Recent advances in fast algorithms for optimal changepoint detection

Toby Dylan Hocking
toby.hocking@nau.edu

April 30, 2022

Constrained Dynamic Programming and Supervised Penalty Learning Algorithms for Peak Detection in Genomic Data

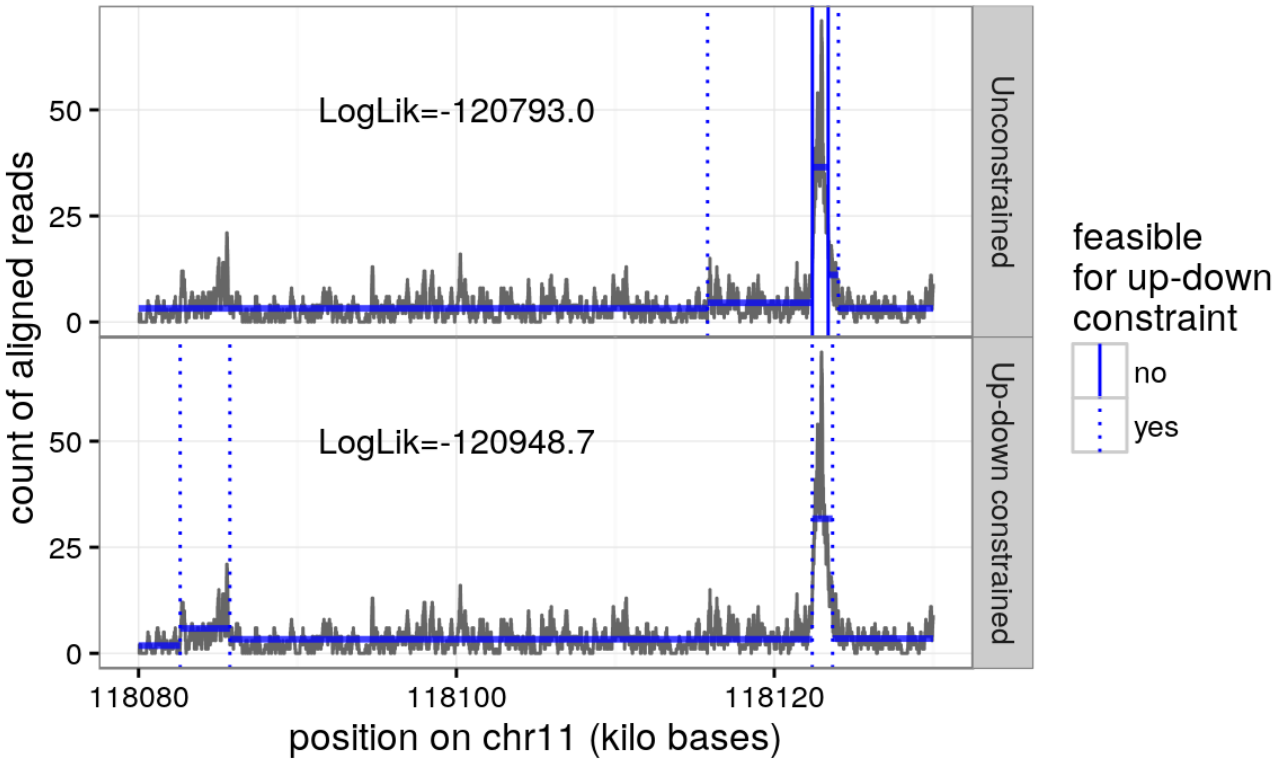
Linear Time Dynamic Programming for Computing Breakpoints in the Regularization Path of Models Selected From a Finite Set

Generalized Functional Pruning Optimal Partitioning (GFPOP) for Constrained Changepoint Detection in Genomic Data

Conclusions

Citation

Hocking TD, Rigaill G, Fearnhead P, Bourque G. Constrained Dynamic Programming and Supervised Penalty Learning Algorithms for Peak Detection in Genomic Data. Journal of Machine Learning Research 21(87):1-40, 2020.



New idea: an optimal algorithm for computing best changepoints which alternate up and down (R package PeakSegOptimal).

Comparison to previous work and novelty

Constraint	No pruning	Functional pruning
None	Dynamic Prog. Algo. (DPA) Optimal, $O(Sn^2)$ time Auger and Lawrence (1989)	Pruned DPA (PDPA) Optimal, $O(Sn \log n)$ time Rigaill (2010); Johnson (2011)
Up-down	Constrained DPA (CDPA) Sub-optimal, $O(Sn^2)$ time Hocking et al. (2015)	Generalized Pruned DPA (GPDPA) Optimal, $O(Sn \log n)$ time **This paper**

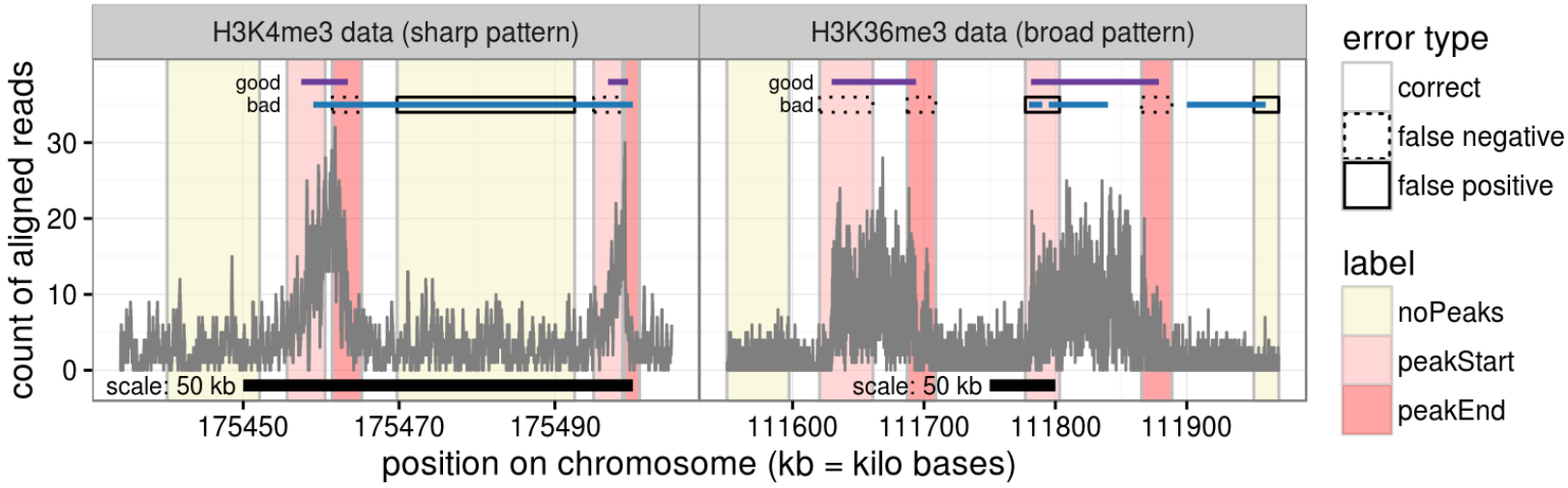
$$\underset{\substack{\mathbf{u}\in\mathbb{R}^S \\ 0=t_0<t_1<\cdots<t_{S-1}<t_S=n}}{\text{minimize}} \sum_{s=1}^{S} \sum_{i=1+t_{s-1}}^{t_s} \ell(u_s, z_i)$$

$$\text{subject to} \quad u_{s-1} \leq u_s \ \forall s \in \{2, 4, \dots\},$$

$$u_{s-1} \geq u_s \ \forall s \in \{3, 5, \dots\}.$$

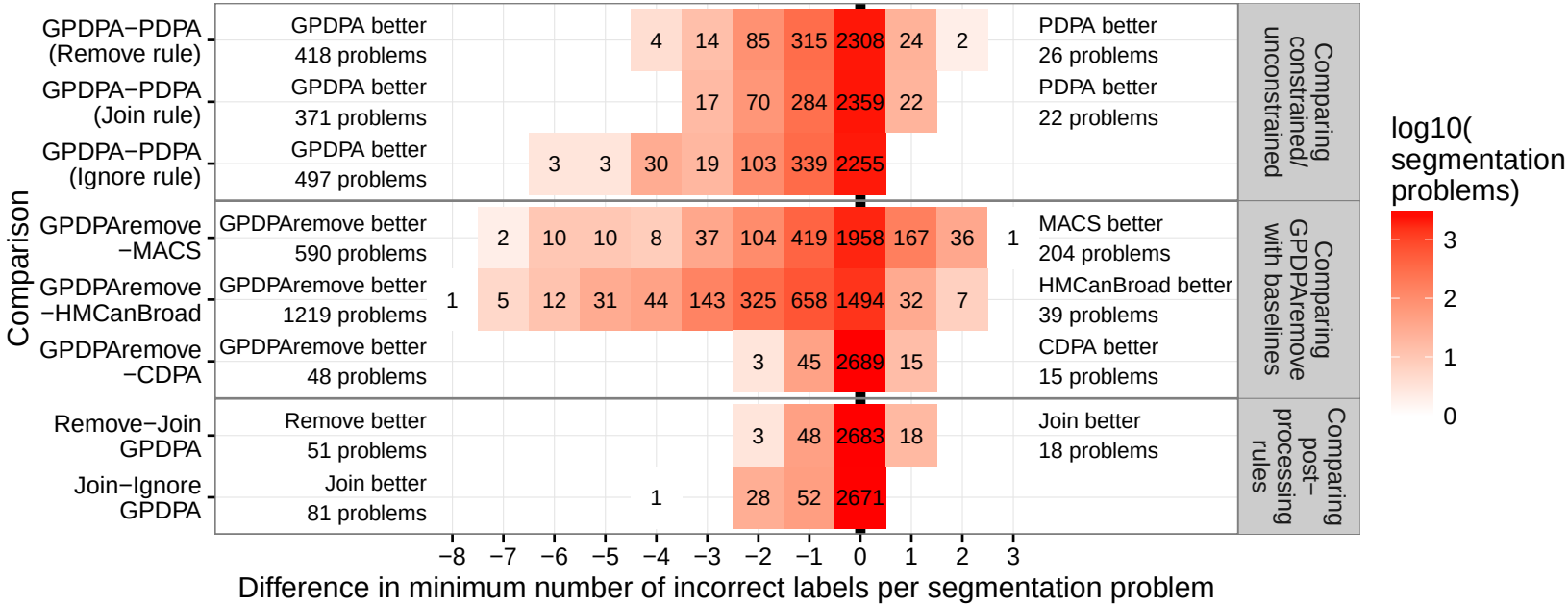
- One hyper-parameter = number of segments $S \in \{1, 3, \dots\}$.
- Hard optimization problem, naively $O(n^S)$ time.
- Previous unconstrained model: not always up-down changes.
- Interpretable: $P = (S-1)/2$ peaks (segments 2, 4, ...).
- H *et al.*, *ICML* 2015: $O(Sn^2)$ time approximate algorithm.
- **This paper**: $O(Sn \log n)$ time optimal algorithm.

Labeled data setting

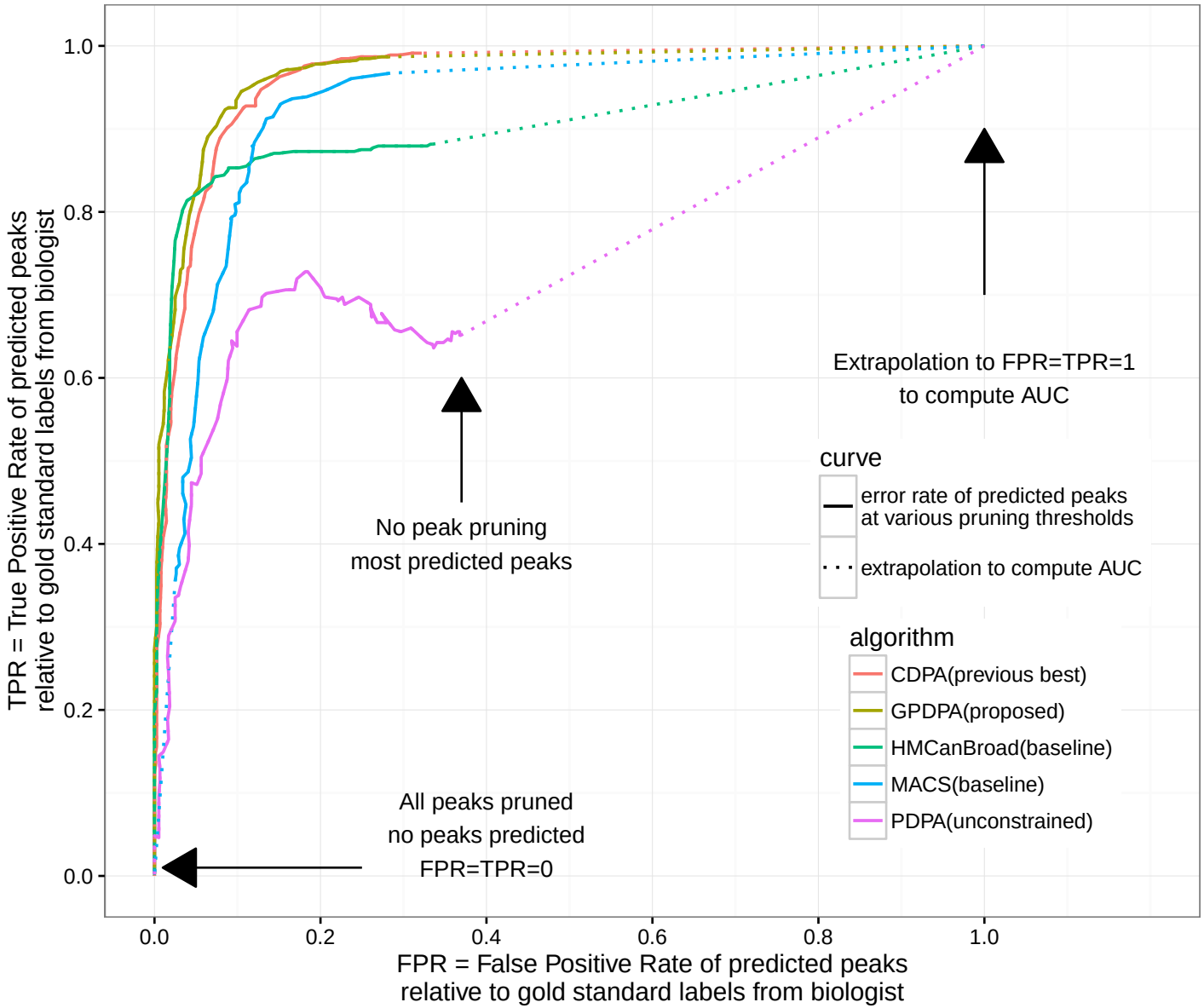


Benchmark data with 2752 labeled data sequences:
`https://rcdata.nau.edu/genomic-ml/chip-seq-chunk-db/`

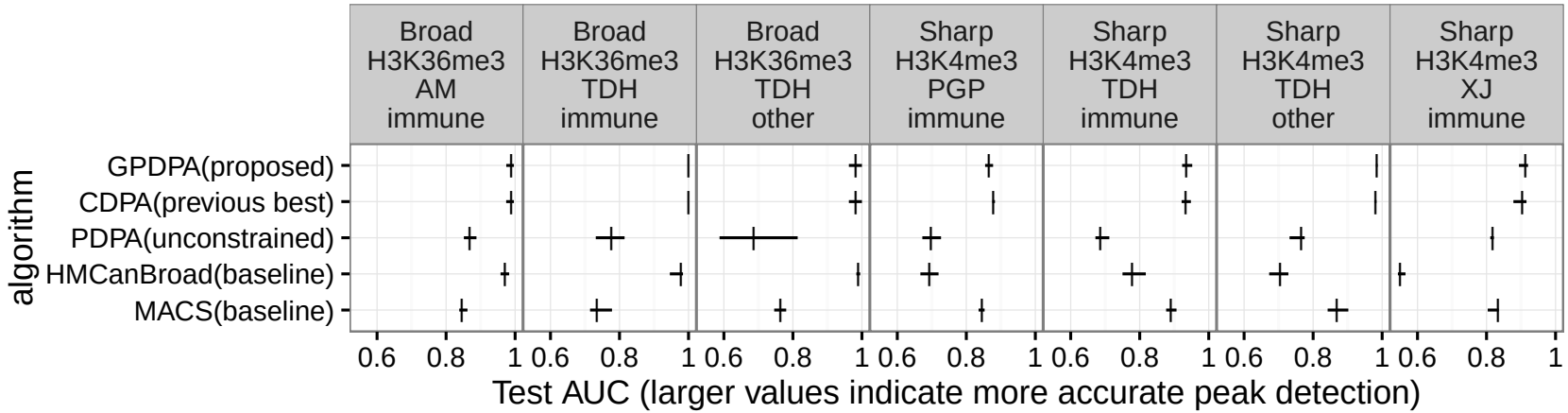
Comparing minimum label errors



Test ROC curves of predicted peaks

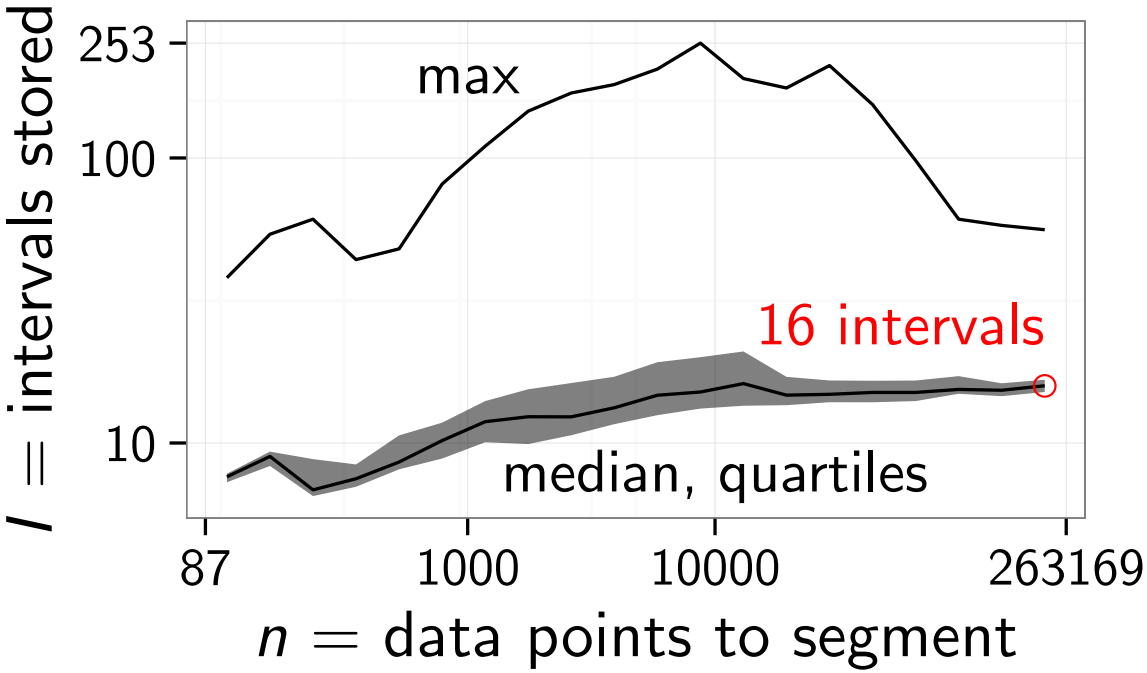


Test AUC of predicted peaks

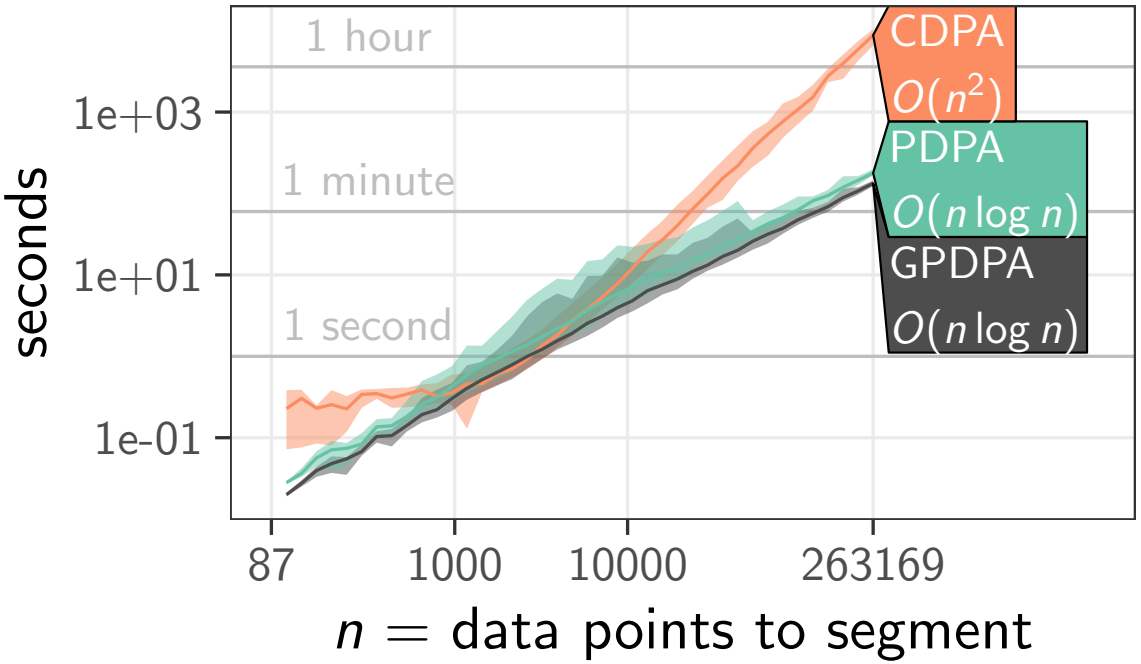


- 4-fold cross-validation: train on 3/4 of labels, test on 1/4.
- All models trained by learning a scalar significance threshold / penalty parameter, which is varied to compute ROC/AUC.
- MACS is highly inaccurate in all data sets.
- HMCanBroad is accurate for broad but not sharp pattern.
- Unconstrained PDPA algorithm not as accurate as up-down constrained algorithms (CDPA, GPDPA).

Time complexity analysis 1



Time complexity analysis 2



Constrained Dynamic Programming and Supervised Penalty Learning Algorithms for Peak Detection in Genomic Data

Linear Time Dynamic Programming for Computing Breakpoints in the Regularization Path of Models Selected From a Finite Set

Generalized Functional Pruning Optimal Partitioning (GFPOP) for Constrained Changepoint Detection in Genomic Data

Conclusions

Citation

Hocking TD, Vargovich J. Linear Time Dynamic Programming for Computing Breakpoints in the Regularization Path of Models Selected From a Finite Set. Journal of Computational and Graphical Statistics (2021).

Constrained Dynamic Programming and Supervised Penalty Learning Algorithms for Peak Detection in Genomic Data

Linear Time Dynamic Programming for Computing Breakpoints in the Regularization Path of Models Selected From a Finite Set

Generalized Functional Pruning Optimal Partitioning (GFPOP) for Constrained Changepoint Detection in Genomic Data

Conclusions

Citation

Hocking TD, Rigaill G, Fearnhead P, Bourque G. Generalized Functional Pruning Optimal Partitioning (GFPOP) for Constrained Changepoint Detection in Genomic Data. Journal of Statistical Software Vol. 101, Issue 10.

Previous work on the Segment Neighborhood problem

Constraint	No pruning	Functional pruning
None	Dynamic Prog. Algo. (DPA) Optimal, $O(SN^2)$ time Auger and Lawrence (1989)	Pruned DPA (PDPA) Optimal, $O(SN \log N)$ time Rigaill (2010); Johnson (2011)
Up-down	Constrained DPA (CDPA) Sub-optimal, $O(SN^2)$ time Hocking et al. (2015)	Generalized Pruned DPA (GPDPA) Optimal, $O(SN \log N)$ time **This paper**

- All algorithms solve the **Segment Neighborhood** "constrained" problem: most likely mean m_i for data z_i, subject to the constraint of S segments ($S-1$ changes).

$$\underset{\mathbf{m}\in\mathbb{R}^N}{\text{minimize}} \quad \sum_{i=1}^{N} \ell(m_i, z_i)$$

$$\text{subject to} \quad \sum_{i=1}^{N-1} I[m_i \neq m_{i+1}] = S-1,$$

...up-down constraints on m.

Previous work on the Optimal Partitioning problem

	no pruning	functional pruning
unconstrained	Opt. Part. Algo. exact $O(N^2)$ Jackson et al 2005	FPOP exact $O(N \log N)$ Maidstone et al 2016
up-down constrained		Generalized FPOP exact $O(N \log N)$ **This work**

- All algorithms solve the **Optimal Partitioning** "penalized" problem: most likely mean m_i for data z_i, penalized by a non-negative penalty $\lambda \in \mathbb{R}_+$ for each change:

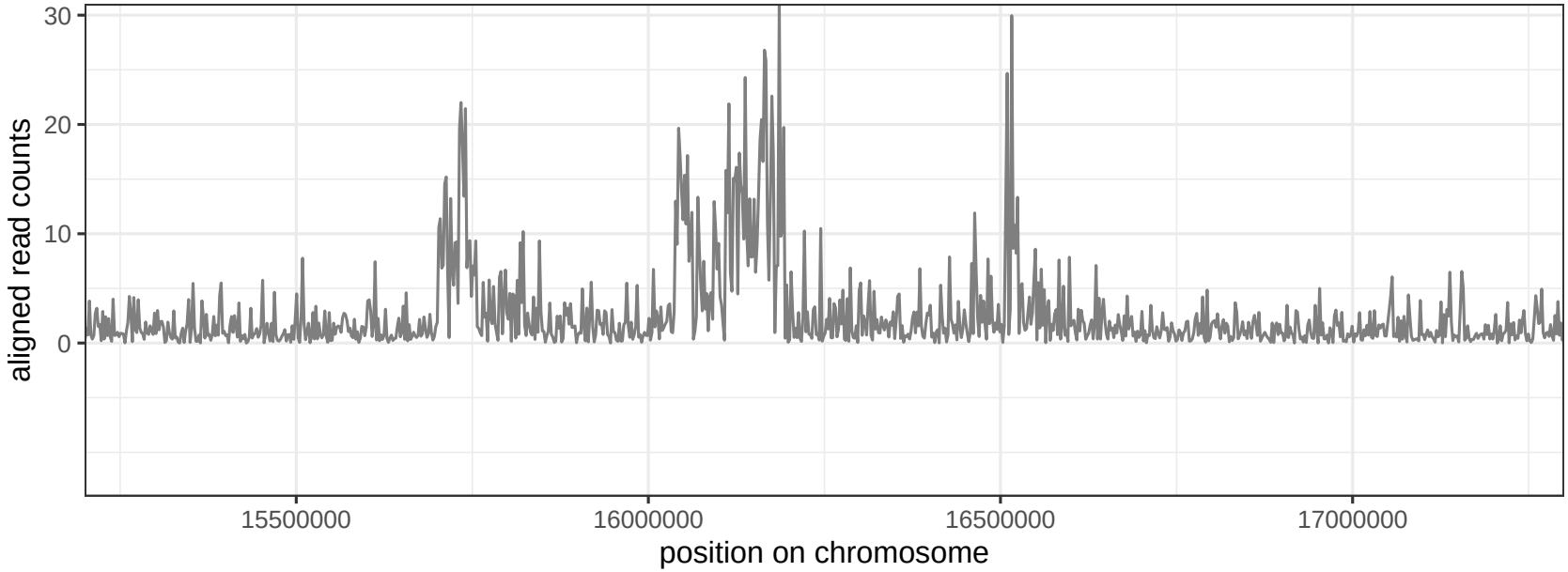
$$\begin{aligned} \underset{\mathbf{m}\in\mathbb{R}^N}{\text{minimize}} \quad & \sum_{i=1}^{N} \ell(m_i, z_i) + \lambda \sum_{i=1}^{N-1} I[m_i \neq m_{i+1}] \\ \text{subject to} \quad & \ldots\text{up-down constraints on } m. \end{aligned}$$

Benchmark of large genomic data sequences

`http://github.com/tdhock/feature-learning-benchmark`

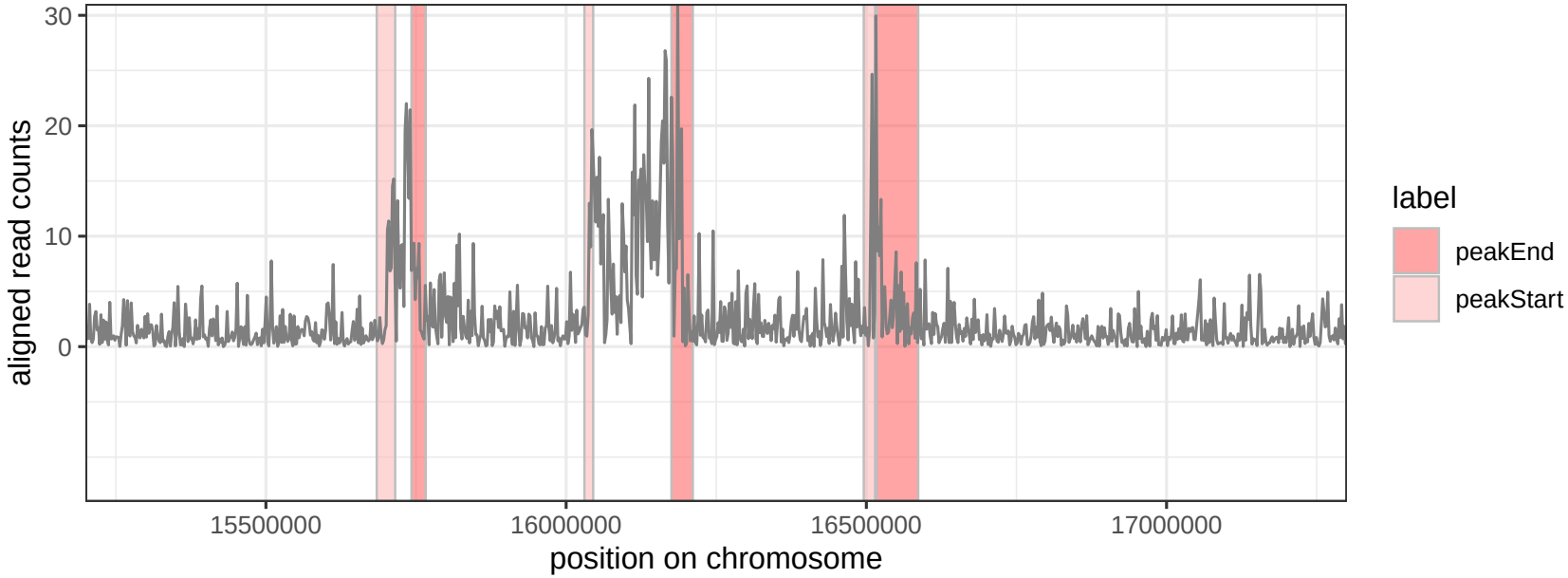
- 4951 data sequences ranging from $N = 10^3$ to $N = 10^7$.
- Ran GFPOP with penalty $\lambda \in (\log N, N)$, resulting in a range of models with different numbers of peaks, for each data set.
- Each data set has **labels** which can be used to determine an appropriate number of peaks.

Labels used to determine optimal number of peaks



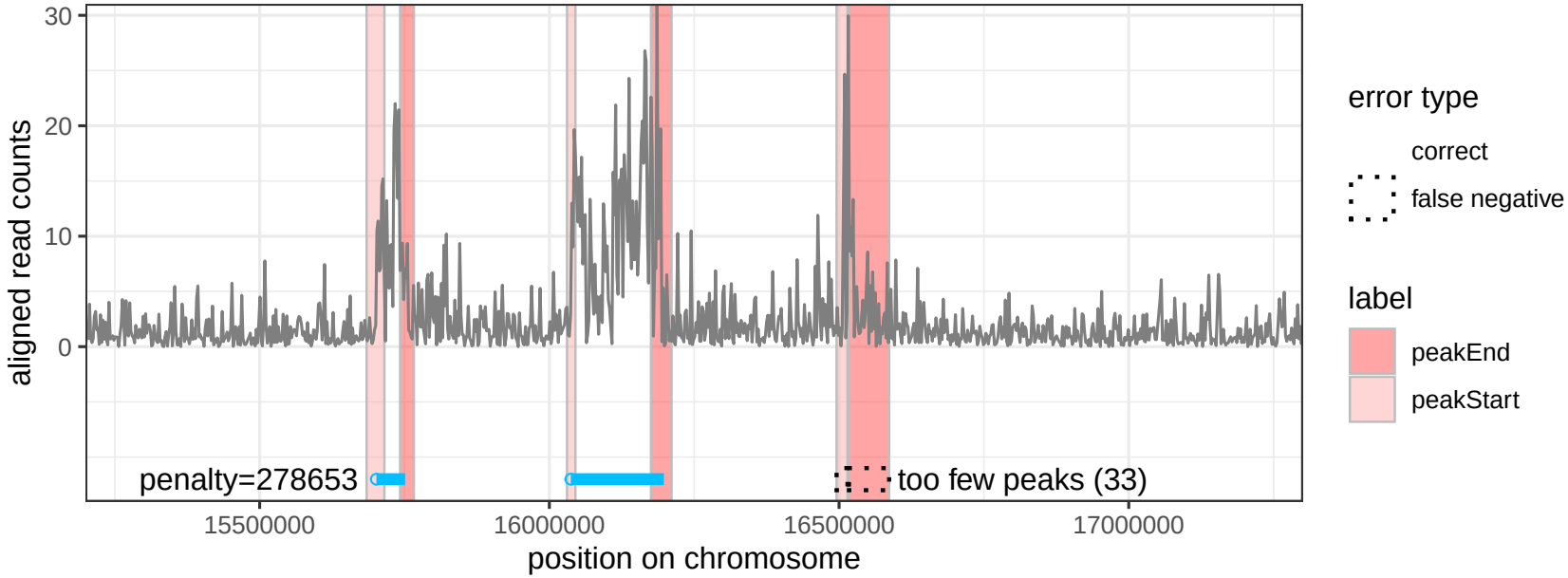
One ChIP-seq data set with $N = 1,254,751$ (only 82,233 shown).

Labels used to determine optimal number of peaks



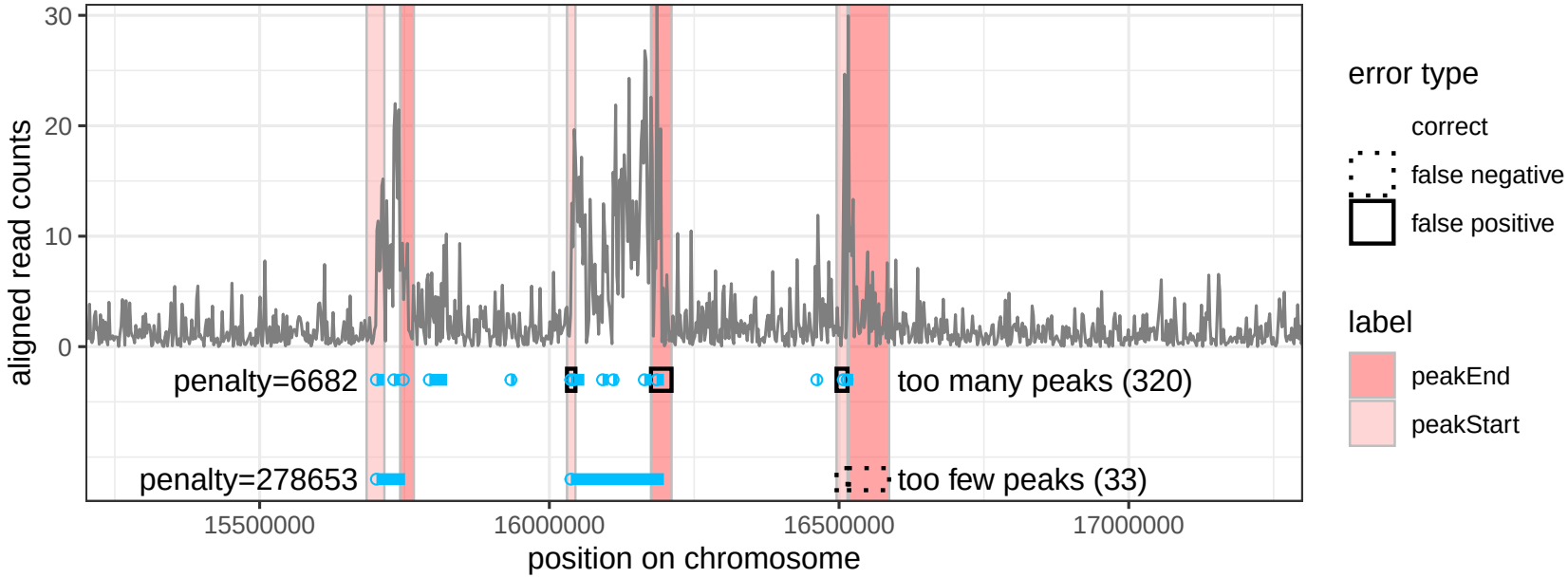
Visually labeled regions (H *et al.*, *Bioinformatics* 2017).

Labels used to determine optimal number of peaks



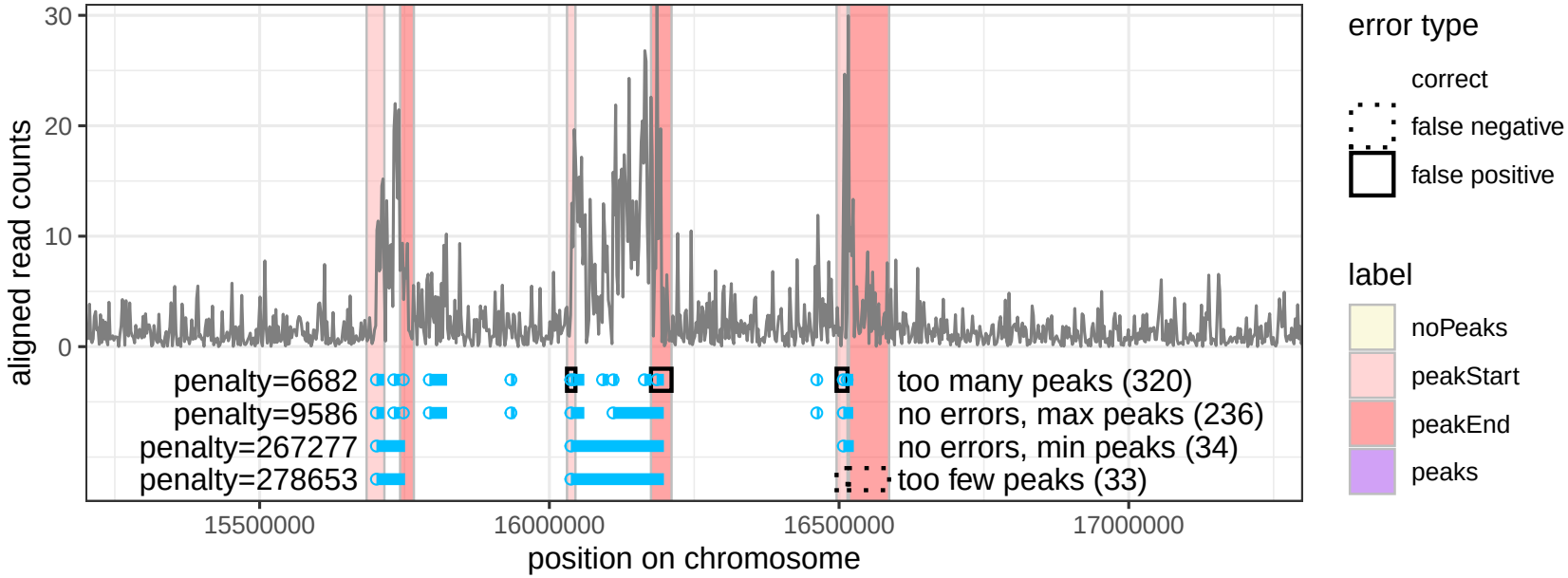
Penalty too large, too few peaks, 2 false negative labels.

Labels used to determine optimal number of peaks



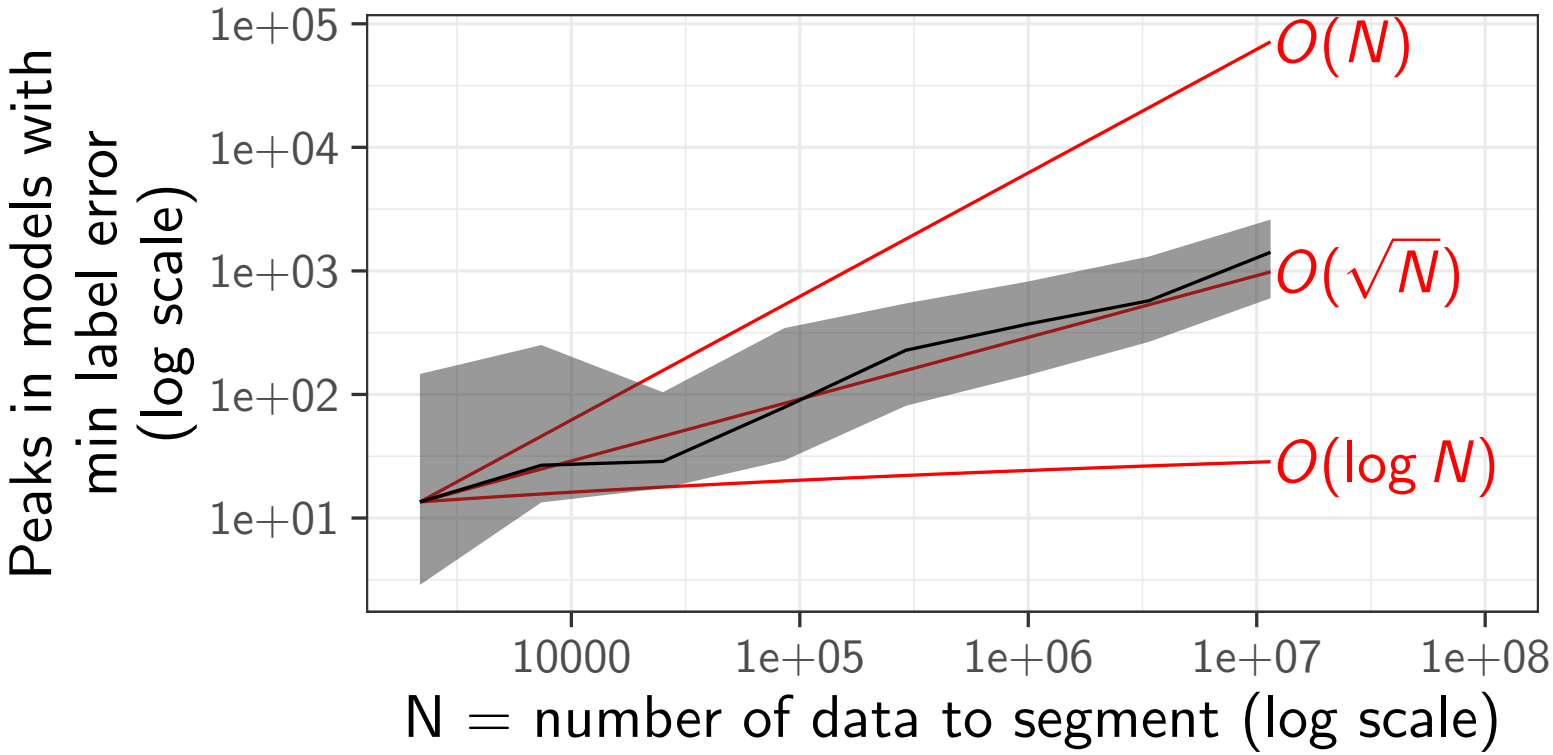
Penalty too small, too many peaks, 3 false positive labels.

Labels used to determine optimal number of peaks



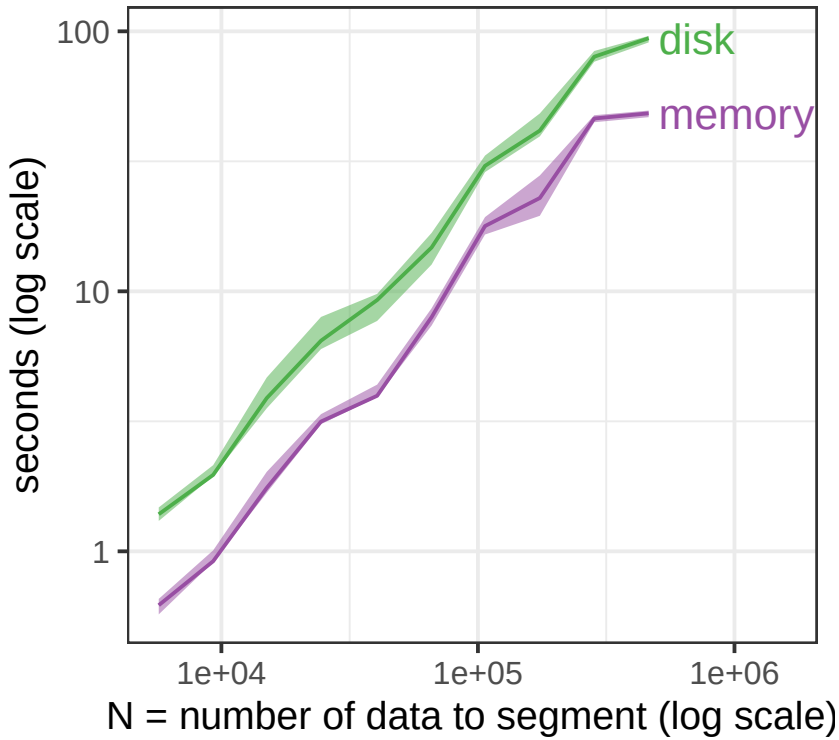
Models with 34–236 peaks have no label errors (midpoint=135).

Segment Neighborhood model too slow for $O(\sqrt{N})$ peaks



- Previous GPDPA: $O(S)$ dynamic programming iterations, each is $O(N \log N)$ time/space.
- If we want $S = O(\sqrt{N})$ segments then the algorithm is $O(N\sqrt{N} \log N)$ time/space – too much for large data.
- For example $N = 10^7$ data. Each $O(N \log N)$ DP iteration takes 1 hour, 80 GB. Overall if we want $S = O(\sqrt{N}) = 2828$ segments we need means 220 TB of storage space and 17 weeks of computation time!

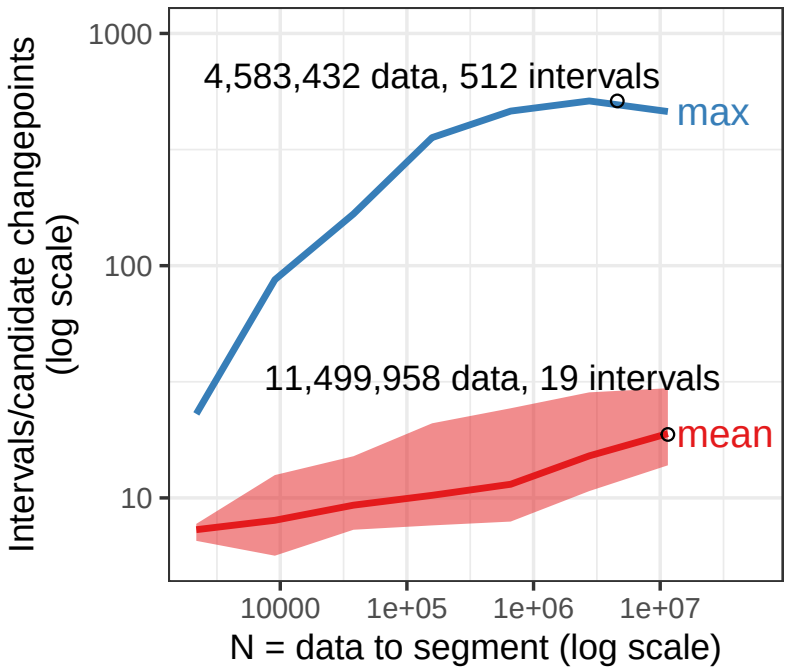
GFPOP Implementation stores cost functions on disk

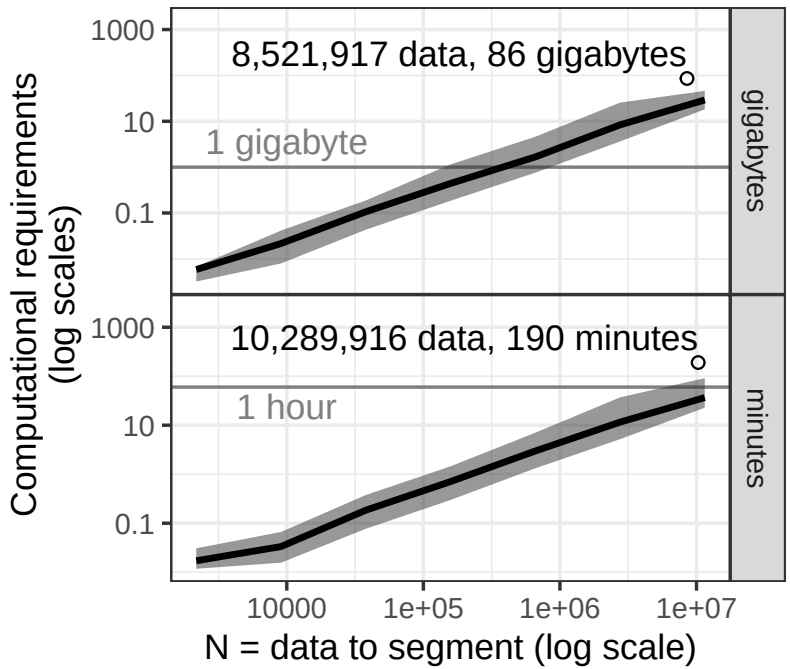


- Disk storage is only a constant factor slower than memory!
- Both are $O(N \log N)$ time.
- Memory implementation: $O(N \log N)$ memory! (too big)
- Disk implementation: $O(\log N)$ memory (< 1GB), $O(N \log N)$ disk.

Time/space to solve one penalty is $O(N \log N)$

Total time/space $= O(NI)$ where I is the number of intervals (candidate changepoints) stored in every optimal cost function $\overline{C_t}(\mu)$.





$I = O(\log N)$ intervals.

Overall $O(N \log N)$ complexity.

But how to compute model with $O(\sqrt{N})$ peaks?

Binary search algorithm using GFPOP to compute most likely model with at most P^* peaks

1: Input: data $\mathbf{z} \in \mathbb{R}^N$, target peaks P^*.
2: $\overline{L}, \overline{p} \leftarrow \text{GFPOP}(\mathbf{z}, \lambda = 0)$ // max peak model
3: $\underline{L}, \underline{p} \leftarrow \text{GFPOP}(\mathbf{z}, \lambda = \infty)$ // 0 peak model
4: While $\underline{p} \neq P^*$ and $\overline{p} \neq P^*$:
5: $\quad \lambda = (\overline{L} - \underline{L})/(\underline{p} - \overline{p})$
6: $\quad L_{\text{new}}, p_{\text{new}} \leftarrow \text{GFPOP}(\mathbf{z}, \lambda)$
7: $\quad$ If $p_{\text{new}} \in \{\underline{p}, \overline{p}\}$: return model with $\underline{p}$ peaks.
8: $\quad$ If $p_{\text{new}} < P^*$: $\underline{p} \leftarrow p_{\text{new}}$
9: $\quad$ Else: $\overline{p} \leftarrow p_{\text{new}}$
10: If $\underline{p} = P^*$: return model with $\underline{p}$ peaks.
11: Else: return model with $\overline{p}$ peaks.

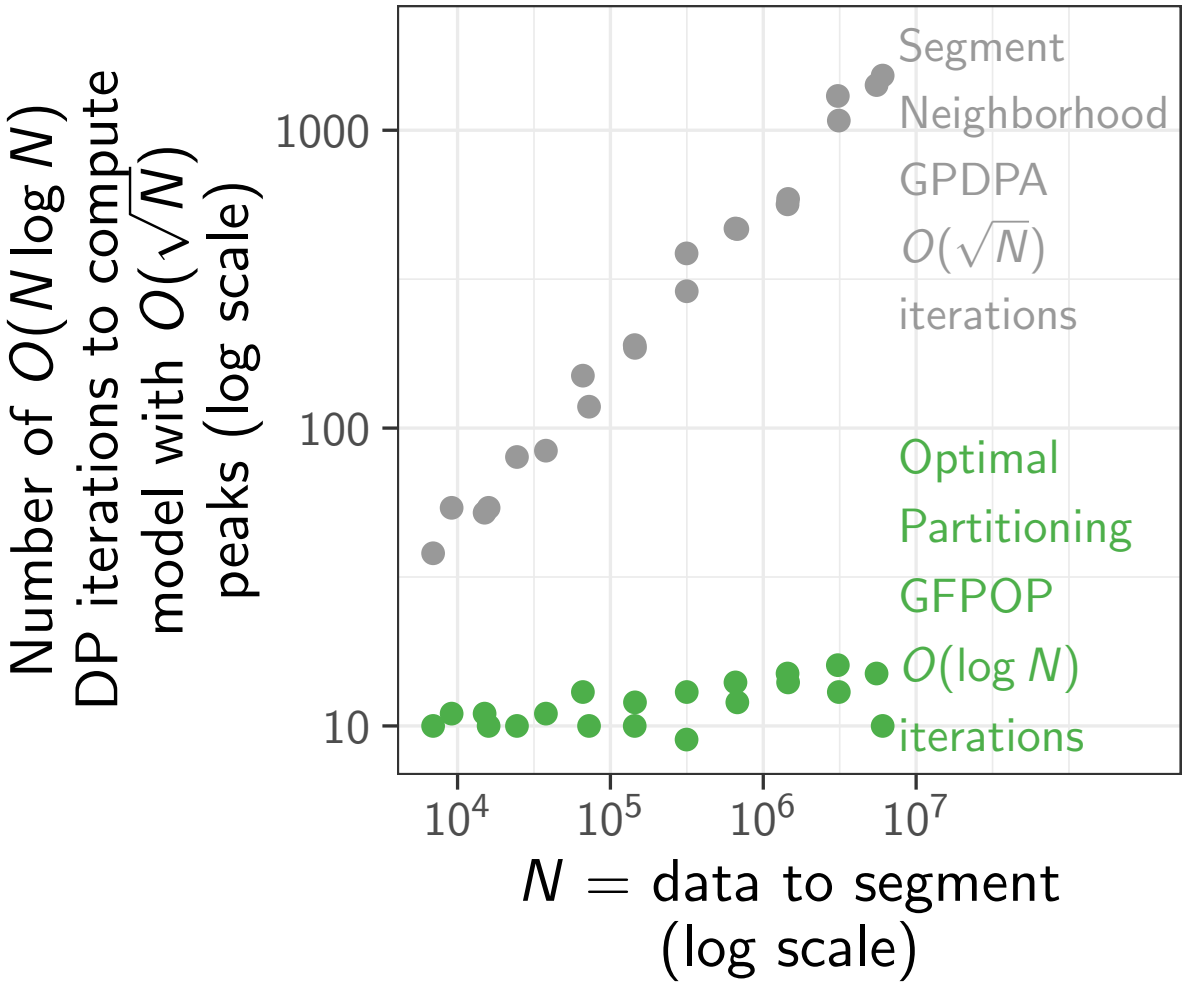
Example run of binary search algorithm using GFPOP

One data set with $P^* = 93$ peaks and $N = 146,186$ data.

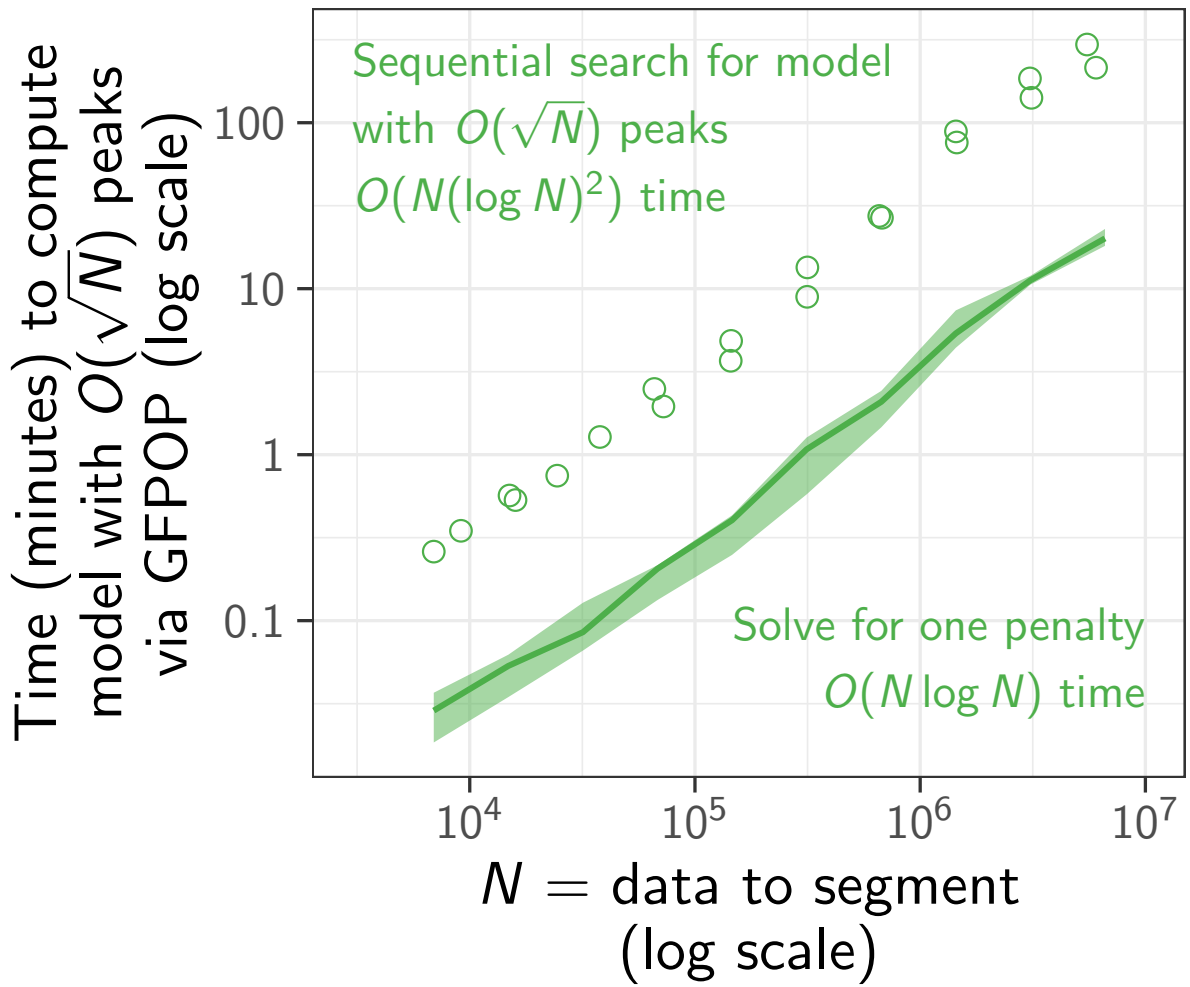
iteration	$\underline{p}$	$\overline{p}$	λ	p_{new}	L_{new}
1			0	68752	-2570319
1			∞	0	14239212
2	0	68752	244.4952	4361	1980119
3	0	4361	2811.0738	188	3676671
4	0	188	56183.7271	55	5330310
5	55	188	12433.3766	98	4108354
6	55	98	28417.5941	72	4584042
7	72	98	18295.6895	83	4336773
8	83	98	15227.9249	90	4218815
9	90	98	13807.6282	95	4146172
10	90	95	14528.5052	92	4188881
11	92	95	14236.0863	94	4160179
12	92	94	14350.6622	93	4174480

12 DP iterations much fewer than $93 \times 2 = 186$ which would be required for Segment Neighborhood!

Only $O(\log N)$ runs of GFPOP to compute $O(\sqrt{N})$ peaks



Binary search only a log factor slower than solving one penalty



For $N = 10^7$ only several hours of computation! (compare with weeks for Segment Neighborhood algorithm)

Constrained Dynamic Programming and Supervised Penalty Learning Algorithms for Peak Detection in Genomic Data

Linear Time Dynamic Programming for Computing Breakpoints in the Regularization Path of Models Selected From a Finite Set

Generalized Functional Pruning Optimal Partitioning (GFPOP) for Constrained Changepoint Detection in Genomic Data

Conclusions

Conclusions

- Previous GPDPA was $O(N\sqrt{N}\log N)$ – much too complex to compute a zero-error model with $O(\sqrt{N})$ peaks for $N = 10^7$ data.
- Proposed GFPOP with binary search is $O(N(\log N)^2)$ time, $O(\log N)$ memory, $O(N \log N)$ disk – optimal models are now possible to compute for large data.
- C++ code with R interface:
  `PeakSegPipeline::PeakSegFPOP_disk`
  `https://github.com/tdhock/PeakSegPipeline`
- Future work: changepoint detection in large ecological data?
- Contact me: toby.hocking@nau.edu
- Thanks!

For some data the desired number of peaks does not exist!

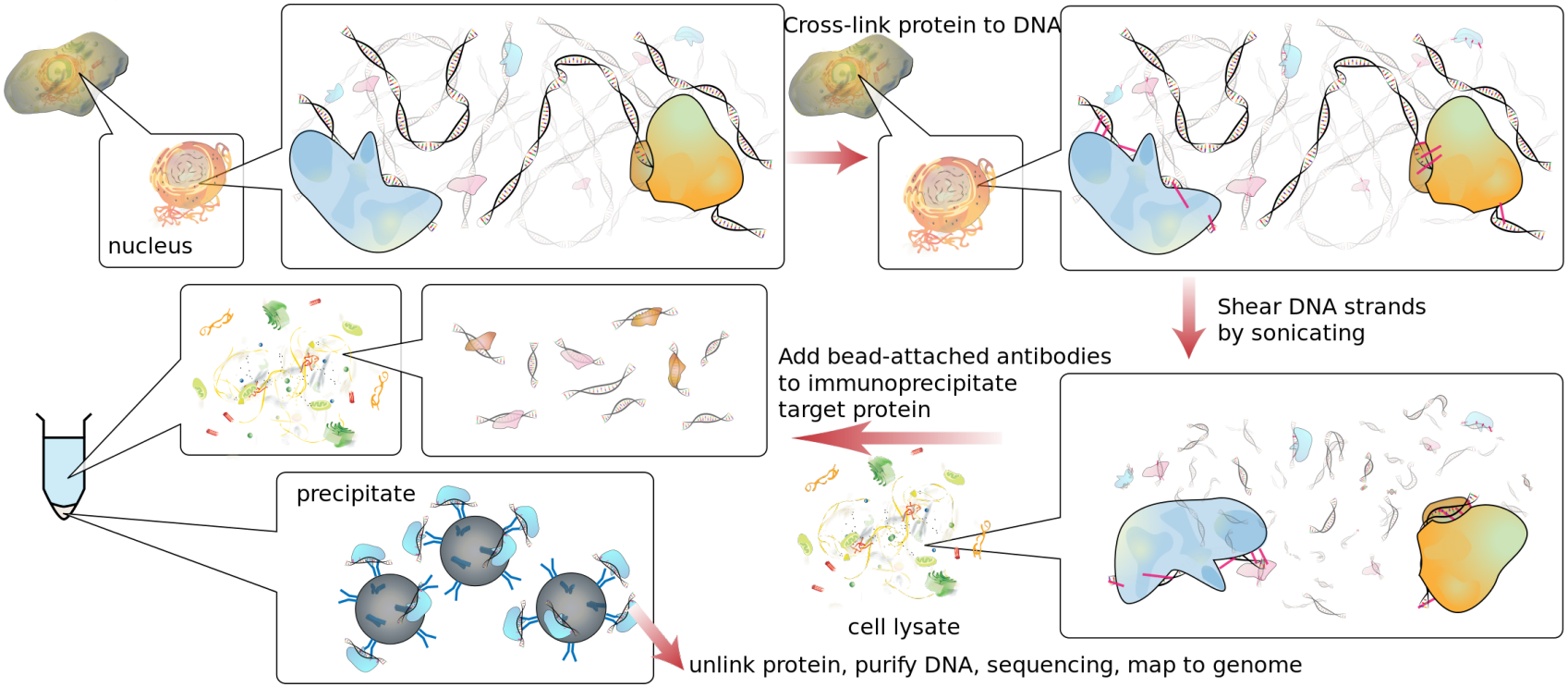
One data set with $P^* = 75$ and $N = 66,031$ data.

iteration	$\underline{p}$	$\overline{p}$	λ	p_{new}	L_{new}
1			0	29681	-1495863.85
1			∞	0	1200631.42
2	0	29681	90.85	3445	-1245181.37
3	0	3445	709.96	401	-446632.57
4	0	401	4107.89	51	3105.03
5	51	401	1284.96	168	-230152.31
6	51	168	1993.65	97	-120725.83
7	51	97	2691.98	68	-53946.18
8	68	97	2302.75	81	-86423.80
9	68	81	2498.28	77	-76891.10
10	68	77	2549.44	71	-61722.09
11	71	77	2528.17	74	-69330.62
12	74	77	2520.16	76	-74374.78
13	74	76	2522.08	76	-74374.78

No model exists between $\underline{p} = 74$ and $\overline{p} = 76$, so algo stops and returns the simpler model with 74 peaks.

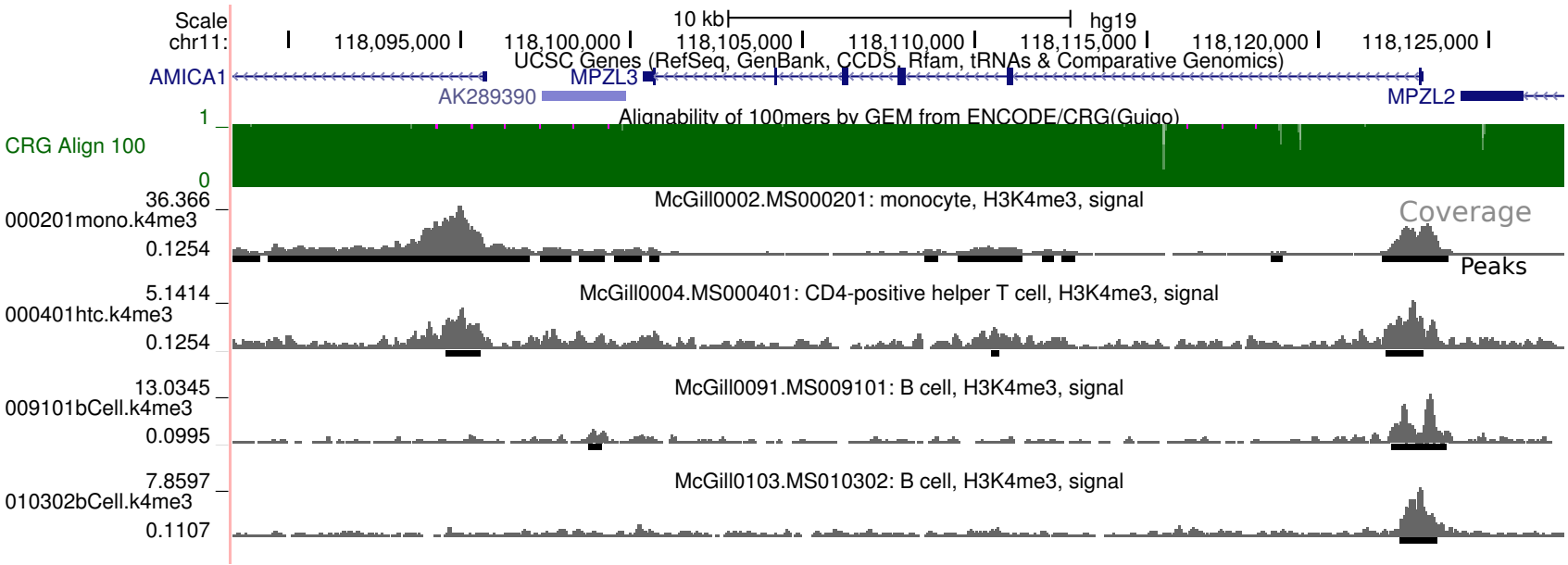
Chromatin immunoprecipitation sequencing (ChIP-seq)

Analysis of DNA-protein interactions.



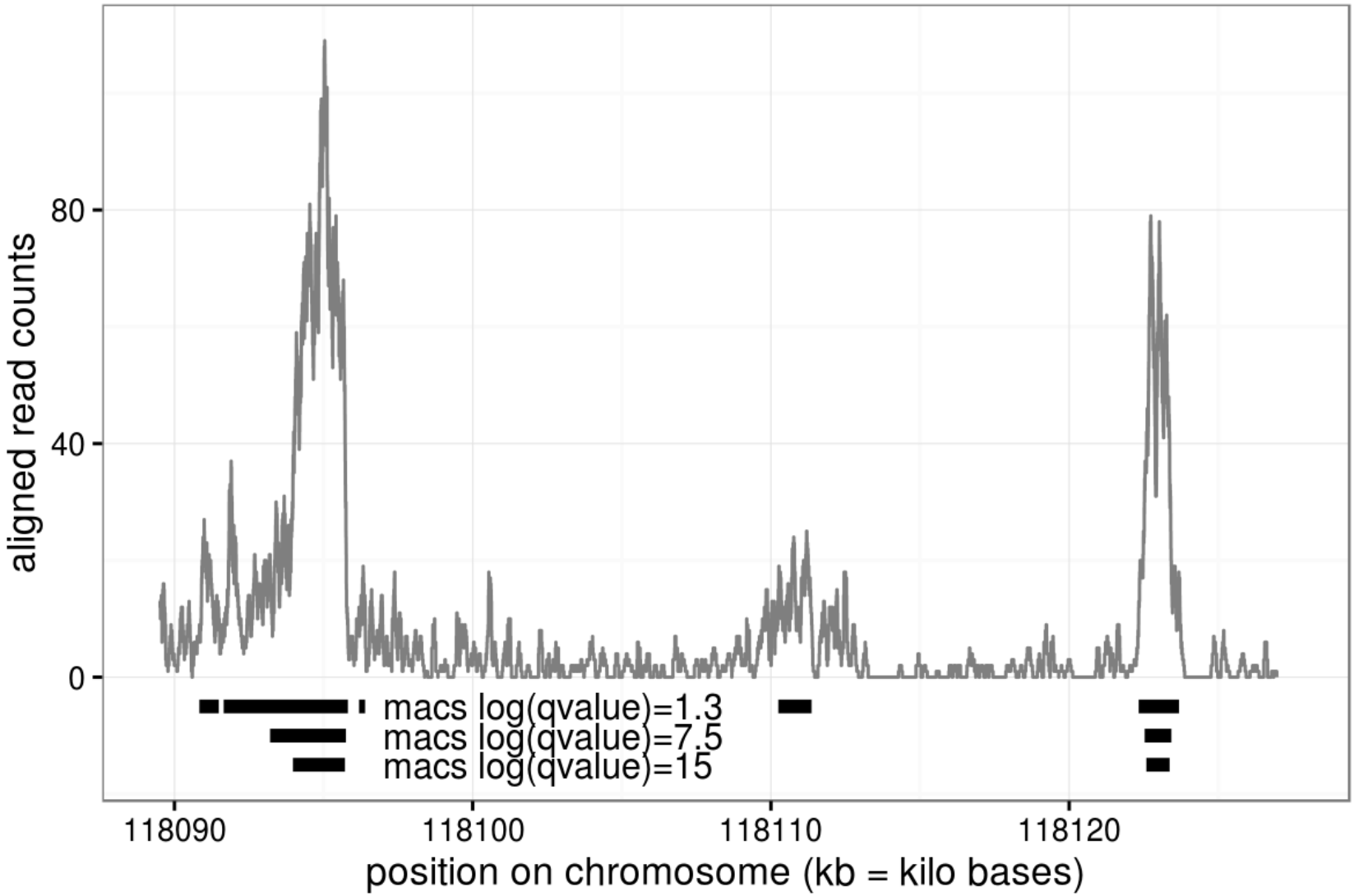
Source: "ChIP-sequencing," Wikipedia.

Problem: find peaks in each of several samples

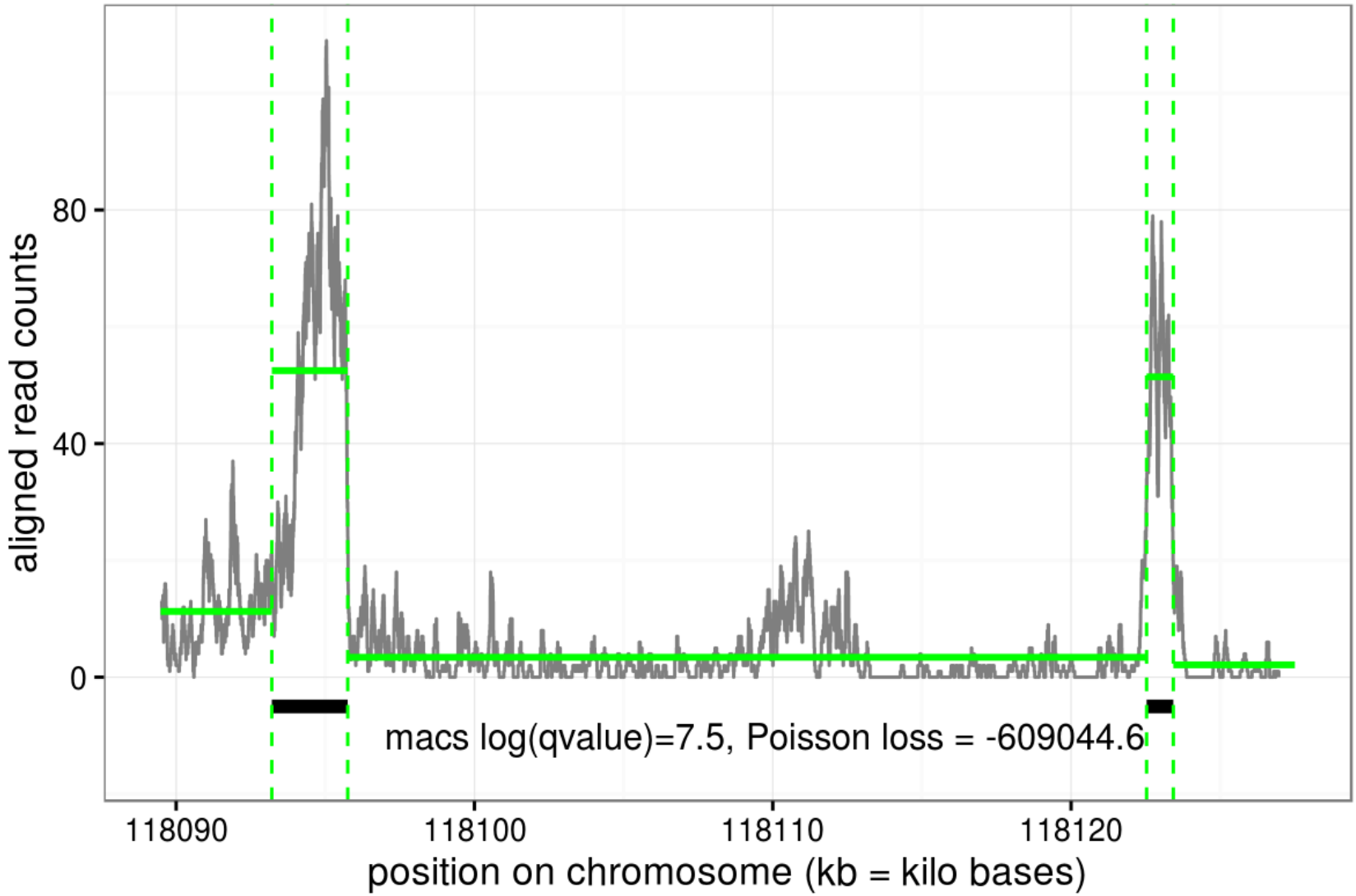


- Grey profiles are noisy aligned read count signals – peaks are genomic locations with protein binding sites.
- Black bars are peaks called by MACS2 (Zhang et al, 2008) – many false positives! (black bars where there is only noise)
- From a machine learning perspective, this is binary classification (positive=peaks, negative=noise).

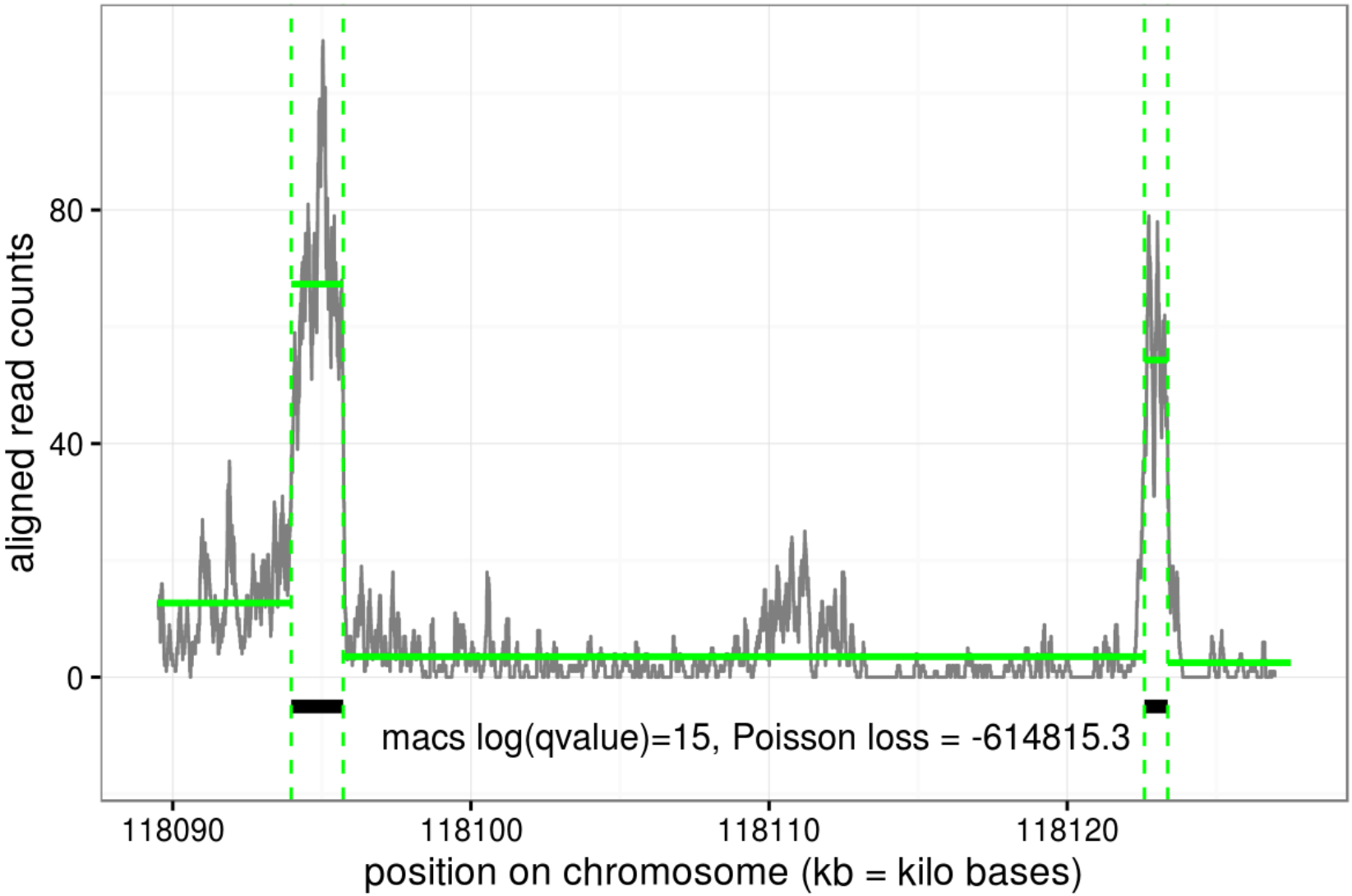
Which macs parameter is best for these data?



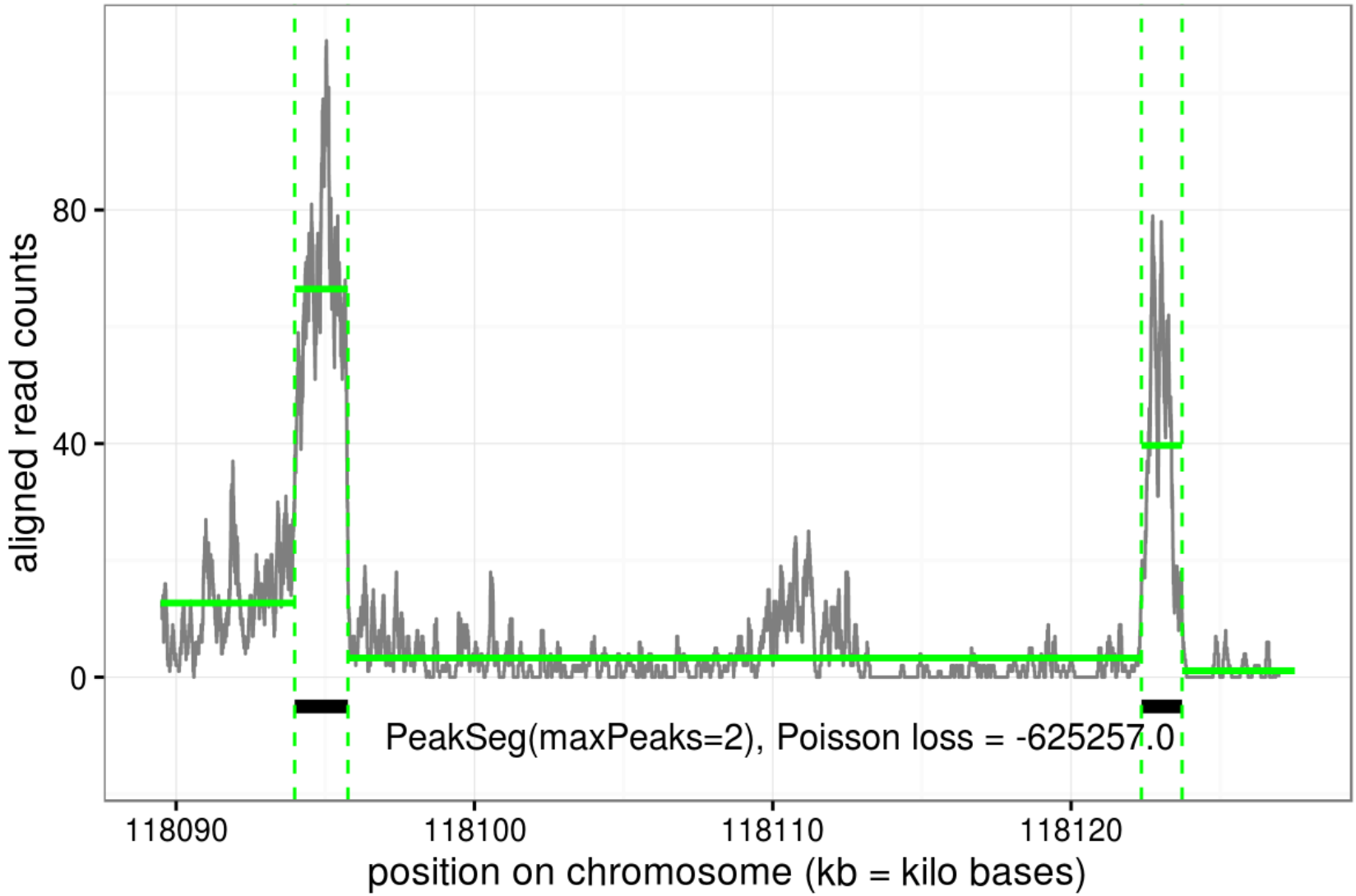
Compute likelihood/loss of piecewise constant model



Idea: choose the parameter with a lower loss



PeakSeg: search for the peaks with lowest loss



Simple model with only one parameter (number of peaks).

Statistical model is a piecewise constant Poisson mean

H *et al.*, ICML 2015.

- We have n count data $z_1, \ldots, z_n \in \mathbb{Z}_+$.
- 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 $u_1, \ldots, u_S \in \mathbb{R}_+$.
- Let $0 = t_0 < t_1 < \cdots < t_{S-1} < t_S = n$ be the segment limits.
- Statistical model: for every segment $s \in \{1, \ldots, S\}$, $z_i \overset{\text{iid}}{\sim} \text{Poisson}(u_s)$ for every data point $i \in (t_{s-1}, t_s]$.
- PeakSeg up-down constraint: $u_1 \leq u_2 \geq u_3 \leq u_4 \geq \cdots$
- Want to find means u_s which maximize the Poisson likelihood: $P(Z = z_i | u_s) = u_s^{z_i} e^{-u_s} / (z_i!)$.
- Equivalent to finding means u_s which minimize the Poisson loss: $\ell(u_s, z_i) = u_s - z_i \log u_s$.

Dynamic programming and functional pruning

Classical dynamic programming for optimal partitioning (Jackson et al 2005) computes the vector of optimal loss values up to N data points, $O(N^2)$ time because each DP iteration needs to consider all $O(N)$ possible changepoints and cost values.

$$\mathcal{C}_1 \quad \mathcal{C}_2 \quad \cdots \quad \mathcal{C}_{N-1} \quad \mathcal{C}_N$$

Functional pruning optimal partitioning (Maidstone 2016) computes a vector of loss **functions**, $O(N \log N)$ because each DP iteration only considers $O(\log N)$ candidate changepoints (the others — which will never be optimal — are pruned).

$$C_1(m_1) \quad C_2(m_2) \quad \cdots \quad C_{N-1}(m_{N-1}) \quad C_N(m_N)$$

Contribution of this work: a new algorithm that applies the functional pruning technique to the up-down constrained model.

Constrained optimal partitioning problem

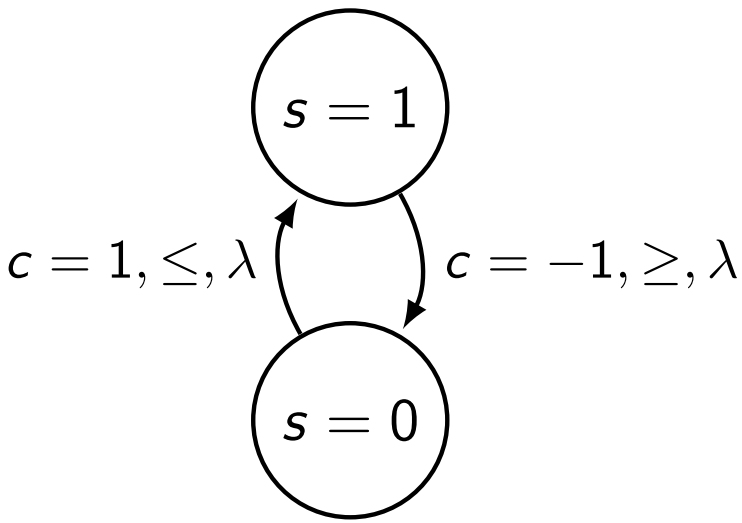
$$\underset{\substack{\mathbf{m}\in\mathbb{R}^N,\ \mathbf{s}\in\{0,1\}^N \\ \mathbf{c}\in\{-1,0,1\}^{N-1}}}{\text{minimize}} \quad \sum_{i=1}^{N} \ell(m_i, z_i) + \lambda \sum_{i=1}^{N-1} I(c_i \neq 0)$$

subject to

no change: $c_t = 0 \Rightarrow m_t = m_{t+1}$ and $s_t = s_{t+1}$

go up: $c_t = 1 \Rightarrow m_t \leq m_{t+1}$ and $(s_t, s_{t+1}) = (0, 1)$,

go down: $c_t = -1 \Rightarrow m_t \geq m_{t+1}$ and $(s_t, s_{t+1}) = (1, 0)$.



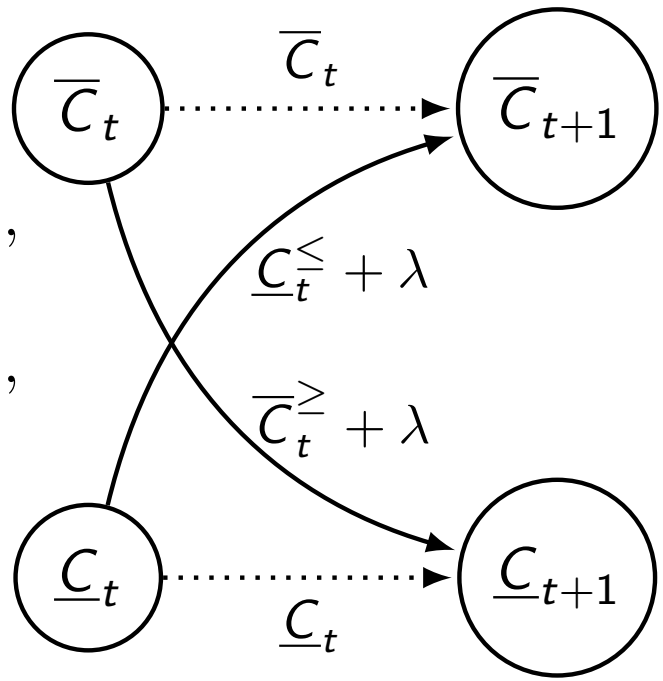
Nodes=states s,
Edges=changes c (constraint, penalty).

Generalized Functional Pruning Optimal Partitioning (GFPOP) algorithm for up-down constrained model

Recursively compute two vectors of real-valued cost functions:

$$\begin{array}{lll} \overline{C}_1(m_1) & \cdots & \overline{C}_N(m_N) \quad \text{optimal cost in peak state } s=1 \\ \underline{C}_1(m_1) & \cdots & \underline{C}_N(m_N) \quad \text{optimal cost in background state } s=0 \end{array}$$

$$\overline{C}_{t+1}(\mu) = \ell(\mu, z_i) + \min\{\overline{C}_t(\mu),\ \underline{C}_t^{\leq}(\mu) + \lambda\},$$

$$\underline{C}_{t+1}(\mu) = \ell(\mu, z_i) + \min\{\underline{C}_t(\mu),\ \overline{C}_t^{\geq}(\mu) + \lambda\},$$

where $f^{\leq}(\mu) = \min_{x \leq \mu} f(x)$,
$f^{\geq}(\mu) = \min_{x \geq \mu} f(x)$.