!)$ term is omitted—it does not depend on μ and cancels in optimization.)

Optimal Partitioning

Given count data $y_1, \ldots, y_n$ and penalty $\beta > 0$, find the segmentation minimizing

$$\sum_{k=1}^{K} \sum_{i \in S_k} \big(\mu_k - y_i \log \mu_k\big) \; + \; \beta\,(K-1)$$

Each μ_k is set to the segment MLE $\bar{y}_{S_k}$.

(The constant $\log(y_i!)$ term is omitted—it does not depend on μ and cancels in optimization.)

Segment neighborhood (`Segmentor3IsBack`): best model for every $K = 1, \ldots, K_{\max}$.

Optimal partitioning: best model for a single β, without computing all K.

From $O(n^2)$ to $O(n)$: The FPOP Idea

Standard DP: let $F_t =$ min penalized cost over $y_1, \dots, y_t$.

$$F_t = \min_{0 \le \tau < t} \left\{ F_\tau + C(y_{\tau+1:t}) + \beta \right\} \qquad \Longrightarrow \qquad O(n^2)$$

From $O(n^2)$ to $O(n)$: The FPOP Idea

Standard DP: let $F_t =$ min penalized cost over $y_1, \ldots, y_t$.

$$F_t = \min_{0 \le \tau < t} \{F_\tau + C(y_{\tau+1:t}) + \beta\} \qquad \Longrightarrow \qquad O(n^2)$$

FPOP (Maidstone et al. 2016): keep cost as a *function* of the segment mean.

$$Q_t(x) = \min\Big\{ \underbrace{Q_{t-1}(x)}_{\text{continue}},\ \underbrace{\min_{x'} Q_{t-1}(x') + \beta}_{\text{switch}} \Big\} \ +\ \ell(y_t, x)$$

where $x = \log \mu$ and $\ell(y, x) = e^x - yx$.

From $O(n^2)$ to $O(n)$: The FPOP Idea

Standard DP: let $F_t =$ min penalized cost over $y_1, \ldots, y_t$.

$$F_t = \min_{0 \le \tau < t} \{ F_\tau + C(y_{\tau+1:t}) + \beta \} \qquad \Longrightarrow \qquad O(n^2)$$

FPOP (Maidstone et al. 2016): keep cost as a *function* of the segment mean.

$$Q_t(x) = \min \Big\{ \underbrace{Q_{t-1}(x)}_{\text{continue}}, \ \underbrace{\min_{x'} Q_{t-1}(x') + \beta}_{\text{switch}} \Big\} \ + \ \ell(y_t, x)$$

where $x = \log \mu$ and $\ell(y, x) = e^x - yx$.

Notice that Q_t is **piecewise** — each piece inherits the $ae^x + bx + c$ structure from the Poisson loss. Candidates that fall entirely above the switch line get pruned $\Rightarrow$ empirically $O(n)$.

Three Operations Per Step

1. **Flat line** at $K_{t-1} + \beta$, where $K_{t-1} = \min_x Q_{t-1}(x)$
   `set_to_unconstrained_min_of`
2. **Min-envelope** of continue curve and flat switch line `set_to_min_env_of`
3. **Add loss** $\ell(y_t, x) = e^x - y_t\, x$

Three Operations Per Step

1. **Flat line** at $K_{t-1} + \beta$, where $K_{t-1} = \min_x Q_{t-1}(x)$
   `set_to_unconstrained_min_of`
2. **Min-envelope** of continue curve and flat switch line `set_to_min_env_of`
3. **Add loss** $\ell(y_t, x) = e^x - y_t\, x$

Each Q_t is stored as a linked list of pieces.

Root-finding (Newton's method) determines where pieces cross.

Backtracking recovers the optimal segmentation from Q_n.

Constrained $\to$ Unconstrained

`PeakSegOptimal`: 2 states (up/down), $N \times 2$ cost array.

- `set_to_min_less_of`: enforces $\mu_k \leq \mu_{k+1}$
- `set_to_min_more_of`: enforces $\mu_k \geq \mu_{k+1}$

Constrained $\rightarrow$ Unconstrained

`PeakSegOptimal`: 2 states (up/down), $N \times 2$ cost array.

- `set_to_min_less_of`: enforces $\mu_k \leq \mu_{k+1}$
- `set_to_min_more_of`: enforces $\mu_k \geq \mu_{k+1}$

My change: **one** state, $N \times 1$ cost array.

- `set_to_unconstrained_min_of`: call `Minimize()`, emit a single constant piece

Constrained $\rightarrow$ Unconstrained

`PeakSegOptimal`: 2 states (up/down), $N \times 2$ cost array.

- `set_to_min_less_of`: enforces $\mu_k \leq \mu_{k+1}$
- `set_to_min_more_of`: enforces $\mu_k \geq \mu_{k+1}$

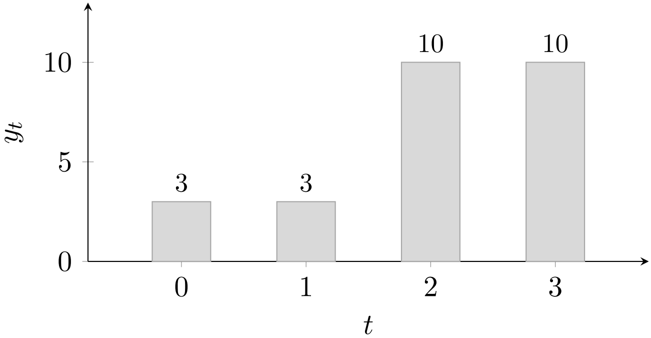
My change: **one** state, $N \times 1$ cost array.

- `set_to_unconstrained_min_of`: call `Minimize()`, emit a single constant piece

Everything else (min-envelope, root-finding, backtracking) stays unchanged.

Worked Example: The Data

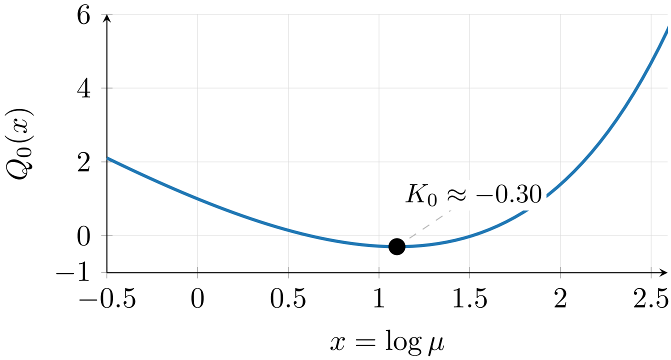
$$\mathbf{y} = [3,\ 3,\ 10,\ 10], \qquad \beta = 2.$$



Expected result: two segments $[3, 3 \mid 10, 10]$ with means $\hat{\mu}_1 = 3$, $\hat{\mu}_2 = 10$.

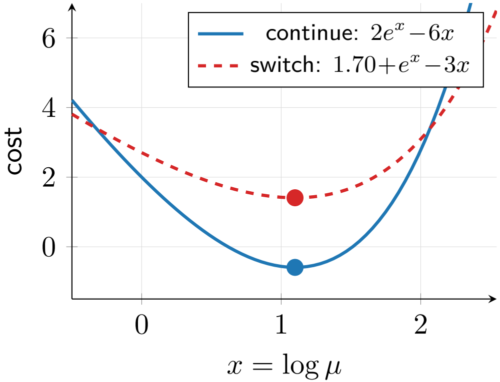
$t = 0$: Initial Cost ($y_0 = 3$)

$$Q_0(x) = e^x - 3x$$



Single piece. Minimum $K_0 \approx -0.30$ at $x = \log 3 \approx 1.10$.

$t = 1$: Continue vs. Switch ($y_1 = 3$)



Continue $[3, 3]$:
$\min \approx -0.59$ at $\mu = 3$

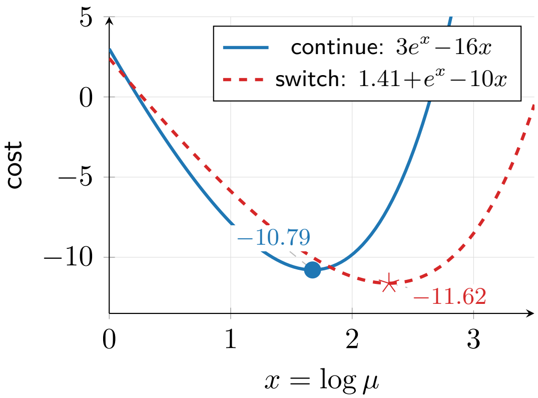
Switch (new seg $[3]$):
$\min \approx 1.41$ at $\mu = 3$

Continue wins by ≈ 2.

Data is consistent—no reason to split.

$K_1 \approx -0.59$

$t = 2$: The Jump ($y_2 = 10$)



Continue $[3, 3, 10]$:
$\min \approx -10.79$ at $\mu \approx 5.3$

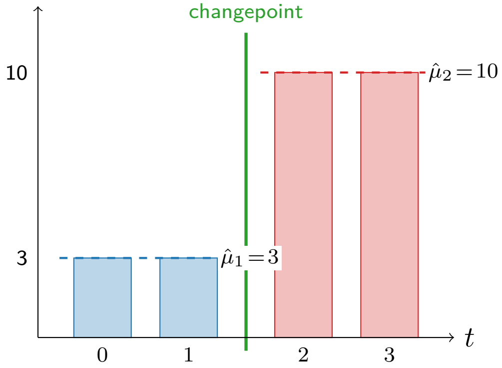
Switch $[3, 3 \mid 10]$:
$\min \approx -11.62$ at $\mu = 10$ $\star$

Switch wins.

$y_2 = 10$ deviates far from $\bar{y}_{0:1} = 3$: paying $\beta = 2$ to start fresh is worth it.

Curves cross at $x \approx 1.82$.

Result



Backtracking:

1. $t = 3$: best mean $\mu = 10$, prev seg end $= 1$
2. $t = 1$: best mean $\mu = 3$, prev seg end $= -1$ (start)

Two segments:

$S_1 = \{y_0, y_1\}$, $\hat{\mu}_1 = 3$
$S_2 = \{y_2, y_3\}$, $\hat{\mu}_2 = 10$

Penalized cost ≈ -24.64.

Why Pruning is Exact

At step t, candidate τ is pruned if

$$Q_t(x, \tau) \;\geq\; K_t + \beta \quad \text{for all } x$$

i.e., continuing from τ is worse than starting fresh, for *every* possible mean.

Why Pruning is Exact

At step t, candidate τ is pruned if

$$Q_t(x, \tau) \ \geq \ K_t + \beta \quad \text{for all } x$$

i.e., continuing from τ is worse than starting fresh, for *every* possible mean.

Claim: a pruned candidate can never become optimal again.

Why Pruning is Exact

At step t, candidate τ is pruned if

$$Q_t(x,\tau) \;\geq\; K_t + \beta \quad \text{for all } x$$

i.e., continuing from τ is worse than starting fresh, for *every* possible mean.

Claim: a pruned candidate can never become optimal again.

1. At time t: candidate τ is replaced by $\tau' = t$, so $Q_t(x,\tau) > Q_t(x,\tau')$ for all x.
2. Future data adds the same loss to both:

$$Q_{t+s}(x,\tau) = Q_t(x,\tau) + L(x) \;>\; Q_t(x,\tau') + L(x) = Q_{t+s}(x,\tau')$$

3. Even if a future K_{t+s} is huge, τ still loses to τ' (which is alive).

Why Pruning is Exact

At step t, candidate τ is pruned if

$$Q_t(x,\tau) \;\geq\; K_t + \beta \quad \text{for all } x$$

i.e., continuing from τ is worse than starting fresh, for *every* possible mean.

Claim: a pruned candidate can never become optimal again.

1. At time t: candidate τ is replaced by $\tau' = t$, so $Q_t(x,\tau) > Q_t(x,\tau')$ for all x.
2. Future data adds the same loss to both:

$$Q_{t+s}(x,\tau) = Q_t(x,\tau) + L(x) \;>\; Q_t(x,\tau') + L(x) = Q_{t+s}(x,\tau')$$

3. Even if a future K_{t+s} is huge, τ still loses to τ' (which is alive).

$\Rightarrow$ Pruned $=$ dead forever. FPOP is **exact**.

Pruning in Practice: Why $O(n)$

Consider a flat signal ($y_t = c$ for all t):

- The original candidate $\tau = 0$ is the global optimum $\to$ never pruned.
- Each new "switch" candidate $\tau = t$ pays penalty β but gains no advantage on flat data.
- As flat data accumulates, the false start's cost rises above $K + \beta$.
- It gets pruned $\to$ active candidate list stays small (often 2–3 pieces).

Pruning in Practice: Why $O(n)$

Consider a flat signal ($y_t = c$ for all t):

- The original candidate $\tau = 0$ is the global optimum $\rightarrow$ never pruned.
- Each new "switch" candidate $\tau = t$ pays penalty β but gains no advantage on flat data.
- As flat data accumulates, the false start's cost rises above $K + \beta$.
- It gets pruned $\rightarrow$ active candidate list stays small (often 2–3 pieces).

In general: the list grows when the signal changes, and shrinks (via pruning) when it stabilizes.

Amortized over the whole sequence: empirically $O(n)$.

Seeing a Pruning Step

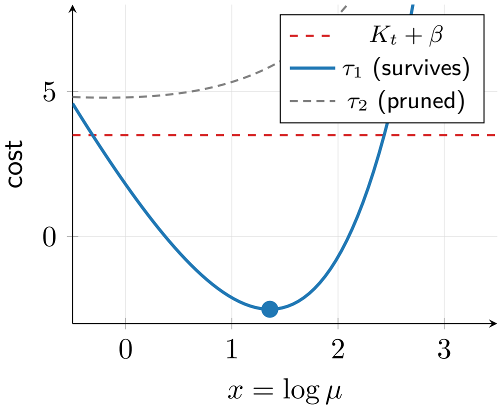
At each step the active set holds one cost curve per surviving candidate.

The flat line at $K_t + \beta$ is the **ceiling**: if a candidate's curve sits entirely above it, that candidate is dead—it can never win for any future μ.

Candidate τ_1 dips below $\to$ survives.

Candidate τ_2 is entirely above $\to$ **pruned**.

On flat signals, new candidates are quickly pruned $\to$ the active set stays small $\to$ empirical $O(n)$.



Implementation

Starting from `PeakSegOptimal`:

- **Reused**: `PoissonLossPieceLog`, `PiecewisePoissonLossLog`, `set_to_min_env_of`, `Minimize`, root-finding, `findMean`
- **New**: `set_to_unconstrained_min_of` (7 lines—just `Minimize` + emit flat piece)
- **Simplified**: 2-state DP loop → 1-state, $N \times 2 \to N \times 1$
- **Added**: Rcpp wrapper, input validation, backtracking

Implementation

Starting from `PeakSegOptimal`:

- **Reused**: `PoissonLossPieceLog`, `PiecewisePoissonLossLog`, `set_to_min_env_of`, `Minimize`, root-finding, `findMean`
- **New**: `set_to_unconstrained_min_of` (7 lines—just `Minimize` + emit flat piece)
- **Simplified**: 2-state DP loop → 1-state, $N \times 2 \rightarrow N \times 1$
- **Added**: Rcpp wrapper, input validation, backtracking

Self-contained: `PoissonFPOP.cpp` (∼760 lines, mostly reused infrastructure).

`sourceCpp("PoissonFPOP.cpp")` — no package install needed.

Validation

Compared against `Segmentor3IsBack::Segmentor(model=1)` using `testthat`.

8 test cases, 54 assertions:

3-segment data	penalties 5, 10, 50, 100
Single changepoint	penalties 10, 50, 200
Constant-rate data	penalty 500 (1 segment)
5-segment data	penalties 1, 5, 20, 100
Data with zeros	penalties 1, 5, 50
$\beta = 0$	every point its own segment
$n = 2$	1-segment and 2-segment outcomes
Bad input	negative data, all-identical data

All breakpoints and segment means match exactly.

Same results as Segmentor's $O(K_{\max}\, n^2)$ segment neighborhood, but in empirical $O(n)$.