# Frontiers in optimal change-point algorithms

Toby Dylan Hocking — toby.dylan.hocking@usherbrooke.ca
Université de Sherbrooke, Département d'Informatique
Collaborators: Charles Truong, Guillem Rigaill, Vincent Runge

Thanks to DATAIA for financing 6 month research visit in Paris!

# Why are change-point algorithms so interesting?

In statistics, we have data, and we want good parameter estimates:

- Linear model coefficients
- Neural networks weights
- Change-points

| | Linear | Neural Net | Change-point |
|---|---|---|---|
| Problem: | Convex | Non-convex | Non-convex |
| Algorithm: | Gradient | Gradient | Dynamic programming |
| Optimality: | Global | Local | Global |

- Change-point detection is a non-convex problem
- with efficient dynamic programming algorithms
- that return the globally optimal model parameters!

This talk: recent computational advances.

1. Drawbacks of classic algorithms

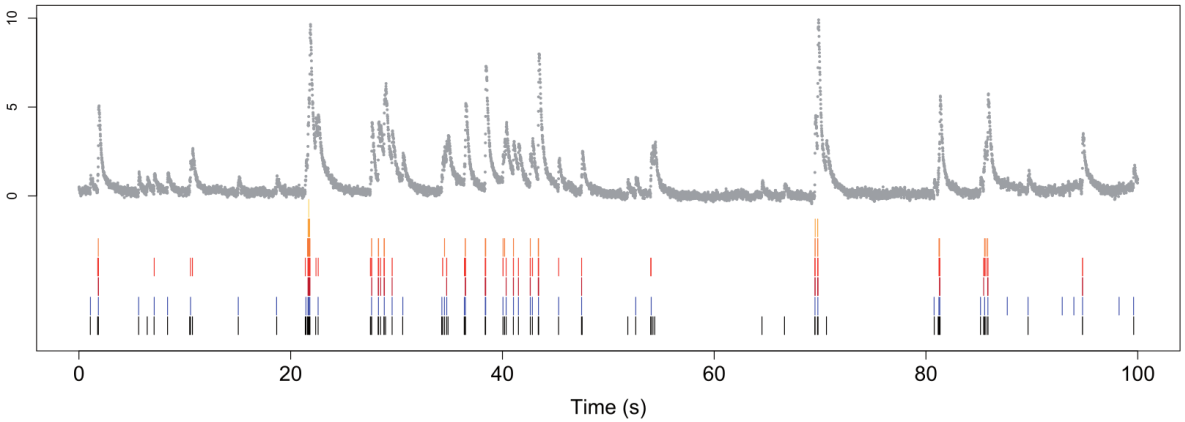
2. Linear time optimal algorithms with pruning

3. Computing constrained models using penalized solver

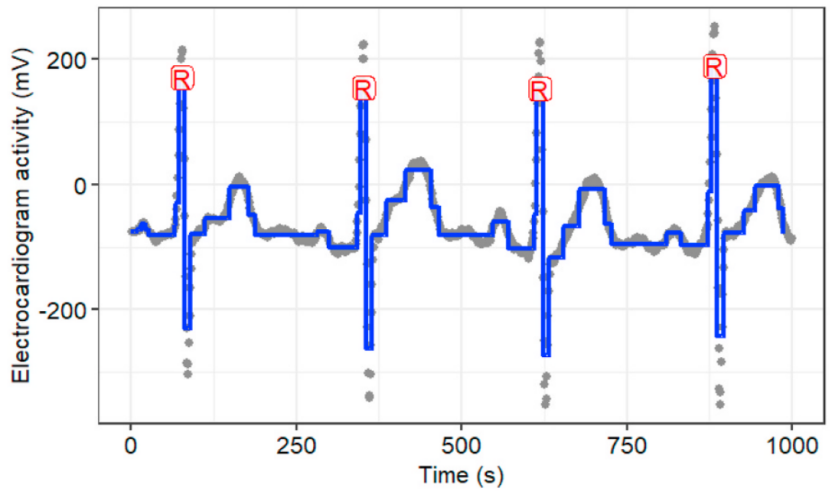
Summary and Discussion

# Changepoint detection algorithms for data over time

Neuron spikes, Jewell *et al.*, Biostatistics 2019.

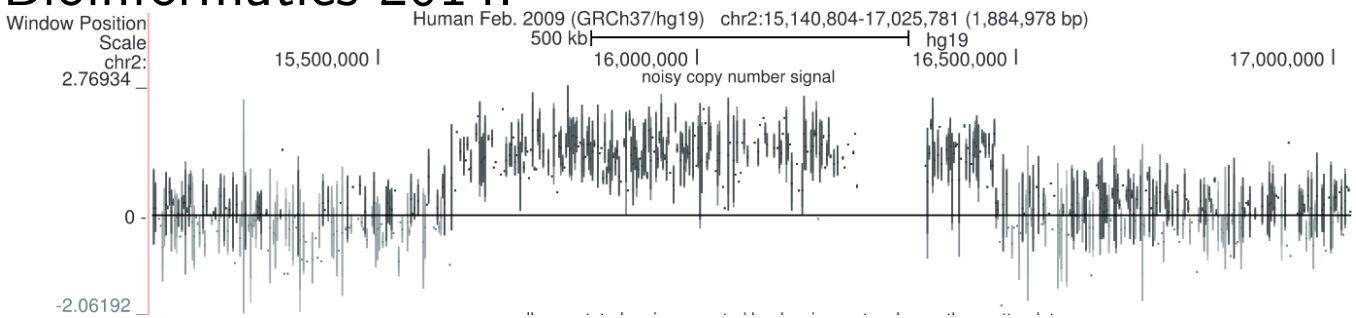



Electrocardiograms (heart monitoring), Fotoohinasab *et al.*, Asilomar conference 2020.

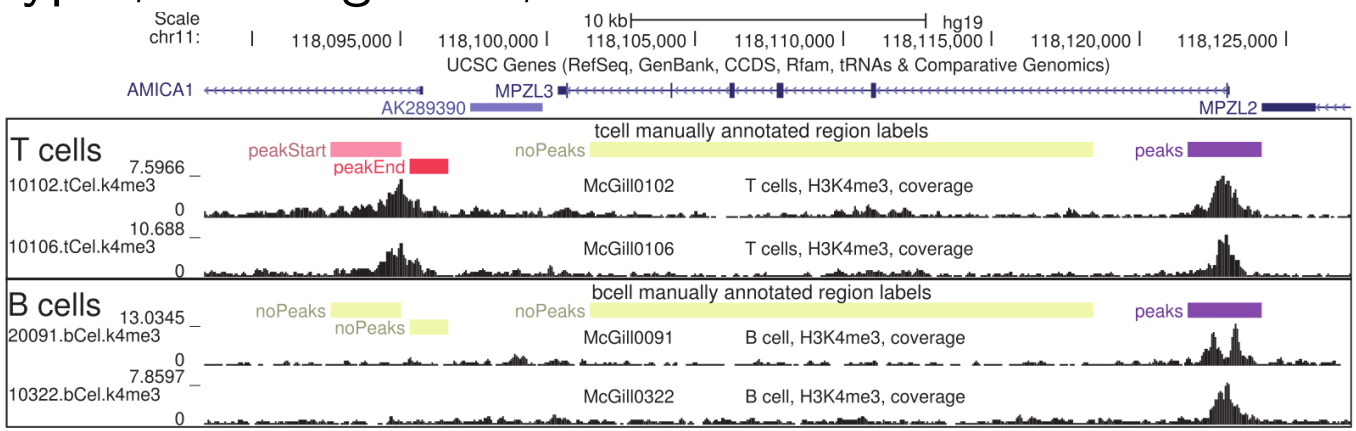


# Changepoint detection algorithms for data over space

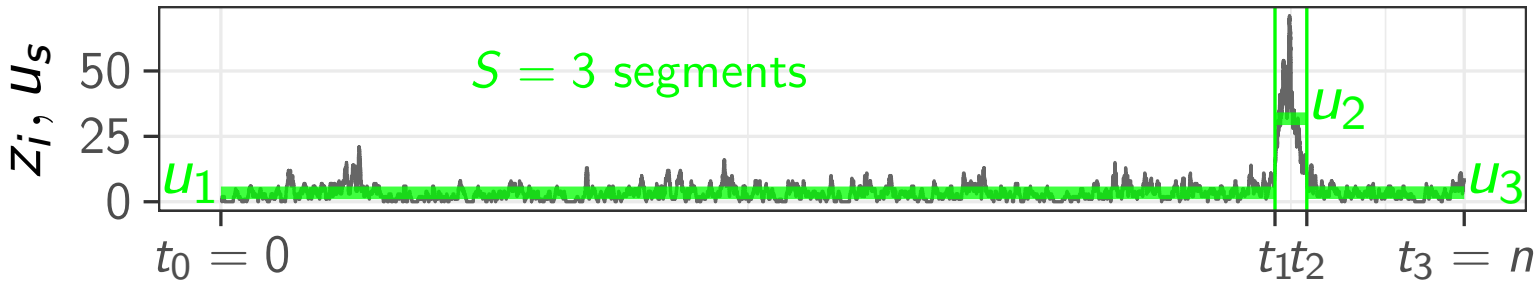
DNA copy number data for cancer diagnosis, Hocking *et al.*, Bioinformatics 2014.




ChIP-seq data for annotating active/inactive regions in different cell types, Hocking *et al.*, Bioinformatics 2017.



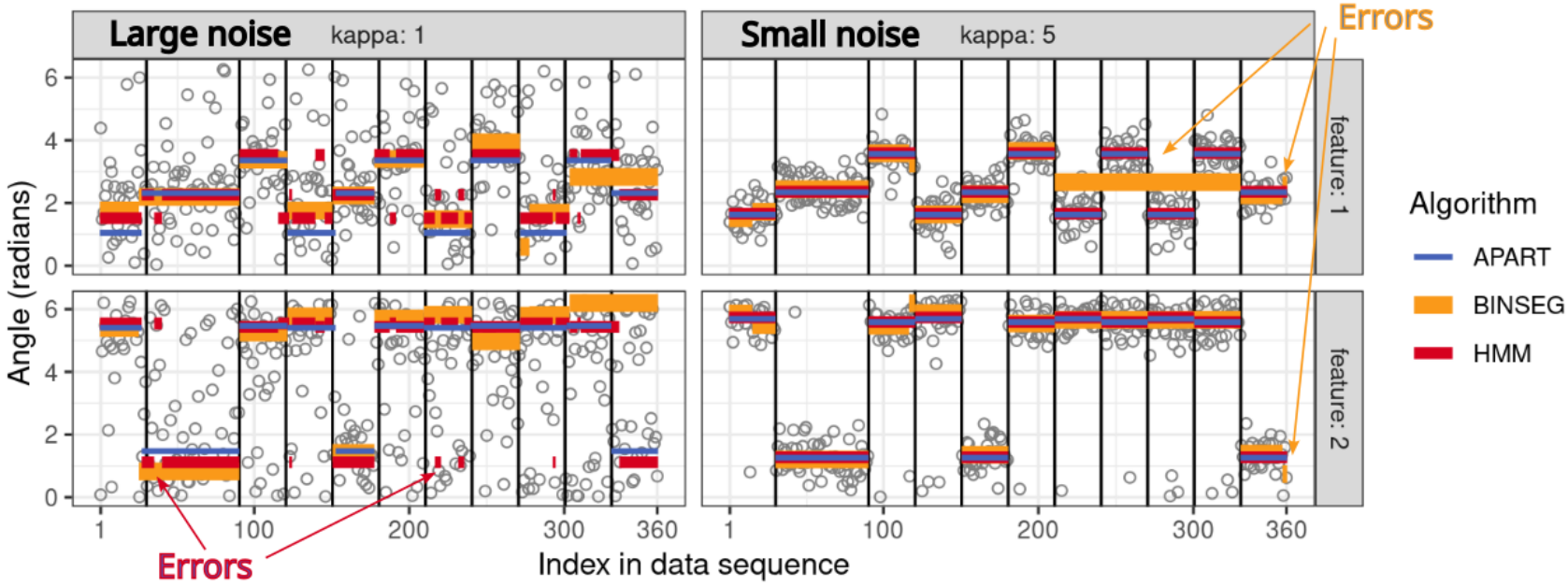
# Optimal changepoint detection problem




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

- Algorithm inputs $n$ data $z_1, \ldots, z_n$ and # of segments $S$.
- Loss function $\ell$ measures fit of means $u_s$ to data $z_i$ (Gaussian, Poisson, exponential, ...).
- Goal is to compute best $S-1$ changepoints $t_1 < \cdots < t_{S-1}$ and $S$ segment parameters $u_1, \ldots, u_S$.
- Non-convex optimization problem, naïvely $O(n^S)$ time.
- Fast approximate heuristic algorithms?

# Fast heuristics can yield inaccurate change-points

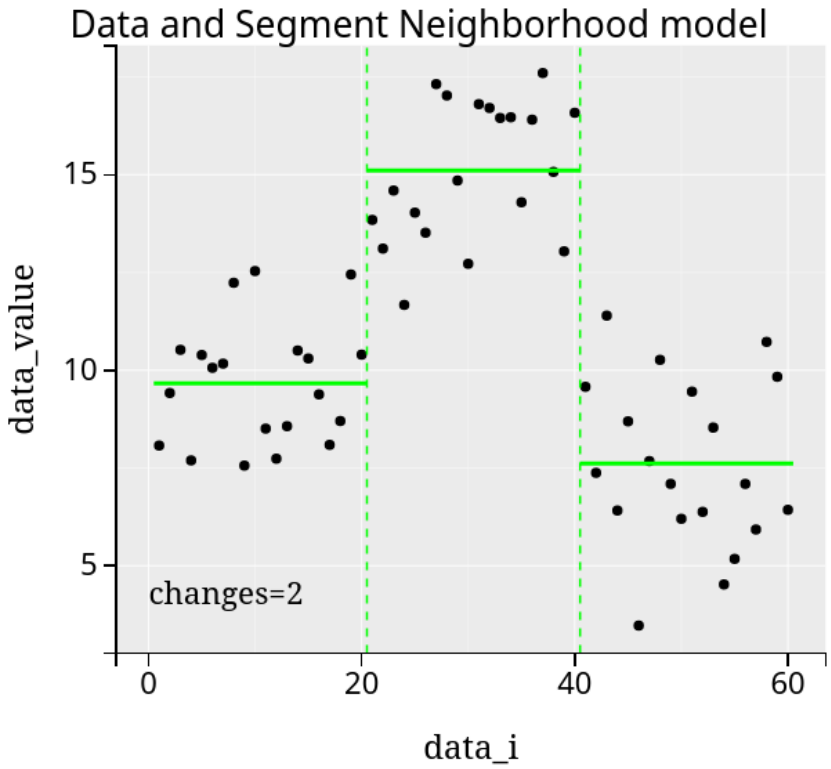


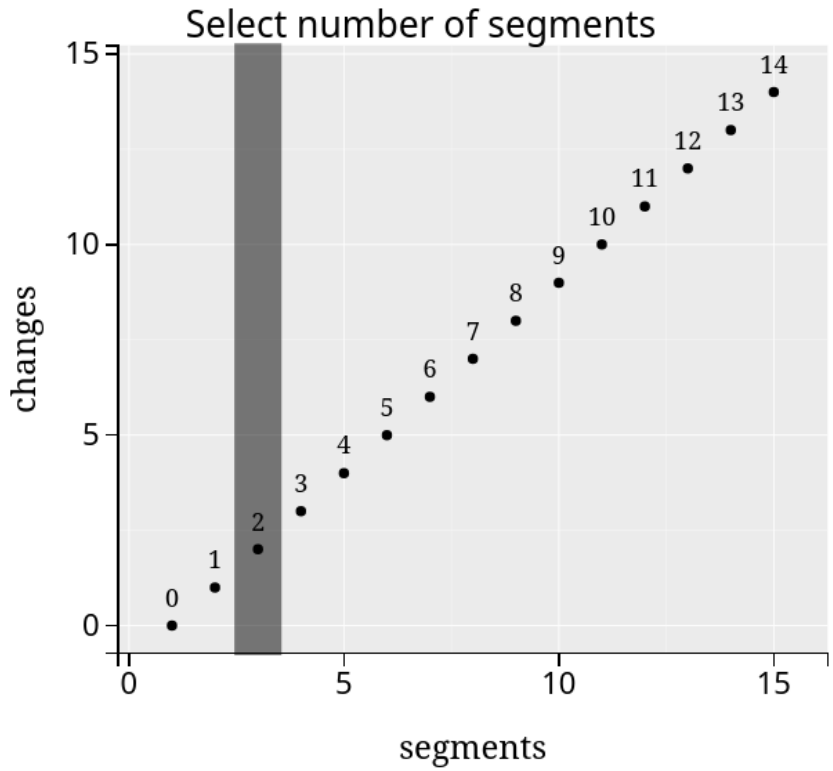

- Simulation: angular data, 2 dimensions/features.
- Heuristics are approximate optimization algorithms.
- Fast: linear $O(n)$ time for $n$ data.
- Not guaranteed to compute change-points with best loss.
- BINSEG = Binary segmentation (Scott and Knott, 1974).
- HMM = Hidden Markov Model (Rabiner, 1989).

# Constrained problem, classic dynamic programming solution

**Constrained** optimal change-point problem:

$$\min_{\mu\in\mathbb{R}^n} \underbrace{\sum_{i=1}^{n} \ell(\mu_i, z_i)}_{\text{Loss (data-fitting)}} \quad \text{subject to} \quad \underbrace{\sum_{i=2}^{n} I[\mu_{i-1} \neq \mu_i] \leq S.}_{\text{Number of change-points (regularization)}}$$




https://tdhock.github.io/SN-demo/

# Constrained problem, classic dynamic programming solution

**Constrained** optimal change-point problem:

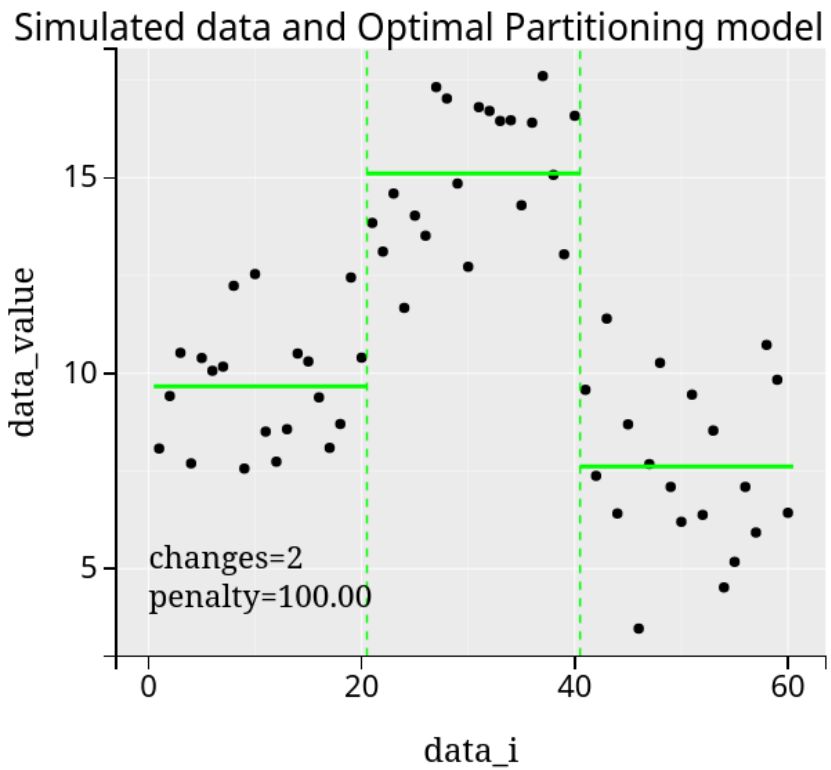
$$\min_{\mu\in\mathbb{R}^n} \underbrace{\sum_{i=1}^{n} \ell(\mu_i, z_i)}_{\text{Loss (data-fitting)}} \quad \text{subject to} \quad \underbrace{\sum_{i=2}^{n} I[\mu_{i-1} \neq \mu_i] \leq S.}_{\text{Number of change-points (regularization)}}$$

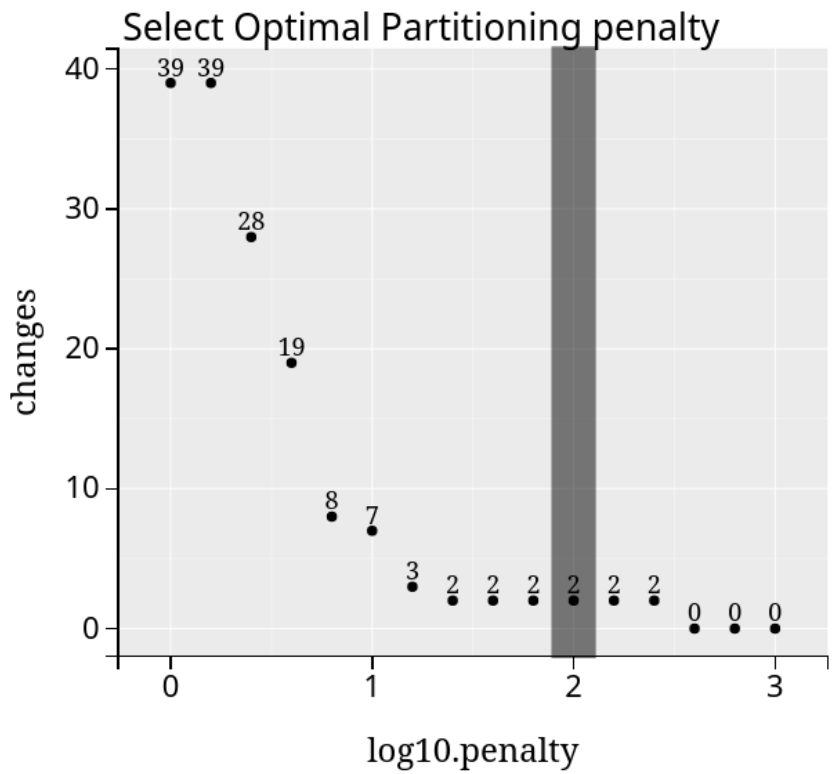
- Auger and Lawrence, Algorithms for the optimal identification of **segment neighborhoods**, Bull Math Biol (1989).
- Optimal recursive updates (dynamic programming algorithm).
- Let $C_{n,S}$ be best cost up to $n$ data and $S$ segments.
- Start by computing $C_{1,1}$ to $C_{1,n}$ (cum sum).
- Then compute $C_{2,2}$ to $C_{2,n}$, $C_{3,3}, \ldots, C_{3,n}$, etc.
- Output all optimal models from 1 to $S$ segments.
- Time complexity $O(Sn^2)$ — faster than naïve $O(n^S)$.
- Still too slow for large data $n$ and model sizes $S$.

# Penalized problem, classic dynamic programming solution

**Penalized** optimal change-point problem:

$$\min_{\mu \in \mathbb{R}^n} \underbrace{\sum_{i=1}^{n} \ell(\mu_i, z_i)}_{\text{Loss (data-fitting)}} + \underbrace{\lambda \sum_{i=2}^{n} I[\mu_{i-1} \neq \mu_i]}_{\text{Number of change-points (regularization)}}.$$




https://tdhock.github.io/OP-demo/

# Penalized problem, classic dynamic programming solution

**Penalized** optimal change-point problem:

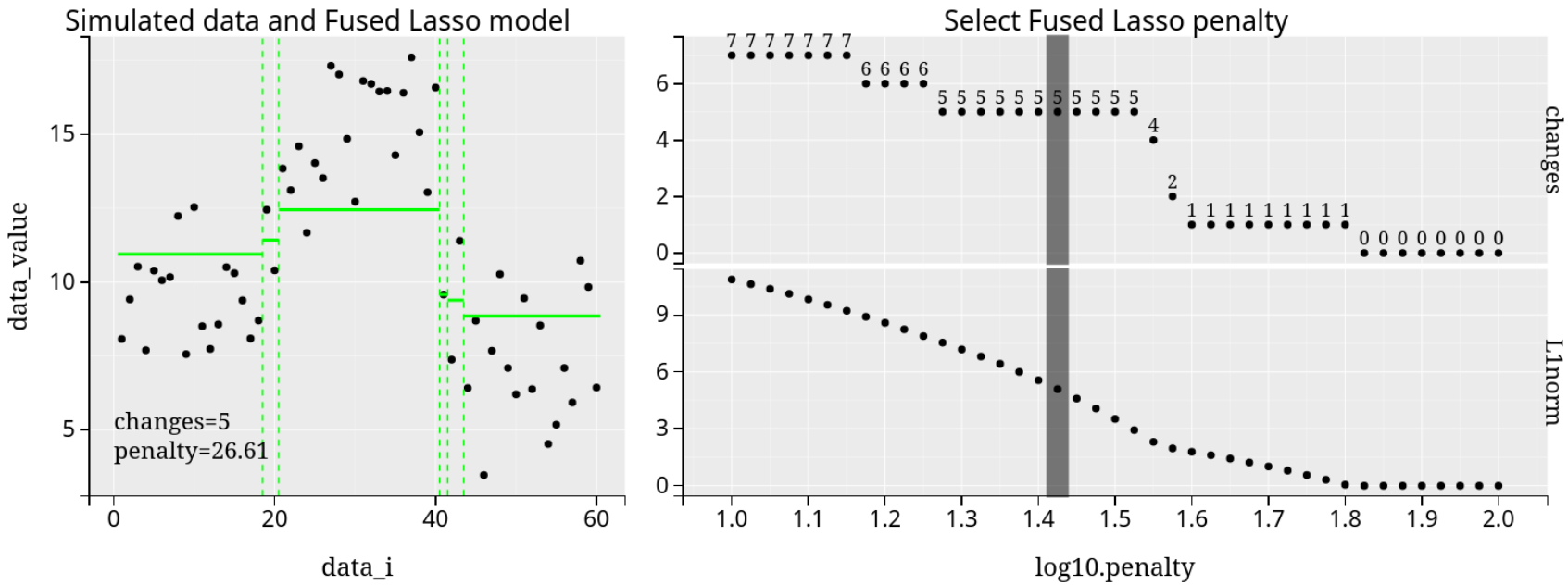
$$\min_{\mu\in\mathbb{R}^n} \underbrace{\sum_{i=1}^{n} \ell(\mu_i, z_i)}_{\text{Loss (data-fitting)}} + \underbrace{\lambda \sum_{i=2}^{n} I[\mu_{i-1} \neq \mu_i]}_{\text{Number of change-points (regularization)}}.$$

- Hyper-parameter is non-negative penalty $\lambda \geq 0$.
- Jackson, *et al.*, An algorithm for **optimal partitioning** of data on an interval, IEEE Sig Proc Lett (2005).
- Optimal recursive updates (dynamic programming algorithm).
- Let $C_n$ be best cost up to $n$ data.
- Recursively compute $C_1, C_2, \ldots, C_n$.
- Time complexity $O(n^2)$ — faster than $O(Sn^2)$.
- User can not directly specify number of segments $S$.
- Output one optimal change-point model (not all in $1, \ldots, S$).
- Quadratic time is still too slow for large data $n$.

# Penalized problem, Fused Lasso heuristic

**Penalized relaxed** change-point problem (fused lasso):

$$\min_{\mu \in \mathbb{R}^n} \underbrace{\sum_{i=1}^{n} \ell(\mu_i, z_i)}_{\text{Loss (data-fitting)}} + \underbrace{\lambda \sum_{i=2}^{n} |\mu_{i-1} - \mu_i|}_{\text{Absolute differences (regularization)}}$$




`https://tdhock.github.io/FL-demo/`

library(flsa) in R.

# Penalized problem, Fused Lasso heuristic

**Penalized relaxed** change-point problem (fused lasso):

$$\min_{\mu \in \mathbb{R}^n} \underbrace{\sum_{i=1}^{n} \ell(\mu_i, z_i)}_{\text{Loss (data-fitting)}} + \underbrace{\lambda \sum_{i=2}^{n} |\mu_{i-1} - \mu_i|}_{\text{Absolute differences (regularization)}}$$

- Hyper-parameter is non-negative penalty $\lambda \geq 0$.
- Indicator function $I[\cdot] \in \{0, 1\}$ is non-convex.
- Substitute absolute difference penalty: $\sum |\cdot|$.
- Fused Lasso for change-points (Tibshirani *et al.*, JRSSB 2005).
- Gradient-based linear time algorithm.
- Lasso selects too many variables (false positive change-points).

# Summary of classic algorithms

Some fast heuristics:

- Binary segmentation (Scott and Knott, 1974).
- Hidden Markov Models (Rabiner, 1989).
- Fused Lasso (Tibshirani, 2005).

Quadratic time optimal algorithms:

- Segment neighborhood (Auger and Lawrence, 1989).
- Optimal partitioning (Jackson, *et al.*, 2005).

1. Drawbacks of classic algorithms

2. Linear time optimal algorithms with pruning

3. Computing constrained models using penalized solver

Summary and Discussion

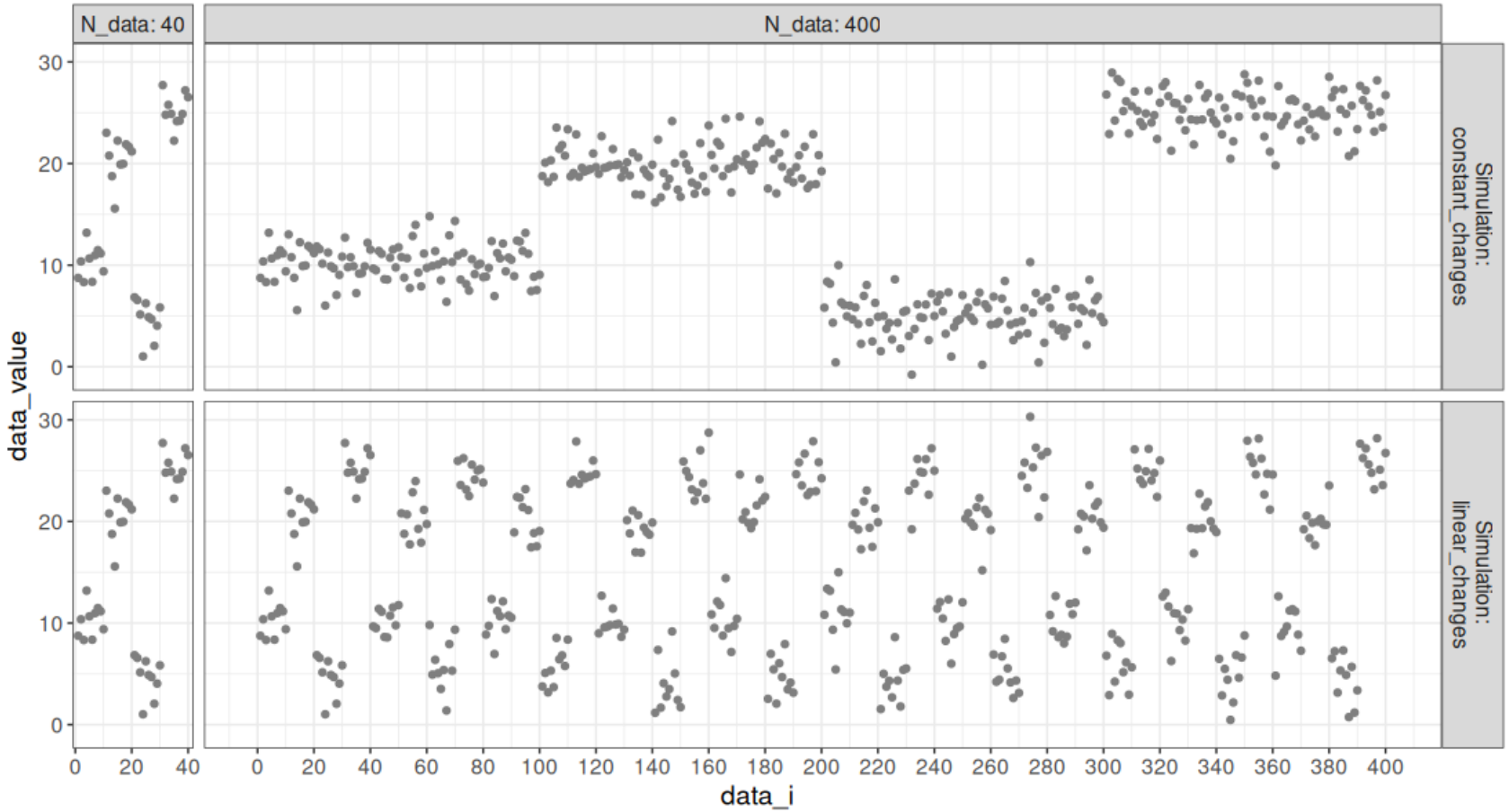
# Penalized problem, PELT algorithm

**Penalized** optimal change-point problem:

$$\min_{\mu \in \mathbb{R}^n} \underbrace{\sum_{i=1}^{n} \ell(\mu_i, z_i)}_{\text{Loss (data-fitting)}} + \underbrace{\lambda \sum_{i=2}^{n} I[\mu_{i-1} \neq \mu_i]}_{\text{Number of change-points (regularization)}}.$$

- Recursively compute $C_1, C_2, \ldots, C_n$.
- Classic DP considers all previous change-points.
- At time $n$, considers all of $C_1, \ldots, C_{n-1}$.
- Pruned Exact Linear Time algo: Killick, *et al.*, JASA (2012).
- At time $n$, TODO subset of candidates $C_1, \ldots, C_{n-1}$.
- Easy to implement, with only 1 additional line of code!
- Same output: one optimal change-point model.
- Time complexity: best $O(n)$, worst $O(n^2)$.
- library(changepoint) in R.

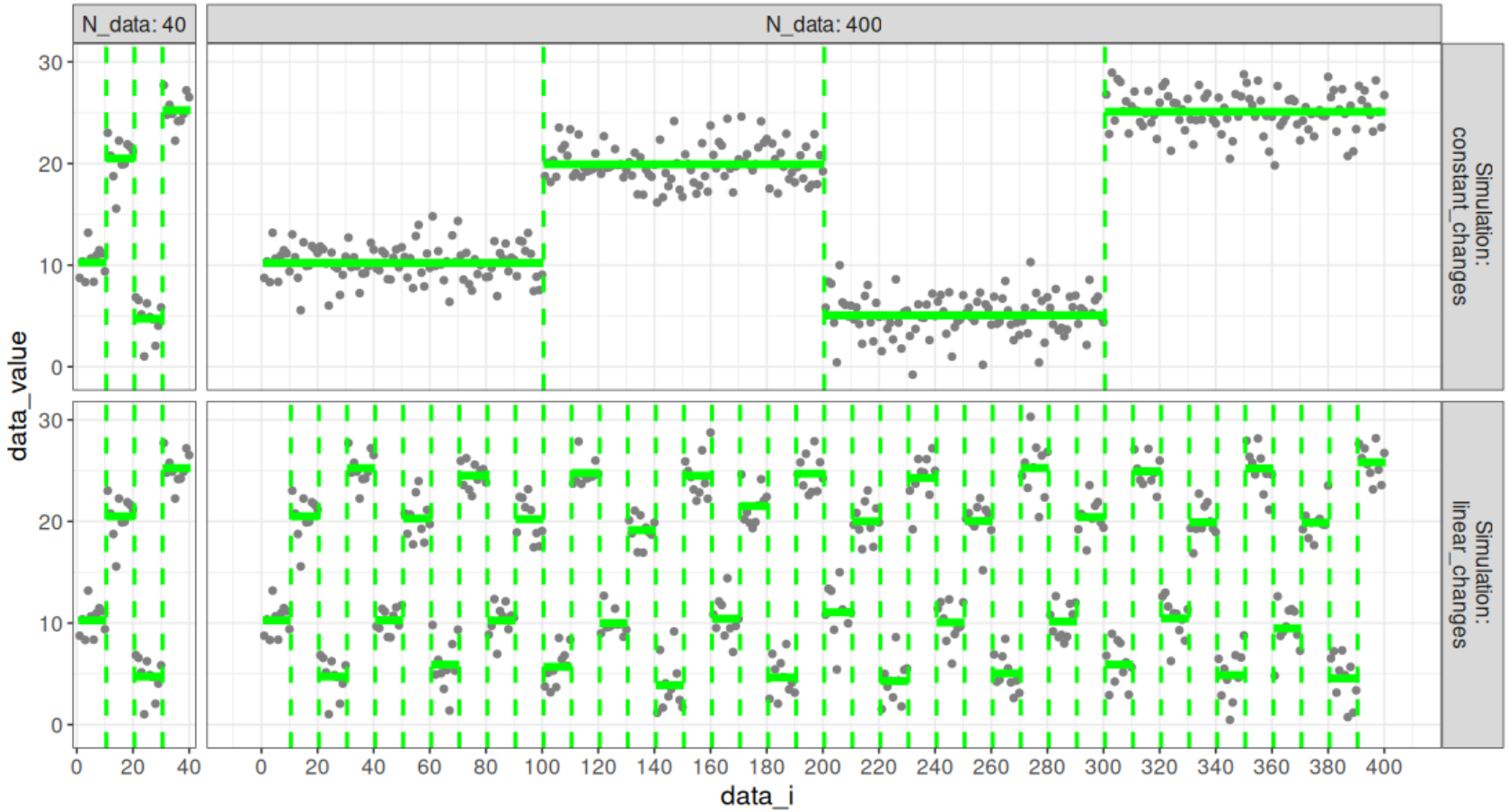
# Simulated data with constant and linear changes




- Constant changes: always 4 segments (for any data size $n$).
- Linear: change every 10 data.

`https://tdhock.github.io/blog/2025/PELT-vs-fpopw/`

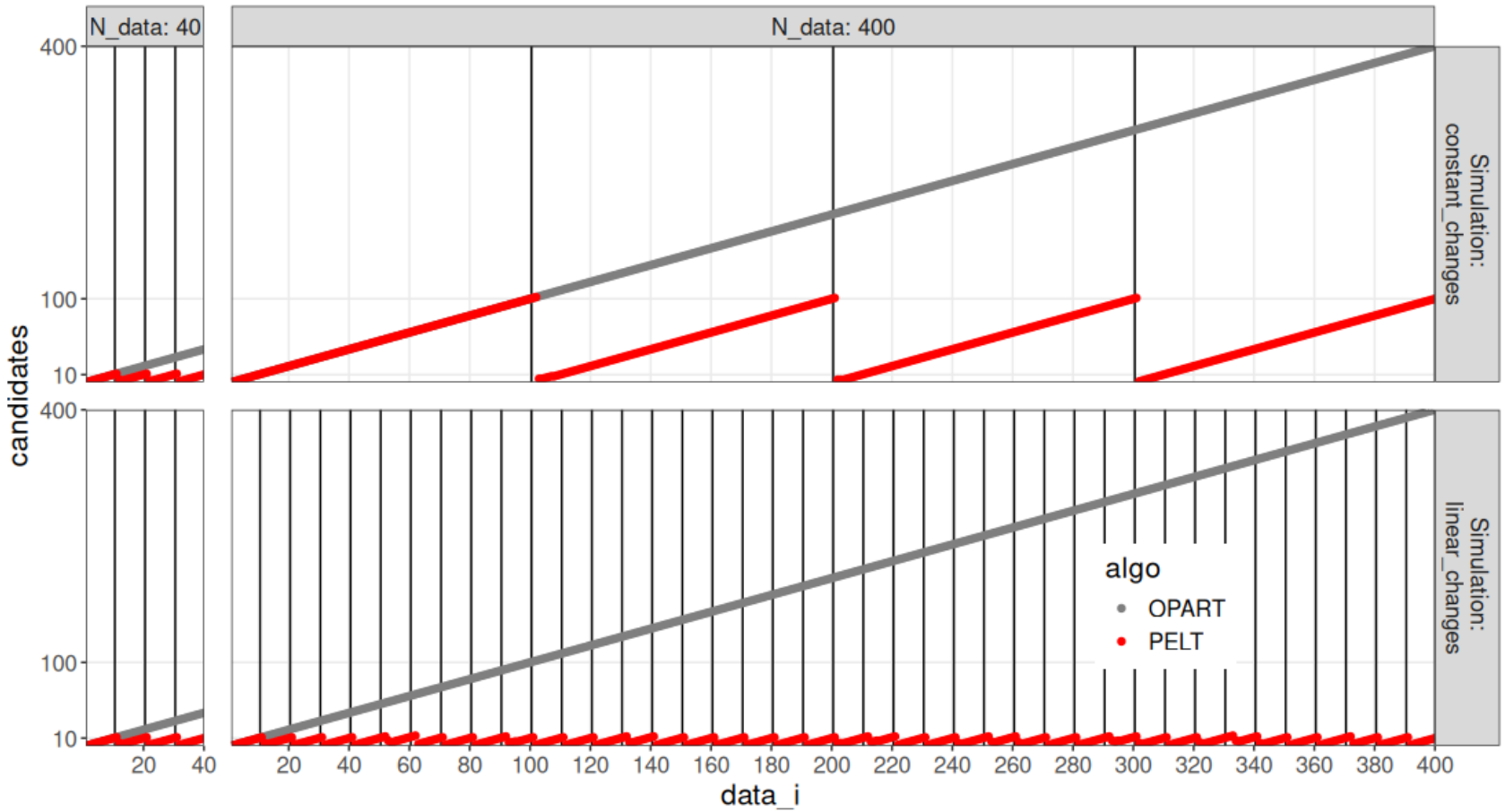
# Simulated data with constant and linear changes




- Constant changes: always 4 segments (for any data size $n$).
- Linear: change every 10 data.

`https://tdhock.github.io/blog/2025/PELT-vs-fpopw/`

# PELT algorithm demonstration




- After each change in data, PELT prunes prior candidates.
- More changes—more pruning—faster.

`https://tdhock.github.io/blog/2025/PELT-vs-fpopw/`

# Penalized problem, PELT algorithm

| # changes in data | Algorithm | # candidates per data | Overall time |
|---|---|---|---|
| Constant $O(1)$ | OPART | $O(n)$ | $O(n^2)$ |
| | PELT | $O(n)$ | $O(n^2)$ |
| Linear $O(n)$ | OPART | $O(n)$ | $O(n^2)$ |
| | PELT | $O(1)$ | $O(n)$ |

- PELT fast/linear for data with frequent changes.
- But slow/quadratic for long runs of data without changes.
- Can we go faster when there are no changes?

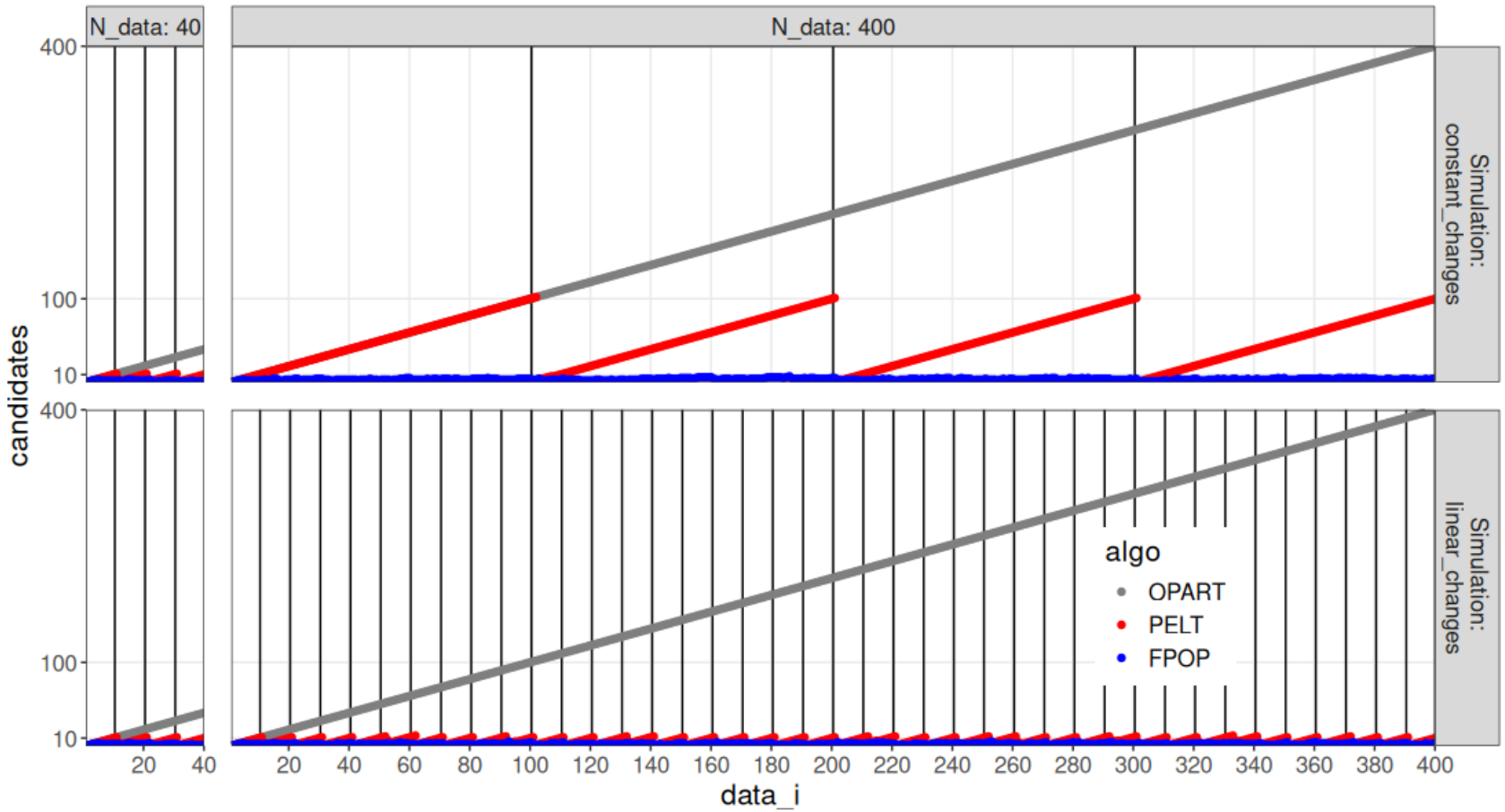
# Yes, faster with FPOP algorithm!

**Penalized** optimal change-point problem:

$$\min_{\mu \in \mathbb{R}^n} \underbrace{\sum_{i=1}^{n} \ell(\mu_i, z_i)}_{\text{Loss (data-fitting)}} + \underbrace{\lambda \sum_{i=2}^{n} I[\mu_{i-1} \neq \mu_i]}_{\text{Number of change-points (regularization)}}.$$

- FPOP algo: Maidstone, *et al.*, Stat. and Comp. (2017).
- Let $C_n(m)$ be the best cost with mean $m$ at $n$ data.
- Recursively compute functions $C_1(m), C_2(m), \ldots, C_n(m)$.
- **Functional pruning** considers a small subset of candidates.
- Same output: one optimal change-point model.
- Same time as PELT in theory: best $O(n)$, worst $O(n^2)$.
- But much faster in practice! (never quadratic)
- library(fpopw) in R.

# FPOP algorithm demonstration




- FPOP always prunes, whether or not there are changes in data.

`https://tdhock.github.io/blog/2025/PELT-vs-fpopw/`

# Penalized problem, FPOP algorithm

| # changes in data | Algorithm | # candidates per data | Overall time |
|---|---|---|---|
| Constant $O(1)$ | OPART | $O(n)$ | $O(n^2)$ |
| | PELT | $O(n)$ | $O(n^2)$ |
| | FPOP | $O(\log n)$ | $O(n \log n)$ |
| Linear $O(n)$ | OPART | $O(n)$ | $O(n^2)$ |
| | PELT | $O(1)$ | $O(n)$ |
| | FPOP | $O(1)$ | $O(n)$ |

- FPOP always fast in 1d data, no matter how many changes.
- Worst case $O(n^2)$ only happens in pathological data.

# Penalized problem, FPOP extensions

Functional pruning can handle:

- Robust loss functions: Fearnhead and Rigaill, JASA (2019).
- Auto-regressive models: Romano *et al.*, JASA (2021).
- Storage on disk: Hocking *et al.*, JSS (2022).
- Inequality/graph constraints: Runge *et al.*, JSS 2023.
- Multi-variate time series: Pishchagina *et al.*, Computo (2024).
- Multi-scale penalties: Liehrmann and Rigaill, JCGS (2025).

But:

- Difficult to implement (100s of lines of C++ code).
- Especially difficult for high dimensional data/models.

Is it possible to go faster?

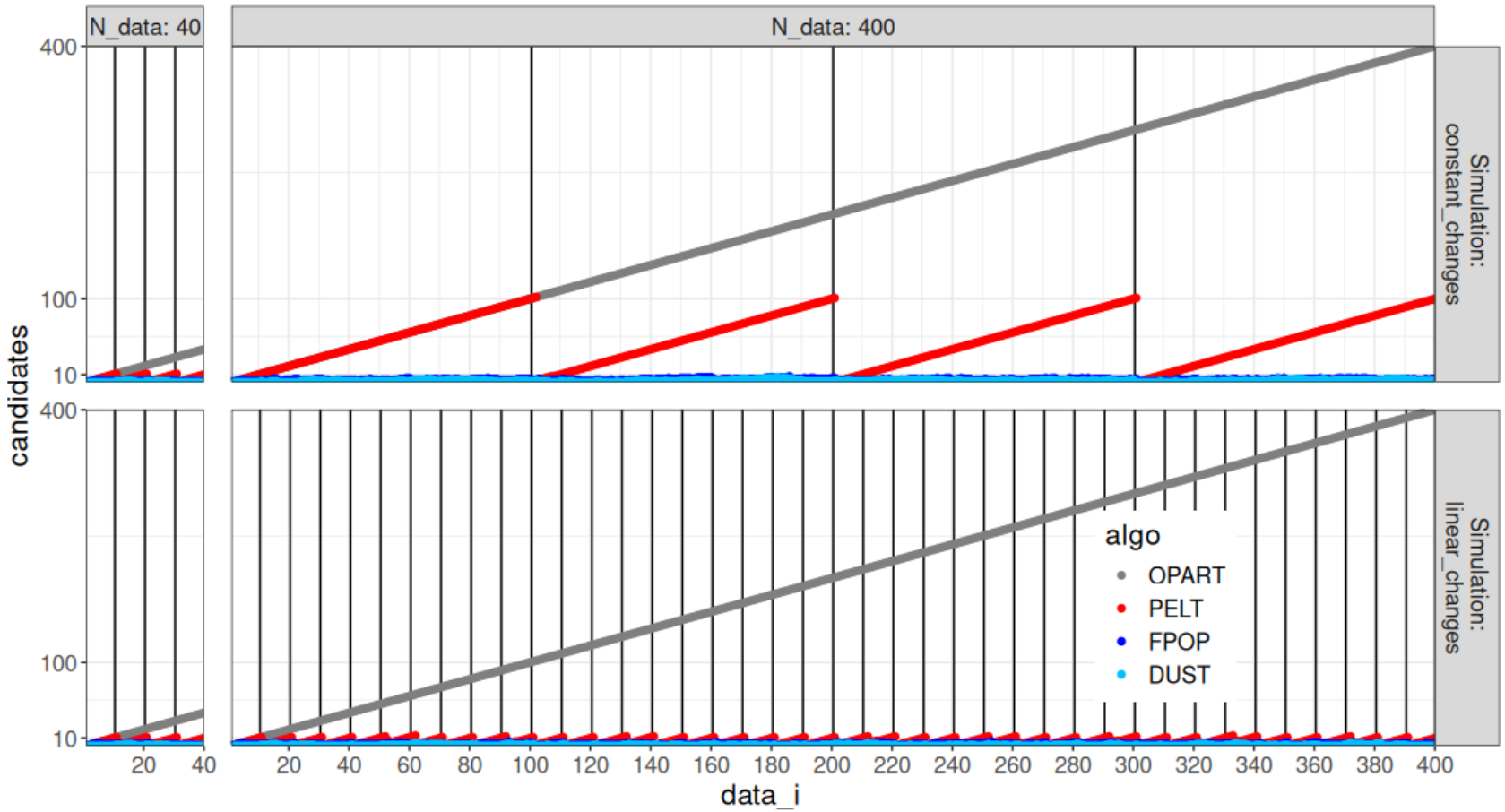
# Yes, faster with DUST algorithm!

**Penalized** optimal change-point problem:

$$\min_{\mu \in \mathbb{R}^n} \underbrace{\sum_{i=1}^{n} \ell(\mu_i, z_i)}_{\text{Loss (data-fitting)}} + \underbrace{\lambda \sum_{i=2}^{n} I[\mu_{i-1} \neq \mu_i]}_{\text{Number of change-points (regularization)}}.$$

- Truong and Runge, An Efficient Algorithm for Exact Segmentation of Large Compositional and Categorical Time Series, Stat (2024).
- DUST algo: DUality Simple Test.
- Combines ideas from PELT and FPOP algos.
- Solve a Langrange dual problem to prune change-points.
- Easier to code than FPOP.
- Same time as FPOP in theory: best $O(n)$, worst $O(n^2)$.
- But much faster in practice! (constant factors)
- library(dust) in R.

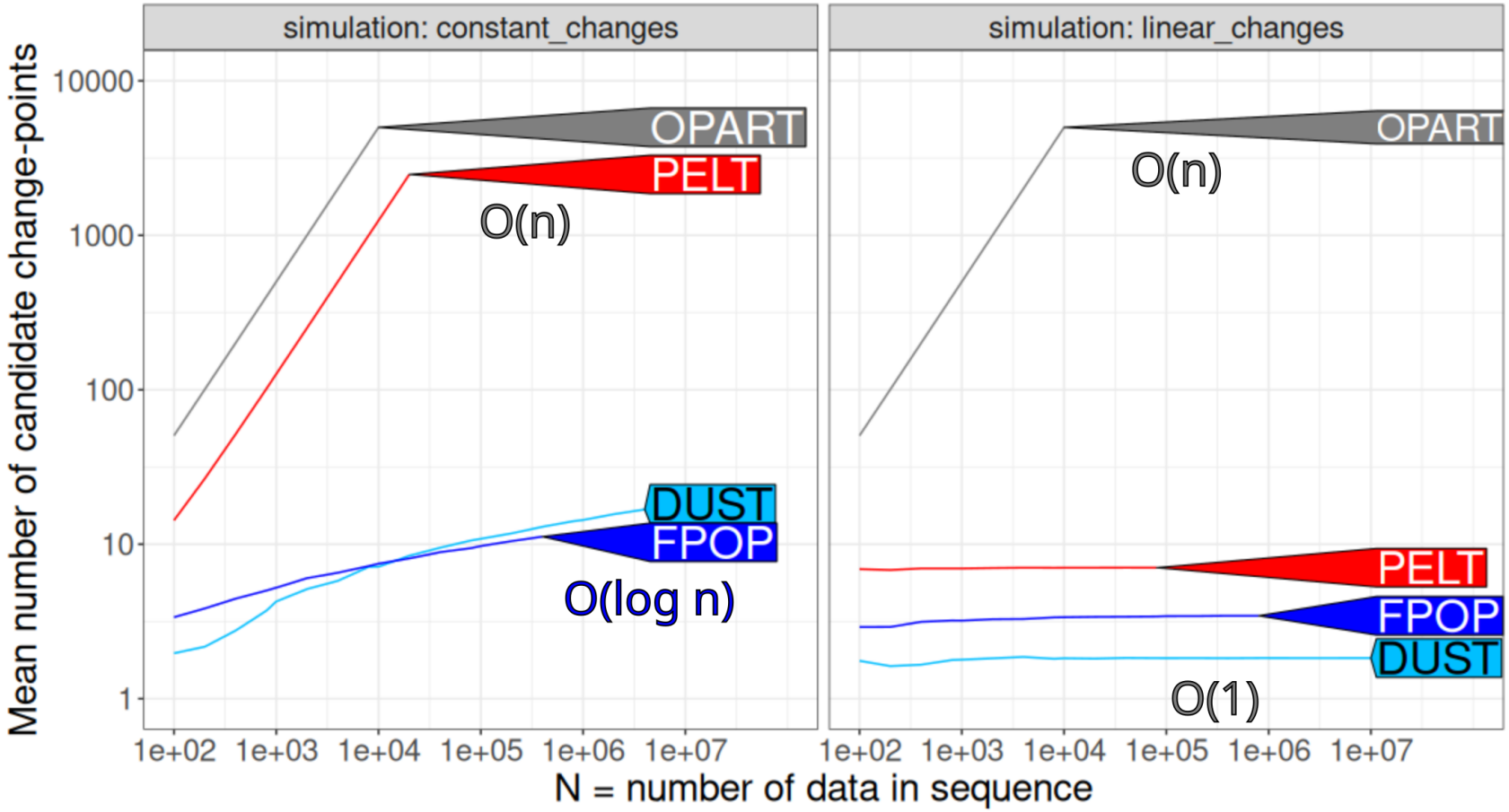
# DUST algorithm demonstration




- DUST always prunes, whether or not there are changes in data.

`https://tdhock.github.io/blog/2025/PELT-vs-fpopw/`

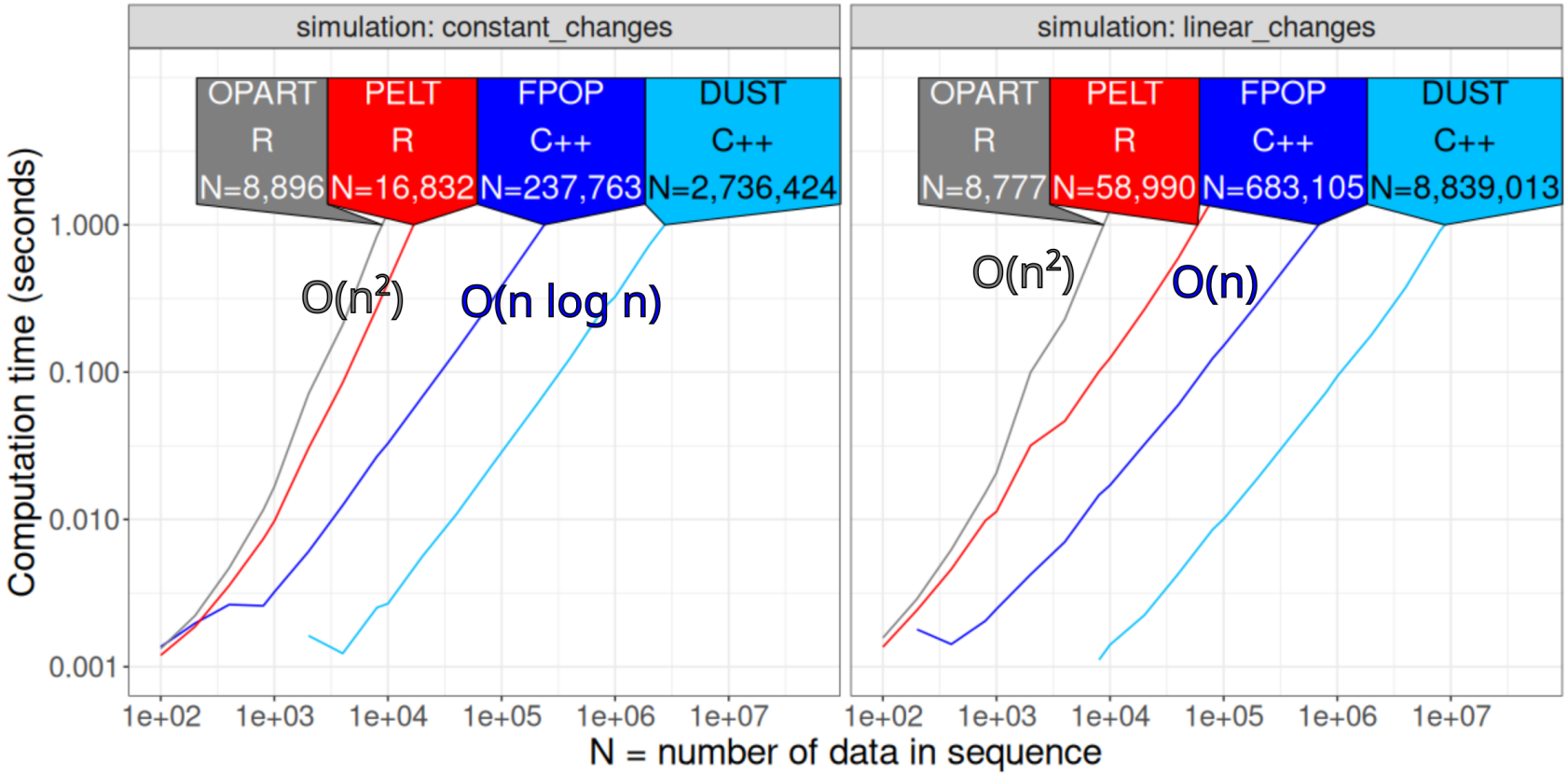
# Number of candidates considered




- Slope on log-log plot indicates asymptotic complexity class.

`https://tdhock.github.io/blog/2025/PELT-vs-fpopw/`

# Overall computation time




- N= data size computable in time limit of 1 second.
- FPOP 10x faster than PELT.
- DUST 10x faster than FPOP.
- library(atime) in R for asymptotic analysis.

`https://tdhock.github.io/blog/2025/PELT-vs-fpopw/`

# Penalized problem, DUST algorithm

**Penalized** optimal change-point problem:

$$\min_{\mu \in \mathbb{R}^n} \underbrace{\sum_{i=1}^{n} \ell(\mu_i, z_i)}_{\text{Loss (data-fitting)}} + \underbrace{\lambda \sum_{i=2}^{n} I[\mu_{i-1} \neq \mu_i]}_{\text{Number of change-points (regularization)}}.$$

| # changes in data | Algorithm | # candidates per data | Overall time |
|---|---|---|---|
| Constant $O(1)$ | OPART | $O(n)$ | $O(n^2)$ |
| | PELT | $O(n)$ | $O(n^2)$ |
| | FPOP | $O(\log n)$ | $O(n \log n)$ |
| | DUST | $O(\log n)$ | $O(n \log n)$ |
| Linear $O(n)$ | OPART | $O(n)$ | $O(n^2)$ |
| | PELT | $O(1)$ | $O(n)$ |
| | FPOP | $O(1)$ | $O(n)$ |
| | DUST | $O(1)$ | $O(n)$ |

- DUST always fast in 1d data, no matter how many changes.
- Is it possible to go faster and stay optimal?

# Penalized problem, DUST algorithm

**Penalized** optimal change-point problem:

$$\min_{\mu \in \mathbb{R}^n} \underbrace{\sum_{i=1}^{n} \ell(\mu_i, z_i)}_{\text{Loss (data-fitting)}} + \underbrace{\lambda \sum_{i=2}^{n} I[\mu_{i-1} \neq \mu_i]}_{\text{Number of change-points (regularization)}}.$$

| # changes in data | Algorithm | # candidates per data | Overall time |
|---|---|---|---|
| Constant $O(1)$ | OPART | $O(n)$ | $O(n^2)$ |
| | PELT | $O(n)$ | $O(n^2)$ |
| | FPOP | $O(\log n)$ | $O(n \log n)$ |
| | DUST | $O(\log n)$ | $O(n \log n)$ |
| Linear $O(n)$ | OPART | $O(n)$ | $O(n^2)$ |
| | PELT | $O(1)$ | $O(n)$ |
| | FPOP | $O(1)$ | $O(n)$ |
| | DUST | $O(1)$ | $O(n)$ |

- DUST always fast in 1d data, no matter how many changes.
- Is it possible to go faster and stay optimal?
- Conjecture: only by constant factors.

1. Drawbacks of classic algorithms

2. Linear time optimal algorithms with pruning

3. Computing constrained models using penalized solver

Summary and Discussion

# Computing one constrained model using penalized solver

How to compute the best model with $P^*$ change-points?

- Example: want $P^* = 100$ change-points.

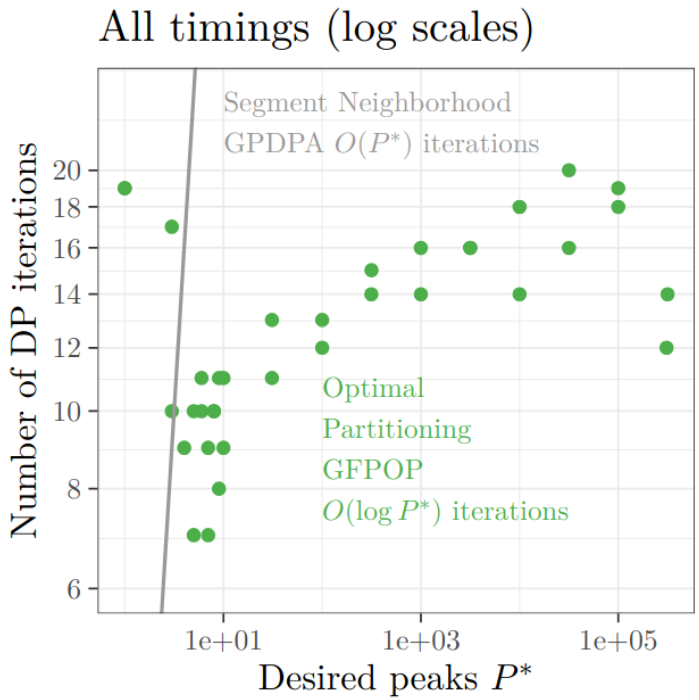
Pruned dynamic programming algo: Rigaill, JSFDS (2015).

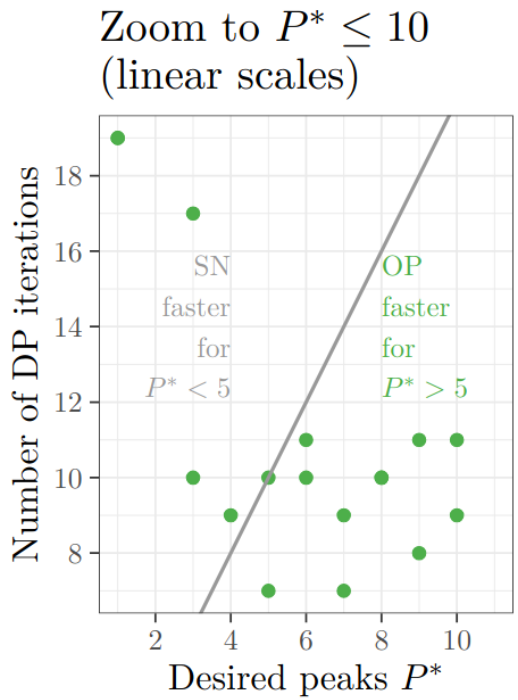
- Run constrained solver to get all model sizes from 0 to 100.
- $O(P^* n \log n)$ time—fast in $n$.
- Slow in $P^*$: we do not want model sizes 0 to 99.
- library(fpopw) in R.

Sequential search algo: Hocking *et al.*, Journal of Statistical Software 101(10) (2022).

- Penalized solver returns best change-points for a given penalty $\lambda \geq 0$ (but $\lambda$ that yields $P^*$ is unknown).
- Run penalized solver for an special sequence of $\lambda$ values.
- $O(\log P^* n \log n)$ time—fast in $n$ and $P^*$! (empirically)
- library(PeakSegDisk) in R.

# Penalized (OP) is faster than constrained (SN)




- Figure: genomic data, $N \approx 10^6$,
- Sequential search repeated runs penalized (OP) solver.
- Example: for $N = 10^7$, desired change-points $P^* = 3000$.
- Constrained (SN) solver: ≈100TB storage, 10 weeks.
- Penalized (OP) solver: 100GB storage, 10 hours.

Hocking *et al.*, Journal of Statistical Software 101(10) (2022).

# Fast computation of large model sizes and ranges

How to compute all models with penalties $\lambda \in [\underline{\lambda}, \overline{\lambda}]$?

- Example: want all models with $\lambda \in [0.1, \overline{1}0.5]$.
- CROPS: Change-points for a Range Of PenaltieS.
- Haynes, *et al.* Journal of Computational and Graphical Statistics, 26(1), 134-143 (2017).
- library(changepoint) in R.

How to compute all models with number of change-points $P \in [\underline{P}, \overline{P}]$?

- Example: want all models from 50 to 60 change-points.
- CROCS: Change-points for a Range Of ComplexitieS.
- Liehrmann *et al.*, BMC Bioinformatics 22(323) (2021).
- library(CROCS) in R.

Both exploit the piecewise linear solution path to obtain an efficient algorithm, linear time in the number of models!

1. Drawbacks of classic algorithms

2. Linear time optimal algorithms with pruning

3. Computing constrained models using penalized solver

Summary and Discussion

# Summary and Discussion

- Optimal changepoint detection in $n$ data is a non-convex problem, naïvely a $O(n^S)$ computation for $S$ segments.
- Pruning algorithms compute a globally optimal change-point model much faster, $O(n \log n)$ or even $O(n)$.

Future work:

- Fast pruning algorithms for piecewise loss functions (L1 etc).
- Proving best/worst case time complexity of sequential search algorithms?

Contact: toby.dylan.hocking@usherbrooke.ca

# References


- Auger IE and Lawrence CE. Algorithms for the optimal identification of segment neighborhoods. Bull Math Biol 51:39–54 (1989).
- G Rigaill. A pruned dynamic programming algorithm to recover the best segmentations with 1 to kmax change-points. Journal de la Société Française de la Statistique, 156(4), 2015.
- **Hocking TD**, Boeva V, Rigaill G, Schleiermacher G, Janoueix-Lerosey I, Delattre O, Richer W, Bourdeaut F, Suguro M, Seto M, Bach F, Vert J-P. SegAnnDB: interactive Web-based genomic segmentation. Bioinformatics (2014) 30 (11): 1539-1546.
- **Hocking TD**, Goerner-Potvin P, Morin A, Shao X, Pastinen T, Bourque G. Optimizing ChIP-seq peak detectors using visual labels and supervised machine learning. Bioinformatics (2017) 33 (4): 491-499.
- Jewell S, **Hocking TD**, Fearnhead P, Witten D. Fast Nonconvex Deconvolution of Calcium Imaging Data. Biostatistics (2019).
- Fotoohinasab A, **Hocking TD**, Afghah F. A Graph-Constrained Changepoint Learning Approach for Automatic QRS-Complex Detection. Asilomar Conference on Signals, Systems, and Computers (2020).
- **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.
- Runge V, **Hocking TD**, Romano G, Afghah F, Fearnhead P, Rigaill G. gfpop: an R Package for Univariate Graph-Constrained Change-point Detection. Journal of Statistical Software 106(6) (2023).

# References

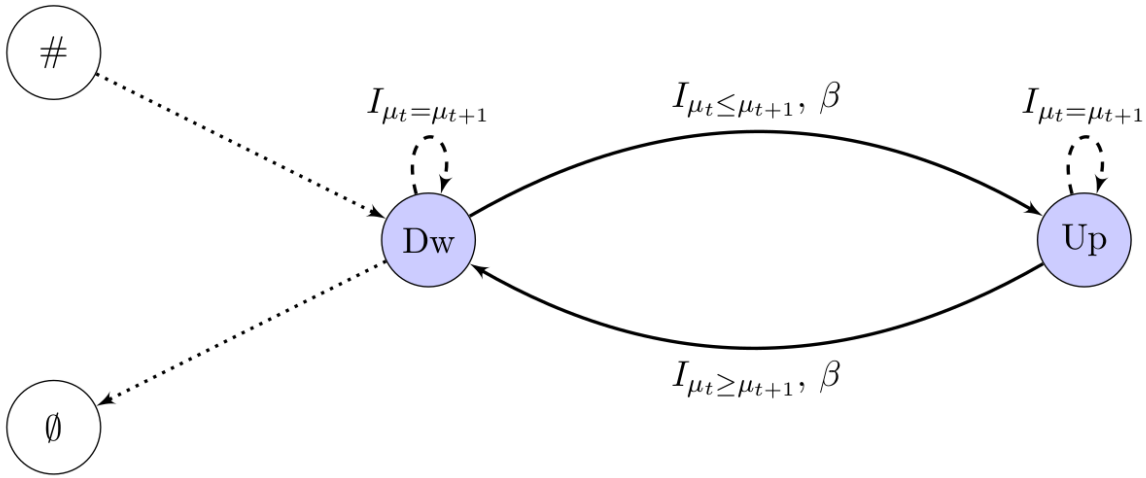

- **Hocking TD**, Rigaill G, Bach F, Vert J-P. Learning sparse penalties for change-point detection using max-margin interval regression. International Conference on Machine Learning 2013.
- Drouin A, **Hocking TD**, Laviolette F. Maximum margin interval trees. Neural Information Processing Systems 2017.
- Barnwal A, Cho H, **Hocking TD**. Survival regression with accelerated failure time model in XGBoost. Journal of Computational and Graphical Statistics 31(4) (2022).
- Hillman J, **Hocking TD**. Optimizing ROC Curves with a Sort-Based Surrogate Loss Function for Binary Classification and Changepoint Detection. Journal of Machine Learning Research 24(70) (2023).

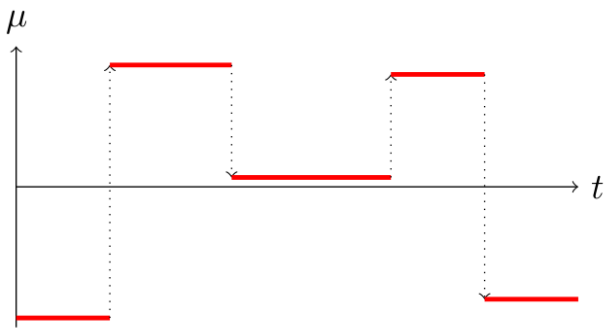
# References


- **Hocking TD**, Rigaill G. SegAnnot: an R package for fast segmentation of annotated piecewise constant signals, Pre-print hal-00759129.
- **Hocking TD**, Srivastava A. Labeled Optimal Partitioning. Computational Statistics 38 (2023).
- Fotoohinasab A, **Hocking TD**, Afghah F. A Greedy Graph Search Algorithm Based on Changepoint Analysis for Automatic QRS-Complex Detection. Computers in Biology and Medicine 130 (2021).

# Optimization constraints defined using a graph

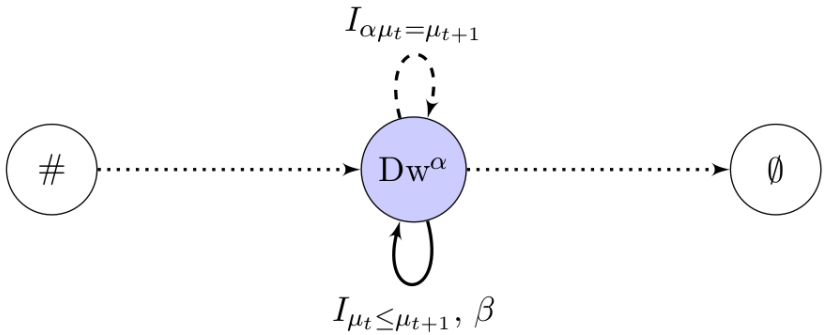
Runge *et al.*, Journal of Statistical Software 2023.

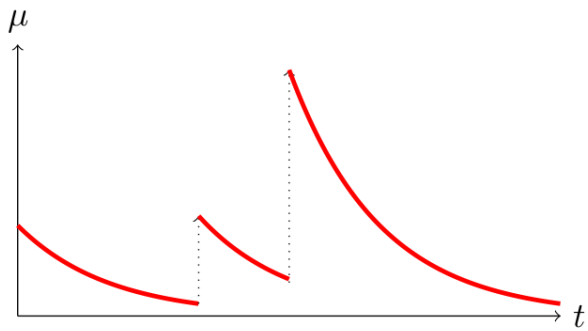


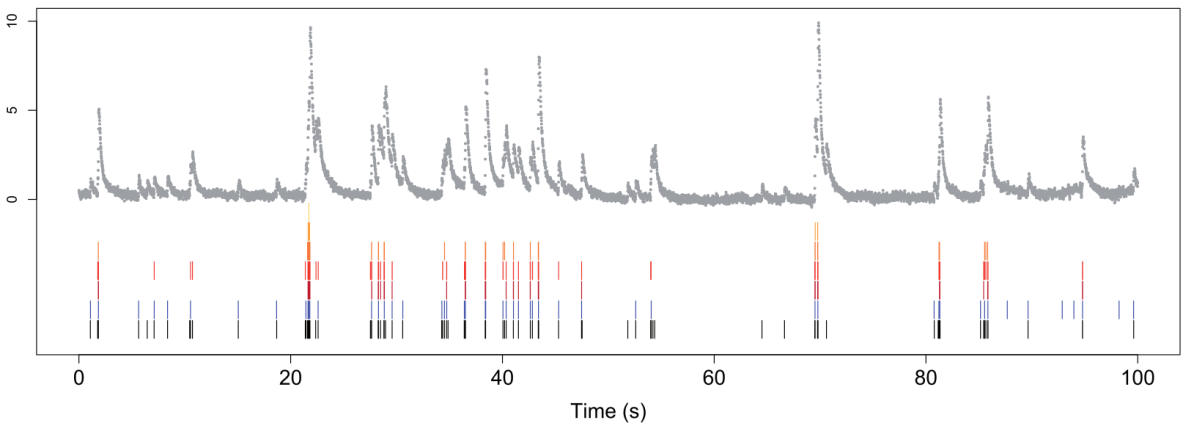

- Purple Dw/Up nodes represent hidden states.
- #/∅ nodes constrain start/end state.
- Edges represent possible state transitions.
- library(gfpop) in R computes optimal change-points for user-defined constraint graphs.

# All up changes, exponential decaying segments



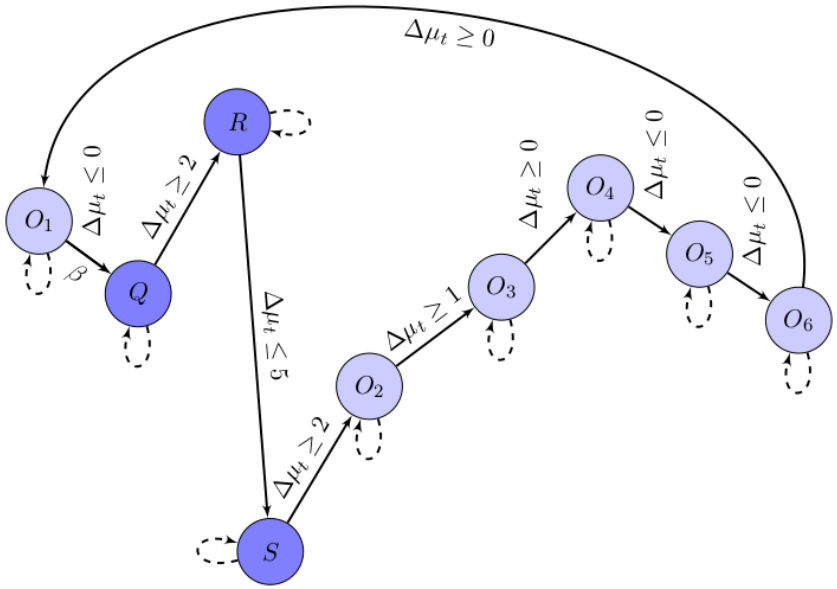



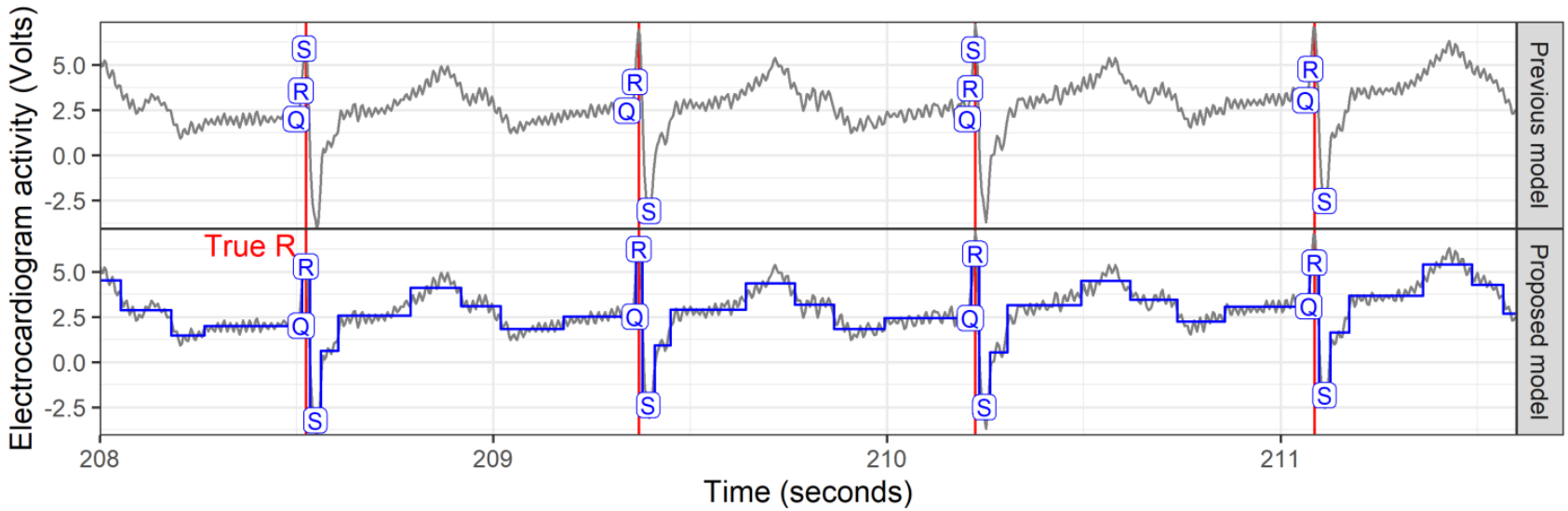

Jewell *et al.*, Biostatistics 2019.




library(FastLZeroSpikeInference) in R.

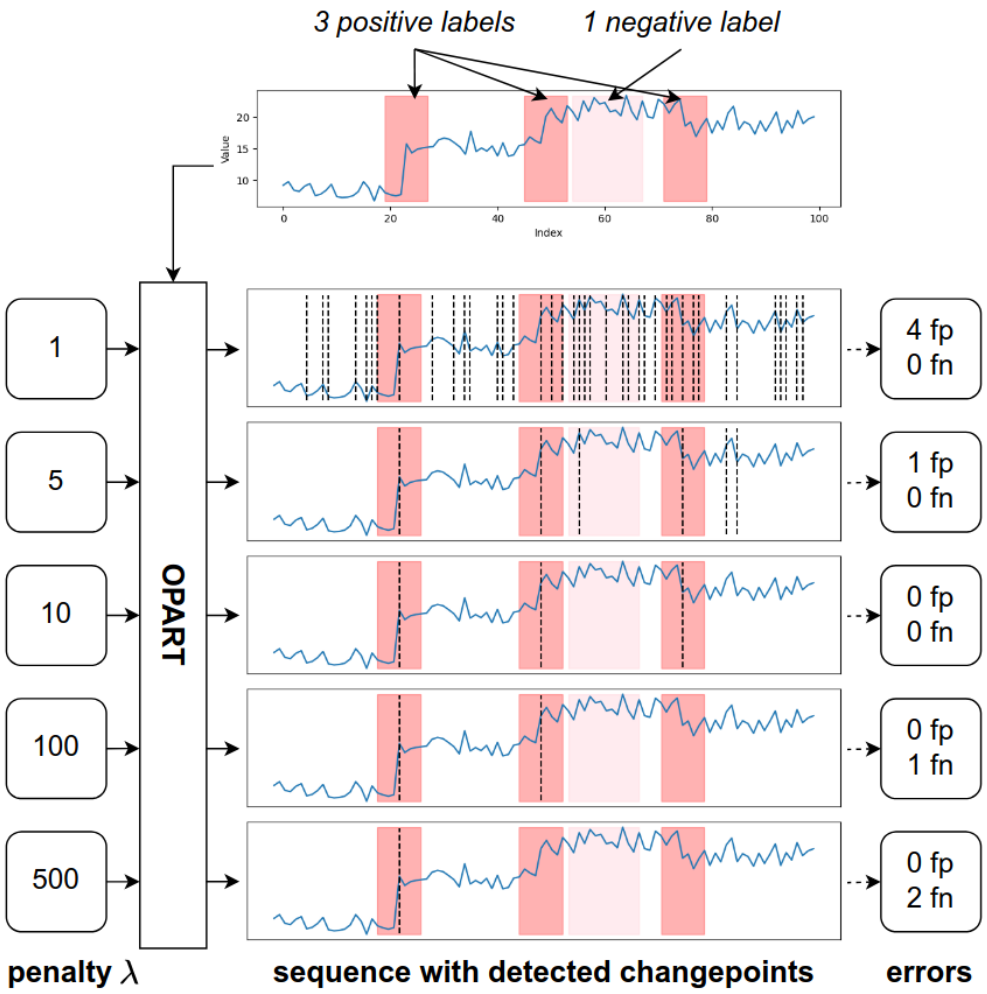
# Complex graph for electrocardiogram data




Fotoohinasab *et al.*, Asilomar conference 2020.

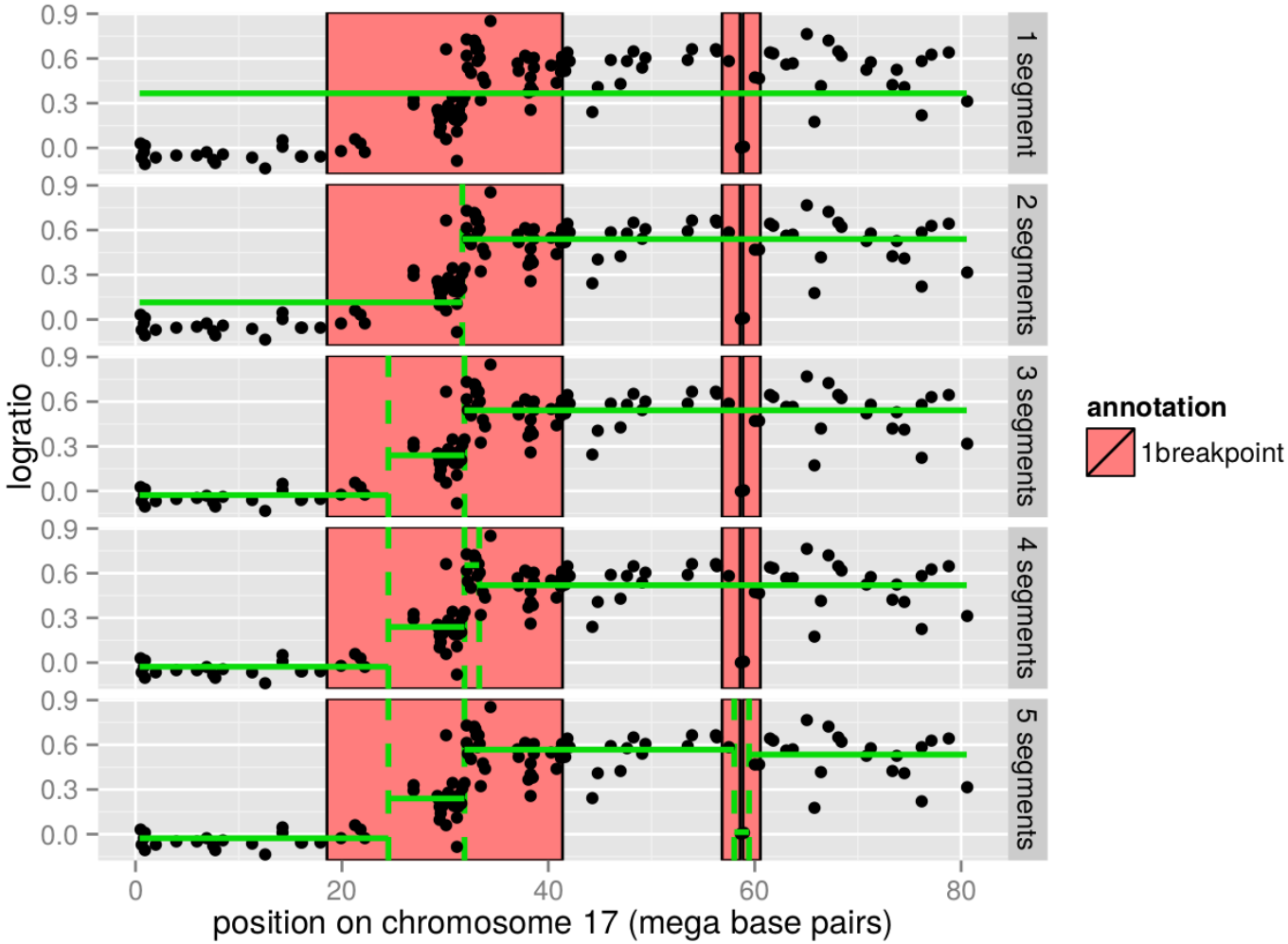
# Fitting change-points to labels from an expert




- Figure: Nguyen and Hocking, arXiv:2505.07413.
- fp=false positive (too many changes).
- fn=false negative (too few changes).
- Try different $\lambda$ values until we find a model with no errors.

# What if no models agree with expert labels?

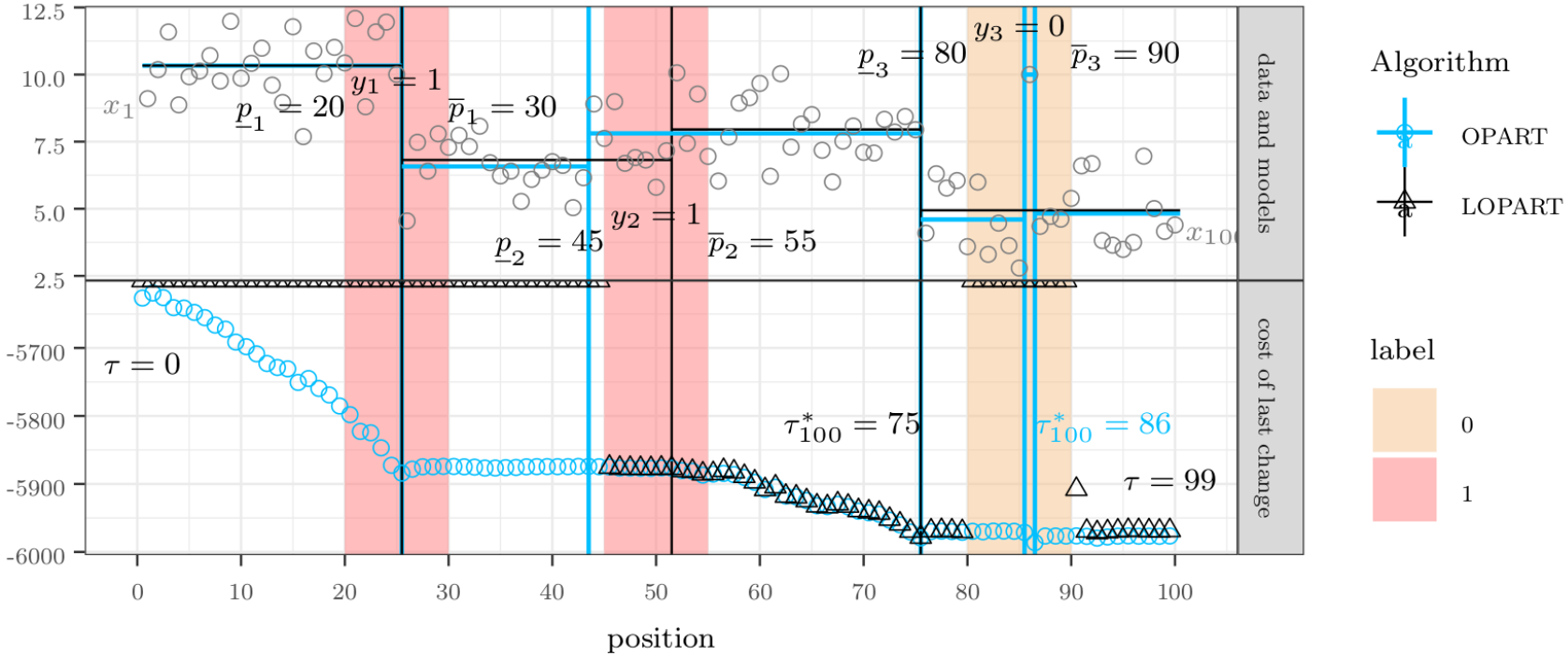
Hocking and Rigaill, Pre-print hal-00759129.




- Expert wants: one changepoint in each label (red rectangle).
- No model is consistent with all three labels.

# Using expert labels as optimization constraints

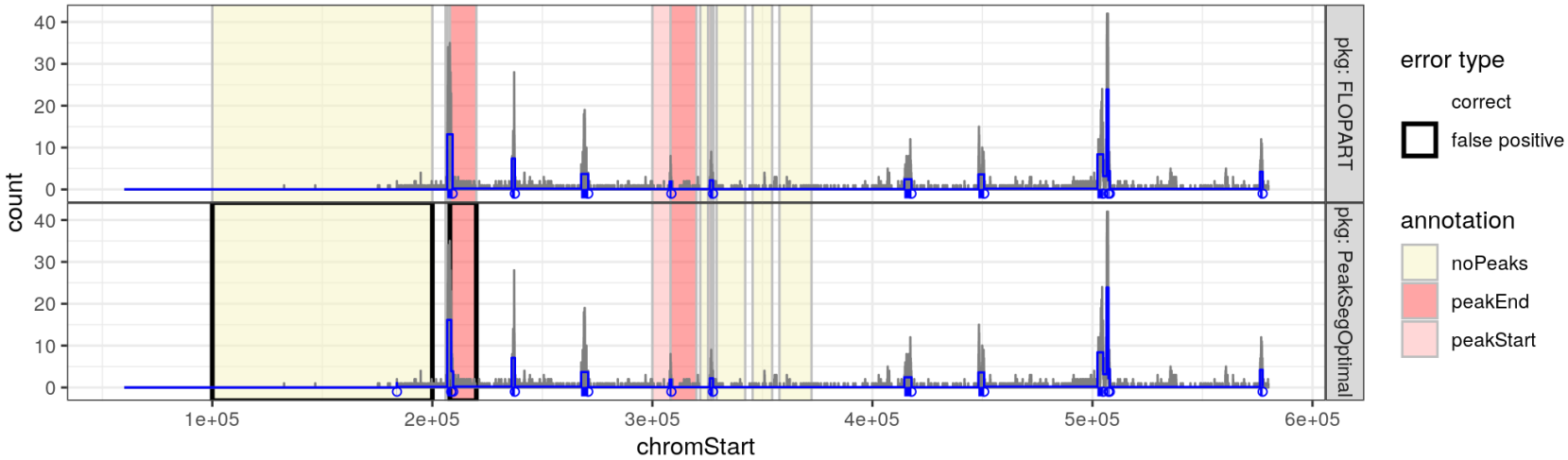
Hocking and Srivastava, Computational Statistics 38 (2023).




- Previous OPART model (blue) ignores labels (two errors).
- Main idea: add optimization constraints to ensure that there is the right number of changepoints predicted in each label.
- Proposed LOPART model (black) consistent with labels.
- library(LOPART) in R.

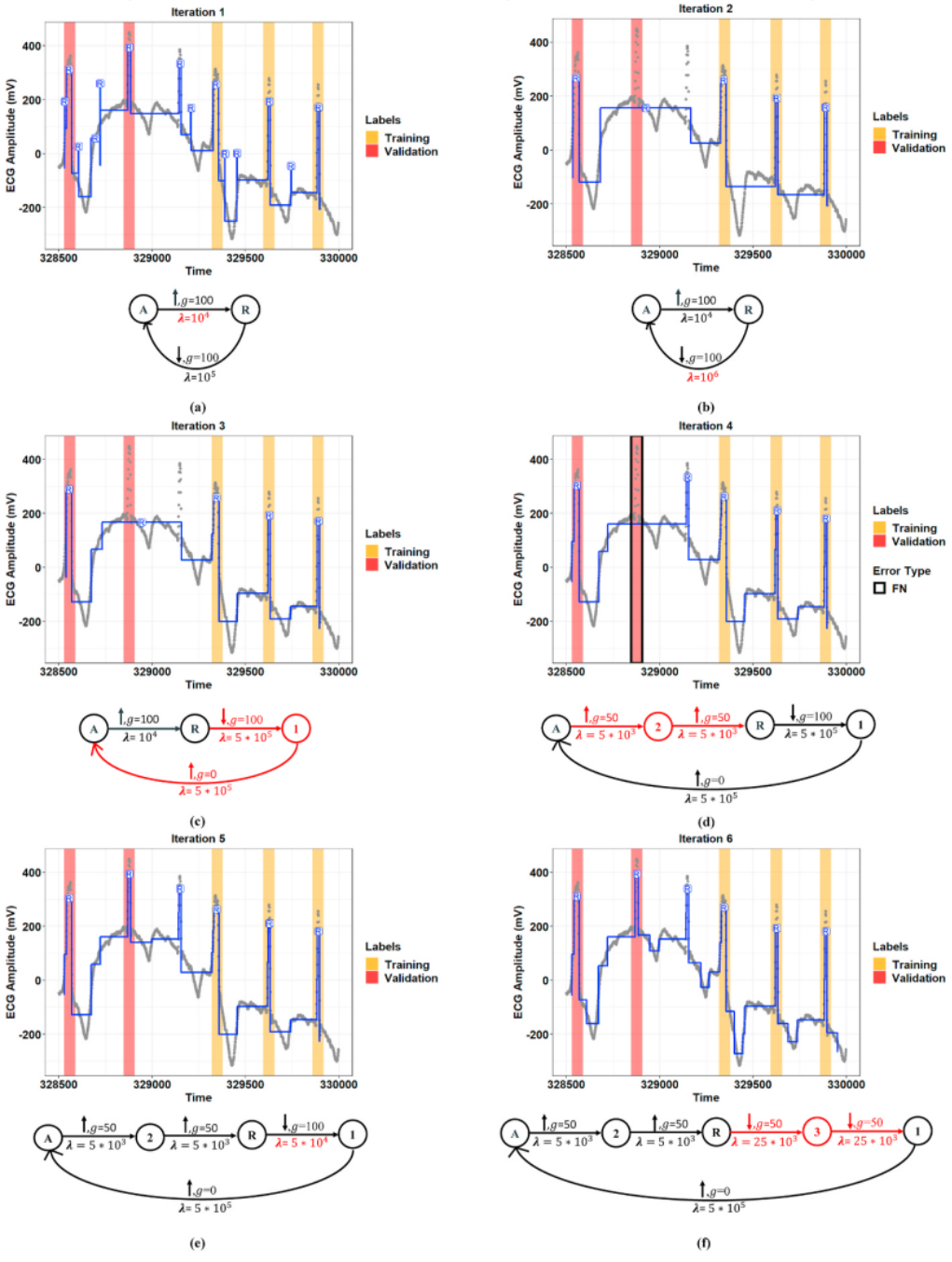
# Label constraints and directional constraints

Kaufman *et al.*, Journal of Computational and Graphical Statistics 33(4) (2024).




- Previous PeakSegOptimal algorithm (bottom) ignores labels (two errors).
- Proposed FLOPART model (top) consistent with labels, and interpretable in terms of up changes to peaks and down changes to background noise.
- library(FLOPART) in R.

# Learning a complex graph using labeled regions




- Fotoohinasab *et al.*, 2021.
- Simple initial graph is iteratively edited (red) to agree with expert labeled regions (orange rectangles).
- Easier for expert to provide labels than graph.

# How to predict the number of changes?

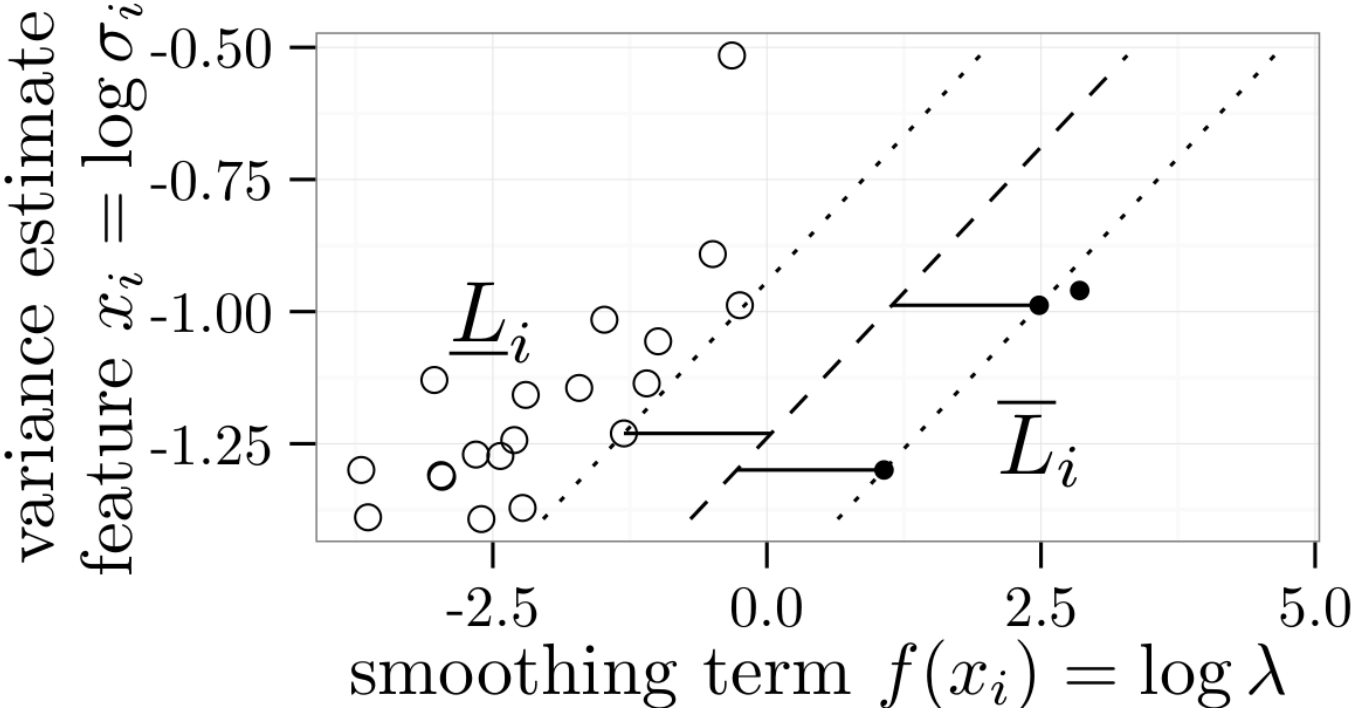
TODO figure.

In practice the penalty $\lambda$ is unknown — what value should we use?

- $z = z_1, \ldots, z_n$ is the data sequence to segment.
- Let $x = \phi(z)$ be a feature vector (fixed, not learned).
- Learn $f(x) = \log \lambda$, predicted penalty.
- Goal: minimize error with respect to labeled regions in test set.

# Learning to predict number of changes similar to SVM

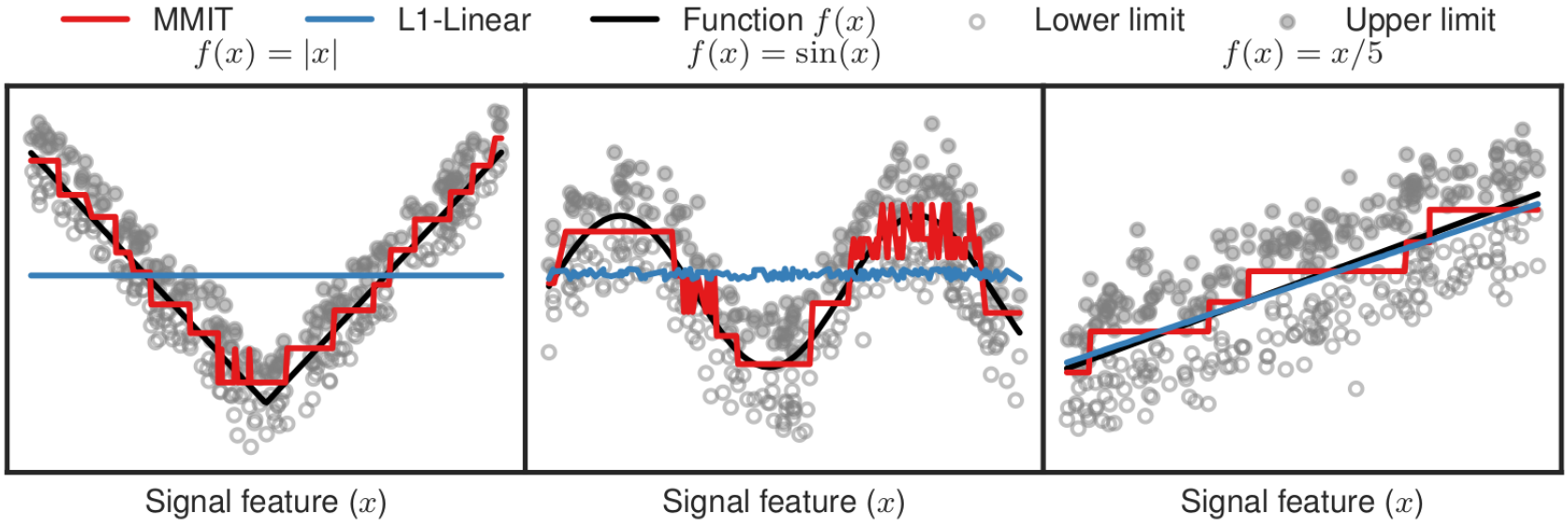
Hocking *et al.*, Int'l Conference on Machine Learning 2013.




- Train on several data sequences with labels (dots).
- Want to compute function between white and black dots.
- SVM margin is multi-dimensional (diagonal).
- Here margin to maximize is one-dimensional (horizontal).
- L1 regularized linear model, library(penaltyLearning) in R.

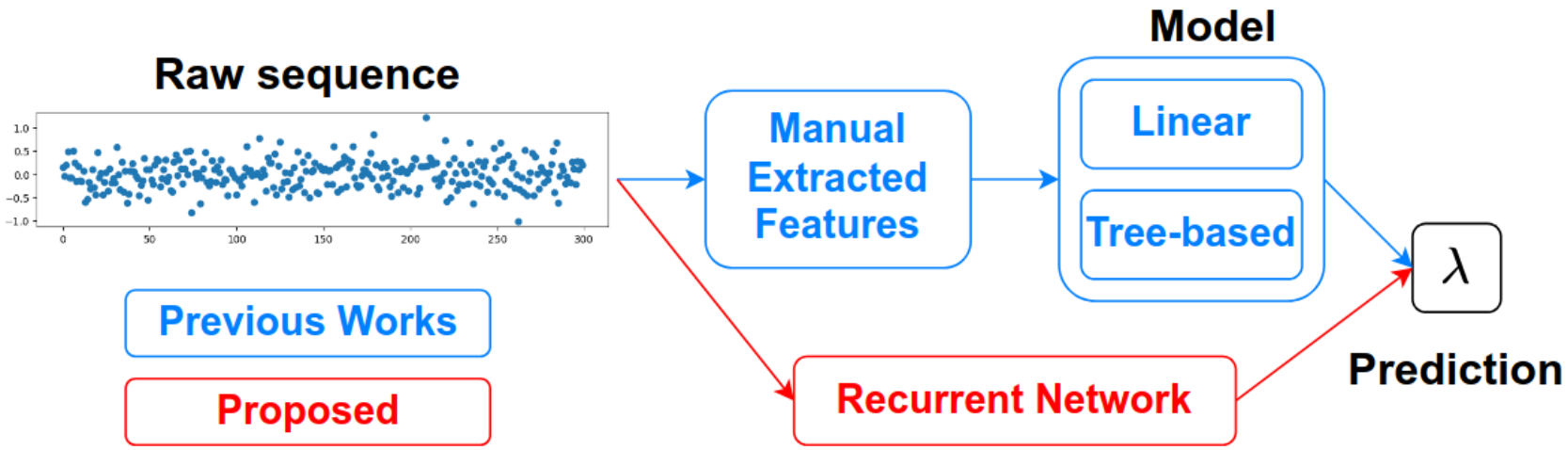
# Decision tree learns non-linear function of inputs

Drouin *et al.*, Neural Information Processing Systems 2017.




- Generalization of classical CART regression tree learning algorithm.
- Can learn non-linear functions of inputs.
- More recently we implemented a similar idea in xgboost, Barnwal *et al.*, Journal of Computational and Graphical Statistics 31(4) (2022).
- library(mmit) and library(xgboost) in R.

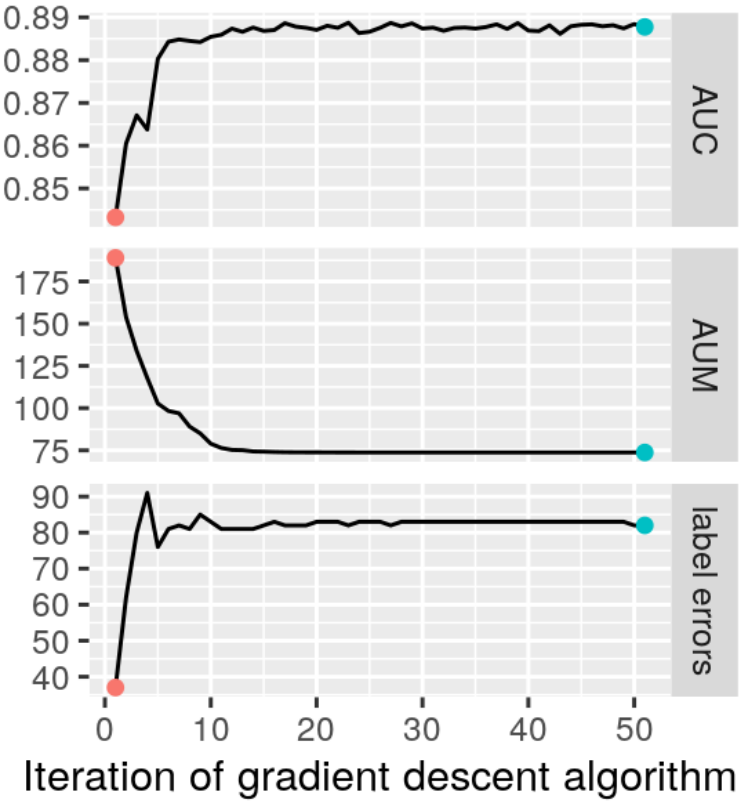
# Penalty prediction using recurrent neural network

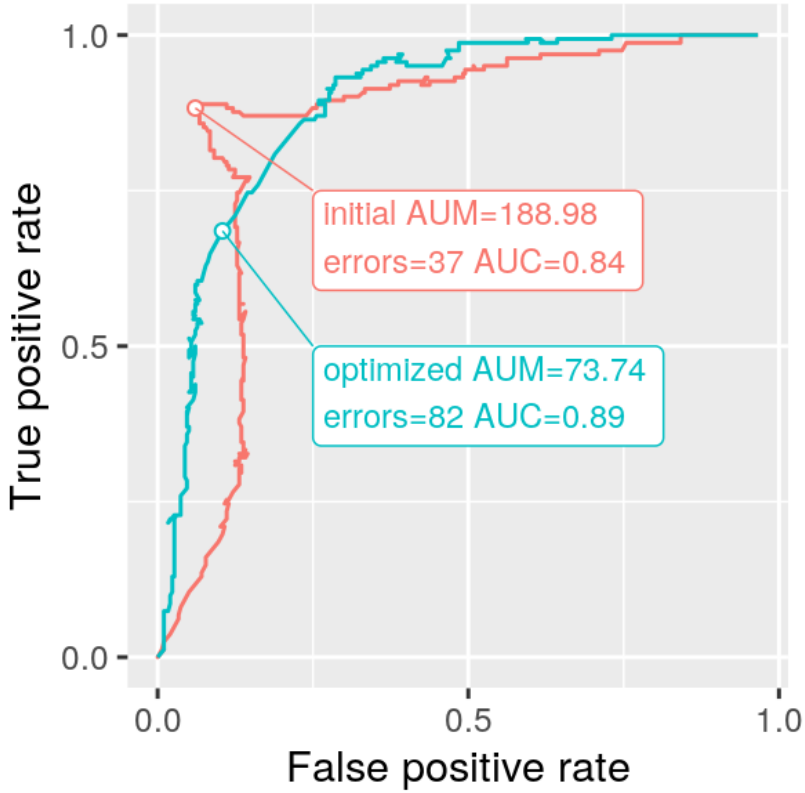



- Nguyen and Hocking, arXiv:2505.07413.

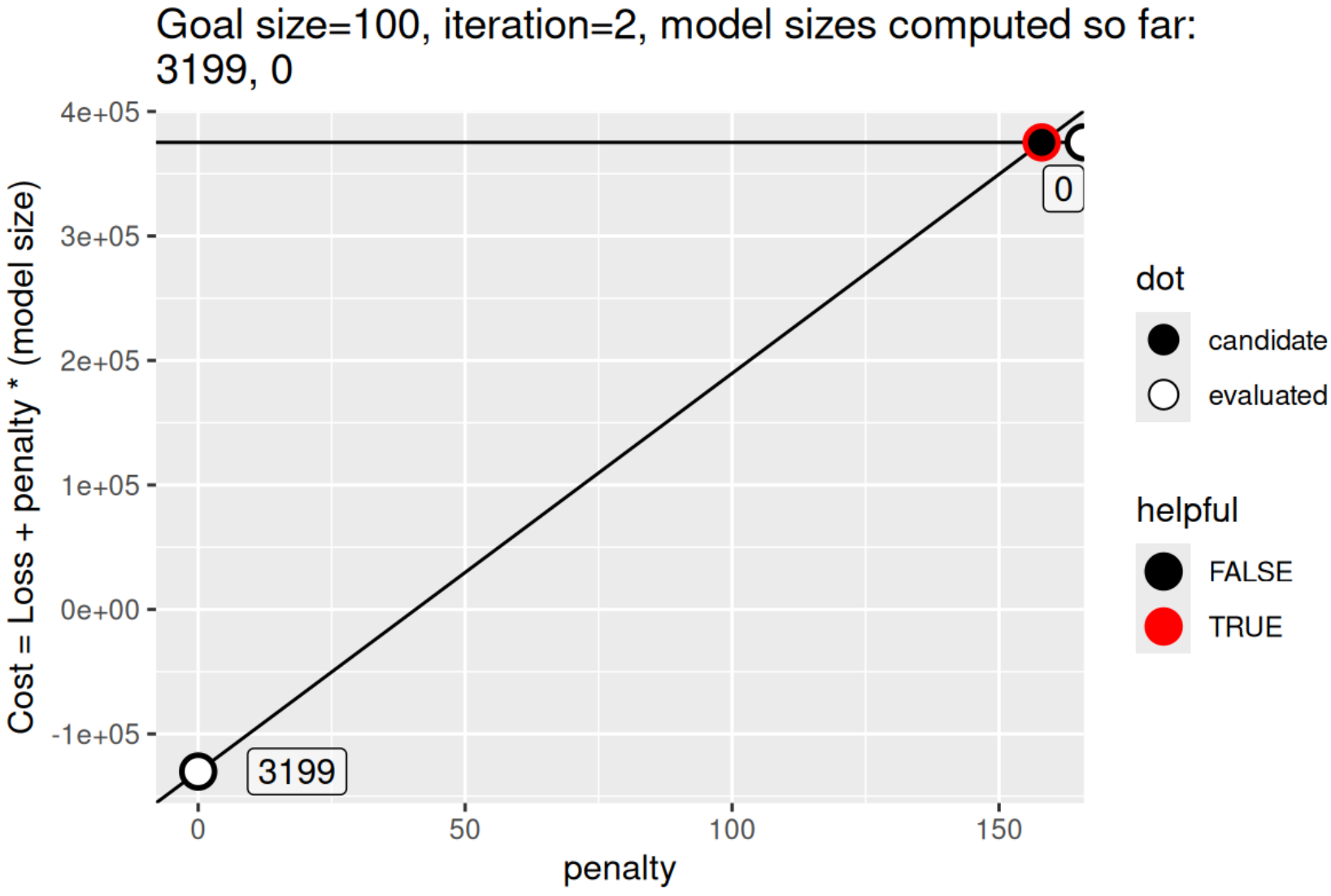
# AUM gradient descent algorithm optimizes AUC

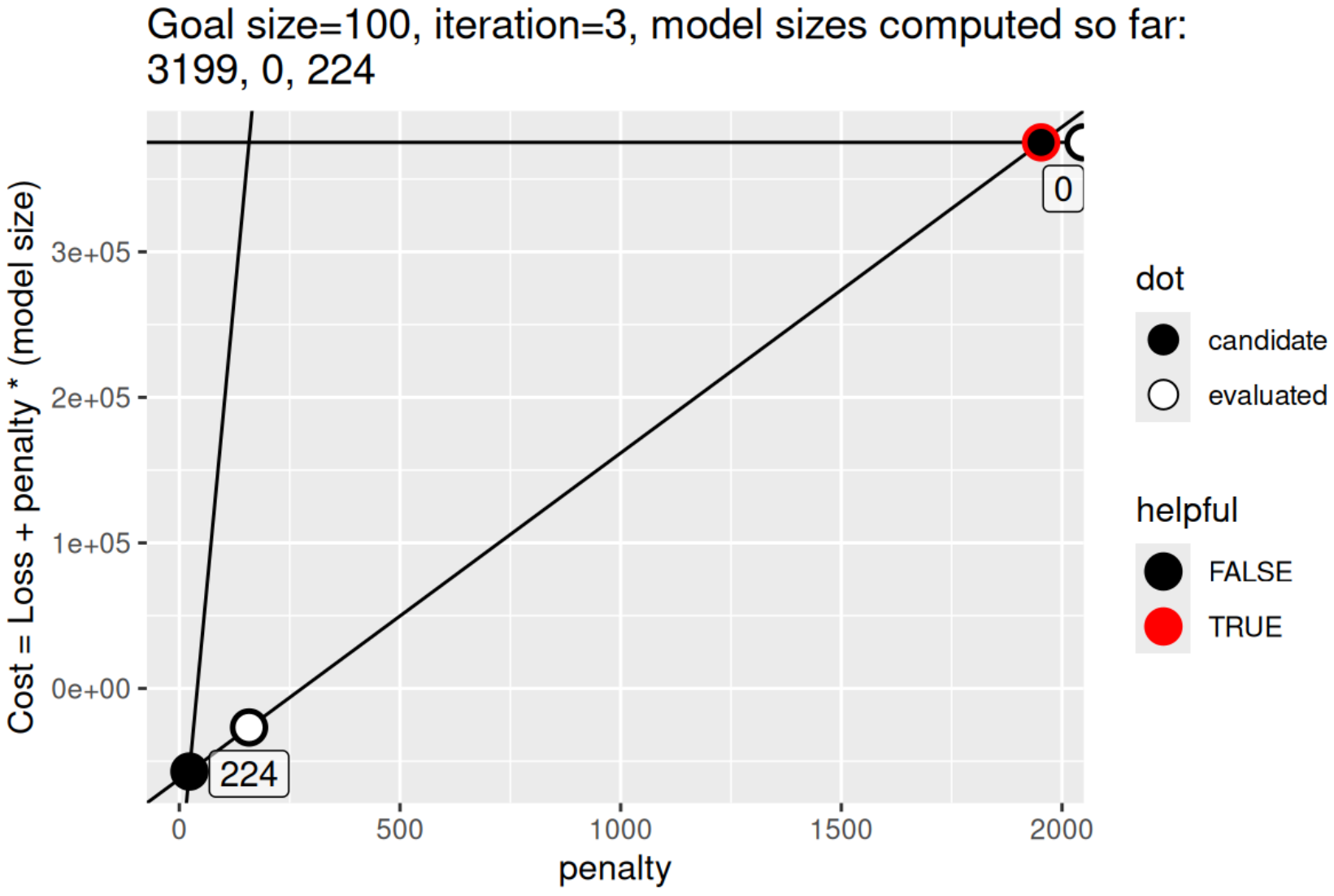
Hillman and Hocking, *Journal of Machine Learning Research* 2023.

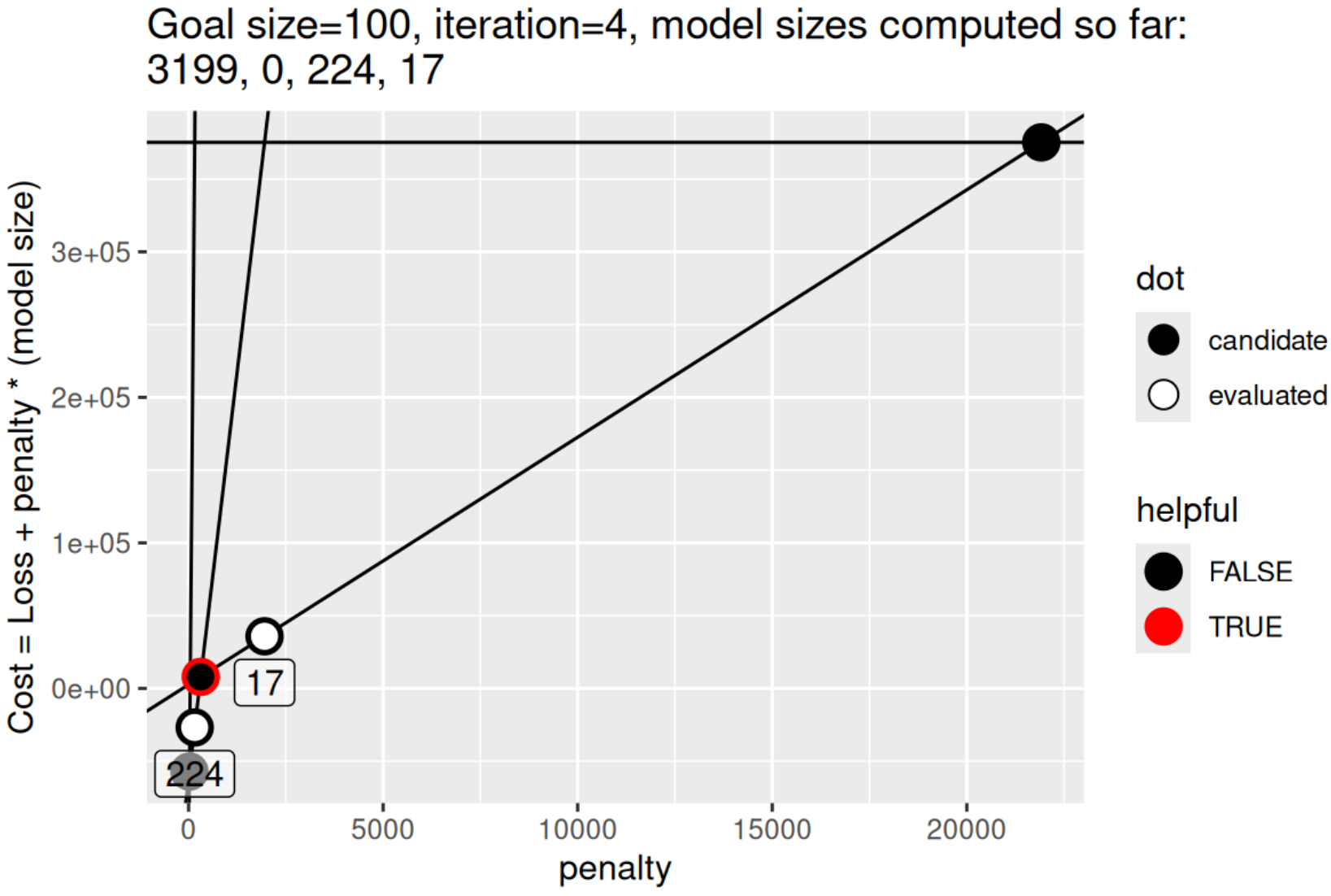


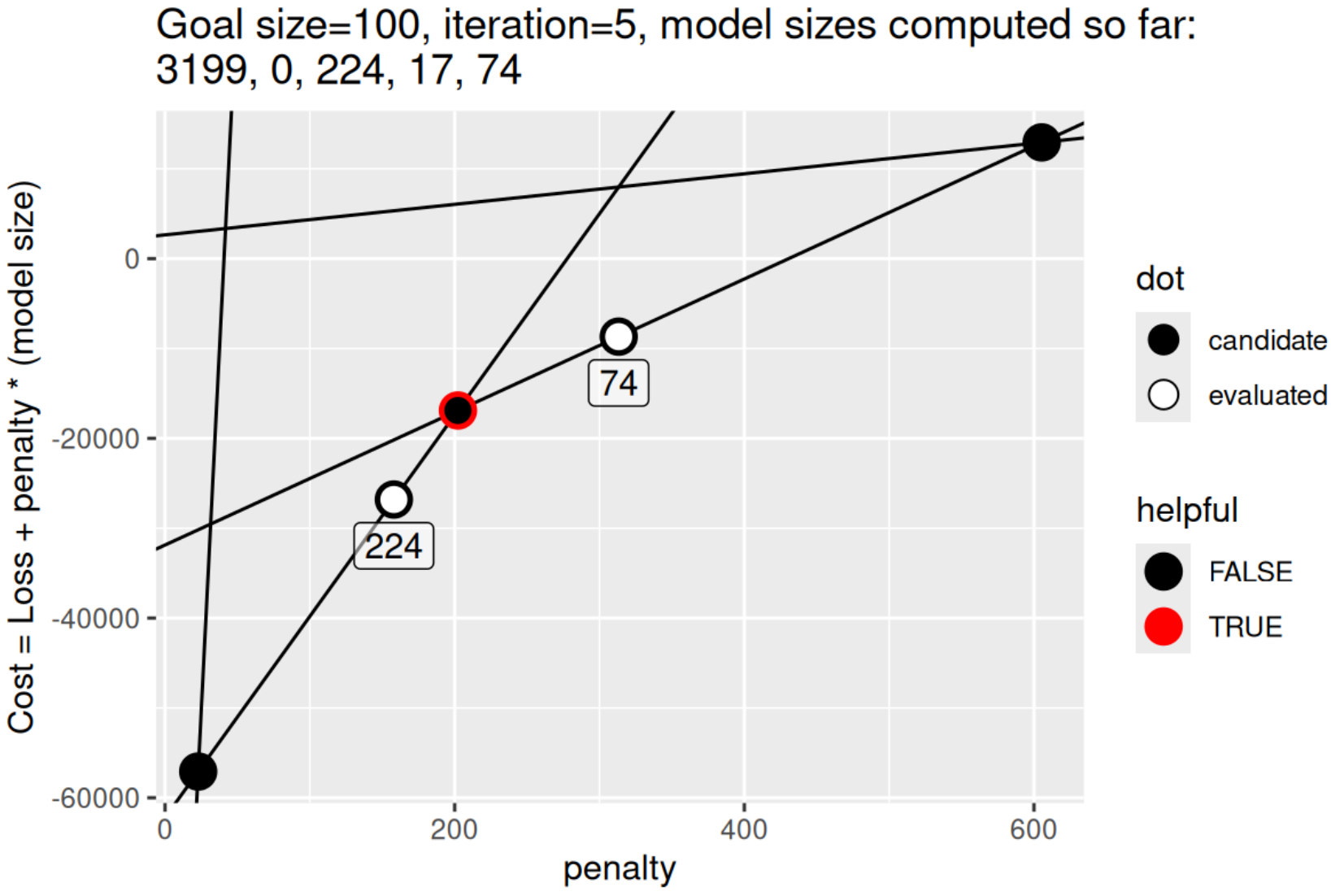



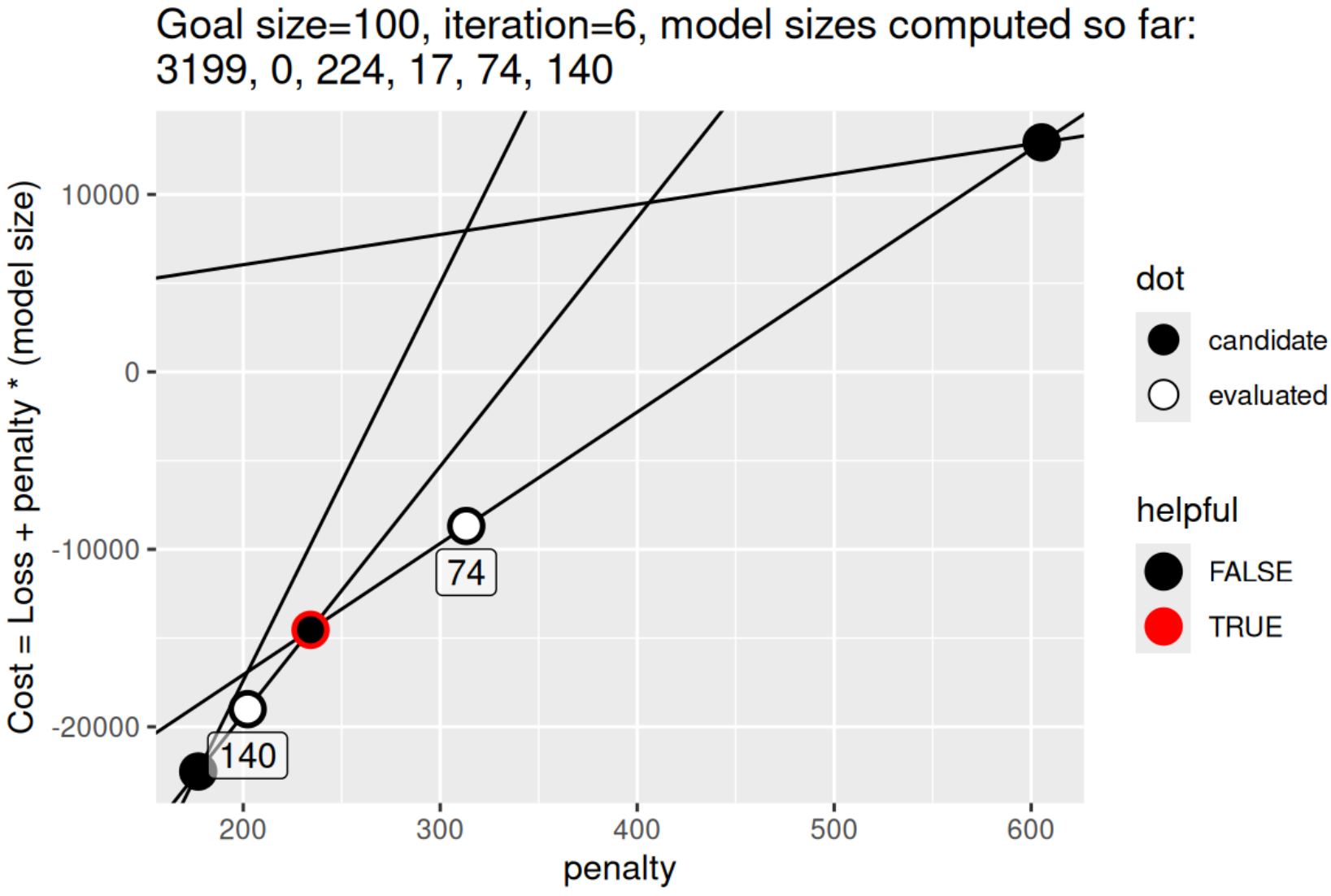

- Initial predictions: minimum label errors.
- Optimized ROC curves are more regular/monotonic.
- Trade-off between AUC and label error optimization.
- library(aum) in R.

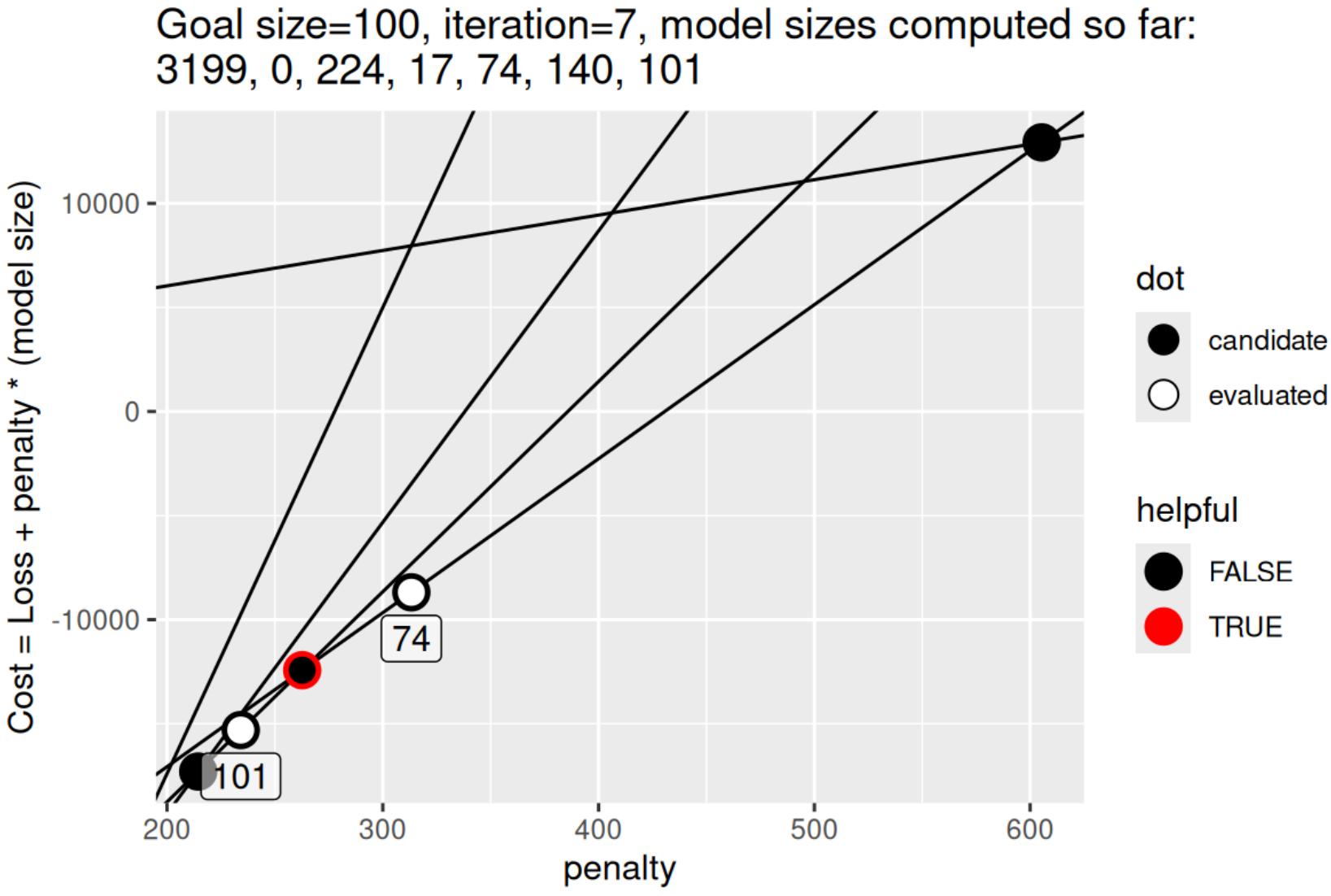

Goal size=100, iteration=2, model sizes computed so far:
3199, 0
Cost = Loss + penalty * (model size)
4e+05
3e+05
2e+05
1e+05
0e+00
-1e+05
0
50
100
150
penalty
3199
0
dot
candidate
evaluated
helpful
FALSE
TRUE

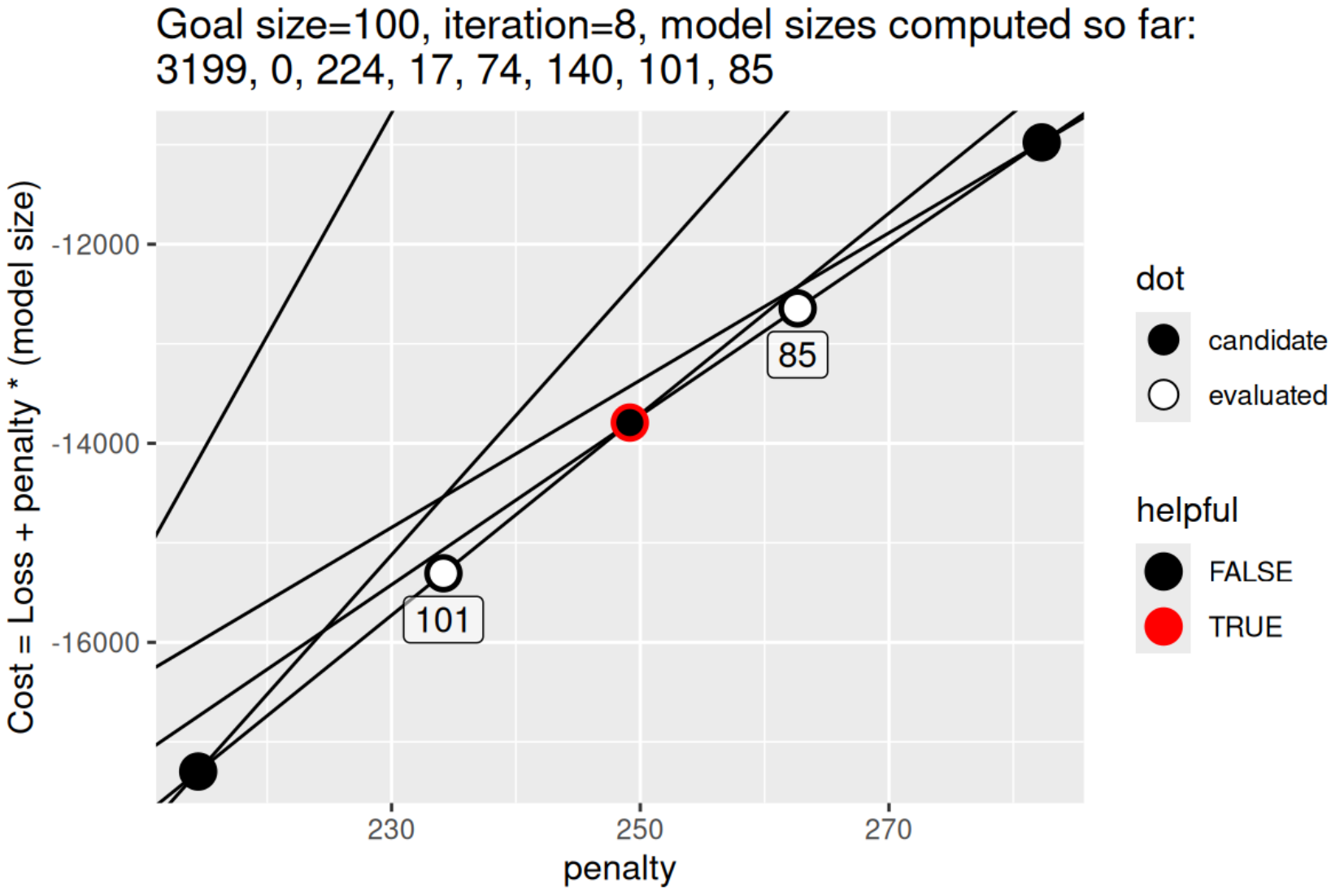

Goal size=100, iteration=3, model sizes computed so far:
3199, 0, 224
Cost = Loss + penalty * (model size)
3e+05
2e+05
1e+05
0e+00
0
224
0
500
1000
1500
2000
penalty
dot
candidate
evaluated
helpful
FALSE
TRUE

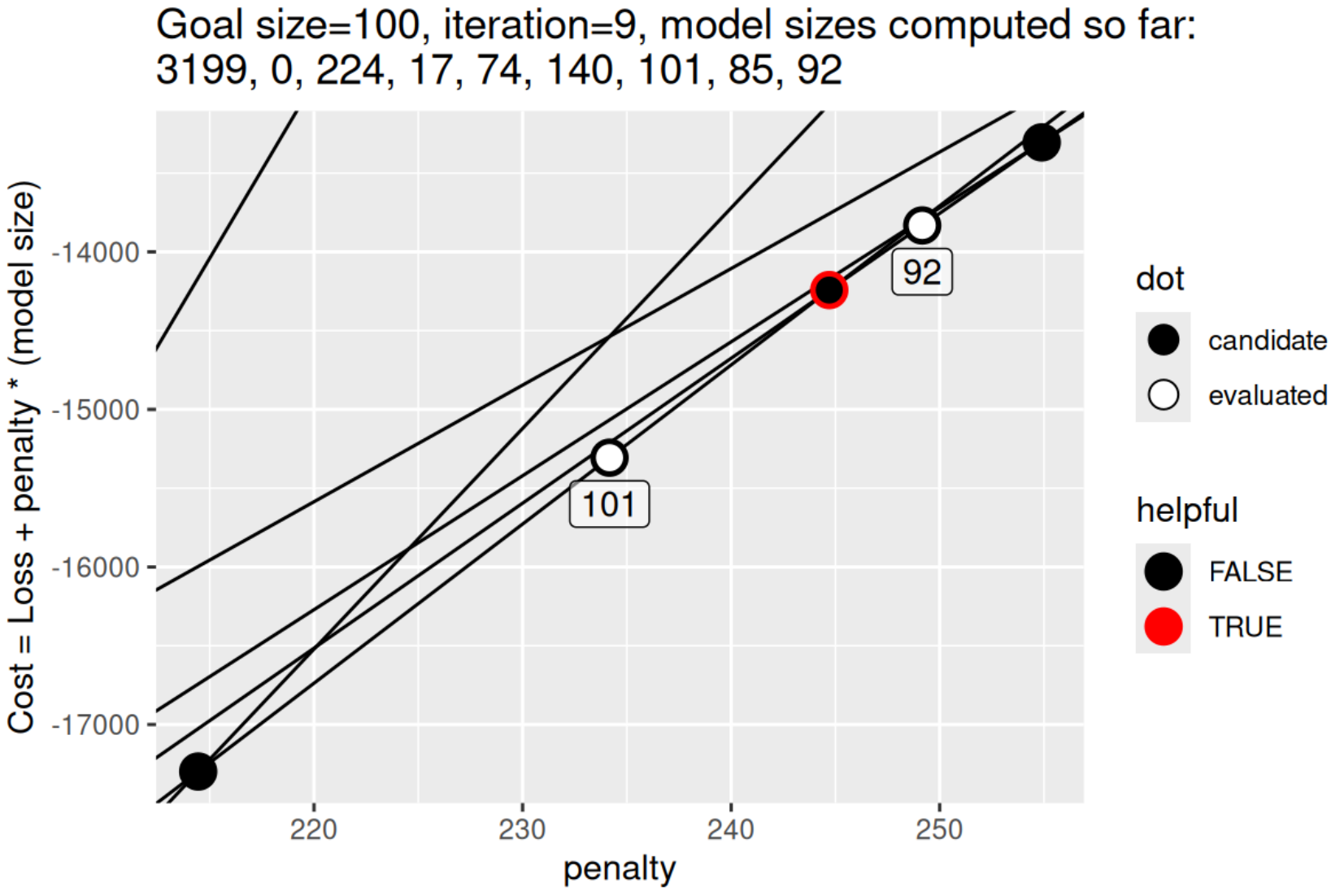

Goal size=100, iteration=4, model sizes computed so far:
3199, 0, 224, 17
Cost = Loss + penalty * (model size)
3e+05
2e+05
1e+05
0e+00
17
224
0
5000
10000
15000
20000
penalty
dot
candidate
evaluated
helpful
FALSE
TRUE

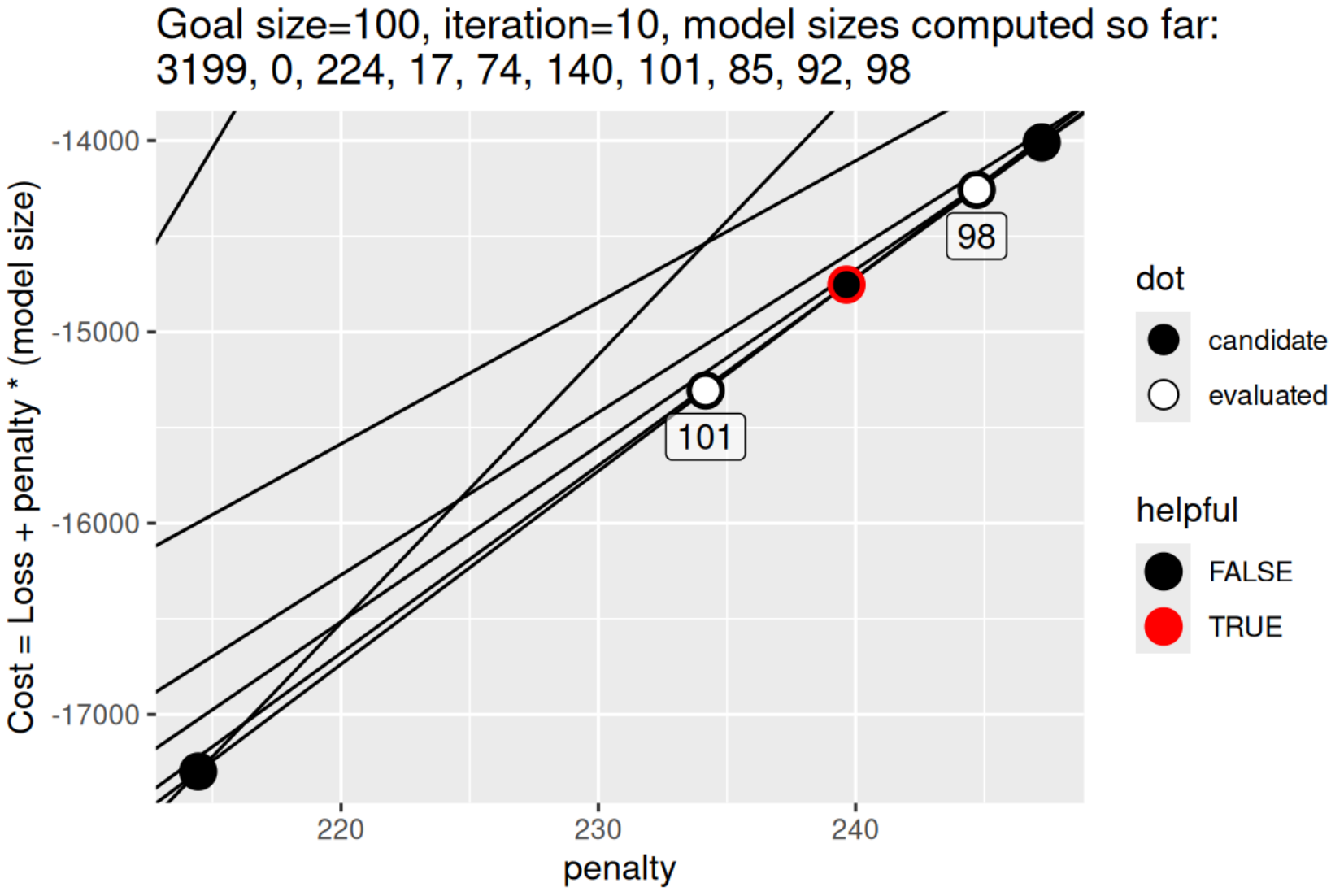

Goal size=100, iteration=5, model sizes computed so far:
3199, 0, 224, 17, 74
Cost = Loss + penalty * (model size)
0
-20000
-40000
-60000
0
200
400
600
penalty
74
224
dot
candidate
evaluated
helpful
FALSE
TRUE

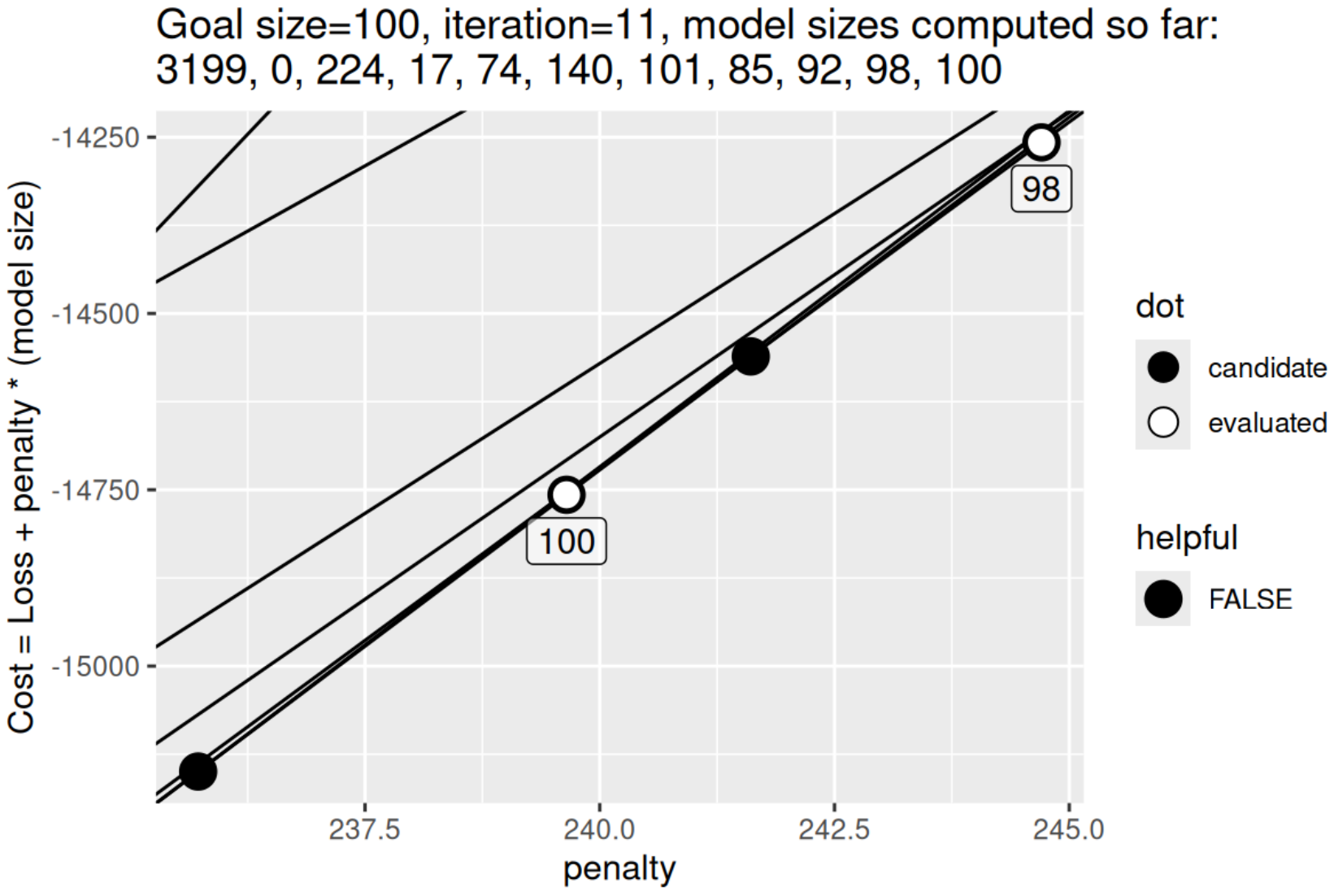

Goal size=100, iteration=6, model sizes computed so far:
3199, 0, 224, 17, 74, 140
Cost = Loss + penalty * (model size)
10000
0
-10000
-20000
200
300
400
500
600
penalty
74
140
dot
candidate
evaluated
helpful
FALSE
TRUE


Goal size=100, iteration=7, model sizes computed so far:
3199, 0, 224, 17, 74, 140, 101
Cost = Loss + penalty * (model size)
10000
0
-10000
200
300
400
500
600
penalty
74
101
dot
candidate
evaluated
helpful
FALSE
TRUE


Goal size=100, iteration=8, model sizes computed so far:
3199, 0, 224, 17, 74, 140, 101, 85
Cost = Loss + penalty * (model size)
-12000
-14000
-16000
230
250
270
penalty
85
101
dot
candidate
evaluated
helpful
FALSE
TRUE


Goal size=100, iteration=9, model sizes computed so far:
3199, 0, 224, 17, 74, 140, 101, 85, 92
Cost = Loss + penalty * (model size)
-14000
-15000
-16000
-17000
220
230
240
250
penalty
92
101
dot
candidate
evaluated
helpful
FALSE
TRUE


Goal size=100, iteration=10, model sizes computed so far:
3199, 0, 224, 17, 74, 140, 101, 85, 92, 98
Cost = Loss + penalty * (model size)
-14000
-15000
-16000
-17000
220
230
240
penalty
98
101
dot
candidate
evaluated
helpful
FALSE
TRUE


Goal size=100, iteration=11, model sizes computed so far:
3199, 0, 224, 17, 74, 140, 101, 85, 92, 98, 100
Cost = Loss + penalty * (model size)
-14250
-14500
-14750
-15000
237.5
240.0
242.5
245.0
penalty
98
100
dot
candidate
evaluated
helpful
FALSE