Journal of Statistical Software
March 2021, Volume 97, Issue 8. doi: 10.18637/jss.v097.i08

# mosum: A Package for Moving Sums in Change-Point Analysis

**Alexander Meier**
Otto von Guericke University Magdeburg

**Claudia Kirch**
Otto von Guericke University Magdeburg

**Haeran Cho**
University of Bristol

**Abstract**

Time series data, i.e., temporally ordered data, is routinely collected and analysed in in many fields of natural science, economy, technology and medicine, where it is of importance to verify the assumption of stochastic stationarity prior to modeling the data. Nonstationarities in the data are often attributed to structural changes with segments between adjacent change-points being approximately stationary. A particularly important, and thus widely studied, problem in statistics and signal processing is to detect changes in the mean at unknown time points. In this paper, we present the R package **mosum**, which implements elegant and mathematically well-justified procedures for the multiple mean change problem using the moving sum statistics.

*Keywords*: MOSUM, change-point analysis, time series.

## 1. Introduction

With its beginnings dating back as far as the 1950s (Page 1954), change-point analysis is still a very active field of research in statistics. It can broadly be classified into procedures for *sequential* or *online* detection of change-points (i.e., monitoring for changes as the data is being observed) and *offline* detection (i.e., searching for changes after all the data is observed). In this work, we present the package **mosum** (Meier, Cho, and Kirch 2021), which provides an implementation of the moving sum (MOSUM) procedure from Eichinger and Kirch (2018) and its multiscale extension for offline detection of multiple changes in the mean. It is available for the statistical computing language R (R Core Team 2021) from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=mosum`.

There exist many theoretical approaches and software implementations to offline change-point analysis. For example, the R package **bcp** (Erdman and Emerson 2007) provides an implementation of the Bayesian approach proposed in Barry and Hartigan (1993). Several R packages implement algorithms based on global optimization of penalized cost functions, such as the package **strucchange** (Zeileis, Leisch, Hornik, and Kleiber 2002) for detecting structural changes in linear regression models (it also contains utility functions for empirical MOSUM processes); **changepoint** (Killick and Eckley 2014) implementing the pruned exact linear time (PELT) algorithm (Killick, Fearnhead, and Eckley 2012a) (which also implements the binary segmentation algorithm (Scott and Knott 1974; Sen and Srivastava 1975) and the segment neighborhood algorithm (Auger and Lawrence 1989; Bai and Perron 1998)); **changepoint.np** (Haynes and Killick 2020) extending the PELT algorithm with nonparametric cost functions; **ecp** (James and Matteson 2014) implementing the nonparametric change-point method for multivariate data from Matteson and James (2014); **Segmentor3IsBack** (Cleynen, Rigaill, and Koskas 2016) and **fpop** (Rigaill, Hocking, Maidstone, and Fearnhead 2019) implementing pruned dynamic programming algorithms using the functional pruning, see also Rigaill (2015) and Maidstone, Hocking, Rigaill, and Fearnhead (2017).

Another branch of methodologies performs multiscale analysis in searching for change-points in local environments (Fang, Li, and Siegmund 2020; Chan and Chen 2017). For software implementations, see e.g., **breakfast** (Anastasiou, Chen, Cho, and Fryzlewicz 2020) implementing the wild binary segmentation (WBS, Fryzlewicz 2014) and the tail-greedy unbalanced Haar (TGUH, Fryzlewicz 2018) algorithms; **not** (Baranowski, Chen, and Fryzlewicz 2019b) extending the WBS for detecting change-like features (Baranowski, Chen, and Fryzlewicz 2019a); **stepR** (Pein, Hotz, Sieling, and Aspelmeier 2020) and **FDRSeg** (Li and Sieling 2017) implementing the methods proposed in Frick, Munk, and Sieling (2014), Li, Munk, and Sieling (2016) and Pein, Sieling, and Munk (2017) for multiscale inference of step functions. For the segmentation of genomic data, there are packages such as **cumSeg** (Muggeo 2020), **DNAcopy** (Seshan and Olshen 2020) and **modSaRa** (Xiao, Niu, Hao, Xu, Jin, and Zhang 2016). In addition, several methods for sequential change-point detection are provided in **cpm** (Ross 2015). For an overview on recent developments in this area, we refer to Cho and Kirch (2020a) and the repository provided by Killick, Nam, Aston, and Eckley (2012b).

The MOSUM procedure in this work can be seen as complementary to previous approaches, and Eichinger and Kirch (2018) showed that its performance is competitive with state of the art procedures. One of the attractive and useful features of the MOSUM framework is that the underlying statistics have a clear and easy interpretation and lend themselves naturally for visual inspection of structural changes. Also, it can accommodate multiscale extensions of the MOSUM procedure for the improved adaptivity. Furthermore, the MOSUM framework does not rely on any distributional assumption of the underlying data generating process and thus is very flexible.

This paper is structured as follows. In Section 2, we explain the MOSUM statistic and the procedures for change-point estimation. In particular, we discuss two algorithms for multiscale change-point estimation employing a range of summation bandwidths. Section 3 gives an introduction to the **mosum** package, by illustrating the methods for the evaluation of the MOSUM statistics, multiple change-point estimation and visualization with short usage examples. Section 4 contains more detailed usage examples with an application to US macroeconomic data, and Section 5 summarizes the contributions made in this work and provides an outlook for further research and development. In Appendix, we discuss some algorithmic and implementation details (Section A), remark on the computational time of MOSUM-based procedures (Section B), and provide the proof of an asymptotic distributional result (Section C).

## 2. MOSUM procedure for multiple changes in the mean

In this section, we discuss the idea of moving sums and describe the procedures for detecting multiple change-points in the mean, which are implemented in the **mosum** package. In Section 2.1, we review the intuition and the mathematical theory behind the use of MOSUM statistics for multiple change-point detection. In Section 2.2, we explain how to produce the estimators for the locations of the change-points from the MOSUM statistics, followed by a brief discussion of MOSUM-based variance estimation in Section 2.3. In Sections 2.5 and 2.6, we present the extensions of the MOSUM procedure with multiple summation bandwidths. Section 2.7 elaborates on how to construct bootstrap confidence intervals for the change-point locations.

### 2.1. MOSUM statistic

Consider observations $X_1, \ldots, X_n$ drawn independently from a distribution with the same mean as an example. By the Law of Large Numbers, it follows that

$$\frac{1}{G}\sum_{t=k+1}^{k+G} X_t - \frac{1}{G}\sum_{t=k-G+1}^{k} X_t \approx 0$$

for a sufficiently large *summation bandwidth* $G > 0$. If, on the other hand, there is a change in the mean of height $d$ at time point $k$, then

$$\frac{1}{G}\sum_{t=k+1}^{k+G} X_t - \frac{1}{G}\sum_{t=k-G+1}^{k} X_t \approx d.$$

Based on this observation, the following MOSUM statistic provides a good tool for change-point detection:

$$T_G := \max_{1\le k\le n} \frac{|T_G(k)|}{\widehat{\sigma}_k} \tag{1}$$

with the MOSUM *detector*

$$T_G(k) = T_G(k; X) := \sqrt{\frac{G}{2}}\left(\frac{1}{G}\sum_{t=k+1}^{k+G} X_t - \frac{1}{G}\sum_{t=k-G+1}^{k} X_t\right), \quad k = G, \ldots, n-G, \tag{2}$$

and a local estimator $\widehat{\sigma}^2_k$ of the innovation variance (see the upcoming Section 2.3). The values $T_G(k)$ for $k$ at the left and right boundaries can be calculated adopting a CUSUM-type *boundary extension*:

$$T_G(k) := \sqrt{\frac{2G}{k(2G-k)}}\sum_{t=1}^{k}\left(\bar{X}_{(1,2G)} - X_t\right), \quad k = 1, \ldots, G-1, \tag{3}$$

with $\bar{X}_{(1,2G)} := (2G)^{-1}\sum_{t=1}^{2G} X_t$ and similarly for $k = n-G+1, \ldots, n$.

The statistical model underlying the MOSUM procedure is a classical change-point location model (c.f. Eichinger and Kirch 2018)

$$X_t = f_t + e_t = \sum_{j=1}^{N+1} \mu_j \mathbb{I}\{k_{j-1} < t \le k_j\} + e_t, \quad t = 1, \ldots, n. \tag{4}$$

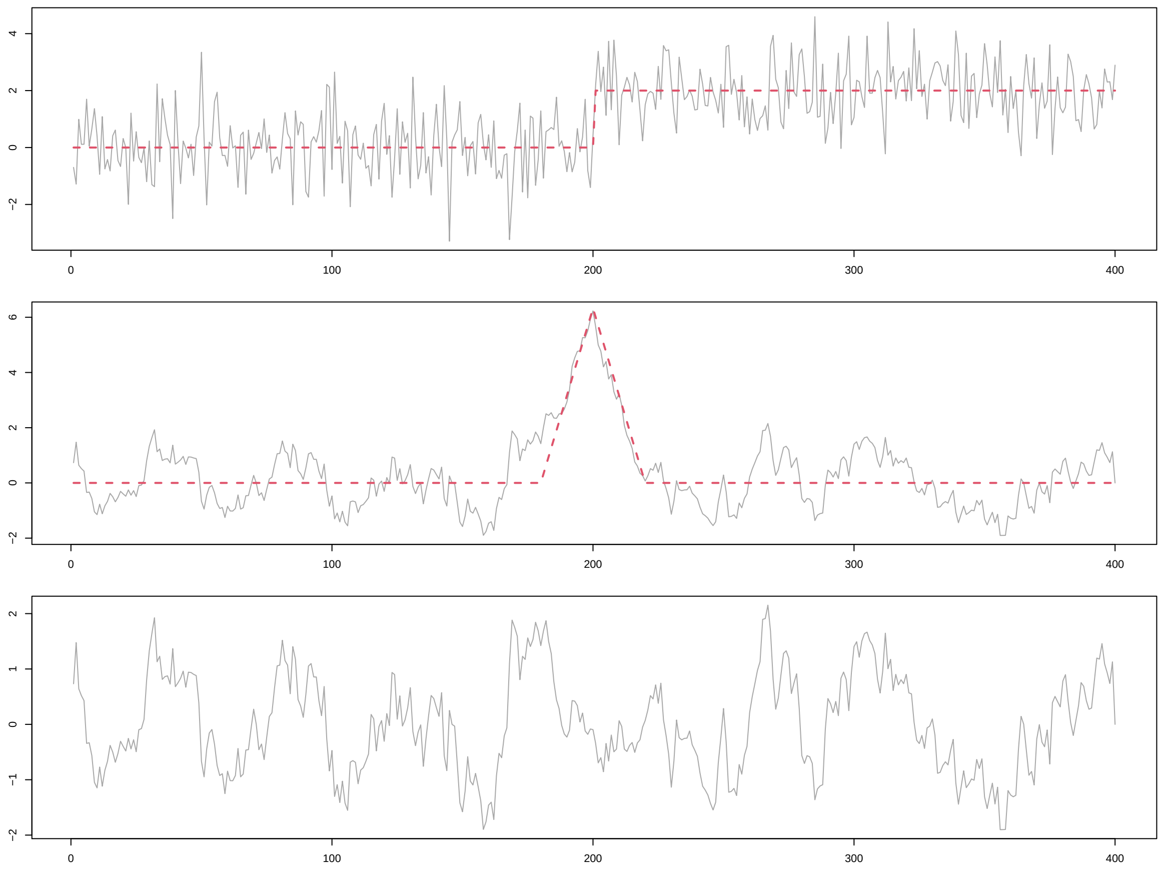

Figure 1: Top: A time series $X_1, \ldots, X_n$ of length $n = 400$ with one mean change of height $d = 2$ at time $k = 200$. The corresponding step signal $f_t$ is overlaid as dashed line. Middle: The MOSUM detector $T_G(k; X)$ for $k = 1, \ldots, n$, where the MOSUM signal $T_G(k; f)$ is overlaid as dashed line. Bottom: $T_G(k; e) = T_G(k; X) - T_G(k; f)$.

The piecewise constant deterministic signal $f_t$ has $N$ change-points at $k_j$, $j = 1, \ldots, N$ (with the notational convention $k_0 := 0$ and $k_{N+1} := n$), and the centred model innovations $e_t$ are assumed to be independent and identically distributed. Note that the MOSUM detector from (2) is decomposed as

$$T_G(k; X) = T_G(k; f) + T_G(k; e). \tag{5}$$

We call $T_G(k; f)$ the *MOSUM signal* and $T_G(k; e)$ the noise term.

The number of change-points $N$ and their locations $k_j$ for $j = 1, \ldots, N$, as well as the heights of jumps $d_j = \mu_{j+1} - \mu_j$, are typically unknown and in many applications, estimation of both the number and locations of changes is of particular interest. The MOSUM detector lends itself naturally for this purpose, since whenever a mean change at time $k$ occurs, the corresponding MOSUM signal $T_G(k; f)$ from (5) will attain a local maximum in its absolute value, which is superimposed by the noise term $T_G(k; e)$ in the detector $T_G(k; X)$. This is illustrated in Figure 1, where the prominent peak of the MOSUM signal at time $k = 200$ is still clearly visible in the noisy $T_G(k; X)$.

In order to make use of this observation for change-point detection, a suitable threshold is needed (for details, see Section 2.2 below). One option is to use a critical value of the corresponding test procedure as threshold. The actual distribution of the MOSUM statistic is not known in general, even for well-known innovation distributions and small sample sizes. One therefore makes use of an asymptotic result to derive a MOSUM-based test for changes in the mean. Indeed, for appropriate bandwidths $G = G(n)$ satisfying $G \to \infty$ as $n \to \infty$ while $G/n \to 0$ (as detailed in Appendix C),

$$a_G \, T_G - b_G \implies \Gamma_2 \quad \text{under } H_0\text{: No mean change,}$$

where $\Rightarrow$ denotes the convergence in distribution, $\Gamma_2$ is a Gumbel-distributed random variable with $P(\Gamma_2 \le z) = \exp(-2\exp(-z))$, and $a_G$ and $b_G$ are sequences of properly chosen scaling and shifting factors depending only on the sample size $n$ and the bandwidth $G$. This asymptotic result gives rise to a MOSUM-based test with asymptotic level $\alpha$, which rejects the null hypothesis $H_0 : N = 0$ against the alternative $H_1 : N > 0$ when the MOSUM statistic $T_G$ from (1) exceeds the asymptotic critical value

$$C_{n,G}(\alpha) := \frac{b_G + Q_{1-\alpha}(\Gamma_2)}{a_G}, \tag{6}$$

where $Q_{1-\alpha}(\Gamma_2)$ is the $(1-\alpha)$-quantile of the Gumbel distribution. The $p$ value corresponding to the test statistic $t$ is given by $p_{n,G}(t) = 1 - \exp(-2\exp(b_G - a_G t))$.

The bandwidth $G$ plays a crucial role in the performance of the methodology in practical applications. We will discuss this issue in Section 2.4.

## 2.2. Change-point estimators

The *absolute MOSUM detector* $|T_G(k)|$ (see (2)) is a powerful tool for visual change-point inspection. The corresponding (absolute) MOSUM signal $|T_G(k; f)|$ (see Figure 1 for an example) is a piecewise linear function, which is equal to zero far away from the change-points, linearly increases as a change-point is approached, and then decreases towards zero after the change-point. Consequently, a jump of the underlying step signal $f_t$ results in a peak in the MOSUM signal, with the location of the jump coinciding with that of the local maximum of $|T_G(k; f)|$.

In practice, $T_G(k; f)$ is not observable and we have to work with the noisy MOSUM detector $T_G(k) = T_G(k; X)$ instead. Therefore, it is natural to apply a threshold to the (scaled) absolute MOSUM detector and construct change-point estimators based on the local maxima of neighborhoods exceeding the threshold. To elaborate, we consider *significant neighborhoods* $(l, r)$ with $l \le k \le r$, such that

$$\begin{aligned} \frac{|T_G(k)|}{\widehat{\sigma}_k} &\ge C_{n,G}(\alpha) \quad \text{for } k = l, \ldots, r, \quad \text{and} \\ \frac{|T_G(k)|}{\widehat{\sigma}_k} &< C_{n,G}(\alpha) \quad \text{for } k = l-1, r+1. \end{aligned}$$

Choosing the maximal point within every significant environment

$$k^{\circ}_{(l,r)} = \arg\max_{l \le k \le r} \frac{|T_G(k)|}{\widehat{\sigma}_k} \tag{7}$$

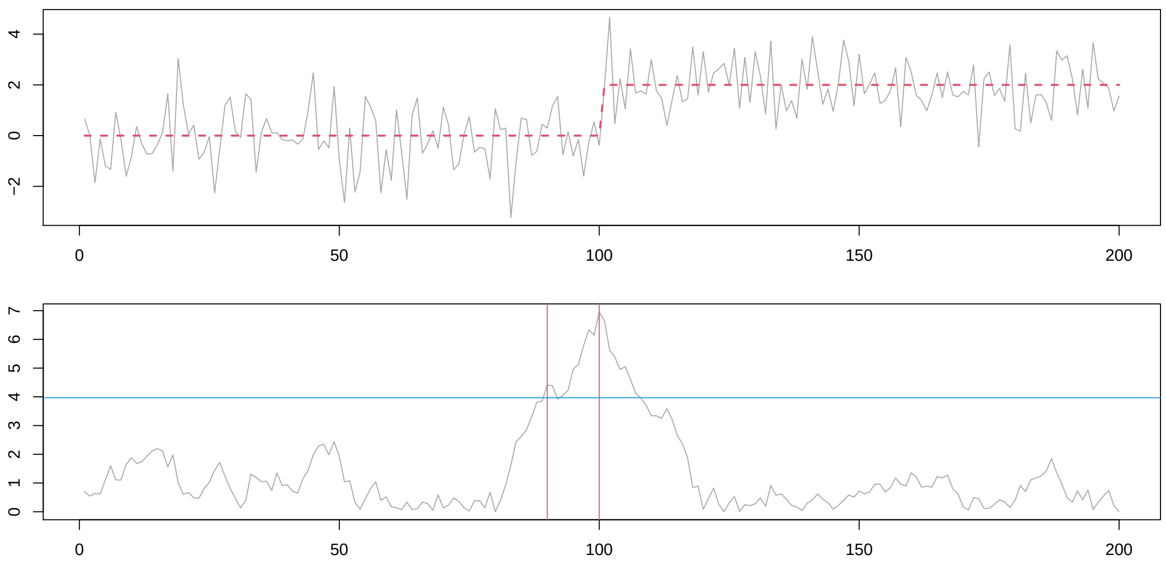

Figure 2: Top: A time series with one mean changes at the location 100, where the underlying step signal $f_t$ is overlaid as dashed line. Bottom: The corresponding absolute MOSUM detector $|T_G(k)|$ for $k = 1, \ldots, n$ with bandwidth $G = 20$. The critical value for level $\alpha = 0.05$ is denoted by a horizontal line, and maximal points within significant environments of $T_G(k)$ are highlighted by the vertical lines.

as a change-point estimator, however, may result in false positives for the following reason. Recall that the MOSUM signal $T_G(k; f)$ is a linear function to the left and right of the true change-point and will cross the threshold. The observed detector on the other hand adds noise to this signal. Therefore it can happen, merely by chance, that the detector falls below the threshold only to be exceeding it again for just one time point or two, before crossing beneath the threshold again. Obviously, this results in additional significant neighborhoods and hence spurious estimators. See Figure 2 for an example, where this phenomenon occurs at about time point 90. Every statistical procedure based on the MOSUM detector thus needs to take this effect into account.

While spurious peaks such as this can usually be distinguished from true changes by visual inspection, the following two mathematical criteria are implemented in the R package **mosum** in order to avoid such systematic over-estimation.

### *The $\varepsilon$-criterion*

Let $0 < \varepsilon < 1/2$ be fixed. A maximal point within a significant environment $k^\circ_{(l,r)}$ as in (7) will be accepted as a change-point estimate if and only if

$$r - l \geq \varepsilon G. \tag{8}$$

This criterion states that the significance environment has to be large enough relative to the summation bandwidth $G$. Otherwise, we argue that the significance is merely due to the influence of a neighboring change (as in Figure 2 at about time point 90), hence it will be discarded. Based on extensive simulation studies, we recommend $\varepsilon = 0.2$ as a reasonable default choice which works well in many situations. The estimators thus obtained for the change-point locations are consistent in rescaled time, as long as the distance between neighboring change-points is, in some asymptotic sense, at least twice the bandwidth, see Eichinger and Kirch (2018) for further details. The $\varepsilon$-criterion is particularly useful if the bandwidth is sufficiently large, i.e., $G \geq 20$.

*The $\eta$-criterion*

Let $\eta \geq 0$ be fixed. The $\eta$-criterion accepts a time point $k^\circ$ as a change-point estimator if and only if it is the maximal point within its own $\eta G$ environment, i.e.,

$$k^\circ = \arg \max_{k^\circ - \eta G \leq k \leq k^\circ + \eta G} \frac{|T_G(k)|}{\widehat{\sigma}_k}. \tag{9}$$

From the simulation studies, we found that a value of $\eta = 0.4$ can be recommended if there is no further knowledge about the mutual distance of changes in the data. Lifting the requirement on the significance of the entire $\eta G$ environment, the $\eta$-criterion is less conservative compared to the $\varepsilon$-criterion. This can be particularly useful when the bandwidth is small; we further compare the two criteria in the following.

*Comparison of the two criteria*

The $\varepsilon$-criterion is conservative in the sense that it not only requires significance of one point (as per the corresponding testing procedure) but of a whole neighborhood of length at least $\varepsilon G$. Therefore, in some situations, the $\varepsilon$-criterion might be too restrictive. In particular, if the bandwidth is small (e.g., $G = 8$), significance environments of length 2 or even of length 1 should not be discarded a priori. See Figure 3 for an example of such a situation, where the changes at time points 10 and 20 are detected with a significant environment of length 1; nonetheless, the peaked shape of the detector clearly indicates the presence of changes.

Furthermore, it can happen for larger bandwidths and certain jump signals that the detector lies entirely above the threshold. In such cases, the $\varepsilon$-criterion only keeps the global maximizer of the detector as a change-point estimator even though it is obvious that only one change-point could not have caused the detector to be significant everywhere. See Figure 4 for an example of such a situation, where the location of changes is visible in the detector (in terms of peaks), but the detector does not fall below the critical value between the changes. The $\eta$-criterion with a suitably chosen $\eta$ on the other hand can correctly identify more than one change-points in this example.

## 2.3. Variance estimation

The standard variance estimator $(n-1)^{-1}\sum_{t=1}^{n}(X_t - \bar{X}_n)^2$ is consistent in the absence of mean changes, but systematically over-estimates the variance in the presence of mean changes and thus is not suitable for change-point estimation procedure. The median absolute deviation (Hampel 1974) or inter-quartiles range estimators are popularly adopted in the presence of change-points, but they have also been observed to over-estimate the variance in the presence of frequent change-points, see Fryzlewicz (2020). An alternative approach to variance estimation is to use a local variance estimator, such as the MOSUM-based variance estimator

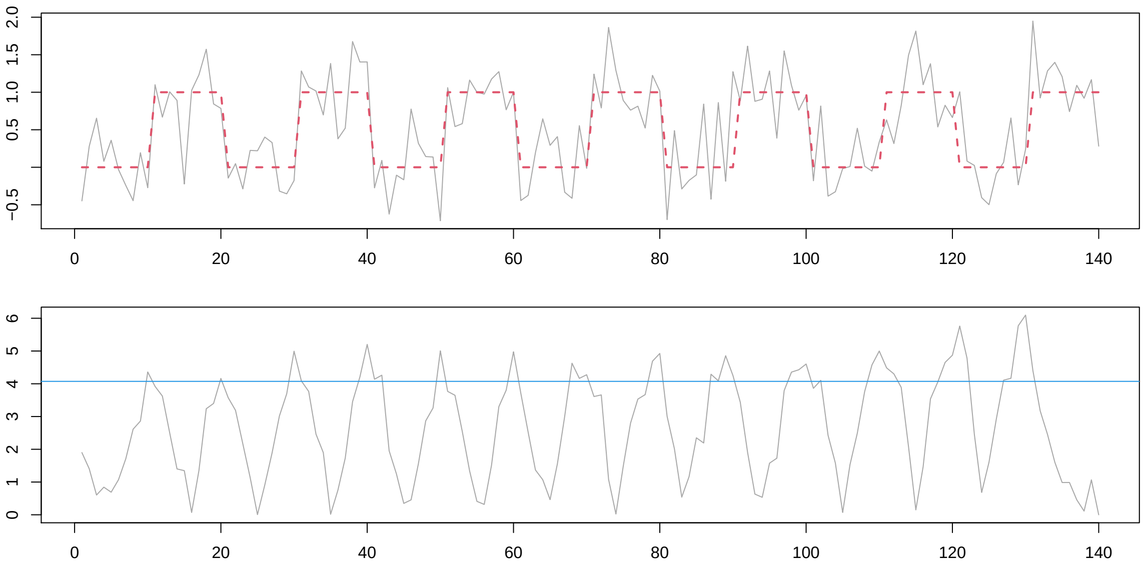

Figure 3: Top: A time series of length 140 with mean changes at locations $10, 20, \ldots, 130$, where the underlying step signal $f_t$ is overlaid as dashed line. Bottom: The corresponding scaled absolute MOSUM detector $\sigma^{-1}|T_G(k)|$ with $G = 8$; the critical value for level $\alpha = 0.05$ is denoted by a horizontal line. The series is a realization of the model `teeth10` from Fryzlewicz (2014) (see Section 3.1).

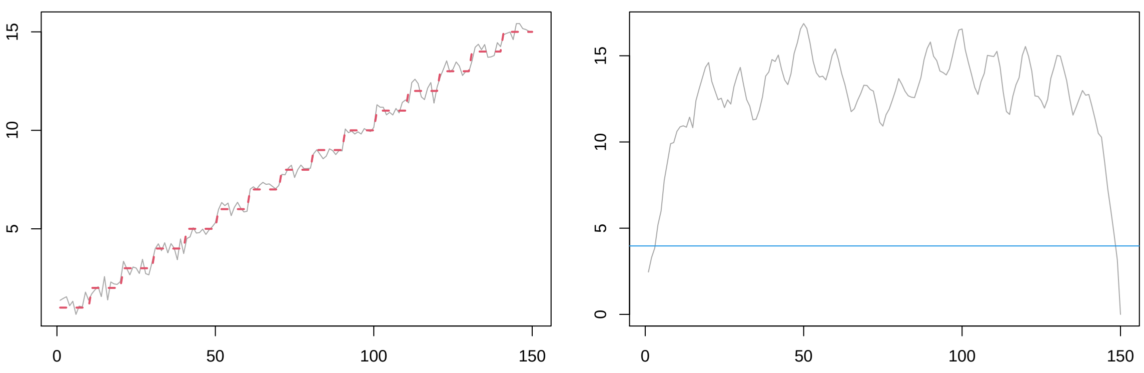

Figure 4: Left: A time series of length 150 with mean changes at locations $10, 20, \ldots, 140$, where the underlying step signal $f_t$ is overlaid as dashed line. Right: The corresponding scaled absolute MOSUM detector $\sigma^{-1}|T_G(k)|$ with $G = 15$; the critical value for level $\alpha = 0.05$ is denoted by a horizontal line. The series is a realization of the model `stairs10` from Fryzlewicz (2014) (see Section 3.1).

defined as

$$\hat{\sigma}^2_{(l,r)} := \frac{1}{r-l+1} \sum_{t=l}^{r} (X_t - \bar{X}_{(l,r)})^2$$

with $\bar{X}_{(l,r)}$ denoting the sample mean of observations $X_l, X_{l+1}, \ldots, X_r$.

Possible choices of $\widehat{\sigma}_k$ for the usage within (1) are the local variance estimators

$$\widehat{\sigma}_k^2 = \frac{1}{2}\left(\widehat{\sigma}^2_{(k-G+1,k)} + \widehat{\sigma}^2_{(k+1,k+G)}\right), \tag{10}$$

$$\widehat{\sigma}_k^2 = \min\{\widehat{\sigma}^2_{(k-G+1,k)}, \widehat{\sigma}^2_{(k+1,k+G)}\}, \quad \text{or} \tag{11}$$

$$\widehat{\sigma}_k^2 = \max\{\widehat{\sigma}^2_{(k-G+1,k)}, \widehat{\sigma}^2_{(k+1,k+G)}\}. \tag{12}$$

The variance estimator in (11) is preferable for the detection of change-points at which not only the mean but also the variance undergoes changes, whereas the estimator in (12) is more robust in the sense that it will prevent the detection of spurious estimators in the presence of time-varying variance; see Sections 4.1 and 4.3 for an example with heteroscedastic data.

It is possible to consider the model (4) with dependent innovations $e_i$ fulfilling weak dependency assumptions. The distributional convergence and thus the thresholds associated with the corresponding critical values as discussed in Section 2.1, remain valid as long as the variance estimators $\widehat{\sigma}_k^2$ are replaced by appropriate estimators for the long-run variance $\tau^2 = \lim_{n\to\infty} \operatorname{var}(\sqrt{n}\bar{e}_n)$ with $\bar{e}_n = n^{-1}\sum_{t=1}^n e_t$. For this, we may use the MOSUM-version of the flat-top kernel estimator (Politis and Romano 1995) calculated with sufficiently large bandwidths (e.g., $G \geq 50$), or the difference-based estimators from Tecuapetla-Gómez and Munk (2017), Dette, Eckle, and Vetter (2020) and Axt and Fried (2019). This is not recommended in the presence of frequent changes, however, as accurate estimation of the long-run variance is typically very difficult and thus estimators based on small and medium-sized samples are not very reliable. An alternative approach is to use the local variance estimators (ignoring the dependence structure) and at the same time slightly increasing the threshold, e.g., using $C_{n,G}(\alpha) \cdot \log(n/G)^\delta$ for some $\delta > 0$ (e.g., $\delta = 0.1$), where $C_{n,G}(\alpha)$ is as in (6).

## 2.4. Choice of bandwidth

In practice, the choice of bandwidth plays a crucial role for the performance of the MOSUM procedure. Smaller bandwidths can detect large jumps even if there are neighboring change-points close by. At the same time, large bandwidths may not be suitable for estimating the locations of such change-points due to *contamination* of the signal by neighboring change-points. Indeed, with large bandwidths, the signal may have a flat top (i.e., the signal has no longer a unique maximum but the maximal value is attained over an interval), see the third and fourth panels in Figure 5 for an example. In this case – while being significant – the maximal point in the corresponding significant neighborhood will effectively be arbitrary on that interval, so it cannot be used for the purpose of change-point localization.

On the other hand, large bandwidths are able to detect small isolated changes, whereas small bandwidths tend to miss such small jumps even if they are surrounded by long stationary stretches. Consider Figure 5 as an example, where a bandwidth of at least $G = 70$ seems to be required to detect the small change at $k_1 = 100$. One remedy for this issue is to adopt multiple bandwidths, which will be discussed in the upcoming Section 2.6.

In a similar manner, for some signals, using the same bandwidth for the left and right summation windows in the MOSUM detector does not provide sufficient flexibility. Consider the case of a small mean change located close to a large mean change, see the top panel of Figure 6 for such an example. The change may be too small to be detected by a small bandwidth, whereas the summation windows will be contaminated by the neighboring large mean change

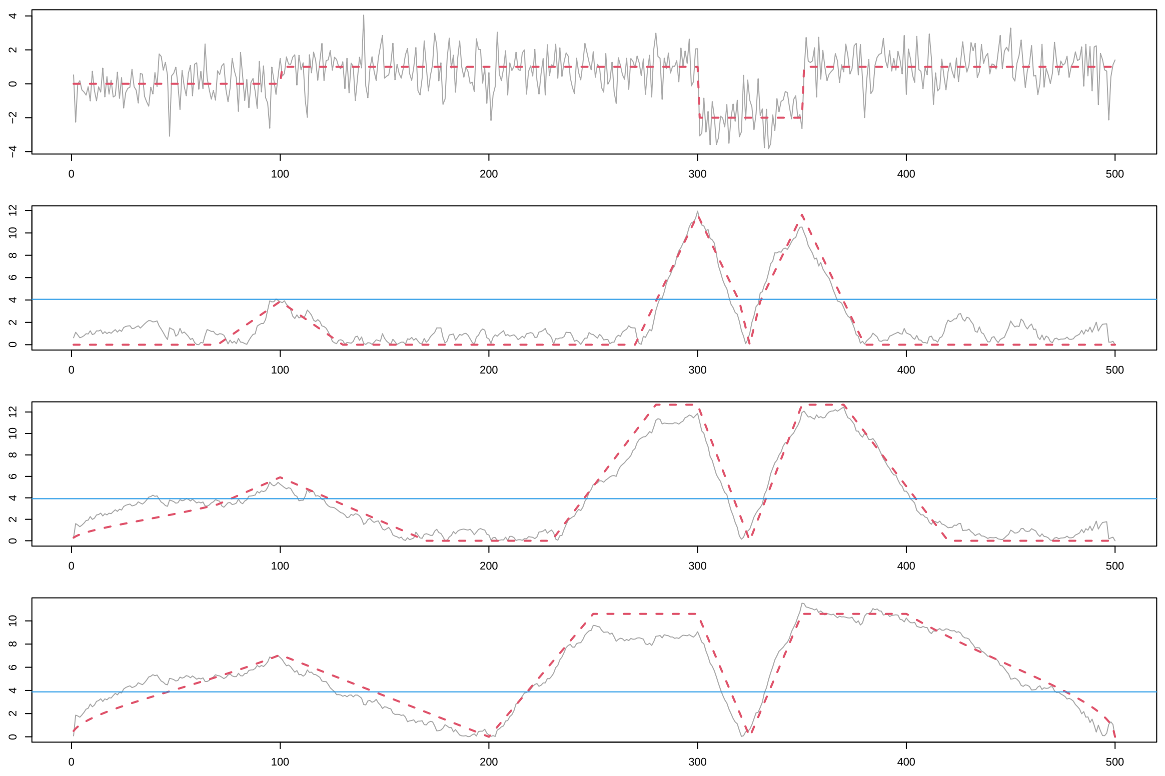

Figure 5: From top to bottom: A time series with mean changes at locations $k_1 = 100$, $k_2 = 300$ and $k_3 = 350$, where the underlying step signal $f_t$ is overlaid as dashed line, and the absolute MOSUM detector $|T_G(k)|$ for $G = 30, 70$ and $100$, where the absolute MOSUM signal $|T_G(k; f)|$ is overlaid as dashed line and the critical value at level $\alpha = 0.05$ is visualized by a solid horizontal line.

when a large bandwidth is used. One way of overcoming this limitation is to consider the use of *asymmetric* bandwidths $\mathbf{G} = (G_\mathrm{l}, G_\mathrm{r})$, which will be discussed in the following Section 2.5.

## 2.5. Asymmetric bandwidths

Let $\mathbf{G} = (G_\mathrm{l}, G_\mathrm{r})$ be a bandwidth, possibly asymmetric (i.e., $G_\mathrm{l} \neq G_\mathrm{r}$). The asymmetric MOSUM detector is defined as

$$T_\mathbf{G}(k) = \sqrt{\frac{G_\mathrm{l} \cdot G_\mathrm{r}}{G_\mathrm{l} + G_\mathrm{r}}} \left( \frac{1}{G_\mathrm{r}} \sum_{t=k+1}^{k+G_\mathrm{r}} X_t - \frac{1}{G_\mathrm{l}} \sum_{t=k-G_\mathrm{l}+1}^{k} X_t \right), \quad k = G_\mathrm{l}, \ldots, n - G_\mathrm{r}. \tag{13}$$

At the left and right boundaries, the values of $T_\mathbf{G}(k)$ can be defined by a CUSUM extension similar to (3):

$$T_\mathbf{G}(k) := \sqrt{\frac{G_\mathrm{l} + G_\mathrm{r}}{k(G_\mathrm{l} + G_\mathrm{r} - k)}} \sum_{t=1}^{k} \left( \bar{X}_{(1, G_\mathrm{l}+G_\mathrm{r})} - X_t \right), \quad k = 1, \ldots, G_\mathrm{l} - 1, \tag{14}$$

and analogously for $k = n - G_\mathrm{r} + 1, \ldots, n$.

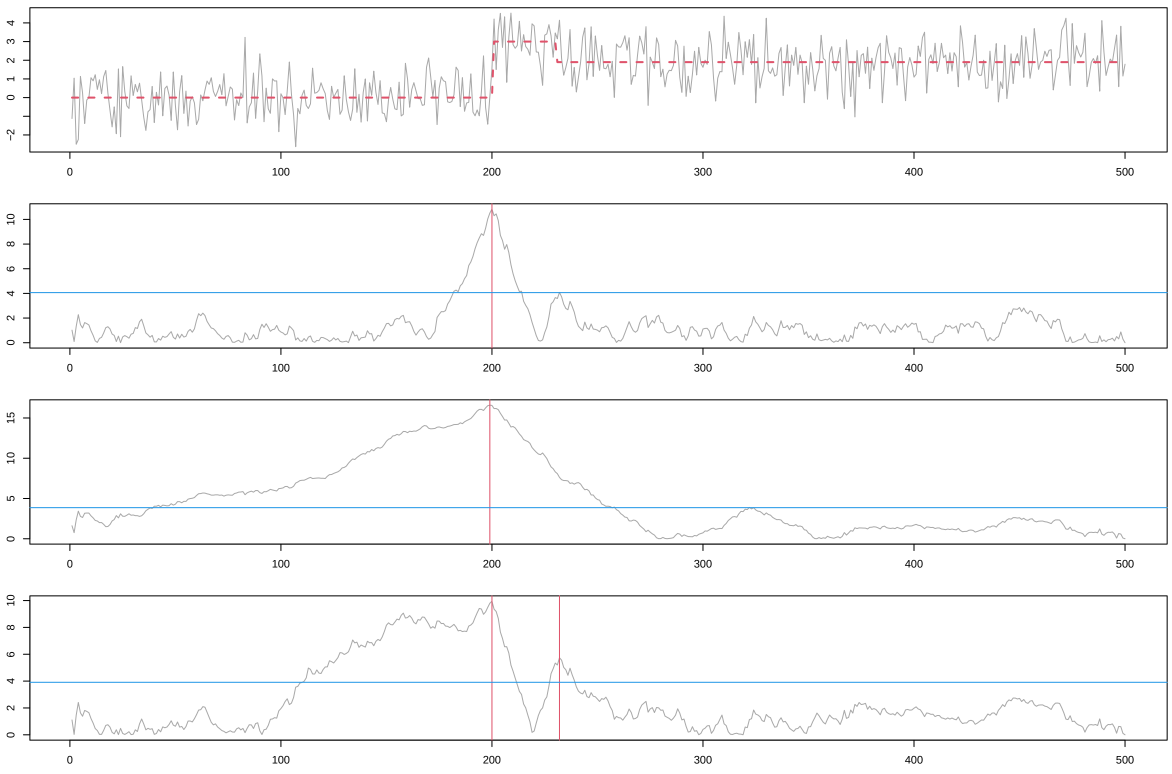

Figure 6: From top to bottom: A time series of length $n = 500$ with two mean changes at the locations 200 and 230 of respective height 3 and $-1.1$, and the corresponding absolute MOSUM detector $|T_{\mathbf{G}}(k)|$ for $k = 1, \ldots, n$ with (symmetric) bandwidths $G = 30, 120$ and $\mathbf{G} = (30, 120)$. Critical value at level $\alpha = 0.05$ is visualized by a solid horizontal line and the change-points estimated according to the $\eta$-rule by solid vertical lines.

The critical value $C_{n,\mathbf{G}}(\alpha)$ for the asymmetric MOSUM test, as well as the corresponding $p$ values $p_{n,\mathbf{G}}$, can be obtained similarly as in the symmetric case, see Section C in the Appendix. As an example, the usefulness of using asymmetric bandwidths is illustrated in the bottom panel of Figure 6.

The extension of the change-point estimators to the asymmetric case is straightforward. The $\varepsilon$-criterion (8) is adjusted to

$$r - l \geq \frac{\varepsilon}{2}(G_{\mathrm{l}} + G_{\mathrm{r}}), \tag{15}$$

and the $\eta$-criterion (9) is adjusted to

$$k^{\circ} = \arg \max_{k^{\circ} - \eta G_{\mathrm{l}} \leq k \leq k^{\circ} + \eta G_{\mathrm{r}}} \frac{|T_{\mathbf{G}}(k)|}{\widehat{\sigma}_k}. \tag{16}$$

The MOSUM-based variance estimators can be defined with asymmetric bandwidths in a similar fashion as the MOSUM detector. When using asymmetric bandwidths, it is advisable to avoid pairs of bandwidths that are too unbalanced, both in view of the asymptotic theory and the finite sample performance as is well-known from the two-sample testing literature. To avoid such situations, one can impose an upper bound on $\max(G_{\mathrm{l}}, G_{\mathrm{r}}) / \min(G_{\mathrm{l}}, G_{\mathrm{r}})$.

**Algorithm 1:** Multiscale MOSUM procedure with bottom-up merging.

```
  input  : Data $(X_1, \ldots, X_n)$, set $\mathcal{G}$ of symmetric bandwidths, $\alpha, \eta \in (0,1)$
1 Initialize $\mathcal{P} \leftarrow \mathcal{K} \leftarrow \emptyset$
  /* $\mathcal{P}$ contains the pool of candidates to be pruned down and $\mathcal{K}$ the final
     estimators                                                              */
  /* Step 1:  Generate candidates                                            */
2 for $G \in \mathcal{G}$ do
3     $\mathcal{P}_G \leftarrow$ set of MOSUM change-point estimators obtained with bandwidth $G$ and
        critical value $C_{n,G}(\alpha)$ according to criterion (9)
4     for $\widehat{k} \in \mathcal{P}_G$ do Add $(\widehat{k}, G)$ to $\mathcal{P}$
5 end
  /* Step 2:  Merging in the increasing order with respect to $G$            */
6 for $(\widehat{k}_\circ, G) \in \mathcal{P}$ in the increasing order with respect to $G$ do
7     if $\min_{\widehat{k} \in \mathcal{K}} |\widehat{k}_\circ - \widehat{k}| \geq \eta G$ then Add $\widehat{k}_\circ$ to $\mathcal{K}$
8 end
  output: $\mathcal{K}$
```

## 2.6. Multiple bandwidths

In Section 2.4, we showed that the use of a single bandwidth (symmetric or asymmetric) may not be adaptive enough to detect different types of changes, which advocates the use of multiple bandwidths. On the other hand, this may introduce spurious change-point estimates and/or multiple estimates that relate to the same underlying true change-point. For these reasons, an additional merging step is necessary. Below, we introduce two procedures for pruning down a set of candidate change-point estimators obtained from applying the MOSUM procedure with multiple bandwidths.

*Multiscale MOSUM procedure with bottom-up merging*

As argued in Section 2.4, one possible issue of large bandwidths is that their summation windows may contain, and thus be contaminated by, several changes, rendering the corresponding estimators unreliable. Motivated by this consideration and following Messer, Kirchner, Schiemann, Roeper, Neininger, and Schneider (2014), it is reasonable to keep all the candidate estimators from the smallest bandwidth and, iteratively moving on to the next smallest bandwidth, only keep those that cannot be accounted for by the previously accepted estimates.

To elaborate, in the order of increasing bandwidths, we check whether there exists a previously accepted change-point $\widehat{k}$ in the final set of estimators $\mathcal{K}$, which is close to the current candidate $\widehat{k}_\circ$ detected with the bandwidth $G$, in the sense that $|\widehat{k}_\circ - \widehat{k}| < \eta G$ for some $\eta \in (0,1)$. If so, we conclude that the change-point estimated by $\widehat{k}_\circ$ has already been detected by $\widehat{k} \in \mathcal{K}$, and thus discard $\widehat{k}_\circ$; otherwise we add it to $\mathcal{K}$. We note that the tuning parameter $\eta$ bears close resemblance to the parameter $\eta$ from the $\eta$-criterion in (9). For this reason, the implementation of the procedure in the R package **mosum** also uses the $\eta$-criterion to obtain the candidate set in this bottom-up merging. Algorithm 1 depicts a comprehensive high-level description of the procedure. See Messer *et al.* (2014) for further details and a more comprehensive discussion of the rationale behind this approach in the context of change-point estimation for point processes.

The advantages of the bottom-up merging include its ease of implementation and computational speed. There are, however, several drawbacks as well: First, this approach only works with multiple *symmetric* bandwidths since a set of asymmetric bandwidths does not provide a canonical ordering. Secondly, the effect of multiple testing should be taken into account, which has been done in Messer *et al.* (2014) for the problem considered therein. More importantly, the bandwidths in consideration need to be large enough so that the critical value from the asymptotic distribution is meaningful. For example, there may be false positives in $\mathcal{P}$, in the sense that for some $(\widehat{k}, G) \in \mathcal{P}$ – particularly for small $G$ – the detection interval $(\widehat{k} - G, \widehat{k} + G]$ does not include any true change-point, and the bottom-up merging may fail to prevent such tuple from entering $\mathcal{K}$. To avoid this, Messer *et al.* (2014) propose to use only bandwidths of order $n$. However, for signals with frequent change-points in some parts but no jumps in other parts, this poses as a limitation.

### *Multiscale MOSUM procedure with localized pruning*

Niu and Zhang (2012) considered using an information criterion to prune down a set of candidate change-point estimators generated by a multiscale MOSUM procedure, say $\mathcal{P}$, by performing an exhaustive search over every possible combination of the candidate estimators, and a similar idea has also been explored by Yau and Zhao (2016). Such an approach quickly becomes computationally infeasible as the number of candidate models grows exponentially with $|\mathcal{P}|$. Moreover, it does not take advantage of that each estimator is detected within a local interval determined by the summation windows. That is, for any estimator $\widehat{k}$ detected with bandwidths $\mathbf{G} = (G_\mathrm{l}, G_\mathrm{r})$, we have the information about its detection interval $\mathcal{I}(\widehat{k}) := (\widehat{k} - G_\mathrm{l}, \widehat{k} + G_\mathrm{r}]$ which can be utilized in the pruning step. In Algorithm 2, we present a comprehensive high-level description of the information criterion-based localized pruning method applicable with the multiscale MOSUM procedure; in contrast to Algorithm 1, it allows the use of asymmetric bandwidths.

In Step 2 of Algorithm 2, we determine the order in which the change-point candidates $(\widehat{k}, \mathbf{G}'' \in \mathcal{P}$ are to be processed. For this, we adopt a sorting function $c$, which assesses the "significance" of each candidate according to the inverse of the $p$ value $(c_p)$ associated with its detection or the (scaled) jump size $(c_J)$ as defined below:

$$c_p(\widehat{k}, \mathbf{G}) = \frac{1}{p_{n,\mathbf{G}}(\widehat{\sigma}_{\widehat{k}}^{-1} |T_\mathbf{G}(\widehat{k})|)} \quad \text{and} \quad c_J(\widehat{k}, \mathbf{G}) = \sqrt{\frac{G_\mathrm{l} + G_\mathrm{r}}{G_\mathrm{l} G_\mathrm{r}}} \frac{|T_\mathbf{G}(\widehat{k})|}{\widehat{\sigma}_{\widehat{k}}}. \tag{17}$$

All change-point candidates are processed one by one in the decreasing order according to the chosen sorting function. In case of ties, we order the candidates with respect to the length of their detection intervals and associated bandwidths, preferring those returned at shorter bandwidths.

In Step 2.2 of Algorithm 2, we identify a local environment $(\widehat{k}_L, \widehat{k}_R]$ around the estimator $\widehat{k}_\circ$ considered at the current iteration, over which we perform the exhaustive search for change-points. It is defined by the currently surviving candidates (which either have already been selected or are still to be processed, contained in $\mathcal{C}$) closest to $\widehat{k}_\circ$, while their detection intervals do not overlap with that of $\widehat{k}_\circ$ (see lines 8–9 of Algorithm 2); any candidate falling within

**Algorithm 2:** Multiscale MOSUM procedure with localized merging.

```
   input  : Data $(X_1, \ldots, X_n)$, set $\mathcal{G}$ of (possibly asymmetric) bandwidths, $\alpha$ and
            $\varepsilon$ or $\eta \in (0,1)$, sorting function $c$ as $c_p$ or $c_J$ from (17)
 1 Initialize $\mathcal{P} \leftarrow \mathcal{C} \leftarrow \mathcal{K} \leftarrow \emptyset$
   /* $\mathcal{P}$ contains the pool of candidates to be pruned down, $\mathcal{C}$ the
      currently surviving candidates and $\mathcal{K}$ the final estimators         */
   /* Step 1: Generate candidates                                                   */
 2 for $\mathbf{G} \in \mathcal{G}$ do
 3     $\mathcal{P}_{\mathbf{G}} \leftarrow$ set of MOSUM change-point estimators obtained with bandwidth $\mathbf{G}$ and
         critical value $C_{n,\mathbf{G}}(\alpha)$ according to criteria (15) or (16)
 4     for $\widehat{k} \in \mathcal{P}_{\mathbf{G}}$ do Add $(\widehat{k}, \mathbf{G})$ to $\mathcal{P}$
 5 end
   /* Step 2: Prune down $\mathcal{P}$ in decreasing order with respect to $c(\cdot)$  */
 6 while $\mathcal{P}$ is not empty do
       /* Step 2.1: Find the candidate of consideration $(\widehat{k}_\circ, \mathbf{G}_\circ)$      */
 7     $(\widehat{k}_\circ, \mathbf{G}_\circ) \leftarrow \operatorname{argmax}_{(\widehat{k},\mathbf{G}) \in \mathcal{P}} c(\widehat{k}, \mathbf{G})$
       /* Step 2.2: Find the local environment $(\widehat{k}_L, \widehat{k}_R]$ and the set of
          conflicting candidates $\mathcal{D}$                                      */
 8     $\widehat{k}_L \leftarrow \max\{\widehat{k} \in \mathcal{C} \cup \{0\}, \widehat{k} < \widehat{k}_\circ \colon \widehat{k} \in \mathcal{K} \text{ or } \mathcal{I}(\widehat{k}) \cap \mathcal{I}(\widehat{k}_\circ) = \emptyset\}$
 9     $\widehat{k}_R \leftarrow \min\{\widehat{k} \in \mathcal{C} \cup \{n\}, \widehat{k} > \widehat{k}_\circ \colon \widehat{k} \in \mathcal{K} \text{ or } \mathcal{I}(\widehat{k}) \cap \mathcal{I}(\widehat{k}_\circ) = \emptyset\}$
10     $\mathcal{D} \leftarrow \{(\widehat{k}, \mathbf{G}) \in \mathcal{P} \colon \widehat{k}_L < \widehat{k} < \widehat{k}_R\}$
       /* Step 2.3: Perform exhaustive search on conflicting candidates */
       /* See Algorithm 3 in Appendix                                   */
11     $\widehat{\mathcal{A}} \leftarrow$ set selected according to (II) with $\mathcal{C}$, $\mathcal{D}$ and $(\widehat{k}_L, \widehat{k}_R]$
       /* Step 2.4: Update $\mathcal{P}$, $\mathcal{C}$ and $\mathcal{K}$ based on the exhaustive search   */
12     $\mathcal{R} \leftarrow \{(\widehat{k}_\circ, \mathbf{G}_\circ)\} \cup \{(\widehat{k}, \mathbf{G}) \in \mathcal{D} \colon \widehat{k} \in [\min \widehat{\mathcal{A}}, \max \widehat{\mathcal{A}}]\}$
13     if $\widehat{k}_L \in \mathcal{K} \cup \{0\}$ then $\mathcal{R} \leftarrow \mathcal{R} \cup \{(\widehat{k}, \mathbf{G}) \in \mathcal{D} \colon \widehat{k} \in (\widehat{k}_L, \min \widehat{\mathcal{A}})\}$
14     if $\widehat{k}_R \in \mathcal{K} \cup \{n\}$ then $\mathcal{R} \leftarrow \mathcal{R} \cup \{(\widehat{k}, \mathbf{G}) \in \mathcal{D} \colon \widehat{k} \in (\max \widehat{\mathcal{A}}, \widehat{k}_R)\}$
15     Add $\mathcal{A}$ to $\mathcal{K}$, remove $\mathcal{R}$ from $\mathcal{P}$ and update $\mathcal{C} \leftarrow \mathcal{P} \cup \mathcal{A}$
16 end
   output: $\mathcal{K}$
```

$(\widehat{k}_L, \widehat{k}_R]$ is regarded as belonging to the set of *conflicting* candidates $\mathcal{D}$, in the sense that their detection bandwidths overlap with that of $\widehat{k}_\circ$.

For the exhaustive search performed on $\mathcal{D}$ in Step 2.3, in which we determine the inclusion or exclusion of each change-point candidate in $\mathcal{D}$, the Schwarz Criterion (SC) is adopted. Specifically, for any $\mathcal{A} \subset \mathcal{D}$, SC is calculated as

$$\mathrm{SC}(\mathcal{A}|\mathcal{C}, (\widehat{k}_L, \widehat{k}_R]) = \frac{n}{2} \log \mathrm{RSS}(\mathcal{A} \cup \{\mathcal{C} \setminus (\widehat{k}_L, \widehat{k}_R)\}) + (|\mathcal{A}| + |\mathcal{C} \setminus (\widehat{k}_L, \widehat{k}_R)|) \cdot p(n)^{\varrho}, \qquad (18)$$

where

$$\mathrm{RSS}(\mathcal{Q}) = \sum_{j=0}^{m} \sum_{t=\widetilde{k}_j+1}^{\widetilde{k}_{j+1}} \left( X_t - \bar{X}_{(\widetilde{k}_j+1, \widetilde{k}_{j+1})} \right)^2 \tag{19}$$

denotes the residual sum of squares given some set $\mathcal{Q} = \{\widetilde{k}_1, \ldots, \widetilde{k}_m\}$ of change-point candidates (with the convention $\widetilde{k}_0 := 0$ and $\widetilde{k}_{m+1} := n$). If the user is confident that a normality assumption on the data is reasonable, or at least that all the moments of $X_t$ exist, we recommend the choice of $p(n) = \log(n)$ with $\varrho$ slightly larger than one (e.g., $\varrho = 1.01$). Otherwise $p(n) = n$ should be used with an exponent $2/\nu < \varrho < 1$, where $\nu$ is the number of moments that one believes to exist for $\mathsf{E}(|e_t|^\nu) < \infty$ (see Kühn 2001).

Among the $2^{|\mathcal{D}|}$ subsets of $\mathcal{D}$, we select the subset $\widehat{\mathcal{A}}$ according to the following rule. Let $\mathcal{F}$ denote the collection of all subsets $\mathcal{A} \subset \mathcal{D}$ satisfying:

$$\text{adding further change-point candidates to } \mathcal{A} \text{ monotonically increases the SC,} \tag{I}$$

and denote by $m^* = \min_{\mathcal{A} \in \mathcal{F}} |\mathcal{A}|$. Then, we select

$$\widehat{\mathcal{A}} = \arg\min\{\mathcal{A} \subset_R \mathcal{A}' \in \mathcal{F} \text{ with } m^* \le |\mathcal{A}'| \le m^* + 2 : \mathrm{SC}(\mathcal{A}|\mathcal{C}, (\widehat{k}_L, \widehat{k}_R])\} \tag{II}$$

where, by $\mathcal{A} \subset_R \mathcal{A}' = \{\widehat{k}_{i_1} < \widehat{k}_{i_2} < \ldots < \widehat{k}_{i_m}\}$, we denote that $\mathcal{A}$ is a "restricted" subset of $\mathcal{A}'$ which contains all *inner* elements of $\mathcal{A}'$ (if any), while the first and the last elements of $\mathcal{A}'$ may or may not be included in $\mathcal{A}$; i.e., $\mathcal{A}' \setminus \mathcal{A} \subset \{\widehat{k}_{i_1}, \widehat{k}_{i_m}\}$. If there are multiple subsets yielding the minimum SC in (II), we choose the one with the minimum cardinality; if there are still ties, we arbitrarily select one. An efficient algorithm implementing the exhaustive search according to (II) is outlined in Section A of the Appendix.

Finally, $\widehat{\mathcal{A}}$ is added to the final set of estimated change-points $\mathcal{K}$ in Step 2.4 of Algorithm 2, while candidates in $\mathcal{D}$ which do not merit further consideration are removed from $\mathcal{P}$, and the set of currently surviving candidates $\mathcal{C}$ is updated accordingly. Repeatedly performing the localized exhaustive search, the algorithm is terminated when $\mathcal{P}$ is empty.

We refer to Cho and Kirch (2020b) for the discussion on the theoretical properties of the multiscale MOSUM procedure with localized pruning, where it is shown to achieve consistency both in the total number and the locations of change points under general conditions.

## 2.7. Bootstrap confidence intervals

Consider a set $\widehat{k}_1, \ldots, \widehat{k}_{\widehat{N}}$ of change-point estimates returned by, e.g., the multiscale MOSUM procedures (Algorithms 1–2) or by the single-bandwidth MOSUM procedure from Section 2.2. Conditional on consistent estimation of the multiple change-points, both in their total number (such that $\widehat{N} = N$) and locations, we can construct confidence intervals for the locations of change-points via bootstrapping. Below we provide the simplest version of the bootstrap scheme for i.i.d. innovations.

Denote the (possibly asymmetric) detection bandwidths associated with $\widehat{k}_j$ by $\widehat{\mathbf{G}}_j = (\widehat{G}_{\mathrm{l}\,j}, \widehat{G}_{\mathrm{r}\,j})$, and let $I_j := \{\widehat{k}_{j-1}+1, \ldots, \widehat{k}_j\}$ for $j = 1, \ldots, N+1$, where the convention $\widehat{k}_0 := 0$ and $\widehat{k}_{N+1} := n$ is employed. Note that $I_j$ represents all time points between the $(j-1)$th and the $j$th estimated change-points, so that the underlying step signal is assumed to be (approximately) constant within this interval. A bootstrap replicate $\{X_t^* : t \in I_j\}$ of the observations $\{X_t : t \in I_j\}$

can be obtained by drawing a random sample of size $|I_j|$ from $\{X_t \colon t \in I_j\}$ (with replacement). Repeating this for all $j = 1, \ldots, \widehat{N} + 1$ yields a bootstrap replicate $\{X_1^*, \ldots, X_n^*\}$ of the observed time series. Then a bootstrap replicate $k_j^*$ of $\widehat{k}_j$ is obtained as

$$k_j^* = \arg\max_{\widehat{k}_j - \widehat{G}_{\mathrm{l}\,j}+1 \le k \le \widehat{k}_j + \widehat{G}_{\mathrm{r}\,j}} |T^*_{\widehat{\mathbf{G}}_j}(k)|,$$

where $T^*_{\widehat{\mathbf{G}}_j}(k) = T_{\widehat{\mathbf{G}}_j}(k; X_1^*, \ldots, X_n^*)$ is defined as in (13), with $X_t$ replaced by its bootstrap replicates $X_t^*$. From this, confidence intervals for the change-point locations can readily be computed.

To elaborate, assume that for each change-point estimate $\widehat{k}_j$, we have $B$ bootstrap replicates $k^*_{j,b}$, $1 \le b \le B$. Denote by $M_j(\alpha)$ the empirical $(1-\alpha/2)$-quantile of $\{|k^*_{j,1} - \widehat{k}_j|, \ldots, |k^*_{j,B} - \widehat{k}_j|\}$. Then a *pointwise* $100(1-\alpha)\%$-confidence interval $K_j^{\mathrm{pw}}(\alpha)$ for $k_j$ can be constructed as

$$K_j^{\mathrm{pw}}(\alpha) := \left[\widehat{k}_j - M_j(\alpha), \widehat{k}_j + M_j(\alpha)\right]. \tag{20}$$

To construct *uniform* confidence intervals for $k_1, \ldots, k_N$, we take into account the multiple testing problem that arises when considering multiple estimates, by computing $M(\alpha)$ as

$$M(\alpha) = \min\left\{c : \frac{1}{B}\sum_{b=1}^{B} \mathbb{I}\left\{\max_{1\le j\le N}\left(\frac{\widehat{d}_j^2}{\widehat{\sigma}_j^2}\left|k^*_{j,b} - \widehat{k}_j\right|\right) \le c\right\} \ge 1-\alpha\right\},$$

where $\widehat{d}_j = \bar{X}_{(\widehat{k}_j+1, \widehat{k}_{j+1})} - \bar{X}_{(\widehat{k}_{j-1}+1, \widehat{k}_j)}$ denotes the estimated height of the jump at $t = k_j$, and

$$\widehat{\sigma}_j^2 = \frac{1}{\widehat{k}_{j+1} - \widehat{k}_{j-1} - 2}\left(\sum_{t=\widehat{k}_j+1}^{\widehat{k}_{j+1}} (X_t - \bar{X}_{(\widehat{k}_j+1, \widehat{k}_{j+1})})^2 + \sum_{t=\widehat{k}_{j-1}+1}^{\widehat{k}_j} (X_t - \bar{X}_{(\widehat{k}_{j-1}+1, \widehat{k}_j)})^2\right)$$

the pooled innovation sample variance. Then, uniform $100(1-\alpha)\%$-confidence intervals for $k_1, \ldots, k_N$ are given by

$$K_j^{\mathrm{unif}}(\alpha) := \left[\widehat{k}_j - \frac{M(\alpha)\widehat{\sigma}_j^2}{\widehat{d}_j^2}, \widehat{k}_j + \frac{M(\alpha)\widehat{\sigma}_j^2}{\widehat{d}_j^2}\right], \quad j = 1, \ldots, N. \tag{21}$$

Note that for each $j = 1, \ldots, N$, if either of the two endpoints of $K_j^{\mathrm{pw}}(\alpha)$ falls outside the detection interval $(\widehat{k}_j - \widehat{G}_{\mathrm{l}\,j}, \widehat{k}_j + \widehat{G}_{\mathrm{r}\,j}]$, we trim off the confidence intervals so that $K_j^{\mathrm{pw}}(\alpha) \subset (\widehat{k}_j - \widehat{G}_{\mathrm{l}\,j}, \widehat{k}_j + \widehat{G}_{\mathrm{r}\,j}]$; a similar step is taken for the uniform confidence intervals $K_j^{\mathrm{unif}}(\alpha)$.

## 3. Introduction to the package

In this section, we introduce the main functions of the **mosum** packages for multiple change-point detection, `mosum`, `multiscale.bottomUp` and `multiscale.localPrune`, as well as functions for producing confidence intervals for change-point locations and visualising the results. We first start with a brief guide on how to generate piecewise stationary time series popularly used as benchmark in the change-point literature in Section 3.1. In Section 3.2, we explain the implementation of the (single-bandwidth) MOSUM procedure described in Sections 2.1–2.2. In the subsequent Sections 3.3 and 3.4, we introduce the implementations of the multiscale MOSUM procedures given in Algorithms 1–2 from Section 2.6, respectively. We present how bootstrap confidence intervals for change-points can be obtained in Section 3.6, and discuss some tools for visualization of the outcome from multiscale change-point analysis in Section 3.7.

## 3.1. Generating piecewise stationary time series

The function `testData` generates piecewise stationary time series with i.i.d. innovations.

```
testData(model = "custom", lengths = NULL, means = NULL,
  sds = NULL, rand.gen = rnorm, seed = NULL, ...)
```

It takes the following arguments:

- `model`: A string specifying the model from which a realization is to be generated. The default choice is `"custom"`, which allows the user to parse the piecewise stationary model with the arguments `lengths`, `means` and `sds`. In addition, five change-point models `"blocks"`, `"fms"`, `"mix"`, `"teeth10"` and `"stairs10"` from Fryzlewicz (2014) have been implemented. If one of these models is chosen, the arguments `lengths`, `means` and `sds` are ignored.
- `lengths`: Lengths of the piecewise stationary segments, represented as an integer vector. Only in use if `model = "custom"`.
- `means`, `sds`: Means and deviation scalings of the piecewise stationary segments, represented as numeric vectors. The i.i.d. innovations generated from the distribution specified by `rand.gen` are multiplied by the respective entry of `sds` over each segment. Note that the entries of `sds` coincides with the standard deviation in case of standard normal innovations (`rand.gen = rnorm`). Only in use if `model = "custom"`.
- `rand.gen`: A function to generate the time series innovations.
- `seed`: A seed value to be parsed to `set.seed` (optional) preceding a call of `rand.gen`. If `seed` is set to `NULL`, then `set.seed` is not called.

The function `testData` returns a list consisting of a numeric vector containing a realization of the specified time series model, as well as the deterministic piecewise constant signal $f_t$ and the scaling vector applied to the innovations. As an example, we consider a realization of the `mix` time series model, visually overlaid by the corresponding step signal:

```
R> td <- testData(model = "mix", seed = 1234)
R> plot(ts(td$x), col = "darkgray")
R> lines(td$mu, col = 2, lty = 2, lwd = 2)
```

The result is plotted in Figure 7.

## 3.2. MOSUM procedure with a single bandwidth

The single-bandwidth MOSUM procedure from Section 2.2 for multiple change-point estimation is implemented in the function `mosum`:

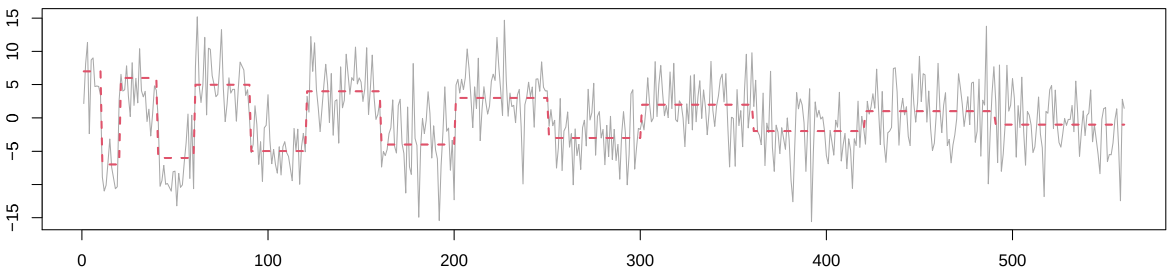

Figure 7: A realization from the `mix` time series model from Fryzlewicz (2014), with the piecewise constant signal overlaid in dashed line.

```
mosum(x, G, G.right = G,
  var.est.method = c("mosum", "mosum.min", "mosum.max", "custom")[1],
  var.custom = NULL,  boundary.extension = TRUE,
  threshold = c("critical.value", "custom")[1], alpha = 0.1,
  threshold.custom = NULL, criterion = c("eta", "epsilon")[1],
  eta = 0.4, epsilon = 0.2, do.confint = FALSE, level = 0.05, N_reps = 1000)
```

The function takes the following arguments:

- `x`: Input data $X_1, \dots, X_n$, i.e., a univariate time series represented as a numeric vector (of length $n$) or an object of class '`ts`'.
- `G`: Bandwidth $G$, i.e., a single positive integer smaller than $n/2$. Alternatively, a single numeric value in the interval $(0, 0.5)$ describing $G$ as a fraction of the data length $n$ can be given.
- `G.right`: Length of the right summation window $G_\mathrm{r}$, i.e., a single positive integer, if an asymmetric bandwidth $\mathbf{G} = (G_\mathrm{l}, G_\mathrm{r})$ is used. As with `G`, it can alternatively be given as a single numeric value in $(0, 0.5)$. When $\max(G_\mathrm{l}, G_\mathrm{r})/\min(G_\mathrm{l}, G_\mathrm{r}) < 4$ while `threshold = "critical.value"`, a warning message is generated; see also Section 2.5 for a brief discussion.
- `var.est.method`: A string encoding how the local variance estimation $\widehat{\sigma}^2_k$ shall be conducted. Currently implemented are:
  - `"custom"`: The local variance estimates supplied by the user; in this case, these values can be parsed as a numeric vector of length $n$ with the argument `var.custom`;
  - `"mosum"`: The MOSUM-based variance estimator from (10) is used;
  - `"mosum.min"`: the MOSUM-based variance estimator from (11) is used;
  - `"mosum.max"`: the MOSUM-based variance estimator from (12) is used.
- `var.custom`: The custom local variance estimates $\widehat{\sigma}^2_k$ for $k = 1, \dots, n$ as a numeric vector of length $n$ containing positive values. Only in use if `var.est.method = "custom"`.
- `boundary.extension`: Logical variable indicating whether the values $T_\mathbf{G}(k)$ for $1 \leq k \leq G_\mathrm{l} - 1$ and $n - G_\mathrm{r} + 1 \leq k \leq n$ shall be padded with CUSUM values, see (3). If `boundary.extension = FALSE`, these values will be evaluated as `NA`.

- `threshold`: A string indicating which threshold should be used to determine significance of the scaled absolute MOSUM detector. By default, the asymptotic critical value $C_{n,G}(\alpha)$ from (6) is used, where the significance level $\alpha$ is given by the parameter `alpha`. Alternatively, with `threshold = "custom"`, it is possible to parse a user-defined numerical value with the argument `threshold.custom`. The latter case might be used e.g., in case of dependent observations, see the discussion at the end of Section 2.3.
- `alpha`: A single numeric value in $(0, 1)$ representing the significance level for the critical value. Only in use if `threshold = "critical.value"`.
- `threshold.custom`: A numeric value greater than 0 to be used as the threshold for the significance of the scaled absolute MOSUM detector. Only in use if `threshold = "custom"`.
- `criterion`: A string indicating which change-point estimation criterion shall be employed. Possible options are `"epsilon"` and `"eta"` for the $\varepsilon$-criterion and $\eta$-criterion, respectively (see (8), (9), (15) and (16)).
- `epsilon`: A numeric value in $(0, 1]$ for $\varepsilon$ in the $\varepsilon$-criterion. Only in use if `criterion = "epsilon"`.
- `eta`: A numeric value greater than 0 for $\eta$ in the $\eta$-criterion. Only in use if `criterion = "eta"`.
- `do.confint`: A boolean argument indicating whether to compute the confidence intervals for change-points.
- `level`: A single numeric value in $(0, 1)$ representing the confidence level for the confidence intervals. Only in use if `do.confint = TRUE`.
- `N_reps` : A single positive integer representing the number of bootstrap replicates to be generated for confidence interval construction. Only in use if `do.confint = TRUE`.

The function `mosum` is flexible in that it allows for the user to supply custom variance and threshold values using the arguments `var.custom` and `threshold.custom`, respectively, as well as providing the theoretically-motivated default choices discussed in Sections 2.1 and 2.3.

When called, `mosum` returns an `S3` object of class '`mosum.cpts`', containing the following entries (apart from the call arguments):

- `stat`: Scaled absolute MOSUM detector $\widehat{\sigma}_k^{-1}|T_{\mathbf{G}}(k)|$ for $1 \le k \le n$, as a numeric vector of length $n$.
- `rollsums`: Unscaled MOSUM detector $T_{\mathbf{G}}(k)$ for $1 \le k \le n$, as a numeric vector of length $n$.
- `var.estimation`: Values of $\widehat{\sigma}_k^2$ as a numeric vector of length $n$ estimated as specified by `var.est.method`.
- `cpts`: A vector containing the locations of the estimated change-points.
- `cpts.info`: A data frame containing the estimated change-points and their respective detection bandwidths, asymptotic $p$ values of the MOSUM statistics and scaled change heights (obtained as in (17)) as columns.

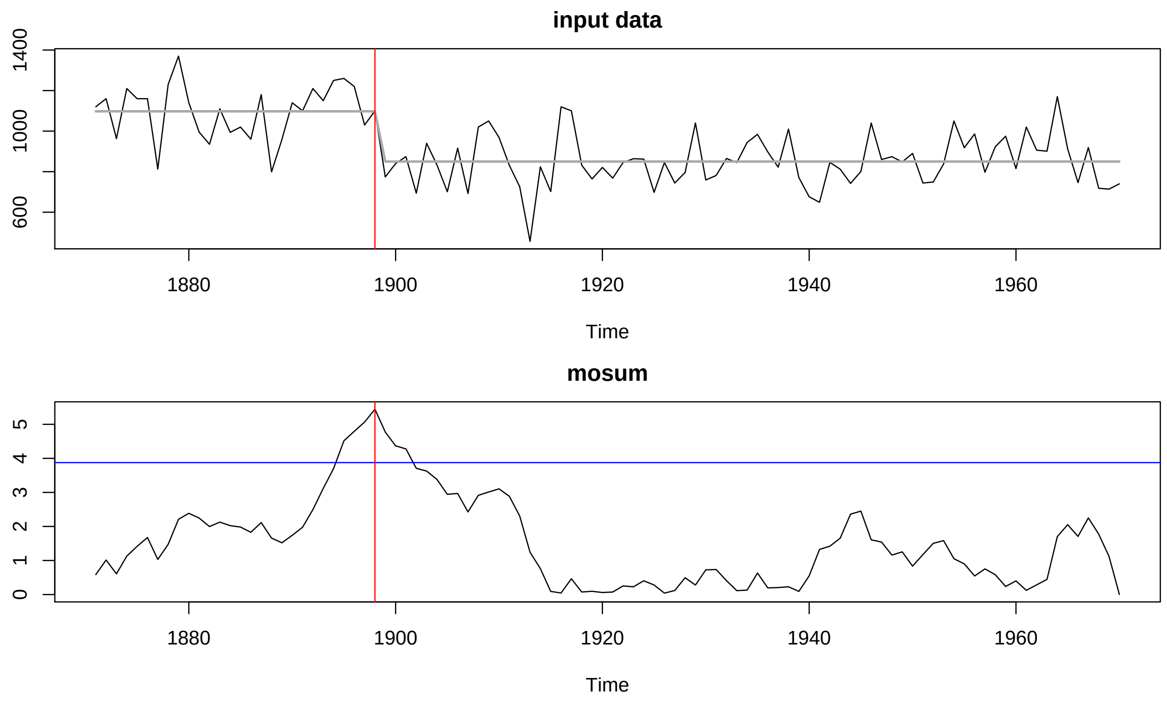

Figure 8: Annual flow of the Nile at Aswan from 1871 to 1970 with the estimated piecewise constant signal (above) and the corresponding values of the scaled absolute MOSUM detector with bandwidth $G = 20$ (below). The critical vale is visualized by a solid horizontal line and the location of the estimated change-point location by a solid vertical line.

- `ci`: An S3 object of class '`cpts.ci`' containing confidence intervals for the change-points. Returned iff `do.confint = TRUE`; see Section 3.6 for further details.

S3 objects of class '`mosum.cpts`' are supported by `plot`, `summary`, `print` and `confint` methods.

As an illustration, we analyse `Nile`, a time series of length $n = 100$ containing the annual flow of the Nile at Aswan from the R package **datasets** (see `help(Nile)` for further information about the dataset). Using the argument `display` of `plot.mosum.cpts`, we can visualize either the input time series or the scaled absolute MOSUM detector along with the estimated change-points.

```
R> m <- mosum(Nile, G = 20,  alpha = 0.05)
R> par(mfcol = c(2, 1), mar = c(4, 2.5, 2.5, 0.5))
R> plot(m, display = "data")
R> plot(m, display = "mosum")
```

The result is shown in Figure 8. It can be seen that the scaled MOSUM detector exceeds the critical value $C_{n,G}(0.05)$ in the years 1895–1901. Applying the $\eta$-criterion described in (9) with the default choice $\eta = 0.4$, a single change-point is estimated at $\alpha = 0.05$ as below.

```
R> summary(m)
```

```
change-points estimated at alpha = 0.05 according to eta-criterion
 with eta = 0.4 and mosum variance estimate:

      cpts  G.left  G.right  p.value   jump
[1,]    28      20       20  0.00308  1.721
```

The estimated change-point location at $k = 28$, where the scaled absolute MOSUM detector attains its maximum, coincides with the year 1898, which is close to the beginning of the construction of the Aswan Low Dam in 1899. Besides the estimated locations of change-points, `summary` of '`mosum.cpts`' objects extract and print the information contained in the entry `cpts.info`.

## 3.3. Multiscale MOSUM procedure with bottom-up merging

The function `multiscale.bottomUp` provides an implementation of the multiscale MOSUM procedure with bottom-up merging described in Algorithm 1 from Section 2.6:

```
multiscale.bottomUp(x, G = bandwidths.default(length(x),
  G.min = max(20, ceiling(0.05*length(x)))),
  threshold = c("critical.value", "custom")[1],
  alpha = 0.1, threshold.function = NULL, eta = 0.4,
  do.confint = FALSE, level = 0.05, N_reps = 1000, ...)
```

Apart from those passed to the function `mosum` (including `...`), it accepts the following arguments:

- `G`: A set $\mathcal{G}$ of (symmetric) bandwidths, represented as a vector of integers smaller than $n/2$, or numeric values in $(0, 0.5)$ describing the bandwidths relative to $n$. When the smallest bandwidth is smaller than $\min(20, 0.05n)$ (0.05 in the case of the relative bandwidth) while `threshold = "critical.value"`, a warning message is generated, see the discussion at the end of Section 2.6.1. By default, it is given by `bandwidths.default` (see Section 3.5), with its arguments set to avoid triggering the warning.
- `threshold`: it is possible to parse a user-defined threshold function with the argument `threshold.function` when `threshold = "custom"`; see Section 4.2 for an example.
- `threshold.function`: A user-specified function of the form `function(G, n, alpha)` for computing a bandwidth-dependent threshold of significance. Only in use if `threshold = "custom"`.

When called, `multiscale.bottomUp` returns an S3 object of class '`multiscale.cpts`', consisting of the following entries in addition to the call arguments:

- `cpts`: Set of change-point estimators $\mathcal{K}$ (see Algorithm 1), represented as an integer vector.
- `cpts.info`: A data frame containing the estimated change-points and their respective detection bandwidths, asymptotic $p$ values of the MOSUM statistics and scaled change heights (obtained as in (17)) as columns.

- `pooled.cpts`: Candidate set $\mathcal{P}$ containing all the estimators from the multiscale MOSUM procedure considered for pruning.
- `ci`: An S3 object of class '`cpts.ci`' containing confidence intervals for the change-points. Returned iff `do.confint = TRUE`; see Section 3.6 for further details.

S3 objects of class '`multiscale.cpts`' are supported by `plot`, `summary`, `print` and `confint` methods. We provide a detailed description of `plot.multiscale.cpts` in Section 3.7.

As an example, we apply the multiscale MOSUM procedure with bottom-up merging and the bandwidth set $\mathcal{G} = \{30, 50, 80, 130\}$, to a piecewise i.i.d. normal time series of length $n = 600$ with mean changes at time points $k_1 = 50, k_2 = 100, k_3 = 300$ of size $d_1 = 1, d_2 = 2$ and $d_3 = -3$, respectively:

```
R> td <- testData(lengths = c(50, 50, 200, 300), means = c(0, 1, 3, 0),
+    sds = rep(1, 4), seed = 123)
R> x <- td$x
R> mbu <- multiscale.bottomUp(x, G = c(30, 50, 80, 130))
R> print(mbu$cpts)
R> print(mbu$pooled.cpts)
```

```
[1]  50 100 300
[1]  50  96 100 300
```

The output $\mathcal{K} = \{50, 100, 300\}$ of the algorithm coincides with the true change-points, whereas the candidate set before merging $\mathcal{P}$ contains an additional estimate 96 which is a duplicate estimate for $k_2 = 100$.

## 3.4. Multiscale MOSUM procedure with localized pruning

The function `multiscale.localPrune` provides an implementation of Algorithm 2 from Section 2.6:

```
multiscale.localPrune(x, G = bandwidths.default(length(x)),
  max.unbalance = 4, threshold = c("critical.value", "custom")[1],
  alpha = 0.1, threshold.function = NULL,
  criterion = c("eta", "epsilon")[1], eta = 0.4, epsilon = 0.2,
  rule = c("pval", "jump")[1], penalty = c("log", "polynomial")[1],
  pen.exp = 1.01, do.confint = FALSE, level = 0.05, N_reps = 1000, ...)
```

It accepts the following arguments, in addition to those accepted by `mosum` and `multiscale.bottomUp`:

- `G`: A set $\mathcal{G}$ of bandwidths, represented as a vector of integers smaller than $n/2$, or numeric values in $(0, 0.5)$ describing the bandwidths relative to $n$. Asymmetric bandwidths obtained as the Cartesian product of the set `G` with itself are used for the multiscale MOSUM procedure. By default, it is given by `bandwidths.default`, see Section 3.5.
- `max.unbalance`: A numeric value greater than equal to one which imposes an upper bound on $\max(G_\mathrm{l}, G_\mathrm{r}) / \min(G_\mathrm{l}, G_\mathrm{r})$ for the bandwidth $\mathbf{G} = (G_\mathrm{l}, G_\mathrm{r})$ to be used for candidate generation. By default, we recommend `max.unbalance = 4`.

- `rule`: A string for the choice of the sorting criterion $c$ to be used in Algorithm 2. Possible values are `"pval"` for the inverse of the $p$ value corresponding to change-point estimates ($c_p$) and `"jump"` for the corresponding jump size ($c_J$), see (17).
- `penalty`: A string indicating which penalty $p(n)$ to be used in the SC, see (18). Possible values are `"log"` for $p(n) = \log(n)$ and `"polynomial"` for $p(n) = n$.
- `pen.exp`: A numeric value for the exponent $\varrho$ in the penalty term of the SC, see (18).

When called, `multiscale.localPrune` returns an S3 object of class '`multiscale.cpts`', as described in Section 3.3.

As a simple example, we continue with the normal time series `x` from Section 3.3, which is analysed using an asymmetric bandwidth grid of size 16 obtained as the Cartesian product of the set $\{30, 50, 80, 130\}$ with itself:

```
R> mlp <- multiscale.localPrune(x, G = c(30, 50, 80, 130))
R> print(mlp$cpts)
R> print(mlp$pooled.cpts)
```

```
[1]  50 100 300
[1]  48  50  86  96 100 300
```

As the example shows, the initial candidate set $\mathcal{P}$ considered by Algorithm 2 tends to be larger than that considered by Algorithm 1 due to the use of asymmetric bandwidths. The localized merging algorithm is successful in removing any spurious or duplicate estimates and returns $\mathcal{K}$ that correctly estimates all the change-points.

## 3.5. Bandwidth generation

The default option for generating bandwidths for the multiscale MOSUM procedure is the function

```
R> bandwidths.default(n, d.min = 10, G.min = 10, G.max = min(n/2, n^(2/3)))
```

which takes as its input the sample size (`n`), the minimal mutual distance between change-points that can be expected (`d.min`), and the minimal and maximal allowed bandwidths (`G.min` and `G.max`). The function returns an integer vector $(G_1, \ldots, G_m)$ where $G_0 = G_1 = \max\{G_{\min}, 2d_{\min}/3\}$ and $G_{j+1} = G_{j-1} + G_j$ for $j = 1, \ldots, m-1$, with $m$ chosen as the largest integer such that $G_m \le G_{\max}$ while $G_{m+1} > G_{\max}$. For `multiscale.bottomUp`, we set `G.min` so that the default bandwidths for the function do not generate a warning message about the smallest bandwidth being too small relative to $n$. For `multiscale.localPrune`, we use the default choices given above.

## 3.6. Bootstrap confidence intervals

Bootstrap confidence intervals of change-points as discussed in Section 2.7 can be computed with the function `confint`, which accepts S3 objects of classes '`mosum.cpts`' or '`multiscale.cpts`' as its input:

```
R> confint(object, parm = "cpts", level = 0.05, N_reps = 1000)
```

- `object`: An object either of class '`mosum.cpts`' or '`multiscale.cpts`'. If `object$do.confint = TRUE`, `object$ci` is returned without further computation.
- `parm`: A string indicating which parameters are to be given confidence intervals; only `parm = "cpts"` is supported. The argument is required for the compatibility with the generic function `confint`.
- `level`: A single numeric value in $(0, 1)$ representing the confidence level for the confidence intervals; corresponds to $\alpha$ used in $K_j^{\text{pw}}(\alpha)$ from (20) and $K_j^{\text{unif}}(\alpha)$ from (21).
- `N_reps`: A positive integer representing the number of bootstrap replicates to be generated for confidence interval construction.

When called, the function returns an S3 object of class '`cpts.ci`' containing the following entry besides the call arguments:

- `CI`: A data frame of five columns, containing the estimated change-points (in column `cpts`) and the end points of the pointwise confidence intervals $K_j^{\text{pw}}(\alpha)$ (in columns `pw.left` and `pw.right`) and the uniform confidence intervals $K_j^{\text{unif}}(\alpha)$ (in columns `unif.left` and `unif.right`).

As an example, we revisit the analysis from Section 3.4:

```
R> mlp_ci <- confint(mlp, level = 0.05, N_reps = 10000)
R> print(mlp_ci$CI)
```

```
  cpts pw.left pw.right unif.left unif.right
1   50      21       80        21         79
2  100      95      105        89        111
3  300     298      302       296        304
```

It shows that e.g., the 95% bootstrap confidence intervals for $k_2 = 100$ are given by $K_2^{\text{pw}}(0.05) = [95, 105]$ (pointwise) and $K_2^{\text{unif}}(0.05) = [89, 111]$ (uniform).

## 3.7. Visualization

The plot of the scaled absolute MOSUM detector is particularly suitable for visually inspecting the data for possible change-points. There is a `plot` method available for S3 objects of class '`mosum.cpts`', which plots either the input time series along with piecewise constant signal constructed using the estimated change-points, or the scaled absolute MOSUM detector along with the critical value and the estimated locations of the change-points; see Figure 8.

S3 objects of class '`multiscale.cpts`' for the output of the multiscale algorithms discussed in Sections 3.3–3.4, are also supported by `plot` method. We can visualize the set of estimated change-point locations (on the $x$-axis) against the input time series or display the significance of the change-point estimators, as well as plotting the confidence intervals of change-points or the detection environments of the estimators. For fair representation of the significance of change-point estimators detected at different scales, we utilize the $p$ values associated with their detection rather than plotting the MOSUM detectors from different scales simultaneously.

To elaborate:

```
plot.multiscale.cpts(x, display = c("data", "significance")[1],
  shaded = c("CI", "bandwidth", "none")[1], level = 0.05, N_reps = 1000,
  CI = c("pw", "unif")[1],  xlab = "Time", ...)
```

It accepts the following arguments:

- `x`: An object of class '`multiscale.cpts`'.
- `display`: A string indicating whether to plot the input time series (`display = "data"`) along with the estimated change-point locations and piecewise constant signal, or to display the significance of change-point estimators (`display = "significance"`), represented by one minus the *p* values associated with their detection.
- `shaded`: A string indicating whether confidence intervals (`shaded = "CI"`) or detection intervals (`shaded = "bandwidth"`) shall be plotted as shaded rectangles around the estimated change-point locations. No shaded rectangle is produced when `shaded = "none"`.
- `level, N_reps`: Arguments used for generating the bootstrap confidence intervals, to be parsed to `confint.multiscale.cpts`. Only in use if `shaded = "CI"`.
- `CI`: A string indicating whether pointwise (`CI = "pw"`) or uniform (`CI = "unif"`) confidence intervals shall be plotted. Only in use if `shaded = "CI"`.

As an example, we revisit the analysis from Section 3.4 and visualize the estimated change-points along with their detection environment and the 95% bootstrap confidence intervals of respective change-points:

```
R> par(mfrow = c(4, 1), mar = c(2, 4, 2, 2))
R> plot(mlp, display = "data", shaded = "none")
R> lines(td$mu, lty = 2, col = 2, lwd = 2)
R> plot(mlp, display = "significance",  shaded = "bandwidth")
R> plot(mlp, display = "significance",  shaded = "CI", CI = "pw")
R> plot(mlp, display = "significance",  shaded = "CI", CI = "unif")
```

The output is shown in Figure 9.

It is possible to obtain a 3D surface plot of *standardized* scaled absolute MOSUM detectors with "continuous" bandwidths using the function `persp3D.multiscaleMosum` function:

```
persp3D.multiscaleMosum(x, mosum.args = list(),
  threshold = c("critical.value", "custom")[1], alpha = 0.1,
  threshold.function = NULL, pal.name = "YlOrRd", expand = 0.2,
  theta = 120, phi = 20, xlab = "G", ylab = "time", zlab = "MOSUM",
  ticktype = "detailed", NAcol = "#800000FF", ...)
```

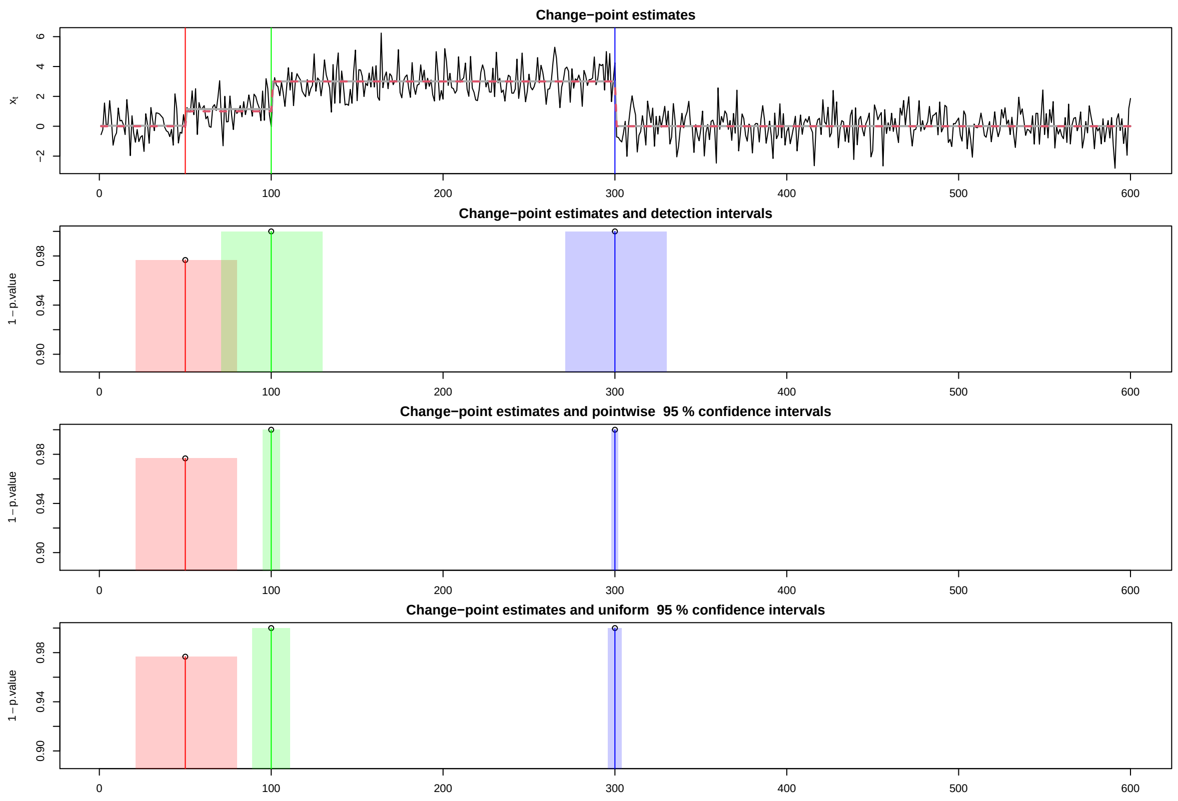

Figure 9: Top: A time series realization `x`, its underlying signal (dashed line), the change-point estimators (vertical lines) and the estimated piecewise constant signal (solid line). Below: Visualization of the significance of estimated change-points where the height of the vertical line at each estimated change-point location represents one minus the $p$ values evaluated at the scaled absolute MOSUM detector value associated with its detection. The detection intervals of estimated change-points, the bootstrap pointwise confidence intervals $K_j^{\mathrm{pw}}(0.05)$ from (20) and the uniform confidence intervals $K_j^{\mathrm{unif}}(0.05)$ from (21) for $j = 1, 2, 3$ are visualized as shaded rectangles surrounding the vertical lines (second to fourth panels).

The purpose of this function is to plot all MOSUM detectors computed with a range of symmetric bandwidths together in one surface plot. To make the graphs from different bandwidths comparable, the MOSUM detectors are standardized with respect to their respective thresholds, e.g., obtained as the critical value at a given significance level. This is particularly useful for visually investigating which features of the data are captured at which bandwidth scale (c.f., the discussion in Section 2.4).

The `persp3D.multiscaleMosum` accepts the following arguments:

- `x`: Input data $X_1, \ldots, X_n$, i.e., a univariate time series represented as a numeric vector (of length $n$) or an object of class '`ts`'.
- `mosum.args`: Further arguments to be parsed to the function `mosum` (see Section 3.2), which may be empty. Note that the bandwidths are chosen by default and should not be given as an argument in `mosum.args`.
- `threshold`, `alpha`, `threshold.custom`: Arguments specifying how to select the thresh-

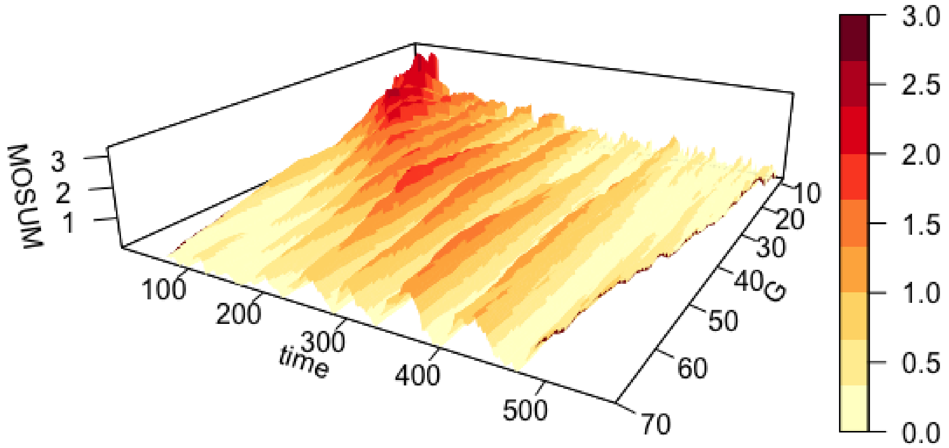

Figure 10: Surface plot visualization of the MOSUM detectors computed with a range of bandwidths on a realization of the `mix` test signal from Section 3.1.

olds for standardising scaled MOSUM detectors $|T_G(k)|/\widehat{\sigma}_k$ computed with different bandwidths $G$, see the description of these arguments in Section 3.3.

- `pal.name`: A string containing the name of the `ColorBrewer` palette from the **RColorBrewer** (Neuwirth 2014) to be used. Sequential palettes are recommended. See `brewer.pal.info` of **RColorBrewer** for further details.

The remaining arguments are graphical parameters parsed to the function `persp3D` of the **plot3D** package (Soetaert 2019), and further information can be found in the R documentation thereof. The range of the color palette is chosen such that the three lightest hues (when a sequential palette is used) indicate insignificant MOSUM values in change-point analysis.

As an example, we consider a visualization of the MOSUM detectors computed on a realization from the test signal `mix` from the literature (see Figure 7 for the plot of the time series):

```
R> x <- testData(model = "mix", seed = 1234)$x
R> persp3D.multiscaleMosum(x, mosum.args = list(boundary.extension = FALSE))
```

The result is shown in Figure 10. Note that a $z$-axis value above one in Figure 10 implies that the respective statistic exceeds the critical value, as indicated by the emergence of the hue of orange. It becomes obvious that the large changes at the beginning are given prominence at smaller bandwidths, whereas the peaks corresponding to smaller changes become significant at larger bandwidth.

## 4. Usage examples

In this section, we provide more detailed examples demonstrating the usage of the **mosum** package for detecting multiple change-points in the mean. We start with the single-bandwidth MOSUM procedure applied to heteroscedastic time series in Section 4.1, followed by the application of the multiscale algorithms from Section 2.6 to test signals with frequent change-points and real-life time series from economics in Sections 4.2 and 4.3.

### 4.1. MOSUM procedure with a single bandwidth

In this first example, we apply the MOSUM procedure with a single, asymmetric bandwidth $\mathbf{G} = (40, 60)$ to a time series of length $n = 800$ with independent normal innovations and two changes in the mean at $k_1 = 200$ and $k_2 = 600$ of respective height $d_1 = 2$ and $d_2 = -1.5$, co-occurring with changes in variance. The variances over each stationary segment is set to be 1, 0.8 and 0.5, respectively. We adopt the local variance estimator (11), which is expected to have more power when there are changes in both mean and variance. By default, the $\eta$-criterion (16) with $\eta = 0.4$ is used for change-point estimation in conjunction with the default significance level $\alpha = 0.1$.

```
R> td <- testData(lengths = c(200, 400, 200), means = c(0, 2, 1),
+    sds = sqrt(c(1, 0.8, 0.5)), seed = 111)
R> x1 <- td$x
R> f1 <- td$mu
R> m <- mosum(x1, G = 40, G.right = 60, var.est.method = "mosum.min")
R> print(m$cpts)
```

```
[1] 205 600
```

The procedure correctly detects the number and locations of the change-points. It may provide further insights to look at the time series in conjunction with the corresponding MOSUM detector and local variance estimator:

```
R> par(mfcol = c(3, 1), mar = c(2, 2, 2, 1))
R> plot(ts(x1))
R> lines(f1, col = 2, lwd = 2, lty = 2)
R> plot(m, display = "mosum")
R> plot(ts(m$var.estimation))
```

The result is shown in Figure 11, which confirms that the estimates are consistent with the visual inspection of the scaled MOSUM detector.

### 4.2. Multiscale MOSUM procedure with bottom-up merging

We apply Algorithm 1 to the piecewise stationary time series `mix` from Section 3.1. The test signal is particularly interesting due to the variety of types of mean changes, from large jumps over short intervals to small jumps over longer stretches of stationarity (see Figure 7). We consider using a dense bandwidth grid $\mathcal{G} = \{10, 11, \ldots, 40\}$ with a slightly increased threshold $C_{n,G}(\alpha) \cdot \log(n/G)^{0.1}$, in order that the use of small bandwidths does not yield spurious estimators, see the discussion at the end of Section 2.6.1.

```
R> x2 <- testData(model = "mix", seed = 1234)$x
R> threshold.custom <- function(G, n, alpha) {
+    mosum.criticalValue(n, G, G, alpha) * log(n/G)^0.1
+  }
R> mbu <- multiscale.bottomUp(x2, G = 10:40, threshold = "custom",
+    threshold.function = threshold.custom)
```

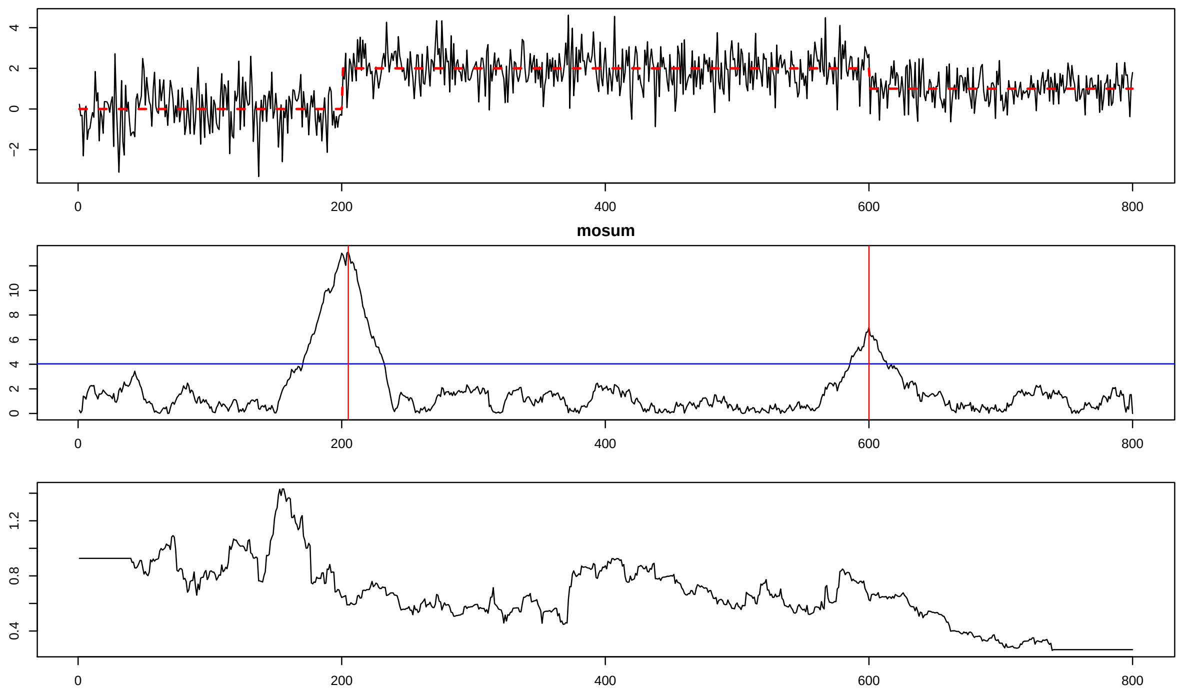

Figure 11: Top: A time series realization `x1`. Middle: The scaled absolute MOSUM detector. Bottom: Local variance estimator.

Although the smallest bandwidth from $\mathcal{G}$ is relatively small compared to the sample size $n = 560$, the use of custom threshold suppresses the issue of the warning message. Intuitively, the large changes with small mutual distance (at the beginning of the signal) should be detected by small bandwidths, whereas the small changes with large mutual distance (towards the end of the signal) should be detected by the large bandwidths. This is indeed verified by inspecting the column `G.left` containing the detection bandwidths in the output of `summary(mbu)`; see also the surface plot Figure 10 in Section 3.7.

```
change-points estimated at alpha = 0.1 according to eta-criterion
 with eta = 0.4

     cpts   G.left   G.right    p.value    jump
1      10       10        10   8.40e-06   3.304
2      20       10        10   1.98e-06   3.531
3      41       10        10   3.31e-12   5.628
4      60       10        10   8.73e-06   3.298
5      89       10        10   4.09e-04   2.691
6     120       10        10   5.22e-04   2.653
7     156       10        10   2.20e-03   2.426
8     200       10        10   3.57e-03   2.349
9     250       10        10   6.03e-03   2.267
10    302       16        16   6.90e-03   1.756
12    363       37        37   3.74e-02   0.970
11    421       30        30   2.74e-02   1.120
```

## 4.3. Multiscale MOSUM procedure with localized pruning

We apply the multiscale MOSUM procedure summarized in Algorithm 2 to the piecewise stationary time series `blocks` (see Section 3.1). We use the default choice of bandwidths returned by `bandwidths.default` as described in Section 3.3. Candidate change-point estimates are generated with the generous choice of $\alpha = 0.4$:

```
R> x3 <- testData(model = "blocks", seed = 123)$x
R> mlp <- multiscale.localPrune(x3, alpha = 0.4, pen.exp = 1.01)
R> print(mlp$cpts)

 [1]  200  266  307  471  511  818  902 1331 1555 1597 1654
```

To see how many candidates have been discarded in the merging step of the algorithm, the set $\mathcal{P}$ of change-point candidates prior to merging may also be of interest:

```
R> print(mlp$pooled.cpts)

 [1]   29   98  148  186  195  200  203  204  205  206  208  266  307
[14]  308  315  316  387  432  438  471  472  489  510  511  512  520
[27]  521  524  783  809  810  818  819  901  902  952 1238 1279 1280
[40] 1322 1331 1340 1347 1353 1460 1469 1546 1547 1548 1555 1556 1557
[53] 1595 1596 1597 1605 1606 1646 1654 1655 1658 1659 1673 1683
```

In addition, we analyse the quarterly US ex-post real interest rate from 1961:Q1 to 1986:Q3 ($n = 103$) from the Citibase data bank, which has been extensively studied in Garcia and Perron (1996); the data is available from the R package **strucchange** (Zeileis *et al.* 2002). We set $\alpha = 0.1$ and use the $\eta$-criterion with $\eta = 0.4$ along with the default bandwidths for the multiscale MOSUM procedure, and use $p(n) = \log n$ and $\varrho = 1.01$ for the penalty function for the localized pruning. The plot of input data series in Figure 12 suggests that there may be changes in the variance. To prevent detecting spurious estimators due to possible heteroscedasticity in the data, we use the local variance estimator of (12).

```
R> data("RealInt", package = "strucchange")
R> mlp <- multiscale.localPrune(RealInt, alpha = 0.1, eta = 0.4,
+     var.est.method = "mosum.max", penalty = "log", pen.exp = 1.01)
R> print(mlp)

change-points estimated with bandwidths
  10 20
at alpha = 0.1 according to eta-criterion with eta = 0.4:
  47 79
```

We note that the outcome did not vary with a range of alternative choices for $\alpha$, $\eta$ and $\varrho$. For comparison, we analyse the same data for change-points using the WBS algorithm with the information criterion-based model selection (Fryzlewicz 2014), the TGUH algorithm (Fryzlewicz 2018), PELT (Killick *et al.* 2012a), S3IB (Rigaill 2015), cumSeg (Muggeo and

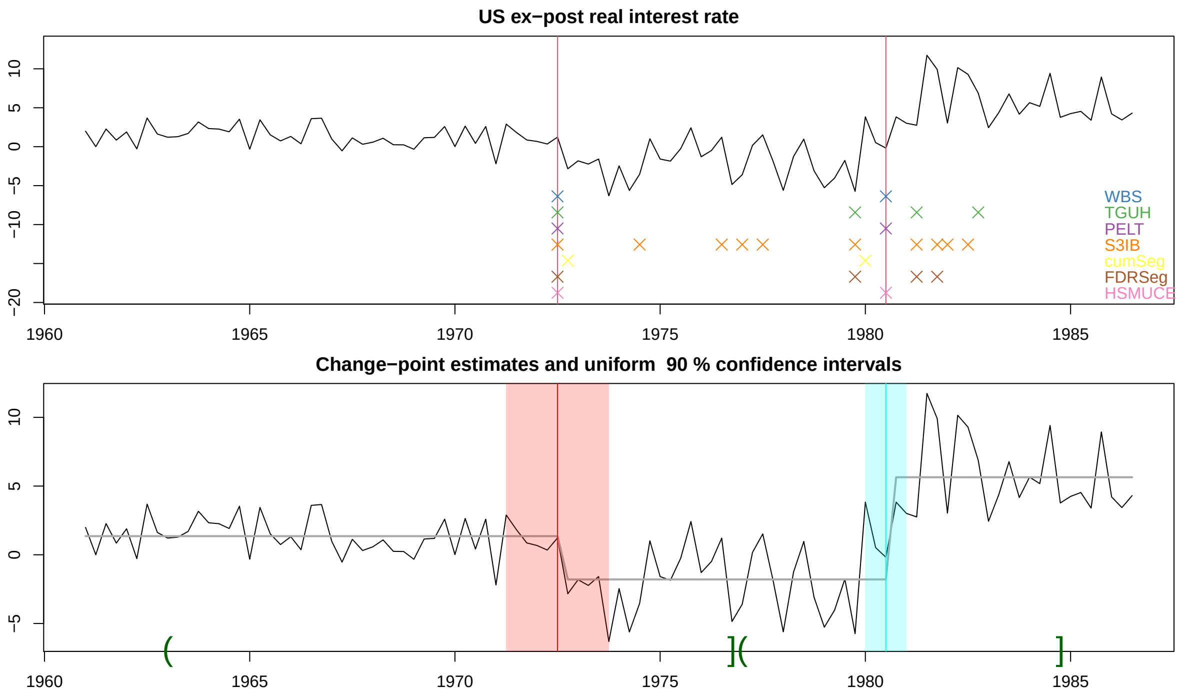

Figure 12: Top: The quarterly US ex-post real interest rate from 1961:Q1 to 1986:Q3, where the vertical lines denote the two change-point estimators returned by `multiscale.localPrune`. Also plotted are change-points estimated by the WBS, TGUH, PELT, S3IB, cumSeg, FDRSeg and H-SMUCE. Bottom: The 90% confidence intervals of change-point locations produced by `confint` (shaded areas) and the function `jumpoint` of **stepR** (plotted on the $x$-axis) against the input time series and the estimated piecewise constant signal.

Adelfio 2010, transforms the data and iteratively fits a linear model for change-point analysis), FDRSeg (Li *et al.* 2016) and H-SMUCE (Pein *et al.* 2017); where relevant, we set the significance level at 0.1 and follow the default setting for any other arguments. Detailed codes for applying the above methods are provided in the online supplementary material.

Top panel of Figure 12 shows the change-points estimated by Algorithm 2 and competitors. Among different methods, cumSeg and H-SMUCE are designed to be robust to heteroscedasticity in the data, and the use of local variance estimator is found effective in this case for our MOSUM-based method. On the other hand, the time-varying variance may have led methods such as TGUH, S3IB and FDRSeg to detect possibly spurious change-points. Most methods return the two change-points identified by Algorithm 2 and in fact, the WBS, PELT and H-SMUCE return the identical estimators. Also, the two change-point estimators are close to the two breaks reported in Garcia and Perron (1996), where the authors related the breaks to the findings in the economics literature.

Unlike `multiscale.localPrune`, WBS, S3IB and cumSeg require the maximal number of change-points as an input, and the output may vary with respect to the choice of this parameter; e.g., when this quantity is set as large as 100, the WBS returns 100 estimators.

We can generate bootstrap confidence intervals around the change-points and plot them:

```
R> plot(mlp, display = "data", shaded = "CI", CI = "unif", level = 0.1)
```

The 90% uniform confidence intervals are plotted in the bottom panel of Figure 12. The **stepR** package supports calculation of confidence intervals of change-point locations, based on the exponential bounds for the probability of under-estimating the change-point set; we plot these confidence intervals in the $x$-axis of the same figure. We note that the latter confidence intervals are considerably wider than the bootstrap confidence intervals generated as above for this data.

## 5. Conclusion and outlook

In this paper, we present the R package **mosum** (Meier *et al.* 2021). Implementations of both single bandwidth and multiscale MOSUM procedures for multiple change-point estimation have been described, as well as tools for visualization and data generation.

Due to its flexibility, the MOSUM framework discussed in this paper can be extended in several ways. First, different MOSUM procedures for the mean change problem can be considered, such as those based on more robust location estimators utilising the median. Secondly, multivariate data analysis could be incorporated. Also, different types of changes can be handled, e.g., parameter changes in (auto-)regressive time series (linear, non-linear or even count time series).

All of the above extensions can be dealt with theoretically in a unified framework based on estimating functions (see e.g., Kirch and Kamgaing 2015, for a discussion in a sequential framework). However, the details of the implementation, including important issues such as covariance estimation, require case-by-case consideration. It is planned to incorporate important extensions into future revisions of the **mosum** package.

## Acknowledgments

This research was supported by the research training group *Mathematical Complexity Reduction* DFG-GRK 2297 and partially supported by DFG grant KI 1443/3-1. Haeran Cho's work was supported by the Engineering and Physical Sciences Research Council grant no. EP/N024435/1. The authors would like to thank Kerstin Reckrühm and Rebecca Killick for the constructive suggestions for improvement, as well as the referees and the Editor for the helpful comments.

## References

Anastasiou A, Chen Y, Cho H, Fryzlewicz P (2020). ***breakfast**: Methods for Fast Multiple Change-Point Detection and Estimation.* R package version 2.1, URL `https://CRAN.R-project.org/package=breakfast`.

Auger IE, Lawrence CE (1989). "Algorithms for the Optimal Identification of Segment Neighborhoods." *Bulletin of Mathematical Biology*, **51**(1), 39–54. `doi:10.1007/bf02458835`.

Axt I, Fried R (2019). "On Scale Estimation under Shifts in the Mean." *Discussion Paper 06/2019*, SFB 823, TU Dortmund University.

Bai J, Perron P (1998). "Estimating and Testing Linear Models with Multiple Structural Changes." *Econometrica*, **66**(1), 47–78. doi:10.2307/2998540.

Baranowski R, Chen Y, Fryzlewicz P (2019a). "Narrowest-over-Threshold Detection of Multiple Change Points and Change-Point-Like Features." *Journal of the Royal Statistical Society B*, **81**(3), 649–672. doi:10.1111/rssb.12322.

Baranowski R, Chen Y, Fryzlewicz P (2019b). ***not**: Narrowest-Over-Threshold Change-Point Detection.* R package version 1.2, URL https://CRAN.R-project.org/package=not.

Barry D, Hartigan JA (1993). "A Bayesian Analysis for Change Point Problems." *Journal of the American Statistical Association*, **88**(421), 309–319. doi:10.2307/2290726.

Chan HP, Chen H (2017). "Multi-Sequence Segmentation via Score and Higher-Criticism Tests." *arXiv 1706.07586*, arXiv.org E-Print Archive. URL https://arxiv.org/abs/1706.07586.

Cho H, Kirch C (2020a). "Data Segmentation Algorithms: Univariate Mean Change and Beyond." *arXiv 2012.12814*, arXiv.org E-Print Archive. URL https://arxiv.org/abs/2012.12814.

Cho H, Kirch C (2020b). "Two-Stage Data Segmentation Permitting Multiscale Change Points, Heavy Tails and Dependence." *arXiv 1910.12486*, arXiv.org E-Print Archive. URL https://arxiv.org/abs/1910.12486.

Cleynen A, Rigaill G, Koskas M (2016). ***Segmentor3IsBack**: A Fast Segmentation Algorithm.* R package version 2.0, URL https://CRAN.R-project.org/src/contrib/Archive/Segmentor3IsBack/.

Dette H, Eckle T, Vetter M (2020). "Multiscale Change Point Detection for Dependent Data." *Scandinavian Journal of Statistics*, **47**(4), 1243–1274. doi:10.1111/sjos.12465.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Eichinger B, Kirch C (2018). "A MOSUM Procedure for the Estimation of Multiple Random Change Points." *Bernoulli*, **24**(1), 526–564. doi:10.3150/16-bej887.

Erdman C, Emerson JW (2007). "**bcp**: An R Package for Performing a Bayesian Analysis of Change Point Problems." *Journal of Statistical Software*, **23**(3), 1–13. doi:10.18637/jss.v023.i03.

Fang X, Li J, Siegmund D (2020). "Segmentation and Estimation of Change-Point Models: False Positive Control and Confidence Regions." *The Annals of Statistics*, **48**(3), 1615–1647. doi:10.1214/19-aos1861.

Frick K, Munk A, Sieling H (2014). "Multiscale Change Point Inference." *Journal of the Royal Statistical Society B*, **76**(3), 495–580. doi:10.1111/rssb.12047.

Fryzlewicz P (2014). "Wild Binary Segmentation for Multiple Change-Point Detection." *The Annals of Statistics*, **42**(6), 2243–2281. doi:10.1214/14-aos1245.

Fryzlewicz P (2018). "Tail-Greedy Bottom-up Data Decompositions and Fast Multiple Change-Point Detection." *The Annals of Statistics*, **46**(6), 3390–3421. doi:10.1214/17-aos1662.

Fryzlewicz P (2020). "Detecting Possibly Frequent Change-Points: Wild Binary Segmentation 2 and Steepest-Drop Model Selection-Rejoinder." *Journal of the Korean Statistical Society*, **49**, 1–7. doi:10.1007/s42952-020-00085-2.

Garcia R, Perron P (1996). "An Analysis of the Real Interest Rate under Regime Shifts." *The Review of Economics and Statistics*, **78**(1), 111–125. doi:10.2307/2109851.

Hampel FR (1974). "The Influence Curve and Its Role in Robust Estimation." *Journal of the American Statistical Association*, **69**(346), 383–393. doi:10.2307/2285666.

Haynes K, Killick R (2020). ***changepoint.np**: Methods for Nonparametric Changepoint Detection*. R package version 1.0.2, URL https://CRAN.R-project.org/package=changepoint.np.

James NA, Matteson DS (2014). "**ecp**: An R Package for Nonparametric Multiple Change Point Analysis of Multivariate Data." *Journal of Statistical Software*, **62**(7), 1–25. doi:10.18637/jss.v062.i07.

Killick R, Eckley IA (2014). "**changepoint**: An R Package for Changepoint Analysis." *Journal of Statistical Software*, **58**(3), 1–19. doi:10.18637/jss.v058.i03.

Killick R, Fearnhead P, Eckley IA (2012a). "Optimal Detection of Changepoints with a Linear Computational Cost." *Journal of the American Statistical Association*, **107**(500), 1590–1598. doi:10.1080/01621459.2012.737745.

Killick R, Nam CFH, Aston JAD, Eckley IA (2012b). "Changepoint.info: The Changepoint Repository." URL http://changepoint.info.

Kirch C, Kamgaing JT (2015). "On the Use of Estimating Functions in Monitoring Time Series for Change Points." *Journal of Statistical Planning and Inference*, **161**, 25–49. doi:10.1016/j.jspi.2014.12.009.

Kühn C (2001). "An Estimator of the Number of Change Points Based on a Weak Invariance Principle." *Statistics & Probability Letters*, **51**(2), 189–196. doi:10.1016/s0167-7152(00)00155-3.

Li H, Munk A, Sieling H (2016). "FDR-Control in Multiscale Change-Point Segmentation." *Electronic Journal of Statistics*, **10**(1), 918–959. doi:10.1214/16-ejs1131.

Li H, Sieling H (2017). ***FDRSeg**: FDR-Control in Multiscale Change-Point Segmentation*. R package version 1.0-3, URL https://CRAN.R-project.org/package=FDRSeg.

Maidstone R, Hocking T, Rigaill G, Fearnhead P (2017). "On Optimal Multiple Changepoint Algorithms for Large Data." *Statistics and Computing*, **27**(2), 519–533. doi:10.1007/s11222-016-9636-3.

Matteson DS, James NA (2014). "A Nonparametric Approach for Multiple Change Point Analysis of Multivariate Data." *Journal of the American Statistical Association*, **109**(505), 334–345. doi:10.1080/01621459.2013.849605.

Meier A, Cho H, Kirch C (2021). ***mosum**: Moving Sum Based Procedures for Changes in the Mean.* R package version 1.2.5, URL https://CRAN.R-project.org/package=mosum.

Messer M, Kirchner M, Schiemann J, Roeper J, Neininger R, Schneider G (2014). "A Multiple Filter Test for the Detection of Rate Changes in Renewal Processes with Varying Variance." *The Annals of Applied Statistics*, **8**(4), 2027–2067. doi:10.1214/14-aoas782.

Muggeo VMR (2020). ***cumSeg**: Change Point Detection in Genomic Sequences.* R package version 1.3, URL https://CRAN.R-project.org/package=cumSeg.

Muggeo VMR, Adelfio G (2010). "Efficient Change Point Detection for Genomic Sequences of Continuous Measurements." *Bioinformatics*, **27**(2), 161–166. doi:10.1093/bioinformatics/btq647.

Neuwirth E (2014). ***RColorBrewer**: ColorBrewer Palettes.* R package version 1.1-2, URL https://CRAN.R-project.org/package=RColorBrewer.

Niu YS, Zhang H (2012). "The Screening and Ranking Algorithm to Detect DNA Copy Number Variations." *The Annals of Applied Statistics*, **6**(3), 1306–1326. doi:10.1214/12-aoas539.

Page ES (1954). "Continuous Inspection Schemes." *Biometrika*, **41**(1/2), 100–115. doi:10.1093/biomet/41.1-2.100.

Pein F, Hotz T, Sieling H, Aspelmeier T (2020). ***stepR**: Multiscale Change-Point Inference.* R package version 2.1-1, URL https://CRAN.R-project.org/package=stepR.

Pein F, Sieling H, Munk A (2017). "Heterogeneous Change Point Inference." *Journal of the Royal Statistical Society B*, **79**(4), 1207–1227. doi:10.1111/rssb.12202.

Politis DN, Romano JP (1995). "Bias-Corrected Nonparametric Spectral Estimation." *Journal of Time Series Analysis*, **16**(1), 67–103. doi:10.1111/j.1467-9892.1995.tb00223.x.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rigaill G (2015). "A Pruned Dynamic Programming Algorithm to Recover the Best Segmentations with 1 to $K_{\max}$ Change-Points." *Journal De La Société Française De Statistique*, **156**(4), 180–205.

Rigaill G, Hocking T, Maidstone R, Fearnhead P (2019). ***fpop**: Segmentation Using Optimal Partitioning and Function Pruning.* R package version 2019.08.26, URL https://CRAN.R-project.org/package=fpop.

Ross GJ (2015). "Parametric and Nonparametric Sequential Change Detection in R: The **cpm** Package." *Journal of Statistical Software*, **66**(3), 1–20. doi:10.18637/jss.v066.i03.

Scott AJ, Knott M (1974). "A Cluster Analysis Method for Grouping Means in the Analysis of Variance." *Biometrics*, **30**(3), 507–512. doi:10.2307/2529204.

Seleznjev OV (1991). "Limit Theorems for Maxima and Crossings of a Sequence of Gaussian Processes and Approximation of Random Processes." *Journal of Applied Probability*, **28**(1), 17–32. doi:10.2307/3214737.

Sen A, Srivastava MS (1975). "On Tests for Detecting Change in Mean." *The Annals of Statistics*, **3**(1), 98–108. doi:10.1214/aos/1176343001.

Seshan VE, Olshen A (2020). ***DNAcopy**: DNA Copy Number Data Analysis.* R package version 1.64.0, URL https://bioconductor.org/packages/DNAcopy/.

Soetaert K (2019). ***plot3D**: Plotting Multi-Dimensional Data.* R package version 1.3, URL https://CRAN.R-project.org/package=plot3D.

Stroustrup B (2000). *The C++ Programming Language.* 4th edition. Pearson Education India.

Tecuapetla-Gómez I, Munk A (2017). "Autocovariance Estimation in Regression with a Discontinuous Signal and m-Dependent Errors: A Difference-Based Approach." *Scandinavian Journal of Statistics*, **44**(2), 346–368. doi:10.1111/sjos.12256.

Xiao F, Niu Y, Hao N, Xu Y, Jin Z, Zhang H (2016). ***modSaRa**: A Computationally Efficient R package for CNV Identification.* R package version 1.0, URL https://publichealth.yale.edu/c2s2/software/modsara/.

Yau CY, Zhao Z (2016). "Inference for Multiple Change Points in Time Series via Likelihood Ratio Scan Statistics." *Journal of the Royal Statistical Society B*, **78**(4), 895–916. doi:10.1111/rssb.12139.

Zeileis A, Leisch F, Hornik K, Kleiber C (2002). "**strucchange**: An R Package for Testing for Structural Change in Linear Regression Models." *Journal of Statistical Software*, **7**(2), 1–38. doi:10.18637/jss.v007.i02.

# A. Localized exhaustive search algorithm

In this section, we discuss an efficient implementation of the exhaustive search performed in Line 11 of Algorithm 2. It seeks for a subset $\widehat{\mathcal{A}} \subset \mathcal{D}$ according to the criterion II from Section 2.6. For the sake of completeness and readability, we present a high-level description thereof in Algorithm 3.

**Algorithm 3:** Localized exhaustive search based on SC.

```
   input : Sets of current candidates $\mathcal{C}$ and conflicting candidates $\mathcal{D} \subset \mathcal{C}$, interval of
           consideration $[s, e]$
 1 Let $M \leftarrow 2^{|\mathcal{D}|}$ and denote by $\mathcal{D}_1, \dots, \mathcal{D}_M$ all $M$ subsets of $\mathcal{D}$ (including $\emptyset$ and $\mathcal{D}$)
 2 Initialize $\ell \leftarrow |\mathcal{D}|$, $\mathcal{F} \leftarrow \widehat{\mathcal{A}} \leftarrow \emptyset$ and $\mathsf{flag}_i \leftarrow \mathsf{true}$ for $i = 1, \dots, M$
   /* Step 1:  Process subsets in descending cardinality                    */
 3 repeat
     /* Step 1.1:  Truncate the search space                                */
 4   for $\mathcal{D}_i$ with $|\mathcal{D}_i| = \ell$ and $\mathsf{flag}_i = \mathsf{false}$ do
 5     Identify $\mathsf{child}(\mathcal{D}_i) = \{j \colon \mathcal{D}_j \subset \mathcal{D}_i$ with $|\mathcal{D}_j| = \ell - 1$ and $\mathsf{flag}_j = \mathsf{true}\}$
 6     for $\mathcal{D}_j \in \mathsf{child}(\mathcal{D}_i)$ do $\mathsf{flag}_j \leftarrow \mathsf{false}$
 7   end
     /* Step 1.2:  Remove one candidate at a time                           */
 8   for $\mathcal{D}_i$ with $|\mathcal{D}_i| = \ell$ and $\mathsf{flag}_i = \mathsf{true}$ do
 9     Update $\mathcal{F} \leftarrow \mathcal{F} \cup \{i\}$ and identify $\mathsf{child}(\mathcal{D}_i)$
10     for $j \in \mathsf{child}(\mathcal{D}_i)$ do
11       if $\mathrm{SC}(\mathcal{D}_i|\mathcal{C}, [s, e]) < \mathrm{SC}(\mathcal{D}_j|\mathcal{C}, [s, e])$ then $\mathsf{flag}_j \leftarrow \mathsf{false}$
12     end
13   end
14   $\ell \leftarrow \ell - 1$
15 until $\ell = 0$
   /* Step 2:  Select final combination according to (II)                   */
16 if $\mathcal{F} \neq \emptyset$ then
17   find $m^* \leftarrow \min_{i \in \mathcal{F}} |\mathcal{D}_i|$
18   identify $i^* \leftarrow \arg\min_{i \colon \mathcal{D}_i \in_R \mathcal{D}_{i'}, i' \in \mathcal{F}, m^* \le |\mathcal{D}_{i'}| \le m^* + 2} \mathrm{SC}(\mathcal{D}_i|\mathcal{C}, [s, e])$
19   set $\widehat{\mathcal{A}} \leftarrow \mathcal{D}_{i^*}$
20 end
   output: $\widehat{\mathcal{A}}$
```

Despite the truncation of the search space performed in Step 1.1 of Algorithm 3, the worst case number of computations within Algorithm 3 is of exponential order $2^{|\mathcal{D}|}$. A careful and efficient implementation is thus needed to make the runtime of the procedure practical. In what follows, we describe some details of our implementation of Algorithm 3 within the **mosum** package. It is based on C++ code, which is integrated into R with the **Rcpp** package (Eddelbuettel and François 2011).

In every iteration of Step 1.2 of Algorithm 3, several SC values have to be computed for comparison. To make these steps less computationally intensive, we use pre-computed sums.

| Set representation $\mathcal{D}_i$ | Binary representation $\vec{\psi}_i$ | Integer representation $\kappa_i$ |
|---|---|---|
| $\emptyset$ | `0000` | 0 |
| $\{\widetilde{k}_1\}$ | `0001` | 1 |
| $\{\widetilde{k}_2\}$ | `0010` | 2 |
| ... | ... | ... |
| $\{\widetilde{k}_1, \widetilde{k}_2, \widetilde{k}_3, \widetilde{k}_4\}$ | `1111` | 15 |

Table 1: Connection between binary representation, set representation and integer representation of all subsets of a candidate set $\mathcal{D} = \{\widetilde{k}_1, \widetilde{k}_2, \widetilde{k}_3, \widetilde{k}_4\}$ of length $|\mathcal{D}| = 4$.

To elaborate, let $\mathcal{P} = \{\widetilde{k}_1 < \ldots < \widetilde{k}_m\}$ denote the set of candidate change-points returned from Step 1 of Algorithm 2. Algorithm 3 is sped up by pre-computing and storing the sums

$$\sum_{t=\widetilde{k}_j+1}^{\widetilde{k}_{j+1}} X_t \quad \text{and} \quad \sum_{t=\widetilde{k}_j+1}^{\widetilde{k}_{j+1}} X_t^2 \quad \text{for } j = 1, \ldots, m,$$

as these sums are used repeatedly in the calculation of the residual sum of squares (19) that are the main ingredient in calculating the information criterion (18).

Another important optimization is attributed to the usage of *bit vectors* for an implicit representation of all the subsets $\mathcal{D}_1, \ldots, \mathcal{D}_M$ of $\mathcal{D} = \{\widetilde{k}_1, \ldots, \widetilde{k}_{|\mathcal{D}|}\}$, where $M = 2^{|\mathcal{D}|}$. The key idea is that every $\mathcal{D}_i \subset \mathcal{D}$ can be represented as a vector $\vec{\psi}_i = (\psi_{i,1}, \ldots, \psi_{i,|\mathcal{D}|})$ of binary variables $\psi_{i,j}$ indicating whether $\widetilde{k}_j$ belongs to $\mathcal{D}_i$ or not, i.e., $\psi_{i,j} \in \{0, 1\}$ and $\psi_{i,j} = 1$ if and only if $\widetilde{k}_j \in \mathcal{D}_i$, for $1 \le i \le M$ and $1 \le j \le |\mathcal{D}|$. In addition, every binary vector $\vec{\psi}_i$ has a canonical representation as a nonnegative integer $\kappa_i = \sum_{j=1}^{|\mathcal{D}|} \psi_{i,j} 2^{j-1} \in \{0, \ldots, M-1\}$. This one-to-one correspondence $\mathcal{D}_i \leftrightarrow \vec{\psi}_i \leftrightarrow \kappa_i$ enables an efficient way of representing the subsets. Table 1 provides an example for the case $|\mathcal{D}| = 4$.

A crucial advantage of this approach is that set operations (acting on $\mathcal{D}_i$) can be translated into binary operations (acting on the bits $\vec{\psi}_i$ of $\kappa_i$), the latter being highly performant when implemented in C++. As an example, the outer loops in Step 1.1 and Step 1.2 of Algorithm 3 translate into a loop over all $\kappa_i$ such that the corresponding $\vec{\psi}_i$ contains exactly $\ell$ non-zero entries. Such a loop can be realized in C++ as follows (see Section 11.1.1 in Stroustrup 2000, for a comprehensive overview on bitwise/logical operators in C++):

```
unsigned int kappa = (1 << l) - 1;
while(kappa < M-1) {
  // ...
  const unsigned int tmp = kappa | (kappa-1);
  kappa = tmp | ((((tmp & -tmp) / (kappa & -kappa)) >> 1) - 1);
}
```

As another example, the inner loops (given $\mathcal{D}_i$) over all subsets $\mathcal{D}_j \subset \mathcal{D}_i$ with $|\mathcal{D}_j| = |\mathcal{D}_i| - 1$ in Step 1.1 and Step 1.2 of Algorithm 3 translate into a loop over all $\kappa_j$ with $\vec{\psi}_j$ having exactly $|\mathcal{D}_i| - 1$ non-zero entries and $\vec{\psi}_j \le \vec{\psi}_i$ holding component-wise among the binary vectors. A possible realization in C++ is as follows:

| signal | Algorithm 1 | Algorithm 2–$c_p$ | Algorithm 2–$c_J$ | WBS2 | PELT | TGUH |
|---|---|---|---|---|---|---|
| `blocks` | 0.038 | 13.947 | 1.187 | 4.045 | 0.305 | 0.796 |
| `fms` | 0.039 | 1.352 | 0.564 | 4.598 | 0.299 | 0.815 |
| `mix` | 0.040 | 24.267 | 0.647 | 4.435 | 0.295 | 0.769 |
| `teeth10` | 0.065 | 0.831 | 0.831 | 4.114 | 0.286 | 0.780 |
| `stairs10` | 0.091 | 1.426 | 5.111 | 4.031 | 0.296 | 0.771 |

Table 2: Execution time of change-point detection algorithms when applied to the dense test signals averaged over 100 replications.

```
// m = log_2(M)
for (unsigned kappa_j_help = 0; kappa_j_help < m; ++kappa_j_help) {
  const unsigned kappa_j_candidate = kappa^(1 << kappa_j_help);
  if (kappa_j_candidate < kappa) {
    // ...
  } // else: discard kappa_j_candidate and continue
}
```

We also benefit from the binary representation when searching for $\mathcal{D}_i \subset_R \mathcal{D}_{i'}$ in Step 2 as this amounts to locating the leftmost and the rightmost index at which $\psi_{i',j} = 1$.

Due to the exponential time and memory consumption, we restrict the candidate set size to $|\mathcal{D}| \leq 24$ in our implementation of Algorithm 3. If a candidate set $\mathcal{D}$ exceeding this size is proposed in Step 2.2 of Algorithm 2, we regard the corresponding tuple $(\widehat{k}_\circ, \mathbf{G}_\circ)$ (from Line 7 in Algorithm 2) as *infeasible*. If an infeasible tuple occurs, we swap $\widehat{k}_\circ$ with the next feasible candidate from $\mathcal{D}$ or, if none of $\mathcal{D}$ is feasible, with the next feasible candidate from $\mathcal{P}$, and leave $\widehat{k}_\circ$ for future consideration. In many cases, changing the order of processing is already sufficient to ensure that the number of conflicting candidates of $\widehat{k}_\circ$ will shrink at some point to a feasible size, because some of the currently conflicting estimators will have been pruned down due to the processing of another overlapping candidate set. If, however, all remaining tuples in $\mathcal{P}$ are infeasible, we use a manual *thinning* step, successively removing the elements of $\mathcal{D}$ until it reaches a feasible size. More precisely, the candidates of $\mathcal{D}$ are processed one by one in decreasing order of mutual distance and removed, until $|\mathcal{D}| = 24$. If this thinning step is necessary in a call of `multiscale.localPrune`, the user will be informed by a warning message as below:

```
Warning: 25 conflicting candidates, thinning manually
```

## B. Execution time of multiscale MOSUM methods

We briefly study the execution time of multiscale MOSUM methods when applied to "dense" test signals of length $n \geq 2 \times 10^4$, which are obtained by repeating the five test signals implemented in the function `testData` until the length of the series exceeds $2 \times 10^4$. Table 2 summarizes the execution time of Algorithms 1–2 implemented in **mosum** on the dense test signals averaged over 100 replications, as well as that of WBS2 (proposed to improve upon the adaptivity of the WBS for signals with possibly frequent changes, see Fryzlewicz (2020) for details), TGUH (implemented in **breakfast**) and PELT (**changepoint**) applied with the default parameters. The code for generating the results in Table 2 is provided in the online supplementary material, and is run on a 4 GHz Intel Core i7 with 16 GB of RAM running macOS Catalina.

As expected, Algorithm 1 can be executed very quickly and its execution time requires only a fraction of a second to handle long time series with frequent jumps. Algorithm 2 with $c_J$ as the sorting function, often takes less than one second to analyse dense test signals generated under different scenarios. An exception is the test signal `stairs10` where the candidates detected at large bandwidths tend to yield large jump size due to the nature of the signal, which results in large search space for Step 2.3 of Algorithm 2 and increased execution time. Similarly, the $p$ values associated with the candidates detected by large bandwidths for `blocks` and `mix` are often set at exactly zero, which creates many ties for the sorting function $c_p$ and leads to large search space. Nonetheless, Algorithm 2 with $c_p$ as the sorting function requires comparable or even smaller execution time than the WBS2 for the test signals `fms`, `teeth10` and `stairs10`.

## C. An asymptotic result

The following result describes the asymptotic distribution of the MOSUM statistic (for symmetric as well as asymmetric bandwidths) under the null hypothesis of no mean changes for i.i.d. innovations, extending the well-known result for symmetric bandwidths reported in Eichinger and Kirch (2018). Extensions to the dependent case are straightforward. The result is important for the described MOSUM procedure to obtain meaningful thresholds for the scaled MOSUM detector.

**Theorem 1.** *Let $X_1, \ldots, X_n$ be independently and identically distributed with mean $\mu$ and finite variance $\sigma^2$. Assume furthermore that $\mathsf{E}|X_1|^{2+\Delta} < \infty$ holds for some $\Delta > 0$. Consider a pair of bandwidths $\mathbf{G} = (G_\mathrm{l}, G_\mathrm{r})$ with $G_{\min} := \min(G_\mathrm{l}, G_\mathrm{r})$ and $G_{\max} := \max(G_\mathrm{l}, G_\mathrm{r})$ depending on $n$, such that $K_n := G_{\min}/G_{\max} \to K > 0$ as well as*

$$\frac{n^\rho}{G_{\min}} \to 0 \quad \textit{and} \quad \frac{n}{G_{\min}} \to \infty$$

*as $n \to \infty$ is fulfilled for some $\rho > \frac{2}{2+\Delta}$. Let*

$$T_\mathbf{G} := \max_{G_\mathrm{l} \le k \le n - G_\mathrm{r}} \frac{|T_\mathbf{G}(k)|}{\widehat{\sigma}_k}$$

*with the MOSUM detector $T_\mathbf{G}(k)$ as defined in (13), and a local estimator $\widehat{\sigma}_k^2$ of the innovation variance $\sigma^2$ fulfilling the following convergence assumption:*

$$\max_{G_\mathrm{l} \le k \le n - G_\mathrm{r}} \left|\widehat{\sigma}_k^2 - \sigma^2\right| = o_p\left((\log(n/G_{\min}))^{-1}\right). \tag{22}$$

*Then, the following distributional convergence holds:*

$$a_\mathbf{G} T_\mathbf{G} - b_\mathbf{G} \Longrightarrow \Gamma_2,$$

*where $\Gamma_2$ is a Gumbel-distributed random variable with its cumulative distribution function $P(\Gamma_2 \le z) = \exp(-2\exp(-z))$ for $z \in \mathbb{R}$, and the sequences $a_\mathbf{G}$ and $b_\mathbf{G}$ are given as*

$$a_\mathbf{G} = \sqrt{2\log\left(\frac{n}{G_{\min}}\right)}, \quad b_\mathbf{G} = 2\log\left(\frac{n}{G_{\min}}\right) + \frac{1}{2}\log\log\left(\frac{n}{G_{\min}}\right) + \log\left(\frac{K_n^2 + K_n + 1}{K_n + 1}\right) - \frac{1}{2}\log\pi.$$

Condition (22) is fulfilled for all the MOSUM-based variance estimators presented in Section 2.3.

*Proof.* We prove the results only for the case of known variance $\sigma^2$. The proof can readily be extended to the general case of a local variance estimator $\widehat{\sigma}_k^2$ fulfilling (22) by the same arguments as in the proof of Theorem 2.1 in Eichinger and Kirch (2018).

Without loss of generality , we can further assume that $\mu = 0$ and $\sigma^2 = 1$ as well as $G_{\min} = G_{\mathrm{l}}$, because of the following distributional equality:

$$T_{(G_{\mathrm{l}},G_{\mathrm{r}})}(k) \overset{\mathcal{D}}{=} T_{(G_{\mathrm{l}},G_{\mathrm{r}})}(n-k+1), \quad k = G_{\mathrm{l}}, \ldots, n - G_{\mathrm{r}}.$$

From $K_n = G_{\mathrm{l}}/G_{\mathrm{r}}$, we get the representation

$$T_{\mathbf{G}}(k) = \frac{1}{\sqrt{(K_n+1)G_{\mathrm{l}}}} \left( K_n \sum_{j=k+1}^{k+G_{\mathrm{r}}} X_j - \sum_{j=k-G_{\mathrm{l}}+1}^{k} X_j \right).$$

By the same arguments as in the proof of Theorem 2.1 in Eichinger and Kirch (2018), the partial sums of length $G_{\mathrm{l}}$ resp. $G_{\mathrm{r}}$ can asymptotically be approximated (after an appropriate change of probability space) by increments of a sequence $W_1, W_2, \ldots$ of standard Wiener processes:

$$\sup_{t\in[1,n/G_{\mathrm{l}}-1/K_n]} \left| \frac{1}{\sqrt{G_{\mathrm{l}}}} \sum_{j=\lfloor tG_{\mathrm{l}}\rfloor - G_{\mathrm{l}}}^{\lfloor tG_{\mathrm{l}}\rfloor} X_j - (W_n(t) - W_n(t-1)) \right| = o_p\left( (\log(n/G_{\mathrm{l}}))^{-1/2} \right)$$

and

$$\sup_{t\in[1,n/G_{\mathrm{l}}-1/K_n]} \left| \frac{1}{\sqrt{G_{\mathrm{l}}}} \sum_{j=\lfloor tG_{\mathrm{l}}\rfloor + 1}^{\lfloor tG_{\mathrm{l}}\rfloor + G_{\mathrm{r}}} X_j - (W_n(t+1/K_n) - W_n(t)) \right| = o_p\left( (\log(n/G_{\mathrm{l}}))^{-1/2} \right).$$

Thus, with

$$Y_n(t) := \frac{1}{\sqrt{K_n+1}} \Big( K_n (W(t+1+1/K_n) - W(t+1)) - (W(t+1) - W(t)) \Big)$$

for a standard Wiener process $(W(t))_{t\geq 0}$, it suffices to show for every $z$

$$P\left( \sup_{t\in[0,n/G_{\mathrm{l}}]} |Y_n(t)| \leq \frac{z+b_{\mathbf{G}}}{a_{\mathbf{G}}} \right) \longrightarrow \exp\left(-2\exp(-z)\right). \tag{23}$$

Note that $Y_n(t)$ is a stationary centred Gaussian process with variance 1. The autocovariance/autocorrelation function $\gamma_{Y_n}(t) = \mathsf{E}[Y_n(t)Y_n(0)]$ of $Y_n$ is given by

$$\gamma_{Y_n}(t) = 1 - K_n \min\left(t, \frac{1}{K_n}\right) + \frac{K_n}{1+K_n} \min\left(t, 1 + \frac{1}{K_n}\right) - \min(1,t).$$

In particular, it is bounded away from $\pm 1$ (for $t > 0$ bounded away from 0), fulfils $\gamma_{Y_n}(t) = 0$ for $t \geq 1 + 1/K_n$ and for $0 \leq t < 1$:

$$\gamma_Y(t) = 1 - \frac{K_n^2 + K_n + 1}{K_n + 1} t.$$

Consequently, the assumptions of Theorem 1 (ii) in Seleznjev (1991) are fulfilled with $T = n/G_{\mathrm{l}}$, $\alpha = 1$ and $u = (z + b_G)/a_G$. Some calculations show that $\tau = \exp(-z)$, proving (23). □

## Affiliation:

Alexander Meier
Institute for Mathematical Stochastics (IMST), Department of Mathematics
Otto von Guericke University Magdeburg
39106 Magdeburg, Germany
E-mail: `meier.alexander@posteo.de`

Claudia Kirch
Institute for Mathematical Stochastics (IMST), Department of Mathematics
Center for Behavioral Brain Science (CBBS)
Otto von Guericke University Magdeburg
39106 Magdeburg, Germany
E-mail: `claudia.kirch@ovgu.de`

Haeran Cho *(corresponding author)*
School of Mathematics
University of Bristol
Bristol BS8 1UG, United Kingdom
E-mail: `haeran.cho@bristol.ac.uk`

*Journal of Statistical Software* `http://www.jstatsoft.org/`
published by the Foundation for Open Access Statistics `http://www.foastat.org/`
March 2021, Volume 97, Issue 8 *Submitted:* 2018-04-06
`doi:10.18637/jss.v097.i08` *Accepted:* 2019-11-02