Detecting possibly frequent change-points: Wild Binary Segmentation 2 and steepest-drop model selection

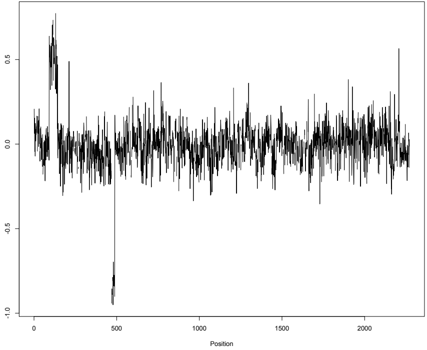
Piotr Fryzlewicz

London School of Economics

University of Kent, 5th December 2019

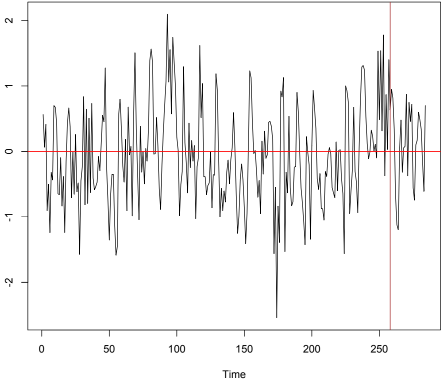
Example problem 1

Identifying locations of gains or losses of DNA copy number. Variations are common in cancer and other diseases. Work with log ratios of test versus reference intensities that look like this:



(Source: R package DNAcopy.)

Difference in monthly UK House Price Index percentage growth between outer and prime London boroughs; brown line marks date of UK EU membership referendum. Possibly frequent change-points? (Source: UK Land Registry.)



Basic model: univariate mean-level shifts

Canonical change-point detection problem (mean-level shifts):

$$X_t = f_t + \varepsilon_t,$$

- where $f_t, \; t = 1, \ldots, T$, piecewise-constant, unknown number of change-points at unknown locations,
- ε_t is (approximately) zero-centred noise with variance σ^2.

Even when ε_t i.i.d. Gaussian, problem not yet solved: many state-of-the-art methods can easily give very different results, especially if f_t is 'challenging', e.g. contains very frequent change-points.

A 'good' change-point detection procedure?

What may we want from a 'good' change-point detection procedure?

- Accurately estimates the number and locations of any change-points present in f_t, for both
  - signals with few/infrequent change-points,
  - and signals with many/frequent change-points;
- is fast to execute and scales well to very long signals;
- has an existing software implementation;
- transfers to other, more complex stochastic models;
- (is easy to explain/implement).

Some existing procedures (and their non-adaptivity)

Very few (if any!) existing procedures tick all these boxes.

Example: in penalised techniques (which minimise the fit to the data + penalty to avoid overfitting), it is clear that some widely-used penalties (e.g. BIC) do not work well for signals with many/frequent change-points. In other words: the penalty does not adapt to the data. Some work to overcome this, e.g. slope heuristics ('data-driven slope estimation', 'dimension jump').

Alternative broad class of methods: those that estimate one change-point at a time. Some examples:

- Binary Segmentation,
- Circular Binary Segmentation,
- Wild Binary Segmentation,
- ...

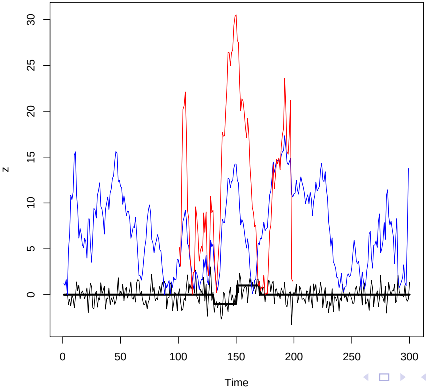
Binary segmentation

Binary segmentation: a well-known and widely-used procedure for multiple change-point detection, in which we first search for one, "most prominent" change-point in the input data.

If one is found, then the sample is split by the estimated change-point, and the same search procedure is then repeated to the left and to the right of it.

Pros: fast to compute, conceptually simple, easy to code. **Cons:** good theoretical performance requires stronger theoretical assumptions, convergence rates poor in more challenging settings.

The following example illustrates potential issues with standard Binary Segmentation. Data X_t in black, global CUSUM in blue, local CUSUM in red (CUSUM is a least-squares measure of the quality of the fit of a piecewise-constant function with one jump to the data):



Wild Binary Segmentation

Clearly, it would have been preferable to use the maximum of the red curve as a locator for a change-point candidate. However, it is obviously not clear a priori what starting point s and end-point e to choose.

Motivated by this, in our earlier work we proposed the following **Wild Binary Segmentation** (WBS) locator statistics

$$WBS = \arg\max_{s^*,b,e^*} |\mathrm{CUSUM}_{s^*,b,e^*}(X)|,$$

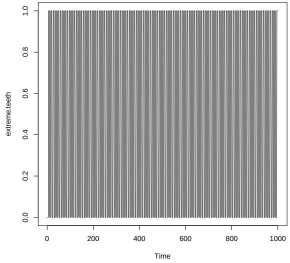
where s^*, e^* are drawn uniformly over the current data segment $[s,e]$ a suitable number (M) of times. Checking all s^*, e^* would have resulted in cubic computational complexity, which would be prohibitive – hence the random draws.

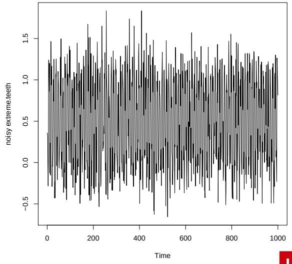
The b that achieves the above maximum is taken as a change-point candidate. The procedure then continues in the usual binary way.

Wild Binary Segmentation – issue with frequent change-points

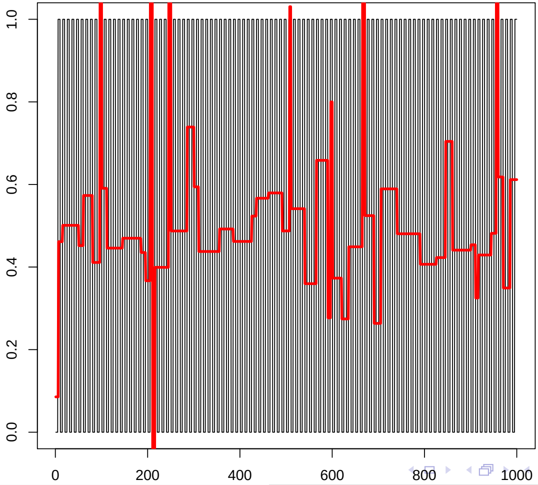
Wild Binary Segmentation is one of the state-of-the-art methods, provided the change-points are not extremely frequent. However, it can fail if change-points occur very frequently.

Consider this "`extreme.teeth`" example.



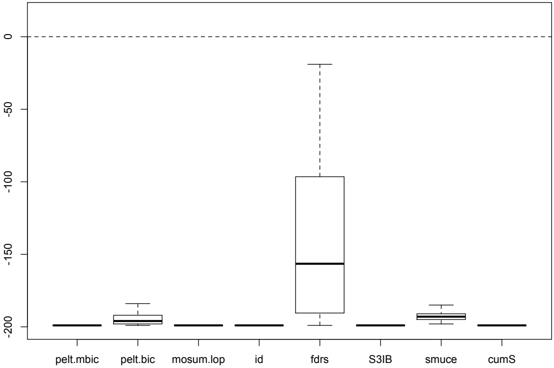


The reconstruction below shows the result of the standard Wild Binary Segmentation applied to the data from the previous slide.



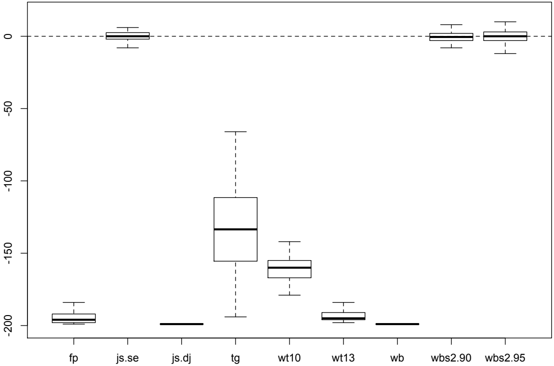
Frequent change-points: performance of the state of the art

WBS is not the only technique struggling in frequent change-point settings. Below are boxplots showing how far off the best methods are in estimating the number of change-points in the above signal.



Frequent change-points: performance of the state of the art (contd)

More methods are shown in the boxplots below. The final two boxplots correspond to WBS2.SDLL, the new technique described in this talk.



Why does WBS misperform in frequent change-point settings?

There are, essentially, two fundamental reasons:

Reason 1.

At each binary stage, WBS only uses the M intervals $[s^*, e^*]$ pre-drawn at the start of the procedure. It does not draw new intervals at each binary stage. As a result, far fewer than M of the original intervals "survive" the binary progression (a point best illustrated on the board) and are therefore able to indicate potential change-point locations. As a result, WBS typically does not compute the entire solution path (i.e. a sequence of estimates of f_t containing $0, 1, \ldots, T-1$ estimated change-points). We refer to this feature (of any change-point detection procedure) as its incompleteness.

Reason 2.

In frequent change-point scenarios, it is unclear what model selection criterion works best:

- In thresholding (i.e. the CUSUMs obtained are tested against a threshold), it is essential to have a good estimate of σ^2. This can be difficult to obtain in frequent change-point scenarios, in which the traditional robust MAD and IQR estimators do not work well.
- Popular penalties, BIC and mBIC do not work well for frequent change-points, and adaptive penalties can be slow to compute and also not very reliable.

Our proposal: WBS2.SDLL

To remedy these two issues (incompleteness and lack of good model selection), we propose **WBS2.SDLL**, a procedure with the following defining characteristics:

- It is complete and it uses recursion to draw intervals adaptively where it needs them, rather than pre-drawing them at the start of the procedure. This allows this solution path algorithm, termed WBS2, to be much faster than a complete or even near-complete version of WBS.
- It uses a new model selection criterion, which we call "Steepest-Drop to Low Levels". It does not use penalties, and relies to a lesser extent on the accurate estimation of $\mathrm{Var}^{1/2}(\varepsilon_t)$ than does classical thresholding.

WBS2: adaptive, recursive interval draws

We now describe the main mechanism by which WBS2 draws intervals:

1. Draw a small number of intervals (say $M = 100$, to fix ideas) and take the argmax of the absolute CUSUMs over these intervals as the first change-point candidate. This is in the hope of detecting one of the change-points present in the data, rather than all of them, as in the classical WBS.
2. Recursively perform the same operation (i.e. again draw M intervals) to the left and to the right of this change-point candidate.
3. On a current domain, only do not draw further intervals if its length $= 1$ (for suitably short domains, draw min(all possible, M) intervals).

WBS2: adaptive, recursive interval draws – contd

WBS2 is by definition a complete procedure: every time point appears in the solution path as a change-point candidate.

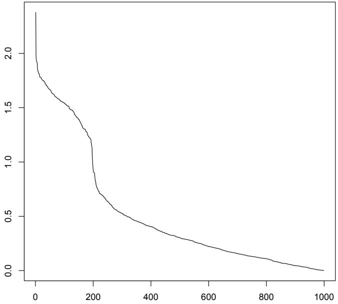
Speed savings with respect to WBS come from the fact that we do not "wastefully" draw intervals only to remove them later at the hierarchical cleaning stage, because WBS2 has no need for such a stage.

WBS draws intervals always in the same way, regardless of input data. WBS2 is adaptive in the way it draws its intervals.

By sorting the change-point candidates according to the magnitudes of (absolute) CUSUMs, from the largest to the smallest, we obtain an entire solution path, i.e. a sequence of estimates with $0, 1, \ldots, T-1$ estimated change-points.

Steepest-Drop model selection along the solution path

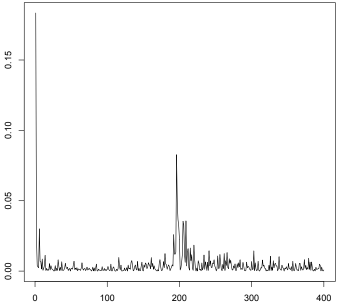
We use an example to motivate and explain our new **Steepest Drop to Low Levels** model selection for WBS2.



Absolute CUSUMs of the noisy `extreme.teeth` signal, produced by WBS2, in decreasing order.

Motivation behind the SDLL model selection

In theory, the CUSUMs that correspond to change-points are of a larger order ($O(\sqrt{T})$) than those that correspond to no change-points / only noise ($O(\sqrt{\log T})$).



The same sequence but now logged and differenced.

Motivation behind the SDLL model selection

The spike at 199 coincides with the number of change-points in the signal and in theory its magnitude $\to \infty$ with T, whereas the magnitudes of all others are bounded.

But choosing the location of the largest spike as the model dimension would not work as that would lead to the model with no change-points.

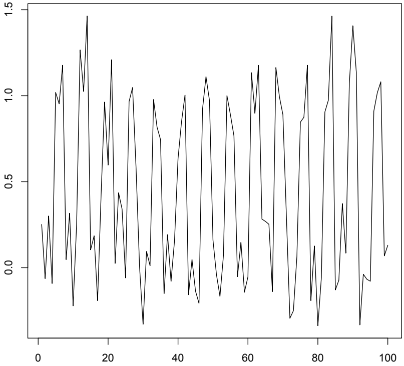
To guard against this, we choose the largest spike such that the CUSUMs "after" the spike fall below a certain threshold; hence the name "Steepest Drop to Low Levels".

Our experience suggests that this threshold does not need to be chosen very well, and even a rough choice leads to good model selection. We tune the threshold so that WBS2.SDLL estimates no change-points for constant signals with 90% or 95% probability.

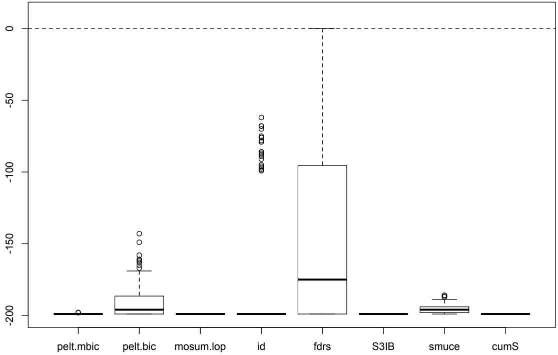
Theorem. Let f have a finite number N of change-points spaced by intervals of length $O(T)$. On a set of probability tending to 1 with T, we have $\hat{N} = N$ and the estimated change-point are near-optimally close to the true ones.

Example 2 – even more frequent change-points

The first 100 observations of a signal (length 700) with even more frequent change-points than before (every 3 and 4 obs), but less noise.

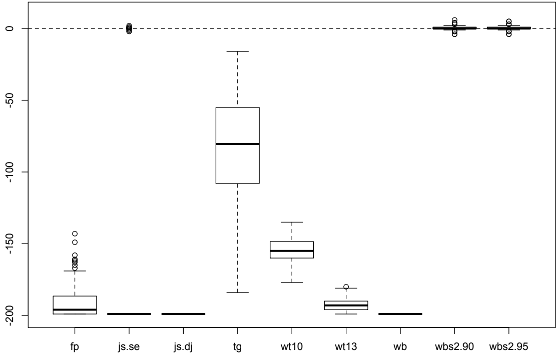


Boxplots of how the different competitors are doing.



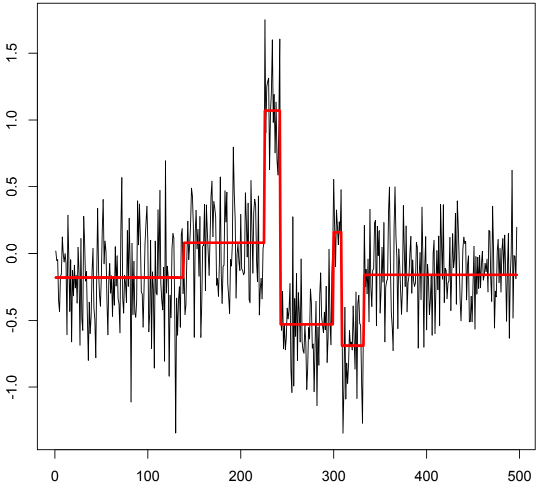
Example 2 – even more frequent change-points – contd

Boxplots of how the different competitors are doing.



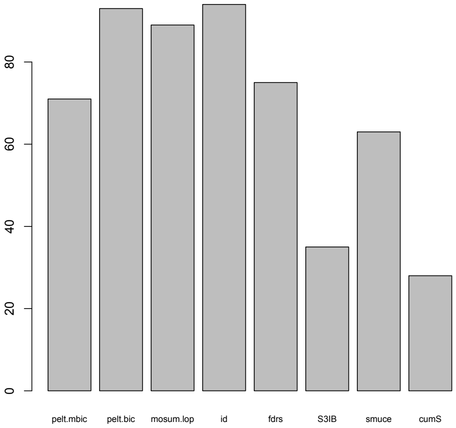
Example 3 – infrequent change-points

Example from Frick, Munk and Sieling (2014):



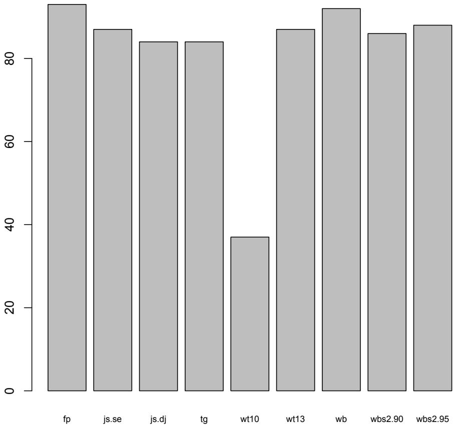
Example 3 – infrequent change-points – contd

Barplots of how many (out of 100) estimators have the correct number of estimated change-points.



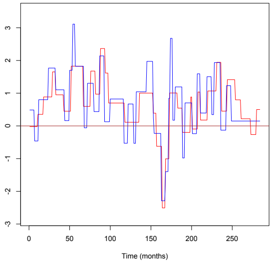
Example 3 – infrequent change-points – contd

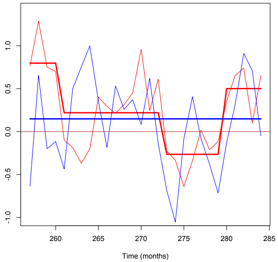
Barplots of how many (out of 100) estimators have the correct number of estimated change-points.



Example: London house prices

UK House Price Index (http://landregistry.data.gov.uk) for London. Blue: prime London; red: outer London; monthly change Feb 1995 – September 2018. Change-point analysis (with WBS2.SDLL) reveals that prices appear to have evolved very differently in the two groups since the Brexit referendum.





References

- Detecting possibly frequent change-points: Wild Binary Segmentation 2 and steepest-drop model selection. P. Fryzlewicz (2019). In submission.
- Wild Binary Segmentation for multiple change-point detection. P. Fryzlewicz (2014). *Annals of Statistics*, 42, 2243-2281.
- R package (on CRAN): **breakfast**.

(Note: the **breakfast** package is currently being expanded and upgraded and it is probably best to wait for the next version.)