
Early version, also known as pre-print

Link to publication record in Explore Bristol Research
PDF-document

University of Bristol - Explore Bristol Research
General rights

This document is made available in accordance with publisher policies. Please cite only the published version using the reference above. Full terms of use are available:
http://www.bristol.ac.uk/pure/about/ebr-terms
mosum: A Package for Moving Sums in Change Point Analysis

Alexander Meier  
OvGU Magdeburg

Claudia Kirch  
OvGU Magdeburg

Haeran Cho  
University of Bristol

Abstract

Understanding changes plays a crucial role in many fields of science, economy, technology and medicine. Whenever time series data, i.e., temporally ordered data, is described by means of a stationary stochastic model, it is of interest to verify the stationarity assumption on the basis of the data, and change point analysis provides mathematical tools for this purpose. A particularly important, and thus widely studied, problem 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 to the multiple mean change problem using moving sum (MOSUM) statistics.

Keywords: MOSUM, change point analysis, time series.

1. Introduction

With its beginnings dating back as far as the 1950s (see Page (1954)), change point analysis is still a very active field of research. 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, which provides an implementation of the moving sum (MOSUM) procedure from Eichinger and Kirch (2018) in addition to an algorithmical multiscale extension for offline detection of multiple changes in the mean. It is available for the statistical computing language R (R Core Team 2018) from the Comprehensive R Archive Network (Meier, Cho, and Kirch 2018).

There exist many theoretical approaches and software implementations to 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), while the package strucchange (Zeileis, Leisch, Hornik, and Kleiber 2002) contains methods for testing structural changes.
in linear regression models; the latter also contains utility functions for empirical MOSUM processes. The \texttt{cumSeg} package (Muggeo 2012) contains methods for detecting changes in genomic sequences, and several methods for sequential change point detection are provided in the R package \texttt{cpm} (Ross 2015). See the repository provided by Killick, Nam, Aston, and Eckley (2012b) for an overview on recent developments in this area.

Meanwhile, there are also several R packages for offline multiple change point detection available. Most notably, implementations of the binary segmentation algorithm (Scott and Knott 1974; Sen and Srivastava 1975), the segment neighbourhood algorithm (Auger and Lawrence 1989; Bai and Perron 1998) and the PELT algorithm (Killick, Fearnhead, and Eckley 2012a) are provided in the \texttt{changepoint} package (Killick and Eckley 2014) and its extension \texttt{change-point.np} (Haynes, Killick, Fearnhead, and Eckley 2016) with nonparametric cost function. Furthermore, the wild binary segmentation algorithm is implemented in the \texttt{wbs} package (Fryzlewicz 2014), and the \texttt{ecp} package (James and Matteson 2014) provides an implementation of the nonparametric approach for multivariate data from Matteson and James (2014). The multiscale regression estimators SMUCE and HSMUCE can be obtained from the \texttt{stepR} package (Pein, Hotz, Sieling, and Aspelmeier 2017), and the \texttt{FDRSeg} package (Li and Sieling 2017) implements the method proposed in Li, Munk, and Sieling (2016) for multiscale inference for step functions with false discovery rate control.

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. The underlying MOSUM statistics have a clear and easy interpretation and lend themselves perfectly for visual inspection. Furthermore, the MOSUM framework is very flexible, since it does not rely on any distributional assumption of the underlying data generating process.

This paper is structured as follows. In Section 2, we explain the MOSUM statistics and the procedures for change point estimation. In particular, we discuss two algorithms for multiscale estimation, employing a range of summation bandwidths. Section 3 gives an introduction to the \texttt{mosum} package. All the methods for the evaluation of the MOSUM statistics, multiple change point estimation and visualisation, are explained and illustrated in short usage examples. Section 4 contains more detailed usage examples, and Section 5 summarises the contributions made in this work and provides an outlook for further research and development. The Appendix contains some algorithmic and implementation details in Section A and the proof of an asymptotic distributional result in Section B.

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

In this section, we will briefly discuss the idea of moving sums and describe the procedures for mean change point estimation which are implemented in the \texttt{mosum} package. In Section 2.1, we will review the intuition and the mathematical theory behind the MOSUM based statistics for multiple change point detection in the case of independent model innovations. In Section 2.2, we will 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 will present the extensions of the MOSUM procedure with multiple summation bandwidths. We will elaborate how to construct confidence intervals for the change point locations using bootstrap methods in Section 2.7.
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 $\delta$ at time point $k$, then

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

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

$$T_G := \max_{1 \leq k \leq n} \left| \frac{T_G(k)}{\hat{\sigma}_k} \right|$$

with the MOSUM detector

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

and a local estimator $\hat{\sigma}_k^2$ of the innovation variance (see the upcoming Section 2.3). The values $T_G(k)$ for $k$ at the left and right boundaries are obtained from a CUSUM-type boundary extension:

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

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

The observations $X_t$ are assumed to follow a classical change point location model (c.f., Eichinger and Kirch (2018))

$$X_t = f_t + e_t = \sum_{i=1}^{N+1} \mu_i \mathbb{I}\{k_{i-1} < t \leq k_i\} + e_t, \quad t = 1, \ldots, n.$$ 

The piecewise constant deterministic signal $f_t$ has $N$ change points at $k_i$, $i = 1, \ldots, N$ (with the 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

$$T_G(k; X) = T_G(k; f) + T_G(k; e).$$

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 $k_i$ for $i = 1, \ldots, N$ as well as the change heights $\delta_i = \mu_{i+1} - \mu_i$ are unknown. In many applications, estimation of both the number and locations of changes are of particular interest. The MOSUM detector lends itself naturally for this purpose,
mosum: A Package for Moving Sums in Change Point Analysis

Figure 1: The above panel shows a time series $X_1, \ldots, X_n$ of length $n = 400$ with one mean change of height $\delta = 2$ at time $k = 200$. The corresponding step signal $f_t$ is overlaid as dashed line. The middle panel shows 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. The lower panel depicts the noise part $T_G(k; e) = T_G(k; X) - T_G(k; f)$.

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 $G = G(n) \to \infty$ for $n \to \infty$ such that $G/n \to 0$ but not too fast (as shown in Appendix B),

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

where $\Rightarrow$ denotes the convergence in distribution, $\Gamma_2$ is a Gumbel-distributed random variable with $P(\Gamma_2 \leq z) = \exp(-2 \exp(-z))$, 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

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

(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 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 neighbourhoods exceeding the threshold. To elaborate, we consider significant neighbourhoods $(l, r)$ with $l \leq k \leq r$, such that

$$\frac{|T_G(k)|}{\hat{\sigma}_k} \geq C_{n,G}(\alpha) \quad \text{for } k = l, \ldots, r, \quad \text{and} \quad \frac{|T_G(k)|}{\hat{\sigma}_k} < C_{n,G}(\alpha) \quad \text{for } k = l-1, r+1.$$  

Choosing the maximal point within every significant environment

$$k_{(l,r)}^{(c)} = \arg \max_{l \leq k \leq r} \frac{|T_G(k)|}{\hat{\sigma}_k}$$  

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 as well as right of the true change point and will cross the threshold. The observed detector on the other hand adds noise to this signal, hence it can happen – merely by chance – that the detector falls beneath this line, then jumps above it again for just one time point or two, before crossing beneath the line again. Obviously, this results in additional significant neighbourhoods 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 $k_{(l,r)}^{(c)}$ as in (7) will be accepted as a change point estimate if and only if

$$r - l \geq \varepsilon G.$$  

(8)
mosum: A Package for Moving Sums in Change Point Analysis

Figure 2: A time series with one mean changes at the location 100 (above), where the underlying step signal \( f_t \) is overlaid as dashed line, and the corresponding absolute MOSUM detector \( |T_G(k)| \) for \( k = 1, \ldots, n \) with bandwidth \( G = 20 \) (below). 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 emphasised with vertical lines.

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 neighbouring change (as in Figure 2 at about time point 90), hence it will be discarded. From preliminary simulation studies, we found that \( \varepsilon = 0.2 \) is 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 neighbouring 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. A time point \( k^o \) in a significant neighbourhood (not necessarily the maximal point of that environment as in (7)) will be accepted as a change point estimator if and only if it is the maximal point within its own \( \eta G \) environment, i.e.,

\[
k^o = \arg \max_{k^o-\eta G \leq k \leq k^o+\eta G} \frac{|T_G(k)|}{\hat{\sigma}_k}.
\]

From preliminary 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.

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 neighbourhood 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.
Figure 3: 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 (above), and the corresponding scaled absolute MOSUM detector $|T_G(k)|/\sigma$ with $G = 8$ (below); the critical value for level $\alpha = 0.05$ is denoted by a horizontal line. The series is a realisation of the model teeth10 from Fryzlewicz (2014) (see Section 3.1).

Figure 4: 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 (left), and the corresponding scaled absolute MOSUM detector $|T_G(k)|/\sigma$ with $G = 15$ (right); the critical value for level $\alpha = 0.05$ is denoted by a horizontal line. The series is a realisation of the model stairs10 from Fryzlewicz (2014) (see Section 3.1).

Note however that the presence of changes is prominently indicated by the peaked shape of the detector.

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 maximiser 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. One solution to this problem is to use a local variance estimator

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

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

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

\[ \hat{\sigma}^2_k = \frac{1}{2} \left( \hat{\sigma}^2_{(k-G+1,k)} + \hat{\sigma}^2_{(k+1,k+G)} \right) \]  \hspace{1cm} (10)

and

\[ \hat{\sigma}^2_k = \min\{ \hat{\sigma}^2_{(k-G+1,k)}, \hat{\sigma}^2_{(k+1,k+G)} \} \]  \hspace{1cm} (11)

with the latter preferable when mean changes co-occur with changes in variance.

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 \( \hat{\sigma}^2_k \) are replaced by appropriate estimators for the long-run variance \( \tau^2 = \lim_{n \to \infty} \text{var} (\sqrt{n} \bar{e}_n) \) with \( \bar{e}_n = n^{-1} \sum_{t=1}^{n} e_t \). This is feasible if sufficiently large bandwidths are used (e.g., \( G \geq 50 \)) and/or if global information can be utilised despite the presence of multiple change points (see e.g., Axt and Fried (2018+)). Otherwise, this is not recommended as the accurate estimation of the long-run variance is typically very difficult and, therefore, estimators based on small and medium sample sizes are not very precise. An alternative approach is to use the local variance estimators (ignoring the dependency 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 neighbouring change points close by. On the other hand, large bandwidths may not be suitable for estimating the locations of such change points due to contamination of the signal by neighbouring 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 neighbourhood will effectively be arbitrary on that interval, so it cannot be used for estimation purposes.

On the plus side, 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.
Figure 5: From top to bottom: A time series with mean changes at locations 100, 300 and 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 visualised by a solid horizontal line.
mosum: A Package for Moving Sums in Change Point Analysis

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 $|T_G(k)|$ for $k = 1, \ldots, n$ with (symmetric) bandwidths $G = 30, 130$ and $G = (30, 130)$.

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: The change may be too small to be detected by a small bandwidth, whereas the summation windows will be contaminated by the neighbouring large mean change when a large bandwidth is used; see the top three panels of Figure 6 for an illustration. One way of overcoming this limitation is to consider the use of asymmetric bandwidths $G = (G_l, G_r)$, which will be discussed in the following Section 2.5.

2.5. Asymmetric bandwidths

Let $G = (G_l, G_r)$ be a bandwidth, possibly asymmetric (i.e., $G_l \neq G_r$). The asymmetric MOSUM detector is defined as

$$T_G(k) = \sqrt{G_l \cdot G_r \left( \frac{1}{G_l} \sum_{t=k+G_l}^{k+G_r} X_t - \frac{1}{G_r} \sum_{t=k-G_r+1}^{k} X_t \right)} , \quad k = G_l, \ldots, n - G_r. \quad (12)$$

At the left and right boundaries, the values of $T_G(k)$ can be defined by a CUSUM extension
similar to (3):

\[ T_G(k) := \sqrt{\frac{G_l + G_r}{k(G_l + G_r - k)} \sum_{t=1}^{k} (X_{t(G_l+G_r)} - X_t), \quad k = 1, \ldots, G_l - 1,} \]

(13)

and analogously for \( k = n - G_r + 1, \ldots, n \).

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

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

\[ r - l \geq \frac{\varepsilon}{2}(G_l + G_r), \]

(14)

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

\[ k^0 = \arg \max_{k^0 - \eta G_l \leq k^0 \leq k^0 + \eta G_r} \left| \frac{T_G(k)}{\hat{\sigma}_k} \right|. \]

(15)

The MOSUM based variance estimators are adjusted in a similar fashion as the MOSUM detector. To avoid bandwidths that are too unbalanced (e.g., \( G = (15, 300) \)), one can use a threshold \( \kappa_0 \) to guarantee \( \max(G_l, G_r) \leq \min(G_l, G_r) \leq \kappa_0 \).

2.6. Multiple bandwidths

In Section 2.4, it has been argued that the use of one single bandwidth (symmetric or asymmetric) may not be adaptive enough to detect different types of changes, advocating 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 merging procedures which are performed on a candidate set \( P \) of possible change points that collects the change point estimators \( K_G \) from each bandwidth \( G \), i.e., \( P = \{(k, G): \hat{k} \in K_G, G \in G \} \) where \( G \) is a set of bandwidths used.

**Multiscale MOSUM procedure with ‘bottom up’ merging**

As argued in Section 2.4, one possible issue of large bandwidths is that their summation window may contain, and thus be contaminated by, several changes, making the corresponding estimators unreliable. At the same time, such change points may be detectable by smaller bandwidths. Motivated by these considerations and following Messer, Kirchner, Schiemann, Roeper, Neininger, and Schneider (2014), it seems reasonable to keep all the estimates obtained from the smallest bandwidth and, iteratively moving on to the next smallest bandwidth, only keep those that cannot be accounted for by previously accepted estimates.

To elaborate, with increasing order of the bandwidth, we check whether there exists a previously accepted change point \( \hat{k} \) that is close to the current candidate \( \hat{k} \) detected with the bandwidth \( G \), in the sense that \( |\hat{k} - \hat{k}| < \eta G \). If so, we conclude that the estimation of \( \hat{k} \) has been influenced by \( \hat{k} \) and thus discard \( \hat{k} \); otherwise we keep it. We note that the tuning parameter \( \eta \) bears close resemblance to the parameter \( \eta \) from the \( \eta \)-criterion in (9). For this
Algorithm 1: Multiscale MOSUM procedure with bottom up merging

\textbf{input}: Data \((X_1, \ldots, X_n)\), set \(\mathcal{G}\) of symmetric bandwidths, \(\alpha, \eta \in (0, 1)\)

\begin{algorithmic}
1 \text{Initialise} \(\mathcal{P} \leftarrow \emptyset, \mathcal{K} \leftarrow \emptyset;\)
2 \text{/* Step 1: Generate candidates */}
3 \textit{for} \(G \in \mathcal{G}\) \text{do}
4 \hspace{1em} \(K_G \leftarrow\) set of MOSUM change point estimates with bandwidth \(G\) and critical value \(C_n,G(\alpha)\) according to criterion (9);
5 \hspace{1em} \textit{for} \(\hat{k} \in K_G\) \text{do Add} \((\hat{k}, G)\) to \(\mathcal{P};\)
6 \text{end}
7 \text{/* Step 2: Merging in increasing order with respect to} G \text{*/}
8 \textit{for} \((\hat{k}, G) \in \mathcal{P}\) \textit{in ascending order with respect to} \(G\) \text{do}
9 \hspace{1em} \text{if} \(\max_{\tilde{k} \in \mathcal{K}} |\tilde{k} - \hat{k}| \geq \eta G\) \text{then Add} \(\hat{k}\) to \(\mathcal{K};\)
10 \text{end}
11 \text{output:} \(\mathcal{K}\)
\end{algorithmic}

Algorithm 2: Multiscale MOSUM procedure with localised pruning

\textbf{input}: Data \((X_1, \ldots, X_n)\), set \(\mathcal{G}\) of symmetric bandwidths, \(\alpha, \eta \in (0, 1)\)

\begin{algorithmic}
1 \text{Initialise} \(\mathcal{P} \leftarrow \emptyset, \mathcal{K} \leftarrow \emptyset;\)
2 \text{/* Step 1: Generate candidates */}
3 \textit{for} \(G \in \mathcal{G}\) \text{do}
4 \hspace{1em} \(K_G \leftarrow\) set of MOSUM change point estimates with bandwidth \(G\) and critical value \(C_n,G(\alpha)\) according to criterion (9);
5 \hspace{1em} \textit{for} \(\hat{k} \in K_G\) \text{do Add} \((\hat{k}, G)\) to \(\mathcal{P};\)
6 \text{end}
7 \text{/* Step 2: Merging in increasing order with respect to} G \text{*/}
8 \textit{for} \((\hat{k}, G) \in \mathcal{P}\) \textit{in ascending order with respect to} \(G\) \text{do}
9 \hspace{1em} \text{if} \(\max_{\tilde{k} \in \mathcal{K}} |\tilde{k} - \hat{k}| \geq \eta G\) \text{then Add} \(\hat{k}\) to \(\mathcal{K};\)
10 \text{end}
11 \text{output:} \(\mathcal{K}\)
\end{algorithmic}

reason, the implementation of the procedure in the R package \texttt{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 \textit{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 advantage of this approach is that it is computationally very efficient and easy to implement. There are, however, several drawbacks as well: First, this idea only works with multiple symmetric bandwidths since a set of asymmetric bandwidths does not obey a canonical ordering. Secondly, the effect of multiple testing should be taken into account, which has been done in Messer \textit{et al.} (2014) for the problem considered therein. More importantly, the bandwidths in consideration need to be large enough for the critical value from the asymptotic distribution to be meaningful, i.e., to avoid false positives in particular for small bandwidths. This is taken care of in Messer \textit{et al.} (2014) by using only bandwidths of order \(n\). However, for signals with dense change points in some parts but no jumps in other parts, this poses as a real limitation.

**Multiscale MOSUM procedure with localised pruning**

A comprehensive high-level description of the multiscale MOSUM procedure with localised merging is given in Algorithm 2 where, in contrast to Algorithm 1, asymmetric bandwidths are allowed. In Step 2 of Algorithm 2, the change point candidates \((\hat{k}, \hat{G}) \in \mathcal{P}\) are processed along with their detection bandwidths \(\hat{G} = (\hat{G}_l, \hat{G}_r)\). In order to determine the order of their consideration, we adopt a criterion function \(c\), for which we use the inverse of the corresponding \(p\)-value \((c_p)\) and the (scaled) jump size \((c_J)\) defined as below:

\[
c_p(\hat{k}, \hat{G}) = \frac{1}{p_{n,G}(T_G(\hat{k}))} \quad \text{and} \quad c_J(\hat{k}, \hat{G}) = \sqrt{\frac{\hat{G}_l + \hat{G}_r}{\hat{G}_l \hat{G}_r} T_G(\hat{k})} \frac{\hat{G}_l + \hat{G}_r}{\hat{G}_l \hat{G}_r}. \quad (16)
\]

All change point candidates are processed one by one in the decreasing order with respect to \(c_p\) or \(c_J\). In case of ties, we order the candidates with respect to the length of their detection
Algorithm 2: Multiscale MOSUM procedure with localised merging

**input**: Data \((X_1, \ldots, X_n)\), set \(G\) of (possibly asymmetric) bandwidths, \(\alpha\) and \\
\(\varepsilon\) or \(\eta \in (0, 1)\), cost function \(c(\cdot) = c_p(\cdot)\) or \(c(\cdot) = c_J(\cdot)\) from (16)

1. Initialise \(C \leftarrow P \leftarrow \emptyset\), \(\mathcal{K} \leftarrow \emptyset\);

   /* Step 1: Generate candidates */

2. Obtain the candidate set \(P\) as in Step 1 of Algorithm 1 and set the current candidate set \\
\(C \leftarrow P\);

   /* Step 2: Merge in decreasing order with respect to \(c(\cdot)\) */

3. while \(P\) is not empty do

   4. \((\hat{k}, \hat{G}) \leftarrow \arg\max_{(\tilde{k}, \tilde{G}) \in P} c(\tilde{k}, \tilde{G})\) ;

      /* Step 2.1: Find set of conflicting candidates \(D\) */

   5. \(\hat{k}_1 \leftarrow \min\{\tilde{k} : (\hat{k}, \hat{G}) \in P, \hat{k} - \hat{k} < \min(\hat{G}_l, \hat{G}_r)\ \text{and} \ \hat{k} > \max\mathcal{K} \cap [1, \hat{k})\};

   6. \(\hat{k}_r \leftarrow \max\{\tilde{k} : (\hat{k}, \hat{G}) \in P, \hat{k} - \hat{k} < \min(\hat{G}_l, \hat{G}_r)\ \text{and} \ \hat{k} < \min\mathcal{K} \cap (\hat{k}, n]\};

   7. \(D \leftarrow \{(\tilde{k}, \tilde{G}) \in P : \hat{k}_1 \leq \tilde{k} \leq \hat{k}_r\}\);

      /* Step 2.2: Perform exhaustive search on conflicting set */

   8. \(s \leftarrow \max\{\tilde{k} \in \mathcal{C} : \tilde{k} < \hat{k}_1\}\), \(e \leftarrow \min\{\tilde{k} \in \mathcal{C} : \tilde{k} > \hat{k}_r\}\);

   9. \(\mathcal{A} \leftarrow \text{set fulfilling (I)-(II) with } \mathcal{C}, \mathcal{D} \text{ and } [s, e]\);

      /* Step 2.3: Merge results from exhaustive search */

   10. \(\mathcal{R} \leftarrow \{(\hat{k}, \hat{G})\} \cup \{(\tilde{k}, \tilde{G}) \in \mathcal{D} : (\hat{k} - \hat{G}_l, \hat{k} + \hat{G}_r) \subset (s, e) \text{ or } \hat{k} \in [\min\mathcal{A}, \max\mathcal{A}]\} ;

   11. Add \(\mathcal{A}\) to \(\mathcal{K}\), remove \(\mathcal{R}\) from \(P\) and afterwards set \(C \leftarrow P \cup \mathcal{A}\);

end

**output**: \(\mathcal{K}\)
mosum: A Package for Moving Sums in Change Point Analysis

intervals and associated bandwidths. For each candidate, the set of conflicting (i.e., with overlapping detection bandwidths) candidates $D$ is computed in Step 2.1 of Algorithm 2. In an exhaustive search performed in Step 2.2 of Algorithm 2, we employ the Schwarz Criterion (SC). It considers the inclusion or exclusion of every change point candidate in $D$ over a local interval $[s,e]$, which is determined by the change point estimates immediately outside $D$ (see line 8 of Algorithm 2). Specifically, for every $A \subset D$, SC is obtained as

$$ SC(A|C, [s,e]) = \frac{n}{2} \log \hat{s}^2(A \cup (C \setminus [s,e])) + (|A| + |C \setminus [s,e]|) p(n)^\rho, $$

(17)

where $C$ denotes the set of current change point candidates (either have already been selected or have not yet been considered), $p(n) = \log(n)$ or $p(n) = n$, and

$$ \hat{s}^2(Q) = \sum_{j=0}^{m} \sum_{t=\tilde{k}_j}^{\tilde{k}_{j+1}} \left( X_t - \bar{X}_{(\tilde{k}_j+1, \tilde{k}_{j+1})} \right)^2 $$

(18)

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

Among the $2^{|D|}$ subsets of $D$, a subset $A$ that meets the following properties is selected by Step 2.2 of Algorithm 2:

(I) adding further candidates from $D$ to $A$ monotonically increases $SC(A|C, [s,e])$,

(II) removing any single element from $A$ increases $SC(A|C, [s,e])$, and

(III) its cardinality is the minimum among all subsets of $D$ satisfying (I)–(II).

If there are several subsets fulfilling (I)–(III), say $A_1, \ldots, A_M$, we choose the one that minimises the information criterion: $A := A_{l^*}$ with $l^* := \arg\min_{1 \leq l \leq M} SC(A_l|C, [s,e])$. Finally, the chosen subset $A \subset D$ is added to the set of estimated change points in Step 2.3 of Algorithm 2, while each candidate $(\tilde{k}, \tilde{G}) \in D$ whose detection environment has been completely contained in the considered set (i.e., $(\tilde{k} - \tilde{G}_l, \tilde{k} + \tilde{G}_r] \subset [s,e]$) is also removed from the candidate set $P$ for the future consideration. An efficient algorithm for the exhaustive search step is outlined in Section A of the Appendix. We defer further algorithmic and mathematical details to Cho and Kirch (2018).

A similar idea has been exploited by Yau and Zhao (2016), who also use an information criterion approach based on a candidate set obtained from a MOSUM procedure. However, their candidate set is obtained from a single-bandwidth MOSUM procedure, where all local maxima (irrespective of whether they are significant or likely to be contaminated) are used in a global minimisation of the information criterion.

2.7. Bootstrap confidence intervals

Consider a set $\hat{k}_1, \ldots, \hat{k}_R$ 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 \((N = N)\) and locations, we can construct confidence intervals for the locations of change points via bootstrapping as outlined below.

Denote the (possibly asymmetric) detection bandwidths associated with \(\hat{k}_j\) by \(\hat{G}_j = (\hat{G}_{1j}, \hat{G}_{rj})\), and let \(I_j := \{k_{j-1} + 1, \ldots, \hat{k}_j\}\) for \(j = 1, \ldots, N+1\), where the convention \(\hat{k}_0 := 0\) and \(\hat{k}_{N+1} := n\) 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 range. 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 \(\{X_t: t \in I_j\}\) (with replacement). Doing this for all \(j = 1, \ldots, N+1\) yields a bootstrap replicate \(\{X_t^*: t \in I\}\) of the observed time series. Then a bootstrap replicate \(k^*_j\) of \(\hat{k}_j\) is obtained as

\[
k^*_j = \underset{\hat{k}_j - \hat{G}_{1j} + 1 \leq k \leq \hat{k}_j + \hat{G}_{rj}}{\arg\max} \left| T^*_G(k) \right|,
\]

where \(T^*_G(k) = T_G(k; X_1^*, \ldots, X_n^*)\) is defined as in (12), with the \(X_t\) replaced by their 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 \(\hat{k}_j\), we have \(B\) bootstrap replicates \(k^*_{j,b}, 1 \leq b \leq B\). Denote by \(M_j(\alpha)\) the empirical \((1 - \alpha/2)\)-quantile of \(|\{k^*_{j,1} - \hat{k}_j, \ldots, k^*_{j,B} - \hat{k}_j\}|\). Then a pointwise 100(1 - \(\alpha\))%-confidence interval \(\hat{K}_j(\alpha)\) for \(k_j\) can be constructed as

\[
\hat{K}_j(\alpha) := [\hat{k}_j - M_j(\alpha), \hat{k}_j + M_j(\alpha)].
\]

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 \(M(\alpha)\) that satisfies

\[
\frac{1}{B} \sum_{b=1}^B \mathbb{I} \left\{ \max_{1 \leq j \leq N} \left( \frac{\delta^2_j}{\hat{\sigma}_j^2} \left| k^*_{j,b} - \hat{k}_j \right| \right) \leq M(\alpha) \right\} \geq 1 - \alpha,
\]

where \(\delta_j = X_{(k_{j+1}, k_{j+1})} - X_{(k_{j-1}, k_{j-1})}\) denotes the estimated jump height at \(t = k_j\), and

\[
\hat{\sigma}_j^2 = \frac{1}{k_{j+1} - k_{j-1} - 2} \left( \sum_{t=k_{j+1}}^{k_{j+1}} (X_t - \tilde{X}_{(k_{j+1}, k_{j+1})})^2 + \sum_{t=k_{j-1}+1}^{k_j} (X_t - \tilde{X}_{(k_{j-1}+1, k_j)})^2 \right)
\]

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

\[
\tilde{K}_j(\alpha) := [\hat{k}_j - M(\alpha)\delta^2_j, \hat{k}_j + M(\alpha)\delta^2_j], \quad j = 1, \ldots, N.
\]

Note that for each \(j = 1, \ldots, N\), if either of the two endpoints of \(\tilde{K}_j(\alpha)\) falls outside the detection interval \((\hat{k}_j - \hat{G}_{1j}, \hat{k}_j + \hat{G}_{rj})\), we trim off the confidence intervals so that \(\tilde{K}_j(\alpha) \subset (\hat{k}_j - \hat{G}_{1j}, \hat{k}_j + \hat{G}_{rj})\); a similar step is taken for the uniform confidence intervals \(\tilde{K}_j(\alpha)\).

The above bootstrap scheme is the simplest (i.e., non-studentised) version for i.i.d innovations; further details and variations are discussed in Cho and Kirch (2018).
3. Introduction to the package

In this section, we will give a detailed introduction to the mosum package. It is structured as follows: We start with a brief guide on how to generate piecewise stationary time series from several models in Section 3.1. In Section 3.2, we explain the implementation of the (single-bandwidth) MOSUM procedure described in Section 2.1–2.2. In the subsequent Sections 3.3 and 3.4, we introduce the implementations of Algorithms 1–2 from Section 2.6, respectively. We present how bootstrap confidence intervals for change points can be obtained in Section 3.5, and discuss some tools for visualisation of multiscale MOSUM statistics in Section 3.6.

The software interface of each function will be discussed in detail, with references to the mathematical description in Section 2 where applicable. We also briefly discuss a small usage example for each function, before elaborating more detailed applications in Section 4.

3.1. Generating piecewise stationary time series

The function testData can be used to generate piecewise stationary time series with i.i.d innovations. It takes the following arguments:

- **model**: A string indicating from which model a realisation is to be generated. The default choice is "custom", in which case the user has to parse the piecewise stationary model with the arguments lengths, means and sds. In addition, five change point models "blocks", "fms", "mix", "stairs10" and "teeth10" from Fryzlewicz (2014) have been implemented. If one of these models is chosen, the arguments lengths, means and sds do not have to be specified.

- **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 numeric vector containing a realisation of the specified time series model. It is also possible to obtain the deterministic piecewise constant underlying step signal $f_t$ by using the function testSignal. As an example, we consider a realisation of the mix time series model, visually overlaid by the corresponding step signal:

```r
R> signal <- testSignal(model = "mix")$mu_t
R> timeSeries <- testData(model = "mix")
R> plot.ts(timeSeries, col = "darkgray")
R> lines(signal, 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`:

```r
R> mosum(x, G, G.right = NA, var.est.method = "mosum", var.custom = NULL,
+   boundary.extension = T, threshold = c("critical.value", "custom")[1],
+   alpha = 0.05, threshold.custom = NULL, criterion = c("eta",
+   "epsilon")[1], eta = 0.4, epsilon = 0.2, do.confint = F, level = 0.05,
+   N_reps = 1000)
```

The function takes 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`.
- `G`: Bandwidth $G$, i.e., a single positive integer. Alternatively, a single numeric value in $(0, 1)$ describing $G$ as a fraction of the data length $n$ can be given.
- `G.right`: Length of the right summation window $G_r$, i.e., a single positive integer, if a symmetric bandwidth $G = (G_l, G_r)$ is used. As with $G$, it can alternatively be given as a single numeric value in $(0, 1)$.
- `var.est.method`: A string encoding how the local variance estimation $\hat{\sigma}_k^2$ 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;
- `var.custom`: The custom local variance estimates $\hat{\sigma}_k^2$ for $k = 1, \ldots, n$ 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 $T_G(k)$ for $1 \leq k \leq G_l - 1$ and $n - G_r + 1 \leq k \leq n$ shall be padded with CUSUM values, see (3). If `boundary.extension = F`, 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 (`threshold = "critical.value"`),
the asymptotic critical value $C_{n,G}(\alpha)$ from (6) is used, where the significance level $\alpha$ is given by the parameter \texttt{alpha}. Alternatively, with \texttt{threshold} = "custom", it is possible to parse a user-defined numerical value with the argument \texttt{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.

- \texttt{alpha}: A single numeric value in $(0, 1)$ representing the significance level for the critical value. Only in use if \texttt{threshold} = "critical.value".
- \texttt{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 \texttt{threshold} = "custom".
- \texttt{criterion}: A string indicating which change point detection criterion shall be employed. Possible options are "epsilon" and "eta" for the $\varepsilon$-criterion and $\eta$-criterion, respectively (see (14) and (15) in Section 2.5).
- \texttt{epsilon}: A numeric value in $(0, 1]$ for $\varepsilon$ in the $\varepsilon$-criterion. Only in use if \texttt{criterion} = "epsilon";
- \texttt{eta}: A numeric value greater than 0 for $\eta$ in the $\eta$-criterion. Only in use if \texttt{criterion} = "eta";
- \texttt{do.confint}: A boolean argument indicating whether to compute the confidence intervals for change-points.
- \texttt{level}: A single numeric value in $(0, 1]$ representing the confidence level for the confidence intervals. Only in use if \texttt{do.confint} = TRUE.
- \texttt{N.reps}: A single positive integer representing the number of bootstrap replicates to be generated for confidence interval construction. Only in use if \texttt{do.confint} = TRUE.

When called, the function \texttt{mosum} returns an S3 object of class \texttt{mosum.cpts}, containing the following entries (apart from the call arguments):

- \texttt{stat}: Scaled absolute MOSUM detector $\hat{\sigma}_k^{-1}|T_G(k)|$ for $1 \leq k \leq n$, as a numeric vector of length $n$.
- \texttt{rollsums}: Unscaled MOSUM detector $T_G(k)$ for $1 \leq k \leq n$, as a numeric vector of length $n$.
- \texttt{var.estimation}: Values of $\hat{\sigma}_k^2$ estimated as specified by \texttt{var.est.method}, as a numeric vector of length $n$.
- \texttt{cpts}: A vector containing the locations of the estimated change points.
- \texttt{ci}: An object of class \texttt{cpts.ci} containing confidence intervals for the change points. Returned iff \texttt{do.confint} = TRUE; see Section 3.5 for further details.

S3 objects of class \texttt{mosum.cpts} are supported by \texttt{plot}, \texttt{summary}, \texttt{print} and \texttt{confint} methods. As an illustration, we analyse \texttt{Nile}, a time series of length $n = 100$ containing the annual flow of the Nile at Aswan from the R package \texttt{datasets} (see \texttt{help(Nile)} for further information about the dataset). First, we visualise the scaled absolute MOSUM detector.
Figure 8: Annual flow of the Nile at Aswan from 1871 to 1970 (above) and the corresponding values of the scaled absolute MOSUM detector with bandwidth $G = 20$ (below). The critical value is visualised by a solid horizontal line and the location of the estimated change point location by a solid vertical line.

```r
R> m <- mosum(Nile, G = 20)
R> par(mfcol = c(2, 1))
R> plot(Nile)
R> plot(m)
```

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. Note that this timespan corresponds to the significant neighbourhood discussed in Section 2.2. Adopting the $\eta$-criterion described in (9) with the default choice $\eta = 0.4$, a single change point is estimated as shown below.

```r
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
```

Note that 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 the corresponding detection bandwidths (`G.left` and `G.right`) as well as the $p$-values (`p.value`) evaluated at the MOSUM detector values, and the corresponding change heights (`jump`).
3.3. Multiscale MOSUM procedure with bottom up merging

The function `multiscale.bottomUp` provides an implementation of Algorithm 1 from Section 2.6:

```r
R> multiscale.bottomUp(x, G = bandwidths.default(length(x)),
+   threshold = c("critical.value", "custom")[1], alpha = 0.05,
+   threshold.function = NULL, eta = 0.4, do.confint = F, level = 0.05,
+   N_reps = 1000, ...)
```

It accepts the following arguments:

- **x**: Input data $X_1, \ldots, X_n$ as a numeric vector (of length $n$) or an object of class `ts`.
- **G**: A set $G$ of (symmetric) bandwidths, represented as a vector of integers or numeric values in $(0, 0.5)$ describing the moving sum bandwidths relative to $n$.
- **threshold**: A string indicating which threshold should be used to determine significance of the scaled absolute MOSUM detector. By default (`threshold = "critical.value"`), 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`. An example for the latter case will be presented later in Section 4.2.
- **alpha**: The level of significance level to be parsed to `mosum` (see Section 3.2).
- **threshold.custom**: A user-specified function of the form `function(G)` for computing a threshold of significance for an integer $G$ representing different bandwidths. Only in use if `threshold = "custom"`.
- **eta**: A numeric value greater than 0 for the $\eta$-criterion adopted in the candidate generation step (see Step 1 in Algorithm 1); to be parsed to `mosum`.
- **do.confint, level, N_reps**: Arguments for generating change point confidence intervals; (see Section 3.2).
- **...**: Further arguments to be parsed to `mosum` function (see Section 3.2).

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

- **cpts**: Estimated change point set $\mathcal{K}$ (see Algorithm 1), represented as an integer vector of length $\hat{N}$.
- **cpts.info**: A data frame containing the estimated change point set $\mathcal{K}$, the respective detection bandwidth, the associated $p$-values and change heights as its columns.
- **pooled.cpts**: Candidate set $\mathcal{P}$ considered by Algorithm 1, represented as an integer vector.
- **ci**: An object of class `cpts.ci` containing confidence intervals for the change points. Returned iff `do.confint = TRUE`; see Section 3.5 for further details.

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

The default option for bandwidth generation is the function
R> bandwidths.default(n, d.min = 10)

which takes two numeric values as its input, representing the sample size \( n \) and the minimal mutual distance between change points that can be expected \( (d.\ min) \). The function returns an integer vector \( (G_1, \ldots, G_m) \) with \( G_0 = G_1 = \max\{8, 2d_{\ \min}/3\} \) and \( G_{j+1} = G_{j-1} + G_j \) for \( j = 1, \ldots, m-1 \), with \( m \) chosen as large as possible such that \( G_m \leq 3\sqrt{n} \) while \( G_{m+1} > 3\sqrt{n} \). It is possible to give the minimum and maximum permitted bandwidths as input arguments \( (G.\ min \ \text{and} \ G.\ max) \) to be used in place of 8 and \( 3\sqrt{n} \), respectively, but this is not recommended.

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 \( k_1 = 50, k_2 = 100, k_3 = 300 \) and mean jump heights \( \delta_1 = 1, \delta_2 = 2, \delta_3 = -3 \):

R> x <- testData(lengths = c(50, 50, 200, 300), means = c(0, 1, 3, 0),
+ sds = rep(1, 4), seed = 123)
R> mbu <- multiscale.bottomUp(x, G = c(30, 50, 80, 130), alpha = 0.1)
R> print(mbu$cpts); print(mbu$pooled.cpts)

\[
\begin{align*}
[1] & 50 \\
[1] & 96 100 300
\end{align*}
\]

It can be seen that the output \( K = \{50, 100, 300\} \) of the algorithm coincides with the ground truth, 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 localised pruning

The function \texttt{multiscale.localPrune} provides an implementation of Algorithm 2 from Section 2.6:

R> multiscale.localPrune(x, G = bandwidths.default(length(x)),
+ threshold = c("critical.value", "custom")[1], alpha = 0.05,
+ 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 = F,
+ level = 0.05, N.reps = 1000, ...)

It accepts the following arguments, in addition to those supplied for \texttt{mosum} and \texttt{multiscale.bottomUp}:

- \textbf{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 \( (c_J, \text{see (16)}) \).

- \textbf{penalty}: A string indicating which penalty \( p(n) \) to be used in the SC (17). Possible values are "log" for \( p(n) = \log(n) \) and "polynomial" for \( p(n) = n \).

- \textbf{pen.exp}: A numeric value for the exponent \( \rho \) in the penalty term of the SC (17).

- \textbf{...}: Further arguments to be parsed to the \texttt{mosum} function (see Section 3.2).
When called, `multiscale.localPrune` returns an S3 object of class `multiscale.cpts`, as described in Section 3.3.

As a simple example, we consider the normal time series \( x \) from Section 3.3, analysed with an asymmetric bandwidth grid \( G \) of size 16 obtained as the Cartesian product of the set \{30, 50, 80, 130\} with itself:

```r
R> mlp <- multiscale.localPrune(x, G = c(30, 50, 80, 130), alpha = 0.1)
R> print(mlp$cpts); print(mlp$pooled.cpts)
[1] 50 100 300
[1] 48 50 86 96 100 300
```

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

### 3.5. 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 class `mosum.cpts` or `multiscale.cpts` as an input:

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

- **object**: An object either of class `mosum.cpts` or `multiscale.cpts` as returned by `mosum`, `multiscale.bottomUp` or `multiscale.localPrune`. If `object$do.confint = TRUE`, `object$ci` is returned without further bootstrapping.
- **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 \(\hat{K}_j(\alpha)\) from (19) and \(\tilde{K}_j(\alpha)\) from (20).
- **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`), the pointwise confidence intervals \(\hat{K}_j(\alpha)\) (in columns `pw.left` and `pw.right`) and the uniform confidence intervals \(\tilde{K}_j(\alpha)\) (in columns `unif.left` and `unif.right`).

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

```r
R> mlp_ci <- confint(mlp, level = 0.05, N_reps = 10000)
R> print(mlp_ci$CI)
```
<table>
<thead>
<tr>
<th></th>
<th>cpt</th>
<th>pw.left</th>
<th>pw.right</th>
<th>unif.left</th>
<th>unif.right</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>50</td>
<td>20</td>
<td>80</td>
<td>22</td>
<td>78</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>95</td>
<td>105</td>
<td>90</td>
<td>110</td>
</tr>
<tr>
<td>3</td>
<td>300</td>
<td>298</td>
<td>302</td>
<td>296</td>
<td>304</td>
</tr>
</tbody>
</table>

It shows that e.g., the 95% bootstrap confidence intervals for \( k_2 = 100 \) are given by \( \hat{K}_2(0.05) = [95, 105] \) (pointwise) and \( \tilde{K}_2(0.05) = [90, 110] \) (uniform).

### 3.6. Visualisation

The plots of the scaled absolute MOSUM detector is particularly suitable for visually inspecting the data for possible change points. There is a `plot` function available for S3 objects of class `mosum.cpts`, which plots the scaled absolute MOSUM detector along with the critical value and estimated change points where applicable; see Figure 6 and Figure 8.

For visualising the output of the multiscale algorithms discussed in Sections 3.3–3.4, the function `plot.multiscale.cpts` can be used. It visualises the set of estimated change points (on the x-axis) and one minus p-values associated with their detection (on the y-axis). Confidence intervals or the detection environments can also be visualised as shaded areas. To elaborate:

```r
R> plot(x, shaded = c("CI", "bandwidth")[1],
+       level = 0.05, N_reps = 10000, CI = c("pw", "unif")[1], ...)
```

It accepts the following arguments:

- `x`: An object of class `multiscale.cpts`.
- `shaded`: A string indicating whether confidence intervals (`shaded = "CI"`) or detection intervals (`shaded = "bandwidth"`) shall be plotted as shaded areas along with the estimated change points.
- `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 visualise the estimated change points along with their detection environment and the 95% bootstrap confidence intervals of respective change points:

```r
R> par(mfcol = c(4, 1), mar = c(2, 4, 2, 2))
R> plot.ts(x, col = "gray")
R> lines(testSignal(lengths = c(50, 50, 200, 300), means = c(0, 1, 3, 0),
+        sds = rep(1, 4))$mu_t, lty = 2, col = 2, lwd = 2)
R> plot(mlp, shaded = "bandwidth")
R> plot(mlp, shaded = "CI", CI = "pw")
R> plot(mlp, shaded = "CI", CI = "unif")
```

The output is shown in Figure 9.

It is possible to obtain a 3D surface plot of standardised scaled absolute MOSUM detectors with ‘continuous’ bandwidths using the function `persp3D.multiscaleMosum` function:
Figure 9: Top: The time series realisation $x$ and its underlying signal (dashed line). Below: Visualisation of the estimated change points from `multiscale.localPrune`, where the $x$-axis represents the change point locations and the $y$-axis depicts one minus the $p$-values evaluated at the scaled absolute MOSUM detector values associated with the estimated change points. The detection intervals of estimated change points, the bootstrap pointwise confidence intervals $\hat{K}_j(0.05)$ from (19) and the uniform confidence intervals $\tilde{K}_j(0.05)$ from (20) for $j = 1, 2, 3$ are visualised as shaded areas surrounding the estimated change points (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 standardised 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 have been 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 thresholds for standardising scaled MOSUM detectors $|T_G(k)|/\hat{\sigma}_k$ 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 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 (2016), and further information can be found in the R documentation thereof. The range of the colour 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 visualisation of the MOSUM detectors for the piecewise stationary reference time series `mix` from the literature (see Figure 7 for a visualisation thereof):

```r
R> x <- testData(model = "mix")
R> persp3D.multiscaleMosum(x, mosum.args = list(boundary.extension = F))
```

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 small changes at the beginning are given a prominence at small bandwidths, whereas the peaks of the large change points slowly become more significant with growing bandwidth.

### 4. Usage examples

In this section, we describe three examples demonstrating the usage of the `mosum` package for detecting multiple change points in the mean. It is structured as follows: We start with
mosum: A Package for Moving Sums in Change Point Analysis

Figure 10: Surface plot visualisation of the MOSUM detectors computed with a range of bandwidths for a realisation of the mix signal from Section 3.1.

a toy example for the single-bandwidth MOSUM procedure from Section 2.2 in Section 4.1, followed by an application of the multiscale algorithms from Section 2.6 to change point reference signals from the literature in Section 4.2 and Section 4.3. In addition, the R code used to generate Figures 1–10 is provided as supplementary online material to this paper for further examples.

4.1. MOSUM procedure with a single bandwidth

In this first example, we apply the MOSUM procedure with a single, asymmetric bandwidth $G = (40, 60)$ to a time series of length $n = 800$ with independent Gaussian innovations and two changes in the mean at the time points 200, 600 of respective height 2 and $-1.5$ co-occurring with changes in variance at the same time. The variances of the stationary segments are set to be 1, 0.8 and 0.5. We adopt the local variance estimator (11), which is well suited for scenarios with changes in both mean and variance. By default, the $\eta$-criterion (15) with $\eta = 0.4$ is used for change point estimation in conjunction with significance level $\alpha = 0.05$.

```R
R> x1 <- testData(lengths = c(200, 400, 200), means=c(0, 2, 1), + sds = sqrt(c(1, .8, .5)), seed = 111)
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:
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 \texttt{mix} from Section 3.1. The test signal is particularly well-suited for a multiscale procedure due to a mix of different types of mean changes, from large jumps over short stretches to small jumps surrounded by longer stretches of stationarity (see Figure 7). To elaborate, we consider a dense bandwidth grid \( G = \{10, 11, \ldots, 40\} \). To account for a possible dependence structure in the data, we use a slightly increased threshold \( C_{\eta,G}(\alpha) \cdot \log(n/G)^{0.1} \) (as discussed in Section 2.3). Furthermore, we use the default choice of \( \eta = 0.4 \) to obtain the candidate set of change point estimates:
mosum: A Package for Moving Sums in Change Point Analysis

Note that the use of $G = \{10, 11, \ldots, 40\}$ generates a warning message due to the smallest bandwidth being too small compared to $n = 560$. 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)`:

<table>
<thead>
<tr>
<th>cpts</th>
<th>G.left</th>
<th>G.right</th>
<th>p.value</th>
<th>jump</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>10</td>
<td>8.40e-06</td>
<td>3.304</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>10</td>
<td>1.98e-06</td>
<td>3.531</td>
</tr>
<tr>
<td>3</td>
<td>41</td>
<td>10</td>
<td>3.31e-12</td>
<td>5.628</td>
</tr>
<tr>
<td>4</td>
<td>60</td>
<td>10</td>
<td>8.73e-06</td>
<td>3.298</td>
</tr>
<tr>
<td>5</td>
<td>89</td>
<td>10</td>
<td>4.09e-04</td>
<td>2.691</td>
</tr>
<tr>
<td>6</td>
<td>120</td>
<td>10</td>
<td>5.22e-04</td>
<td>2.653</td>
</tr>
<tr>
<td>7</td>
<td>156</td>
<td>10</td>
<td>2.20e-03</td>
<td>2.426</td>
</tr>
<tr>
<td>8</td>
<td>200</td>
<td>10</td>
<td>3.57e-03</td>
<td>2.349</td>
</tr>
<tr>
<td>9</td>
<td>250</td>
<td>10</td>
<td>6.03e-03</td>
<td>2.267</td>
</tr>
<tr>
<td>10</td>
<td>302</td>
<td>16</td>
<td>6.90e-03</td>
<td>1.756</td>
</tr>
<tr>
<td>12</td>
<td>366</td>
<td>40</td>
<td>1.90e-02</td>
<td>0.997</td>
</tr>
<tr>
<td>11</td>
<td>421</td>
<td>33</td>
<td>1.53e-02</td>
<td>1.126</td>
</tr>
</tbody>
</table>

While the first nine change points have been detected by the bandwidths $G = 10$, a larger bandwidth was required to detect the change points estimated at positions 250, 302, 366 and 421, see also the surface plot Figure 10 in Section 3.6.

4.3. Multiscale MOSUM procedure with localised pruning

We apply Algorithm 2 to the piecewise stationary time series `blocks` (see Section 3.1), a well-known test signal widely analysed in the literature. 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 $\alpha = 0.4$, since merging is performed with the aid of SC:

```R
R> x3 <- testData(model = "blocks", seed = 1234)
R> mlp <- multiscale.localPrune(x3, alpha = .05, pen.exp = 1.01)
R> print(mlp$cpts)
```

```
[1] 204 266 307 464 511 819 903 1329 1556 1593 1656
```

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

```R
R> print(mlp$pooled.cpts)
```
5. Conclusion and outlook

In this paper, we presented the R package `mosum` (Meier et al. 2018). Implementations of both single bandwidth and multiscale MOSUM procedures for multiple change point estimation have been presented, as well as tools for visualisation and data generation.

The methodology 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 e.g., the median. Secondly, multivariate data analysis could be incorporated. Finally, different types of changes can be considered, 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 of this 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 include important extensions in future revisions of the `mosum` package.

Acknowledgements

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 for pointing out a mistake in the code and Rebecca Killick for several constructive suggestions for improvement.

References


A. Localised exhaustive search algorithm

In this section, we discuss an efficient implementation of a search algorithm for identifying a subset \( \mathcal{A} \subset \mathcal{D} \) of a set of conflicting change point estimates fulfilling the criteria (I)–(III) from Section 2.6 (see Line 9 of Algorithm 2). For the sake of completeness and readability, we present a high-level description thereof in Algorithm 3.

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.
Algorithm 3: Local exhaustive SC search

**input**: Current candidates \( \mathcal{C} \), locally conflicting candidates \( \mathcal{D} \subset \mathcal{C} \), interval of consideration \([s, e]\)

1. Let \( M \leftarrow 2^{\lvert \mathcal{D} \rvert} \) and denote by \( \mathcal{D}_1, \ldots, \mathcal{D}_M \) all \( M \) subsets of \( \mathcal{D} \) (including \( \emptyset \) and \( \mathcal{D} \));

2. Initialize final combinations \( \mathcal{F} \leftarrow \emptyset \) and truncationFlag\((i)\) \( \leftarrow 0 \) for \( i = 1, \ldots, M \);

/* Step 1: Process subsets in descending size */

3. for \( l = \lvert \mathcal{D} \rvert, \ldots, 1 \) do

   /* Step 1.1: Truncate the search space */

   4. for \( \mathcal{D}_i \) with \( \lvert \mathcal{D}_i \rvert = l \) and truncationFlag\((i)\) \( = 1 \) do

      5. for \( \mathcal{D}_j \) with \( \lvert \mathcal{D}_j \rvert = l - 1 \) and \( \mathcal{D}_j \subset \mathcal{D}_i \) do

         truncationFlag\((j)\) \( \leftarrow 1 \);

   end

   /* Step 1.2: Remove one candidate at a time */

   7. for \( \mathcal{D}_i \) with \( \lvert \mathcal{D}_i \rvert = l \) and truncationFlag\((i)\) \( = 0 \) do

      8. Let \( \mathcal{I}_{l,l-1} \leftarrow \{ j : \lvert \mathcal{D}_j \rvert = l - 1 \) and \( \mathcal{D}_j \subset \mathcal{D}_i \) and truncationFlag\((j)\) \( = 0 \} \);

      9. for \( j \in \mathcal{I}_{l,l-1} \) do

         10. if SC\((\mathcal{D}_i|\mathcal{C}, [s, e]) < \text{SC}(\mathcal{D}_j|\mathcal{C}, [s, e])\) then

             truncationFlag\((j)\) \( = 1 \);

      end

   end

/* Step 2: Select final combination minimizing SC */

15. if \( \mathcal{F} \neq \emptyset \) then Letting \( \ell^* \leftarrow \min_{\mathcal{F}} \lvert \mathcal{D}_i \rvert \), select \( i^* \leftarrow \text{argmin}_{\mathcal{F}: \lvert \mathcal{D}_i \rvert = \ell^*} \text{SC}(\mathcal{D}_i|\mathcal{C}, [s, e]) \) and \( \mathcal{A} \leftarrow \mathcal{D}_{i^*} \);

16. else Set \( \mathcal{A} \leftarrow \emptyset \);

**output**: \( \mathcal{A} \)
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 (see 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. To elaborate, let \( C = \{\tilde{k}_1 < \ldots < \tilde{k}_m\} \) 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

\[
\sum_{t=\tilde{k}_{j+1}}^{\tilde{k}_{j+1}} X_t \quad \text{and} \quad \sum_{t=\tilde{k}_{j+1}}^{\tilde{k}_{j+1}} X_t^2
\]

for \( j = 1, \ldots, m \), as these sums are used repeatedly in the calculation of the residual sum of squares \( (18) \) that are the main ingredient in calculating the information criterion \( (17) \).

Another important optimisation is attributed to the usage of bit vectors for an implicit representation of all the subsets \( \mathcal{D}_1, \ldots, \mathcal{D}_M \) of \( \mathcal{D} = \{\tilde{k}_1, \ldots, \tilde{k}_{|D|}\} \), where \( M = 2^{|D|} \). The key idea is that every \( \mathcal{D}_i \subset \mathcal{D} \) can be represented as a vector \( \vec{\psi}_i = (\psi_{i,1}, \ldots, \psi_{i,|D|}) \) of binary variables \( \psi_{i,j} \) indicating whether \( \tilde{k}_j \) belongs to \( \mathcal{D}_i \) or not, i.e., \( \psi_{i,j} \in \{0,1\} \) and \( \psi_{i,j} = 1 \) if and only if \( \tilde{k}_j \in \mathcal{D}_i \), for \( 1 \leq i \leq M \) and \( 1 \leq j \leq |D| \). On the other hand, every binary vector \( \vec{\psi}_i \) has a canonical representation as a nonnegative integer \( \kappa_i = \sum_{j=1}^{|D|} \psi_{i,j} 2^{j-1} \in \{0, \ldots, M-1\} \). 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 \( |D| = 4 \). 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 \( l \) non-zero entries. Such a loop can be realised in C++ as follows (see Section 11.1.1 in Stroustrup (2000) for a comprehensive overview on bitwise/logical operators in C++):

```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 $D_i$) over all subsets $D_j \subset D_i$ with $|D_j| = |D_i| - 1$ in Step 1.1 and Step 1.2 of Algorithm 3 translate into a loop over all $\kappa_j$ with $\psi_j$ having exactly $|D_i| - 1$ non-zero entries and $\psi_j \preceq \psi_i$ holding component wise among the binary vectors. A possible realisation in C++ is as follows:

```cpp
// 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
}
```

Due to the exponential time and memory consumption, we restrict the candidate set size to $|D| \leq 24$ in our implementation of Algorithm 3. If a candidate set $D$ exceeding this size is proposed in Step 2.1 of Algorithm 2, we denote the corresponding tuple $(\hat{k}, \hat{G})$ (from Line 4 in Algorithm 2) as infeasible. If an infeasible tuple occurs, we swap $\hat{k}$ with the next feasible candidate from $D$ or, if none of $D$ is feasible, with the next feasible candidate from $P$ for future consideration. In many cases, this changed order of processing is already sufficient to ensure that the size of neighbours of $\hat{k}$ will shrink at some point to a feasible size (because some of its conflicting change-point estimators will have been merged or removed due to the processing of another overlapping candidate set). If, however, all remaining tuples in $P$ are infeasible, we use a manual thinning step, in which successively change point candidates are removed from $D$ until it reaches a feasible size. More precisely, the candidates of $D$ are processed one by one in decreasing order of mutual distance (the minimal distance from the change locations to any other remaining candidate in $D$) and removed, until $|D| = 24$. 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. 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 independent and identically distributed with mean $\mu$ and finite variance $\sigma^2$. Assume furthermore that $E[|X_1|^{2+\Delta} < \infty$ holds for some $\Delta > 0$. Consider a pair of bandwidths $G = (G_l, G_r)$ with $G_{\min} := \min(G_l, G_r)$ and $G_{\max} := \max(G_l, G_r)$ depending on $n$, such that $K_n := G_{\min}/G_{\max} \rightarrow K > 0$ as well as

$$\frac{G_{\min}}{n^\theta} \rightarrow 0 \quad \text{and} \quad \frac{n}{G_{\min}} \rightarrow \infty$$
as \( n \to \infty \) is fulfilled for some \( \rho > \frac{2}{\pi + \sqrt{2}} \). Let

\[
    T_G := \max_{G_1 \leq k \leq n - G_r} \frac{|T_G(k)|}{\hat{\sigma}_k}
\]

with the MOSUM detector \( T_G(k) \) as defined in (12), and a local variance estimator \( \hat{\sigma}_k^2 \) fulfilling the following convergence assumption:

\[
    \max_{G_1 \leq k \leq n - G_r} |\hat{\sigma}_k^2 - \sigma^2| = o_p \left( \left( \log(n/G_{\min}) \right)^{-1} \right).
\]  

(21)

Then, the following distributional convergence holds:

\[
    a_G T_G - b_G \Rightarrow \Gamma_2,
\]

where \( \Gamma_2 \) is a Gumbel-distributed random variable with its cumulative distribution function

\[
    P(\Gamma_2 \leq z) = \exp\left(-2 \exp(-z)\right) \text{ for } z \in \mathbb{R},
\]

and the sequences \( a_G \) and \( b_G \) are given as

\[
    a_G = \sqrt{2 \log \left( \frac{n}{G_{\min}} \right)}, \quad b_G = 2 \log \left( \frac{n}{G_{\min}} \right) + \frac{1}{2} \log \log \left( \frac{n}{G_{\min}} \right) + \log \left( K_n^2 + K_n + 1 \right) - \frac{1}{2} \log \pi.
\]

Condition (21) 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 \( \hat{\sigma}_k^2 \) fulfilling (21) 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 \( G_{\min} = G_1 \), because of the following distributional equality:

\[
    T_{(G_1, G_1)}(n - k + 1), \quad k = G_1, \ldots, n - G_r.
\]

From \( K_n = G_l/G_r \), we get the representation

\[
    T_G(k) = \frac{1}{\sqrt{(K_n + 1)G_1}} \left( K_n \sum_{j=k+1}^{k+G_1} X_j - \sum_{j=k-G_1+1}^{k} X_j \right).
\]

By the same arguments as in the proof of Theorem 2.1 in Eichinger and Kirch (2018), the partial sums of length \( G_1 \) resp. \( G_r \) 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:

\[
    \sup_{t \in [1, n/(G_1 - 1/K_n)]} \left| \frac{1}{\sqrt{G_1}} \sum_{j=G_1}^{G_1 + t} X_j - (W_n(t) - W_n(t - 1)) \right| = o_p \left( (\log(n/G_1))^{-1/2} \right)
\]

and

\[
    \sup_{t \in [1, n/(G_1 - 1/K_n)]} \left| \frac{1}{\sqrt{G_1}} \sum_{j=G_1+1}^{G_1+G_r} X_j - (W_n(t + 1/K_n) - W_n(t)) \right| = o_p \left( (\log(n/G_1))^{-1/2} \right).
\]
Thus, with
\[ Y_n(t) := \frac{1}{\sqrt{K_n} + 1} \left( K_n (W(t + 1 + 1/K_n) - W(t + 1)) - (W(t + 1) - W(t)) \right) \]
for a standard Wiener process \((W(t))_{t \geq 0}\), it suffices to show for every \(z\)
\[ P \left( \sup_{t \in [0,n/G]} |Y_n(t)| \leq \frac{z + b_G}{a_G} \right) \longrightarrow \exp(-2\exp(-z)). \] (22)

Note that \(Y_n(t)\) is a stationary centered Gaussian process with variance 1. The autocovariance/autocorrelation function \(\gamma_{Y_n}(t) = E[Y_n(t)Y_n(0)]\) 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 ±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 \(T = n/G, \alpha = 1\) and \(u = (z + b_G)/a_G\). Some calculations show that \(\tau = \exp(-z)\), proving (22).

**Affiliation:**

Alexander Meier*, Claudia Kirch*† *(corresponding author)*
* Institute for Mathematical Stochastics (IMST), Department of Mathematics
† Center for Behavioral Brain Science (CBBS)
Otto von Guericke University Magdeburg
39106 Magdeburg, Germany
E-mail: alexander.meier@ovgu.de, claudia.kirch@ovgu.de

Haeran Cho
School of Mathematics
University of Bristol
Bristol BS8 1TW, UK
E-mail: haeran.cho@bristol.ac.uk

---

**Journal of Statistical Software**
published by the Foundation for Open Access Statistics
http://www.jstatsoft.org/

Submitted: yyyy-mm-dd
doi:10.18637/jss.v000.i00

Accepted: yyyy-mm-dd