Changepoint Detection

Lauren Durrell, Kajal Dodhia, Harry Newton, Rui Zhang

STOR-i, Lancaster University

An Example of Changepoint Detection

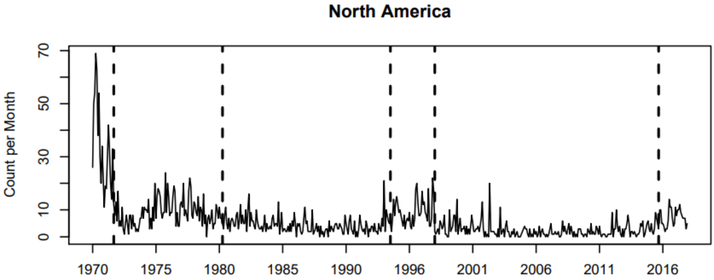
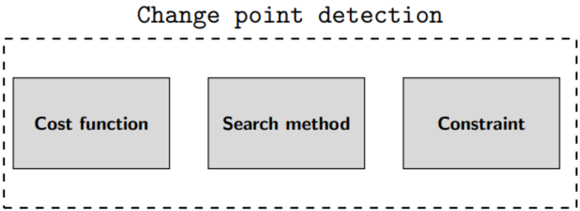
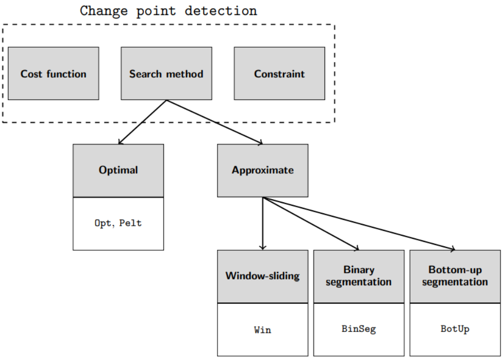


Figure 1: Terrorist Attack Counts in North America [5]

Changepoint Detection Algorithm [6]



Changepoint Detection Algorithm [6]



Outline

Search Methods tradeoff

Exact methods: more thorough but time-consuming
Approximate methods: fast but crude

Outline

Search Methods tradeoff

Exact methods: more thorough but time-consuming
Approximate methods: fast but crude

Game Plan

1. Exact Search via Pruning (PELT, FPOP)
2. Approximate Search via Binary Segmentation (BS, WBS, SBS)
3. Numerical Comparison

Optimal Partitioning

Consider all possible changepoint number k and respective location $\boldsymbol{\tau}$, minimise the total cost (+ penalty)

$$F(t) = \min_{k,\boldsymbol{\tau}} \sum_{i=1}^{m+1} \left[\mathcal{C}\left(y_{(\tau_{i-1}+1):\tau_i} \right) + \beta \right] - \beta$$

Optimal Partitioning

Consider all possible changepoint number k and respective location $\boldsymbol{\tau}$, minimise the total cost (+ penalty)

$$F(t) = \min_{k,\boldsymbol{\tau}} \sum_{i=1}^{m+1} \left[\mathcal{C}\left(y_{(\tau_{i-1}+1):\tau_i}\right) + \beta\right] - \beta$$

A simpler version:

$$F(t) = \min_{\tau} \left[F(\tau) + \mathcal{C}\left(y_{(\tau+1):t}\right) + \beta\right]$$

$$\tau^* = \arg\min_{\tau} \left[F(\tau) + \mathcal{C}\left(y_{(\tau+1):t}\right) + \beta\right]$$

$$\text{cp}(t) = (\text{cp}(\tau^*), \tau^*)$$

Optimal Partitioning

Consider all possible changepoint number k and respective location $\boldsymbol{\tau}$, minimise the total cost (+ penalty)

$$F(t) = \min_{k,\boldsymbol{\tau}} \sum_{i=1}^{m+1} \left[\mathcal{C}\left(y_{(\tau_{i-1}+1):\tau_i}\right) + \beta\right] - \beta$$

A simpler version:

$$F(t) = \min_{\tau} \left[F(\tau) + \mathcal{C}\left(y_{(\tau+1):t}\right) + \beta\right]$$

$$\tau^* = \arg\min_{\tau} \left[F(\tau) + \mathcal{C}\left(y_{(\tau+1):t}\right) + \beta\right]$$

$$\text{cp}(t) = (\text{cp}(\tau^*), \tau^*)$$

A LOT OF iterations !!!

Pruning

A lot of iterations, and some are more important than others.

Pruning

A lot of iterations, and some are more important than others.

Idea: Prune out the bad timestamps.

Pruning

A lot of iterations, and some are more important than others.

Idea: Prune out the bad timestamps.

Solutions: PELT, FPOP, ...

PELT [1]

$$F(t) = \min_{\tau} \left[F(\tau) + \mathcal{C}\left(y_{(\tau+1):t}\right) + \beta \right] =: \min_{\tau} V(\tau)$$

PELT [1]

$$F(t) = \min_{\tau} \left[F(\tau) + \mathcal{C}\left(y_{(\tau+1):t}\right) + \beta \right] =: \min_{\tau} V(\tau)$$

If we have $V(a) > V(b)$ for $a < b$, then we can prune out a at time t.

PELT [1]

$$F(t) = \min_{\tau} \left[F(\tau) + \mathcal{C}\left(y_{(\tau+1):t}\right) + \beta \right] =: \min_{\tau} V(\tau)$$

If we have $V(a) > V(b)$ for $a < b$, then we can prune out a at time t.

Not very useful (yet).

PELT [1]

$$F(t) = \min_{\tau} \left[F(\tau) + \mathcal{C}\left(y_{(\tau+1):t}\right) + \beta \right] =: \min_{\tau} V(\tau)$$

If we have $V(a) > V(b)$ for $a < b$, then we can prune out a at time t.

Not very useful (yet).

Condition 1: Added changepoint always decreases cost by a constant $K \geq 0$.

PELT [1]

$$F(t) = \min_{\tau} \left[F(\tau) + \mathcal{C}\left(y_{(\tau+1):t}\right) + \beta \right] =: \min_{\tau} V(\tau)$$

If we have $V(a) > V(b)$ for $a < b$, then we can prune out a at time t.

Not very useful (yet).

Condition 1: Added changepoint always decreases cost by a constant $K \geq 0$.

After some math ... if $F(s) + \mathcal{C}(y_{(s+1):t}) + K \geq F(t)$, then we can ignore s for consideration at any future time $T \geq t$.

PELT [1]

$$F(t) = \min_{\tau \in R(t)} \left[F(\tau) + \mathcal{C}\left(y_{(\tau+1):t}\right) + \beta \right]$$

$$\tau^* = \arg\min_{\tau} \left[F(\tau) + \mathcal{C}\left(y_{(\tau+1):t}\right) + \beta \right]$$

$$\text{cp}(t) = (\text{cp}(\tau^*), \tau^*)$$

PELT [1]

$$F(t) = \min_{\tau \in R(t)} \left[F(\tau) + \mathcal{C}\left(y_{(\tau+1):t}\right) + \beta \right]$$

$$\tau^* = \arg\min_{\tau} \left[F(\tau) + \mathcal{C}\left(y_{(\tau+1):t}\right) + \beta \right]$$

$$\text{cp}(t) = (\text{cp}(\tau^*), \tau^*)$$

where $R(t) \subset [1 : t]$ is the pruned time set at time t according to $F(s) + \mathcal{C}(y_{(s+1):t}) + K \geq F(t)$.

FPOP [3]

Condition 2: Cost can be written down as a decomposable minimisation problem, i.e.

$$\mathcal{C}(y_{(s+1):t}) = \min_{\mu} \sum_{i=s+1}^{t} \gamma(y_i, \mu).$$

FPOP [3]

Condition 2: Cost can be written down as a decomposable minimisation problem, i.e.

$$\mathcal{C}(y_{(s+1):t}) = \min_{\mu} \sum_{i=s+1}^{t} \gamma(y_i, \mu).$$

Stronger than **Condition 1**.

FPOP [3]

Condition 2: Cost can be written down as a decomposable minimisation problem, i.e.

$$\mathcal{C}(y_{(s+1):t}) = \min_{\mu} \sum_{i=s+1}^{t} \gamma(y_i, \mu).$$

Stronger than **Condition 1**.

With this condition, we can rewrite the overall cost as a double minimisation problem (over τ and μ).

FPOP [3]

Condition 2: Cost can be written down as a decomposable minimisation problem, i.e.

$$\mathcal{C}(y_{(s+1):t}) = \min_{\mu} \sum_{i=s+1}^{t} \gamma(y_i, \mu).$$

Stronger than **Condition 1**.

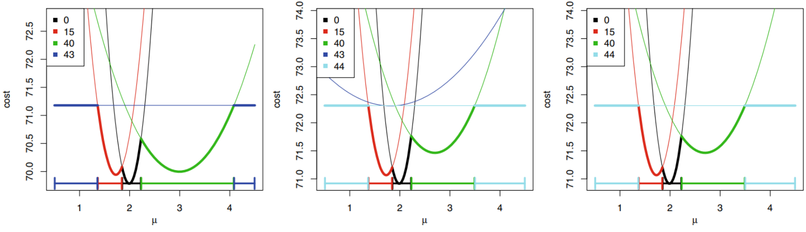
With this condition, we can rewrite the overall cost as a double minimisation problem (over τ and μ).

$$F(t) = \min_{\tau} \min_{\mu} \text{Cost}_t^{\tau}(\mu) = \min_{\mu} \min_{\tau} \text{Cost}_t^{\tau}(\mu) = \min_{\mu} \text{Cost}_t^{*}(\mu)$$

FPOP [3]

1. $\text{Cost}_t^*(\mu)$ consists of components from a few τ
2. Each component is used in the overall cost at some values of μ
3. Updating the cost only involves checking these limited sets

FPOP [3]



(a) (b) (c)

PELT v.s. FPOP

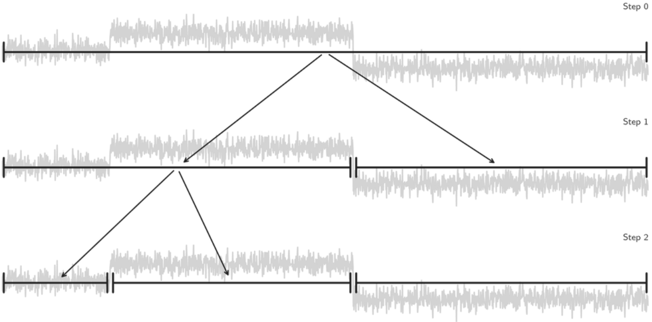
1. Condition needed for PELT is weaker than FPOP
2. FPOP requires more memory than PELT (deteriorates increasingly for higher dimensions)
3. FPOP prunes all the points pruned by PELT, and (much) more

Types of Binary Segmentation

All method use AMOC, but vary the method of breaking down the data into smaller intervals

- Standard
- Wild
- Seeded

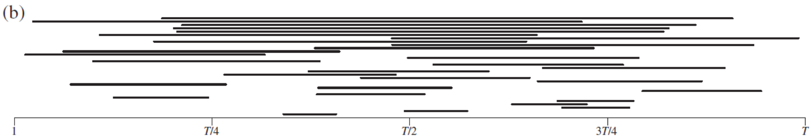
Standard Binary Segmentation [4]



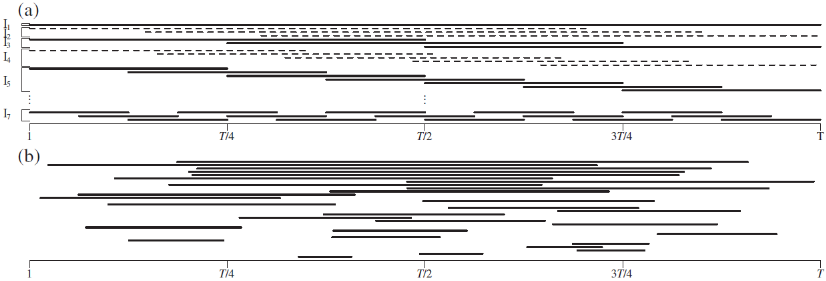
Schematic view of the binary segmentation algorithm

Wild Binary Segmentation [2]

Random interval selection:



Seeded Binary Segmentation [2]



2. Layers of seeded intervals (top) versus random intervals ordered according to length (bottom). In the top panel the dashed lines are created additionally when $a = (1/2)^{1/2}$ compared to $a = 1/2$ (solid lines).

Seeded Binary Segmentation Method

Algorithm 1 Seeded Binary Segmentation

Require: Data X with length T, decay parameter $a \in (\frac{1}{2}, 1]$, minimal segment length $m \geq 2$, and a selection method.

Create a collection of seeded intervals $\mathcal{I}$ with a decay a, which cover $m \geq 2$ observations.

for i in $(1, |\mathcal{I}|)$ **do**

  Take the ith interval in $\mathcal{I}$ and denote the boundaries l and r.

  Calculate the CUSUM statistic $T_{(l,r]}(s) fors = l+1, \ldots, r-1$.

  Apply the chosen selection method to $T_{(l,r]}(\cdot), (l,r) \in \mathcal{I}$,to output the final changepoint estimates.

end for

Seeded Selection Methods

- Greedy selection
- Narrowest-over-threshold

Greedy Selection

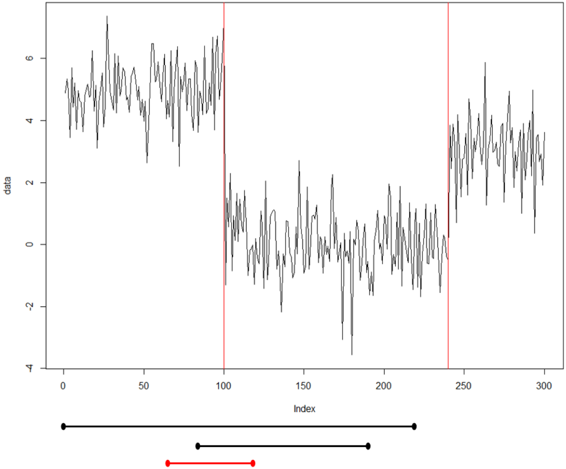
- Ranks the intervals according to their gains $G_{(l,r]}$
- Takes the first $\hat{s}$ of the interval with the highest gains
- Removes from $\mathcal{I}$ all other intervals that contain $\hat{s}$
- repeats these two steps, picking the next best $\hat{s}$
- Stops when no more intervals remain with a gain G over the chosen threshold.

Narrowest-over-threshold

- Takes all intervals $(l, r] \in \mathcal{I}$ where the gains are above the threshold
- The narrowest interval containing the $\hat{s}$ are chosen, and all others are removed from $\mathcal{I}$
- This is repeated until no remaining intervals contain a gain over the threshold

The typical thresholds used for univariate Gaussian models as $C \log^{\frac{1}{2}} T$.

Interval Selection



Why we chose seeded

Benefits:

- Faster - near linear run times
- Reproducibility
- Asymptotically minimax optimal
- More flexible and easier to implement than dynamic programming
- More smaller intervals and fewer longer intervals than the random method
- The speed is comparable to FPOP

Experiment Setup

- Data is generated randomly 50 times but we set a seed.
- The gap between changepoints is 30.
- Accuracy measures
- Computational time
- The mean of these are taken over 50 iterations

Accuracy Measures

Three measures:

- Annotation error: $|\hat{K} - K^*|$
- Hausdorff Distance:

$$\max\left(\max_{\hat{t} \in \hat{\mathcal{T}}} \min_{t^* \in \mathcal{T}^*} |\hat{t} - t^*|, \max_{t^* \in \mathcal{T}^*} \min_{\hat{t} \in \hat{\mathcal{T}}} |\hat{t} - t^*|\right)$$

- Rand Index: Amount of agreement between two segmentations. Set of grouped indices and set of non-grouped indices. A pair of indexes is grouped if they are in the same segment according to both segmentations.

$$\text{RANDINDEX}\left(\mathcal{T}^*, \widehat{\mathcal{T}}\right) := \frac{\left|\text{gr}(\widehat{\mathcal{T}}) \cap \text{gr}\left(\mathcal{T}^*\right)\right| + \left|\text{ngr}(\widehat{\mathcal{T}}) \cap \text{ngr}\left(\mathcal{T}^*\right)\right|}{T(T-1)}$$

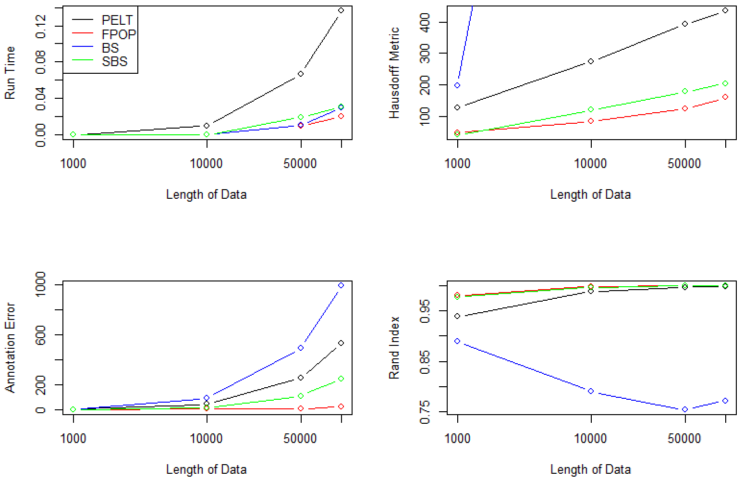
Experiment 1

- We change the length of the data

$$N : 10^3, 10^4, 5x10^4, 10^5$$

- We fix the number of change points to be $\frac{N}{100}$. This is so we have a we consider a linearly increasing number of changepoints.

Experiment 1

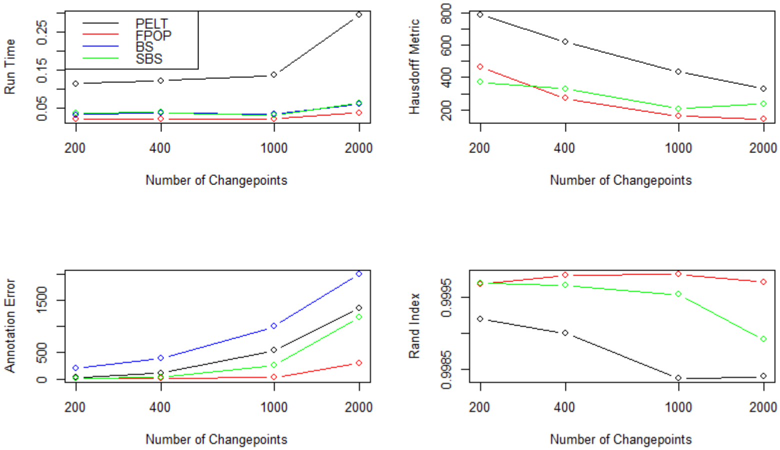


Experiment 2

- We fix the length of the data to $N = 10^5$.
- We vary the number of change points by the following

$$\frac{N}{50}, \frac{N}{100}, \frac{N}{250}, \frac{N}{500}$$

Experiment 2



Conclusion

- Overall, both experiments we find that FPOP seems to perform the best in all metrics and SBS is also pretty good.
- Metric choice should be taken with careful consideration; metrics perform poorly or accurately based on the nature of the data.
- Adjusted Rand index which corrects for chance.

Strengths and Weakness

- The strengths of BS is its computational time to find the change points
- The weakness of BS it doesn't find the changepoints very well
- The weakness of FPOP is that its cost function can only take one parameter, hence less versatile.
- The strengths of SBS is that it is very versatile and more accurate then the exact method PELT which should be more thorough in how it finds its changepoints. It also gives a higher level of reproducibility.
- SBS is not as accurate as FPOP

Further Research

- More comparisons for example to WBS, sliding window (in particular for online) and other methods.
- Consider using the F1 Score
- Consider the penalty terms in more detail, for example, SIC for PELT and FPOP. The penalty constant increases with the amount of data. Overfitting the data may yield too many changepoints.
- Varying the noise-to-signal ratio, magnitude (vary the range of the means) a gap between changepoints.

Reference

[1] R. Killick, P. Fearnhead, and I. A. Eckley.
Optimal detection of changepoints with a linear computational cost.
Journal of the American Statistical Association, 107(500):1590–1598, October 2012.
URL: http://dx.doi.org/10.1080/01621459.2012.737745,
doi:10.1080/01621459.2012.737745.

[2] Solt Kovács, P Bühlmann, H Li, and Axel Munk.
Seeded binary segmentation: a general methodology for fast and optimal changepoint detection.
Biometrika, 110(1):249–256, 2023.

[3] Robert Maidstone, Toby Hocking, Guillem Rigaill, and Paul Fearnhead.
On optimal multiple changepoint algorithms for large data, 2014.
arXiv:1409.1842.

[4] Charles N/a.
Binary segmentation (binseg).
URL: https://centre-borelli.github.io/ruptures-docs/user-guide/detection/binseg/.

[5] S. O. Tickle, I. A. Eckley, and P. Fearnhead.
A computationally efficient, high-dimensional multiple changepoint procedure with application to global terrorism incidence.
Journal of the Royal Statistical Society: Series A Statistics in Society, 184(4):1303–1325, October 2021.
doi:10.1111/rssa.12695.

[6] Charles Truong, Laurent Oudre, and Nicolas Vayatis.
Selective review of offline change point detection methods.
Signal Processing, 167:107299, 2020.