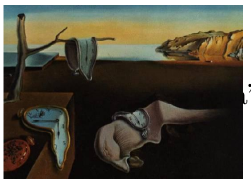
”.2

Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. http://www.jstatsoft.org/

SNSeg: An R Package for Change-Point Analyses and Time Series Segmentation via Self-Normalization

Shubo Sun
University of Miami

Zifeng Zhao
University of Notre Dame

Feiyu Jiang
Fudan University

Xiaofeng Shao
University of Illinois

Abstract

The R package **SNSeg** provides functions to perform change-point segmentation and estimation for univariate and multivariate time series through Self-Normalization (SN). While many existing nonparametric approaches support changes in only one type of parameters (e.g. mean or variance) of a univariate time series and are tuning-parameter-dependent, SN-based algorithms are offered with more convinence in any user-chosen number or smooth parameter, robust to temporal dependence, and can implement change-point estimation based on changes in more than one type of parameters (e.g. mean, variance and acf) of both univariate and multivariate time series. The nested local-window segmentation algorithm (SNCP) is applied to conduct multiple change-point estimation, and an extension of SNCP, named SNHD, is designed for change-point estimation in high-dimensional time series. Graphics for time series and SN test statistics segmentation plots are available with certain options. Detailed examples are given in simulations of different MAR processes.

Keywords: undecided.

1. Introduction

In recent years, change-point analysis has become a research area in statistics and other related fields, including finance, economics, signal processing, and medical research to study diseases such as COVID-19. (Here needs a large number of citations and some other complement stuff) There exist a number of R packages available for change-point analysis. The package **bcp** (Erdman and Emerson 2007) implements Bayesian methods proposed by Barry and Hartigan (1993); the package **strucchange** (Zeileis, Leisch, Hornik, and Kleiber 2002) detects structural changes in linear regression models using algorithms in Zeileis, Kleiber, Krämer, and Hornik (2003); Zeileis (2006); the package **changepoint** (Killick and Eckley 2014) applies the pruned exact linear time (PELT) algorithm described in Killick, Fearnhead, and Eckley (2012); the package **cpm** (Ross 2015) utilizes methods for sequential change-points detection which are first introduced by Hawkins, Qiu, and Kang (2003); Hawkins and Zamba (2005) with multiple extensions (see Ross (2014) for a summary of the extensions); the package **ecp** (James and Matteson 2014) implements the nonparametric change-point methods for multivariate time series from Matteson and James (2014); the package **mosum** (Meier, Kirch, and Cho 2021) performs moving sum (MOSUM) procedure from Eichinger and Kirch (2018) and the package (Wang and Zou 2022) focuses on a variety of parametric generalized linear models with the sample-splitted strategy from Zou, Wang, and Li (2020).

In this paper, we concern the R package **SNSeg** to apply the nonparametric Self-normalization (SN, hereafter) framework for multiple change-points estimation based on the segmentation algorithm proposed by Zhao, Jiang, and Shao (2021). SN was first introduced by Shao (2010) for confidence interval construction in time series. It has been extended to retrospective change-point testing by Shao and Zhang (2010), Hoga (2018), Betken and Wendler (2018), Lavitas and Zhang (2018), and Dette and Gösmann (2020), and to sequential change-point monitoing by Dette, Kokot, and Volgushev (2020) and Chan, Ng, and Yau (2021).

The SN-based segmentation procedures can be seen as novelty to previous approaches on segmenting time series. It proposes a nested local-window segmentation algorithm with SN tests for multiple change-points estimation. Zhao *et al.* (2021) shows that this method is fully nonparametric, robust to temporal dependence, enjoys effortless tuning, and works universally for various parameters of interest for a multivariate time series. Inaddition, new theoretical arguments are developed to establish the consistency and convergence rate of the proposed change-point estimation procedure, which seems to be the first in the SN literature. Furthermore, it does not rely on any distributional assumption of the underlying data. Also, it enables users for free choices of the paremeter of interests for SN tests and the corresponding trimming parameter for the local-window algorithm and thus is very flexible.

The rest of the paper is organized as follows. We first provide the SN-based statistic and procedures for change-point estimation in Section 2. Specifically, we discuss the nested local-window segmentation algorithm employing a broad range of trimming parameters, as well as some theoretical results of threshold of the SN tests for multiple change-points estimation. Building upon the SN change-point detection methods, we introduce the package **SNSeg** in Section 3 to demonstrate the methods for change-point estimation on univariate and multivariate time series based on change in a broad class of parameters, along with visualization with some examples. More detailed usage examples with applications to simulated data are provided in Section **??**. Finally, a summary of the package and contributions are presented in Section 5.

2. SNCP Change-Point Estimation Framework

2.1. Introduction to CUSUM-based SN Framework

Suppose our interest is to test for a change-point in the mean of a univariate time series

$\{X_t\}_{t=1}^n$, which is

$$H_0 : E(X_1) = \cdots = E(X_n) = \mu \tag{1}$$

versus

$$H_a : E(X_1) = \cdots = E(X_k) \neq E(X_{k+1}) = \cdots = E(X_n) \tag{2}$$

where $k = 2, \cdots, n-1$ is unknown. A commonly used test statistic is based on the CUSUM process, which is

$$D_n(\lfloor nr \rfloor) = \frac{1}{\sqrt{n}} \sum_{t=1}^{\lfloor nr \rfloor} (X_t - E(X_n)), \qquad r \in [0,1] \tag{3}$$

Under appropriate moment and weak dependence assumptions on X_t (see Phillips (1987)) and under the null hypothesis, $D_n(\lfloor nr \rfloor) \Rightarrow \sigma(B(r) - rB(1))$, where $B(r)$ is a standard one-dimensional Brownian motion and $\sigma^2 = \lim_{n\to\infty} Var(E(X_n)) = \sum_{k\in Z} Cov(X_0, X_k)$ (see Shao and Zhang (2010)) is the long-run variance.

The Kolmogorov–Smirnov test statistic $K_n = \sup_{r\in[0,1]} |D_n(\lfloor nr \rfloor)/\hat{\sigma^2}| = \sup_{k=1,\cdots,n-1} |D_n(k)/\hat{\sigma}^2|$ can be used for change-point detection using the CUSUM process, where $\hat{\sigma}^2$ is a consistent estimator of σ^2 that

$$\hat{\sigma}^2 = \sum_{k=-l_n}^{l_n} \hat{\gamma}(k) K(k/l_n) \tag{4}$$

where $\hat{\gamma}(j) = \frac{1}{n}\sum_{t=1}^{n-|j|}(X_t - \bar{X}_n)(X_{t+|j|} - \bar{X}_n)$ is the sample ACF at lag j with referring to the sample mean, and $K(\cdot)$ is a kernel function involving the choice of a bandwidth parameter l_n. The selection of l_n may yield serious size and power problems. See Shao and Zhang (2010)) for more discussion.

To bypass this, Shao and Zhang (2010) combined the SN theory (Lobato (2001); Shao (2010)) and the CUSUM process. Let $S_{t_1,t_2} = \sum_{j=t_1}^{t_2} X_j$ for $t_1 < t_2$. The CUSUM-based SN statistic was defined as

$$T_n = \sup_{k=1,\cdots,n-1} D_n(k)' V_n^{-1}(k) D_n(k) \tag{5}$$

where $V_n(k) = \frac{1}{n^2}\left[\sum_{t=1}^{k}(S_{1,t} - (t/k)S_{1,k})^2 + \sum_{t=k+1}^{n}(S_{t,n} - \frac{n-t+1}{n-k}S_{k+1,n})^2\right]$. Note that the normalization factor $V_n(k)$ is proportional to the problematic term σ^2 and σ^2 gets cancelled compared to the Kolmogorov–Smirnov statistic K_n. Meanwhile, the partial sum terms within $V_n(k)$ are invariant to $E(X_{k+1}) - E(X_k)$. As $E(X_{k+1}) - E(X_k)$ increases, both $D_n(k)$ and $T_n(k)$ also increase, and hence leads to the detection of the true change-point at k. SN claims an estimated change-point $\hat{k}$ as

$$\hat{k} = \arg\max_{k=1,\cdots,n} T_n. \tag{6}$$

The package **SNSeg** performs the SNCP algorithms which are an extension to the CUSUM-based SN framework. The following sections discuss the single and multiple change-points estimation using the SNCP framework.

2.2. Single Change-Point Estimation

We start with a single change-point estimation in a general parameter $\theta = \theta(F_t)$ for a univariate time series $\{Y_t\}_{t=1}^n$, where F_t denotes the CDF of Y_t and $\theta(\cdot)$ denotes a general functional. Our interest is to recover all unknown change-points at any t. The global SNCP test statistic SN_n is defined as

$$SN_n = \max_{k=1,\cdots,n-1} T_n(k), \qquad T_n(k) = D_n(k)^2/V_n(k), \tag{7}$$

where

$$D_n(k) = \frac{k(n-k)}{n^{3/2}}(\hat{\theta}_{1,k} - \hat{\theta}_{k+1,n}) \tag{8}$$

is denoted as the contrast statistic at lag k and

$$V_n(k) = \sum_{i=1}^{k} \frac{i^2(k-i)^2}{n^2k^2}(\hat{\theta}_{1,i} - \hat{\theta}_{i+1,k})^2 + \sum_{i=k+1}^{n} \frac{(n-i+1)^2(i-k-1)^2}{n^2(n-k)^2}(\hat{\theta}_{i,n} - \hat{\theta}_{k+1,i-1}). \tag{9}$$

is defined as the normalization factor at lag k. Here $\hat{\theta}_{a,b}$ is the nonparametric estimator of $\theta(\hat{F}_{a,b})$ for any $1 < a < b < n$ where $\hat{F}_{a,b}$ is the empirical distribution of $\{Y_t\}_{t=a}^b$.

If $\theta(\cdot)$ is a mean functional, SN_n reduces to the CUSUM-based SN test statistic in Section 2.1. If it is not a mean functional (e.g., variance, ACF, etc.), SN_n is not equivalent to the CUSUM-based test statistic. We refer to Lavitas and Zhang (2018) and Zhao *et al.* (2021) for more details. $T_n(k)$ is referred as the SNCP test statistic at time point k. For a pre-specified threshold K_n, we declare no change point when $SN_n < K_n$. Given that SN_n exceeds the threshold K_n, we estimate the single change-point via

$$\hat{k} = \arg\max_{k=1,\cdots,n-1} T_n(k). \tag{10}$$

Intuitively, we estimate the change-point as the location of the maximum test statistic among all observations if it exceeds the SN threshold. Note that this SNCP procedure is a general framework since it can be implemented with any functional $\theta(\cdot)$ with a nonparametric estimator based on the empirical distribution. Assumptions and theoretical justifications can be found in Zhao *et al.* (2021).

2.3. Multiple Change-Point Estimation

We furtherly extend the SNCP test to multiple change-point estimation. To proceed, we assume $m_0 \geq 0$ unknown number of change points with $0 < k_1 < \cdots < k_{m_0} < n$ that partition Y_t into $m_0 + 1$ stationary segments with CDF $F^{(i)}$. Define $k_0 = 0$ and $k_{m_0+1} = n$, and then the i_{th} segment, denoted as

$$Y_t = Y_t^{(i)}, k_{i-1} + 1 \leq t \leq k_i, \text{ for } i = 1, \cdots, m_0 + 1, \tag{11}$$

contains observations between two change points $t = k_{i-1}+1$ and $t = k_i$, which share common feature characterized by the functional θ_i, for $i = 1, \cdots, m_0 + 1$.

To recover the unknown locations of the change points, we start by introducing some notations. Similar to the single change-point estimation framework, for $1 \leq t_1 < k < t_2 \leq n$, we define

$$T_n(t_1, k, t_2) = D_n(t_1, k, t_2)^2 / V_n(t_1, k, t_2), \tag{12}$$

where $D_n(t_1, k, t_2) = \frac{(k-t_1+1)(t_2-k)}{(t_2-t_1+1)^{3/2}}(\hat{\theta}_{t_1,k} - \hat{\theta}_{k+1,t_2})$, $V_n(t_1, k, t_2) = L_n(t_1, k, t_2) + R_n(t_1, k, t_2)$ and

$$L_n(t_1, k, t_2) = \sum_{i=t_1}^{k} \frac{(i-t_1+1)^2(k-i)^2}{(t_2-t_1+1)^2(k-t_1+1)^2}(\hat{\theta}_{t_1,i} - \hat{\theta}_{i+1,k}), \tag{13}$$

$$R_n(t_1, k, t_2) = \sum_{i=k+1}^{t_2} \frac{(t_2-i+1)^2(i-1-k)^2}{(t_2-t_1+1)^2(t_2-k)^2}(\hat{\theta}_{i,t_2} - \hat{\theta}_{k+1,i-1}). \tag{14}$$

where $\hat{\theta}_{a,b} = \theta(\hat{F}_{a,b})$ and $\hat{F}_{a,b}$ measures the empirical distribution of $\{Y_t\}_{t=a}^{b}$. Here $T_n(t_1, k, t_2)$ represents the SN test statistic defined on the subsample $\{Y_t\}_{t=t_1}^{t_2}$. In other words, we find the SNCP test statistic at each time point within the interval $[t_1, t_2]$. Set $t_1 = 1$ and $t_2 = n$, $T_n(t_1, k, t_2) = T_n(1, k, n)$ reduces to the single global test change-point estimation framework in Section 2.2.

Then we combine the SN framework with a nested local-window segmentation algorithm proposed by Zhao *et al.* (2021). For each k, instead of one global test for $T_n(1, k, n)$ using all observations, we compute a maximal SNCP test statistic based on a collection of nested windows covering k. Specifically, we fix a small trimming papameter $\epsilon \in (0, 1/2)$ and define the window size $h = \lfloor n\epsilon \rfloor$. For each $k = h, \cdots, n-h$, we define the nested local-window set $H_{1:n}(k)$ as

$$H_{1:n}(k) = (t_1, t_2) | t_1 = k - j_1 h + 1, j_1 = 1, \cdots, \lfloor k/h \rfloor; t_2 = k + j_2 h, j_2 = 1, \cdots, \lfloor (n-k)/h \rfloor. \tag{15}$$

Note that for $k < h$ and $k > n - h$, we have $H_{1:n}(k) = \emptyset$.

For each $k = 1, \cdots, n$, based on its nested local-window set $H_{1:n}(k)$, we define a maximal SN test statistic such that

$$T_{1,n}(k) = \max_{(t_1,t_2) \in H_{1:n}(k)} T_n(t_1, k, t_2), \tag{16}$$

where we set $\max_{(t_1,t_2)\in\emptyset} T_n(t_1, k, t_2) = 0$. Intuitively, the nested local-window framework ensures that there exist some local window sets (t_1, t_2) regarding each k as the only change-point. This means each k will have several SNCP test statistics based on each of its local window sets. The maximal SN test statistic $T_{1,n}(k)$ for $k = 1, \cdots, n$ is then defined as

$$T_{1,n}(k) = \max_{1 \leq t_1 \leq k \leq t_2 \leq n} T_n(t_1, k, t_2). \tag{17}$$

Starting from the full sample $\{Y_t\}_{t=1}^{n}$, we compute the test statistic $T_{1,n}(k)$ for each k. With a properly-chosen threshold K_n, a change-point will be detected via

$$\hat{k} = \arg \max_{k=1,\cdots,n} T_{1,n}(k) \tag{18}$$

if this maximal test statistic $T_{1,n}(k)$ is above K_n; otherwise no change point is detected. We recursively perform SNCP on the subsamples $\{Y_t\}_{t=1}^{\hat{k}}$ and $\{Y_t\}_{t=\hat{k}+1}^{n}$ until all change points are declared.

The SNCP multiple change-points detection procedures can also be applied to multivariate time series, with the mean or covariance matrix as potential functionals. Different from univariate time series cases, the functionals for multivariate time series are vector-valued. Suppose a functional for multivariate time series is defined as $\theta_{a,b}$, then the subsample SNCP statistic is defined as

$$T_n(t_1,k,t_2) = D_n(t_1,k,t_2)^T V_n(t_1,k,t_2)^{-1} D_n(t_1,k,t_2) \tag{19}$$

where we modify $D_n(t_1,k,t_2)$ and the self-normalizer $V_n(t_1,k,t_2)$ of univariate SNCP cases in Section 2.2 such that

$$D_n(t_1,k,t_2) = \frac{(k-t_1+1)(t_2-k)}{(t_2-t_1+1)^{3/2}}(\hat{\theta}_{t_1,k} - \hat{\theta}_{k+1,t_2}), \quad V_n(t_1,k,t_2) = L_n(t_1,k,t_2) + R_n(t_1,k,t_2),$$

$$L_n(t_1,k,t_2) = \sum_{i=t_1}^{k} \frac{(i-t_1+1)^2(k-i)^2}{(t_2-t_1+1)^2(k-t_1+1)^2}(\hat{\theta}_{t_1,i} - \hat{\theta}_{i+1,k})(\hat{\theta}_{t_1,i} - \hat{\theta}_{i+1,k})^T,$$

$$R_n(t_1,k,t_2) = \sum_{i=k+1}^{t_2} \frac{(t_2-i+1)^2(i-1-k)^2}{(t_2-t_1+1)^2(t_2-k)^2}(\hat{\theta_{i,t_2}} - \hat{\theta}_{k+1,i-1})(\hat{\theta_{i,t_2}} - \hat{\theta}_{k+1,i-1})^T.$$

Similar to univariate time series cases, a maximal test statistic $T_{1,n}(k)$ is denoted as $T_{1,n}(k) = \max_{1\le t_1\le k\le t_2\le n} T_n(t_1,k,t_2)$ for $k = 1,\cdots,n$ and SNCP detects a change-point through

$$\hat{k} = \arg\max_{k=1,\cdots,n} T_{1,n}(k)$$

if this maximal $T_{1,n}(k)$ surpasses the pre-specified threshold K_n.

Denote $W_{s,e} = \{(t_1,t_2)|s \le t_1 \le t_2 \le e\}$ and $H_{s:e}(k) = H_{1:n}(k)\bigcap W_{s,e}$, which is the nested window set of k on the subsample $\{Y_t\}_{t=s}^{e}$. Define the subsample maximal test statistic as $T_{s,e}(k) = \max_{(t_1,t_2)\in H_{s:e}(k)} T_n(t_1,k,t_2)$. Algorithm 1 states the formal description of SNCP. The reason to apply the nested local-window algorithms is related to self-normalizer $V_n(t_1,k,t_2)$ under the multiple change-point scenario. The intuition is as follows. Suppose k is a change-point and $\{Y_t\}_{t=1}^{n}$ has other change-points besides k. $V_n(t_1,k,t_2)$ may observe severe inflation since $L_n(t_1,k,t_2)$ and $R_n(t_1,k,t_2)$ are based on contrast statistics and their value can significantly inflate because of the change-points apart from k. By applying the nested local-window algorithms, with a relatively small trimming parameter ϵ, there exists at least one nested window $(\tilde{t_1},\tilde{t_2}) \in H_{1:n}(k)$ that contains k as the only change-point for any change-point k. This avoids the inflation of $L_n(t_1,k,t_2)$ and $R_n(t_1,k,t_2)$. Therefore, the maximal statistic $T_n(t_1,k,t_2)$ remains effective thanks to $T_n(\tilde{t_1},k,\tilde{t_2})$.

2.4. Multiple Change-Point Estimation for High-Dimensional Data

Algorithm 1: SNCP procedures for multiple change-points detection

```
Input: Time Series $\{Y_t\}_{t=1}^n$, threshold $K_n$, window size $h = \lfloor n \rfloor$
Output: Estimated change-point set $\hat{\mathbf{k}} = (\hat{k}_1, \cdots, \hat{k}_{\hat{m}})$
Initialization: SNCP$(1, n, K_n, h)$, $\hat{k} = \emptyset$
Procedure: SNCP$(s, e, K_n, h)$ if $e - s + 1 < 2h$ then
  | Stop
else
  | $\hat{k} = \arg\max_{k=s,\cdots,e} T_{s,e}(k)$;
  | if $T_{s,e}(\hat{k}^*) \leq K_n$ then
  |   | Stop
  | else
  |   | $\hat{\mathbf{k}} = \hat{\mathbf{k}} \cup \hat{k}^*$;
  |   | SNCP$(s, \hat{k}^*, K_n, h)$; SNCP$(\hat{k}^* + 1, e, K_n, h)$;
  | end
end
```

In this section, we modify the SNCP framework in Section 2.3 to design a new algorithm called SNHD for multiple change-points estimation for high-dimensional independent data based on the change of a mean functional. Different from the SNCP subsample test statistic $T_n(t_1, k, t_2)$ used for a low-dimensional time series, SNHD is designed based on the high-dimensional U-statistic proposed by Wang et al. (2019). Given a p-dimensional time series $\{Y_t \in R\}_{t=1}^n$, we define the subsample contrast statistic

$$D_n^*(t_1, k, t_2) = \sum_{\substack{t_1 \leq j_1, j_3 \leq k \\ j_1 \neq j_3}} \sum_{\substack{k+1 \leq j_2, j_4 \leq j_2 \\ j_2 \neq j_4}} (Y_{j_1} - Y_{j_2})^T (Y_{j_3} - Y_{j_4}). \tag{20}$$

Note that $D_n^*(t_1, k, t_2)$ is a two-sample U-Statistic calculating the difference between the mean of $\{Y_t\}_{t=t_1}^k$ and $\{Y_t\}_{t=k+1}^{t_2}$. We define the normalization factor

$$V_n^*(t_1, k, t_2) = \frac{1}{n} \Big[\sum_{t=t_1+1}^{k-2} D_n^*(t_1, t, k)^2 + \sum_{k+2}^{t_2-2} D_n^*(k+1, t, t_2)^2 \Big]. \tag{21}$$

The subsample SNHD test statistic at time point k is defined as

$$T_n^*(t_1, k, t_2) = D_n^*(t_1, k, t_2)^2 / V_n^*(t_1, k, t_2) \tag{22}$$

and the maximal test statistic for k $T_{1,n}^*(k)$ is then defined as

$$T_{1,n}^*(k) = \max_{(t_1, t_2) \in H_{1:n}(k)} T_n^*(t_1, k, t_2). \tag{23}$$

With a pre-specified threshold K_n^*, a change-point is detected at $\hat{k} = \arg\max_{k=1,\cdots,n} T_{1,n}^*(k)$ if this maximal test statistic is above K_n^*. For multiple change-points estimation, SNHD proceeds recursively as SNCP in Section 2.3.

2.5. Theoretical Results of SN-based Test Threshold

For hypothesis testing, it is essential to study the theoretical properties of the threshold K_n. For multiple change-point estimation, we first assume a univariate time series case whose dimension is $d = 1$. For any $u \in (\epsilon, 1-\epsilon)$, where ϵ is a trimming parameter, we define the scaled limit of $H_{1:n}(k)$ by $H_\epsilon(u) = (u_1, u_2)|u_1 = u - j\epsilon, j = 1, \cdots, \lfloor u/\epsilon \rfloor;\ u_2 = u + j\epsilon, j = 1, \cdots, \lfloor (1-u)/\epsilon \rfloor$. We also define $\triangle(u_1, u, u_2) = B(u) - B(u_1) - \frac{u-u_1}{u_2-u_1}[B(u_2) - B(u_1)]$ where $B(\cdot)$ is a standard Brownian motion. Then under the no-change-point case, we have

$$\max_{k=1,\cdots,n} T_{1,n}(k) \xrightarrow{D} G_\epsilon = \sup_{u\in(\epsilon,1-\epsilon)} \max_{(u_1,u_2)\in H_\epsilon(u)} D(u_1, u, u_2)^2 / V(u_1, u, u_2), \tag{24}$$

where $D(u_1, u, u_2) = \frac{1}{\sqrt{u_2-u_1}} \triangle(u_1, u, u_2)$ and $V(u_1, u, u_2) = \frac{1}{(u_2-u_1)^2}\big(\int_{u_1}^{u} \triangle(u_1, s, u)^2 ds + \int_{u}^{u_2} \triangle(u, s, u_2)^2 ds\big)$.

This characterizes the asymptotic behavior of SNCP under no change-point environment and thus provides a natural choice of threshold K_n. For any given window size $h = \lfloor n\epsilon \rfloor$, G_ϵ is a pivotal distribution and its critical values can be obtained via simulation. This theorem can be extended to multivariate time series. See Zhao *et al.* (2021) and its supplement texts for details.

3. Introduction to the Package

In this section, we introduce the functions within the **SNSeg** package for multiple change-points detection, `SNSeg_Uni_single_para`, `SNSeg_Uni_multi_para`, `SNSeg_Multi` and `SNSeg_HD` as well as a function to plot the SN test statistic segmentation graph, `max_SNsweep`. The change-point estimation functions also provide options to plot the time series and observe the location of the change-points. We initially begin with a brief introduction on ways to generate multivariate time series served as examples for change-point detection in Section 3.1. In Section 3.3, we introduce critical value tables for SN tests under various scenarios, and a tool to compute the maximal test statistic at each k in the subsequent section 3.2. In Section 3.4, we bring in methods to perform SN-based tests on a univariate time series with the change in a single parameter. In Section 3.5, we describe the method developed for a univariate time series based on the change in multiple parameters simultaneously. and discuss SN-based methods for multivariate time series in Section 3.6 and 3.7. Tools for visualization of maximal test statistics is provided in Section 3.3 and time series under different cases are offered in Section 3.4, 3.5, 3.6 and 3.7.

3.1. Generating Multivariate Autoregressive Time Series

We describe and provide the functions available for generating multivariate autoregressive time series. The function `MAR` simulates a multivariate time series from a VAR(1) process under the no-change-point setting to test the change in a single mean or a single quantile.

```
MAR(n, reptime, rho)
```

It takes the following arguments:

- `n`: Length of the time series Y_t to be generated where $t = 1, 2, \cdots, n$
- `reptime`: Dimension of the time series to be generated, i.e., `reptime = 2` will generate a 2-d time series. Each individual time series within the multivariate process is independent of each other. A univariate time series can be given with `reptime = 1`.
- `rho`: Value of the coefficient for the generated AR(1) time series, i.e., `rho = 0.1` is equivalent to $\rho = 0.1$.

Note that the function `MAR` returns a matrix of which each row represents the time points and each column is a univariate time series.

As an example, we introduce a simulation setting from Zhao *et al.* (2021) using `MAR`, which generates a multivariate time series for change-point estimation purposes. It simulates a stationary d-dimensional stationary time series $\{X_t = (X_{t1}, \cdots, X_{td})\}_{t=1}^n$ from a VAR(1) process with $X_t = \rho \mathbf{I}_{dt-1} + \epsilon_t$ where ϵ_t are i.i.d.standard d-variate normal $N(0, \mathbf{I}_d)$ and $\mathbf{I}_d$ represents the d-dimensional identity matrix. We generate time series $\{Y_t\}_{t=1}^n$ with piecewise constant mean with the following case:

$$n = 2000, \quad \rho = -0.7 \quad , Y_t = \begin{cases} 0.4/\sqrt{d} + Y_t & t \in [1, 1000], [1501, 2000], \\ 0 + Y_t & t \in [1001, 1500]. \end{cases} \tag{25}$$

We conduct a realization of this with $d = 1$ and visualize it:

```
set.seed(7)
ts <- MAR(n = 2000, reptime = 2, rho = 0.7)
# create change points through the mean shift
# locations of proposed change points
cp_sets <- round(2000*c(0,cumsum(c(0.5,0.25)),1))
mean_shift <- c(0.4,0,0.4)
no_seg <- length(cp_sets)-1

for(index in 1:no_seg){
  tau1 <- cp_sets[index]+1
  tau2 <- cp_sets[index+1]
  ts[tau1:tau2,] <- ts[tau1:tau2,] + mean_shift[index]
}
# select a univariate time series out of the d-dimensional time series
ts <- ts[,2]
df <- cbind(1:length(ts), ts)
colnames(df)[1] <- 't'
df <- data.frame(df)
# visualization
ggplot(df, aes(x=t, y=ts)) + geom_line()
```

The result is plotted in Figure 1.

The function `MAR_Variance` simulates a multivariate time series for the change in variance or autocorrelation. This function generates times series based on simulation studies from Section S.2.5 of the supplementary materials in Zhao *et al.* (2021).

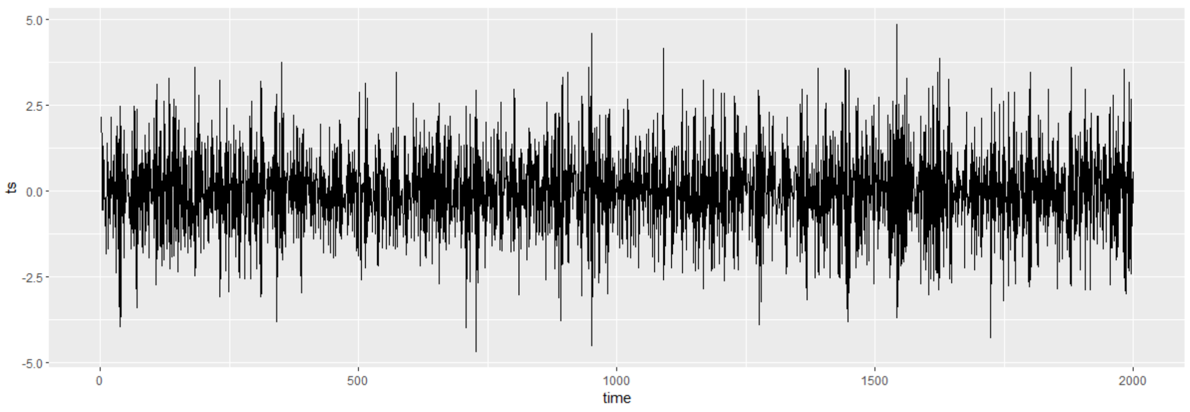

Figure 1: A realization of the time series model with the change in mean from simulation studies of Zhao *et al.* (2021).

```
MAR_Variance(reptime, type = 'V3')
```

It takes the following arguments:

- `reptime`: Dimension of the time series to be generated
- `type`: The type of time series to generated. Currently implemented types are:
  - `V1`: The "V1" type time series is an AR(1) process with moderate temporal dependence adapted from Cho and Fryzlewicz (2012).
  - `V2`: "V2" is an ARMA(1,1) process from Korkas and PryzlewiczV (2017).
  - `V3`: "V3" is an AR(2) process with strong positive dependence from Cho and Fryzlewicz (2012).
  - `A1`: "A1" is an AR(1) process taken from Cho and Fryzlewicz (2012).
  - `A2`: "A2" is an AR(1) process taken from Cho and Fryzlewicz (2012).
  - `A3`: "A3" is an ARMA(1,1) process adapted from Korkas and PryzlewiczV (2017). The default type is `V3`.

When called, `MAR_Variance` returns a matrix whose rows are time lags and columns are each of the univariate time series from the multivariate ones.

For the change in variance, we consider time series $\{Y_t\}_{t=1}^n$ with piecewise constant variance from the types `V1`, `V2` and `V3`. The detailed models (for each individual univariate time series) are:

$$\text{V1} : n = 1024, Y_t = \begin{cases} 0.5Y_{t-1} + \epsilon_t & t \in [1, 400], \\ 0.5Y_{t-1} + 2\epsilon_t & t \in [401, 750], \\ 0.5Y_{t-1} + \epsilon_t & t \in [751, 1024]. \end{cases} \tag{26}$$

$$\text{V2} : n = 1024, Y_t = \begin{cases} 0.7Y_{t-1} + \epsilon_t + 0.6\epsilon_{t-1} & t \in [1, 125], \\ 0.3Y_{t-1} + \epsilon_t + 0.3\epsilon_{t-1} & t \in [126, 532], \\ 0.9Y_{t-1} + \epsilon_t & t \in [533, 704], \\ 0.1Y_{t-1} + \epsilon_t + -0.5\epsilon_{t-1} & t \in [705, 1024]. \end{cases} \tag{27}$$

$$\text{V3}: n = 1024, Y_t = \begin{cases} 0.9Y_{t-1} + \epsilon_t & t \in [1, 512], \\ 1.69Y_{t-1} - 0.81Y_{t-2} + \epsilon_t & t \in [513, 768], \\ 1.32Y_{t-1} - 0.81Y_{t-2} + \epsilon_t & t \in [769, 1024]. \end{cases} \tag{28}$$

and ϵ_t is i.i.d. $N(0, 1)$.

For the change in autocorrelation, we consider time series $\{Y_t\}_{t=1}^n$ with piecewise constant autocorrelation from the types `A1`, `A2` and `A3`. The detailed models (for each individual univariate time series) are:

$$\text{A1}: n = 1024, Y_t = \begin{cases} 0.5Y_{t-1} + \epsilon_t & t \in [1, 400], \\ 0.9Y_{t-1} + \epsilon_t & t \in [401, 750], \\ 0.3Y_{t-1} + \epsilon_t & t \in [751, 1024]. \end{cases} \tag{29}$$

$$\text{A2}: n = 1024, Y_t = \begin{cases} 0.75Y_{t-1} + \epsilon_t & t \in [1, 50], \\ -0.5Y_{t-1} + \epsilon_t & t \in [51, 1024]. \end{cases} \tag{30}$$

$$\text{A3}: n = 1024, Y_t = \begin{cases} -0.9Y_{t-1} + \epsilon_t + 0.7\epsilon_{t-1} & t \in [1, 512], \\ 0.9Y_{t-1}\epsilon_t & t \in [513, 768], \\ \epsilon_t - 0.7\epsilon_{t-1} & t \in [769, 1024]. \end{cases} \tag{31}$$

and ϵ_t is i.i.d. $N(0, 1)$. As an illustration, we generate a ”V1” type time series and visualize it:

```r
set.seed(7)
ts <- MAR_Variance(2, "V1")
# draw a univariate time series out of the simulated 2-d time series
ts <- ts[,2]
df <- cbind(1:length(ts), ts)
colnames(df)[1] <- 't'
df <- data.frame(df)
# visualization
ggplot(df, aes(x=t, y=ts)) + geom_line()
```

The result is displayed in Figure 2.

The last function to generate time series is `MAR_MTS_Covariance`. This function simulates time series for the change in multiple parameters from a multi-dimensional VAR(1) process.

```
MAR_MTS_Covariance(n, reptime, rho_sets, cp_sets, sigma_cross)
```

Denote ρ be a vector of autocorrelations of the multivariate time series and Σ be a covariance matrix of this time series. The function `MAR_MTS_Covariance` takes the following arguments:

- `n`: Length of time series to be generated.
- `reptime`: Dimension of time series to be generated.

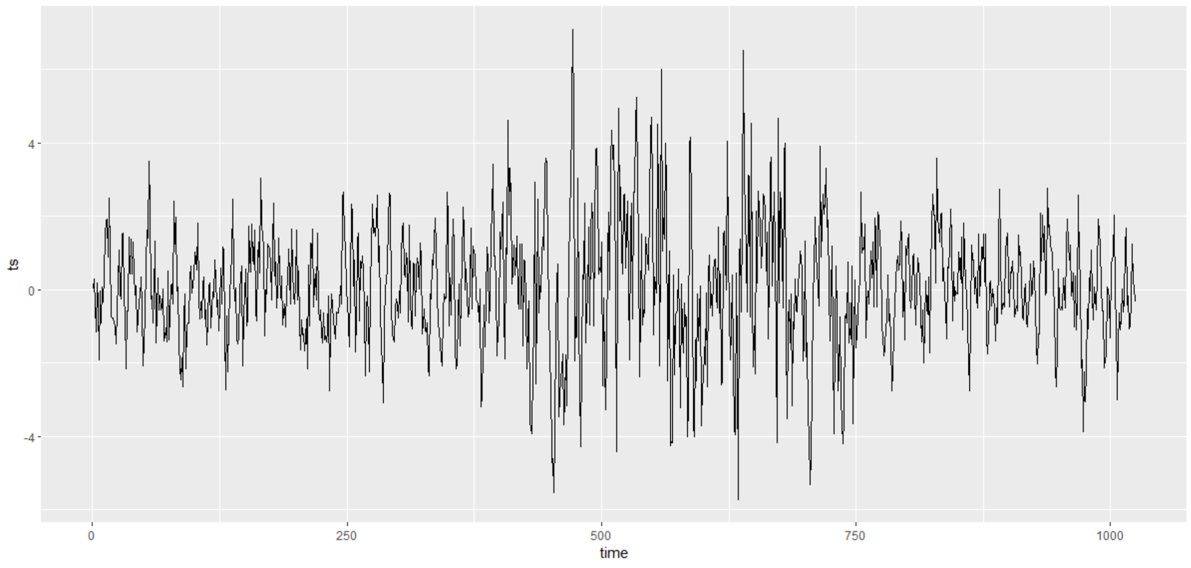

Figure 2: A realization of the "V1" type time series model with the change in variance from simulation studies of Zhao *et al.* (2021).

- `rho_sets`: ρ, a vector of autocorrelations.
- `cp_sets`: Locations of change points to be selected.
- `sigma_cross`: Covariance matrix Σ.

Note that `Mar_MTS_Covariance` returns a list of multivariate time series with the same length and dimension. Each multivariate time series is a matrix of which the columns are time lags. The argument `reptime` is the dimension of each multivariate time series. Also, the true dimension of all multivariate time series is not the value of `reptime`.

As an example, we use the simulation setting "C3" in supplementary textxs of Zhao *et al.* (2021) for the change in covariance matrices. We aim to detect the change in covariance matrices of a 4-dimensional time series $\{Y_t = (Y_{t1}, \cdots, Y_{t4})\}_{t=1}^n$ with $n = 1000$. The number of parameters in the covariance matrix is $d = (4 \times 5)/2 = 10$. Denote Σ_ρ as an exchangeable covariance matrix with unit variance and equal covariance ρ, we consider

$$Y_t = 0.3\mathbf{I}_4 Y_{t-1} + \epsilon_t, \epsilon_t \overset{\text{i.i.d.}}{\sim} N(0, \Sigma_{0.5}), t \in [1, 1000]. \tag{32}$$

Detailed descriptions of the simulation studies for `MAR_MTS_Covariance` can be found in Zhao *et al.* (2021). We visualize this by adopting the following code:

```
exchange_cor_matrix <- function(d, rho){
  tmp <- matrix(rho, d, d)
  diag(tmp) <- 1
  return(tmp)
}
```

```
set.seed(10)
reptime <- 2
d <- 4
n <- 1000
sigma_cross <- list(exchange_cor_matrix(d,0.2),
2*exchange_cor_matrix(d,0.5), 4*exchange_cor_matrix(d,0.5))
rho_sets <- c(0.3,0.3,0.3)
mean_shift <- c(0,0,0) # with mean change
cp_sets <- round(c(0,n/3,2*n/3,n))
ts <- MAR_MTS_Covariance(n, reptime, rho_sets, cp_sets, sigma_cross)
noCP <- length(cp_sets)-2
no_seg <- length(cp_sets)-1
for(rep_index in 1:reptime){
  for(index in 1:no_seg){ # Mean shift
    tau1 <- cp_sets[index]+1
    tau2 <- cp_sets[index+1]
    ts[[rep_index]][,tau1:tau2] <- ts[[rep_index]][,tau1:tau2] + mean_shift[index]
  }
}
# we plot one of the multivariate time series
ts <- ts[[1]]
```

3.2. SN Critical Value Tables

The package **SNSeg** contains three SN critical value tables: `critical_values_single`, `critical_values_multi` and `critical_values_HD`. The critical values are computed according to simulation studies of Zhao *et al.* (2021).

The table `critical_values_single` records critical values of SNCP tests for change in single parameter. It has the following columns:

- Column `"epsilon"`: Value of trimming parameter ϵ referred in Section 2. Available choice of ϵ includes 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.11, 0.12, 0.13, 0.14, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45 and 0.5.
- Columns `0.9`, `0.95`, `0.99`, `0.995` and `0.999`: Confidence level of SN tests.

The table `critical_values_multi` records critical values of SNCP tests for change in multiple parameters. Apart from the same columns described by `critical_values_single`, `critical_values_multi` has an additional column `p`:

- `p`: Dimension of parameters. Available choice of `p`: integers from 2 to 10. For a univariate time series, the dimension is equal to the number of parameters being tested on. For multivariate time series, this might not be the case. Thus, users should determine the true dimension of parameters and choose the correct p when using this table.

The table `critical_values_HD` records critical values of SNHD tests. It only works for multivariate time series with dimension greater than ten. This table has the same columns as those of `critical_values_single` but with one major difference:

- The column `epsilon` of `critical_values_HD` only offers critical values for $\epsilon = 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45$ and 0.5.

The critical values are utilized by functions `SNSeg_Uni_single_para`, `SNSeg_Uni_multi_para`, `SNSeg_Multi` and `SNSeg_HD`. We will introduce these functions in the following sections.

3.3. SN Test Statistics Segmentation

As shown in Section 2, we can acquire a list of maximal SN test statistics $T_{1:n}(k)$ or $T^*_{1,n}(k)$ for $k = 1, \cdots, n$. The function `maxSNsweep` allows users to calculate and plot these test statistics along with the detected change points.

```
    max_SNsweep(SN_sweep_result, plot_SN = TRUE)
```

It takes the following arguments:

- `SN_sweep_result`: A list of matrices with length n, containing the SN test statistics T_n or T^*_n, potential change-point locations k, lower bounds and upper bounds of scanning windows h. `SN_sweep_result` is among the output of functions `SNSeg_Uni_single_para`, `SNSeg_Uni_multi_para`, `SNSeg_Multi` and `SNSeg_HD`, and we will introduce these functions in the remaining sections of Section 3.
- `plot_SN` A boolean value to return a SN test statistics segmentation plot if `plot_SN = TRUE`.

We will display an example of the function `max_SNsweep` in Section 3.4.

3.4. SNCP Procedures for a Univariate Time Series with Change in a Single Parameter: SNSeg_Uni_single_para

The SNCP change-points estimation for a univariate time series based on the change in a single parameter can be implemented through the function `SNSeg_Uni_single_para`. It also detects change points with the change in correlation between two univariate time series.

```
SNSeg_Uni_single_para(ts, type = "mean", confidence = 0.9,
  quantile_level = 0.1, grid_size_scale = 0.05,
  grid_size = NULL, plot_SN = FALSE)
```

It takes the following arguments:

- `ts`: Input time series $\{Y_t\}_{t=1}^n$, i.e., a univariate time series represented as a numeric vector with length n.
- `type`: Type of the parameter of which SN algorithms test for, presented as a string. Only one of the parameters can be selected by users. Available choice of `type` includes:
  - `"mean"`: The function performs SN tests based on the change in mean of the time series.
  - `"variance"`: The function performs SN tests based on the change in variance.

  - `"acf"`: The function performs SN tests based on the change in autocorrelation.
  - `"bivcor"`: The function performs SN tests based on the change in bivariate correlation. Since the SN procedures test on the correlation change between two univariate time series, the input `ts` must be a two-dimensional matrix.
  - `"quantile"`: The function performs SN tests based on the quantile change. Note that the argument `quantile_level` is only used when `type = "quantile"`.

- `confidence`: Confidence level of SN tests as a numeric value. Available choice of confidence levels contains `0.9`, `0.95`, `0.99`, `0.995` and `0.999`. The critical value table of each confidence level can be achieved through the table `critical_values_single` of the package **SNSeg**. Default value of `confidence` is 0.9.

- `quantile_level`: The quantile value as a numeric value that SN tests upon if users set the argument `type = "quantile"`. `quantile_level` is ignored if `type` is not `"quantile"`. The quantile value should be within [0,1]. Default value of `quantile_level` is 0.1.

- `grid_size_scale`: A numeric value of the trimming parameter ϵ. The table `critical_values_single` records the critical value for `grid_size_scale` (the column `"epsilon"`). The function will set any input ϵ less than 0.05 to be exactly 0.05 and ϵ greater than 0.5 to be 0.5. A warning that "Detected the grid_size_scale is greater than 0.5" or "less than 0.05" will be posted in this case. The default value of `grid_size_scale` is 0.05. Only in use if `grid_size = NULL`.

- `grid_size`: Local window size h. Note that $h = n \times \epsilon$ (i.e., `grid_size`$= n \times$`grid_size_scale`.) The default value of `grid_size` is NULL, and thus the function calculates the critical value using `grid_size_scale`. However, if `grid_size` is set by a user, the function calculates `grid_size_scale` first by dividing n from `grid_size` then finds the critical value.

- `plot_SN`: Boolean value to plot the time series or not. The default setting is FALSE.

The function `SNSeg_Uni_single_para` is flexible in that it allows users to select the parameter type ("mean", "variance", etc.) for change-points estimation, and set the window size or select the appropriate ϵ to achieve the theoretical critical value, as discussed in Section 2. It also enables users to casually set up input values for the arguments `grid_size_scale` and `grid_size`. If `epsilon` calculated by any of these two arguments is within [0.05, 0.5] but not supplied by the table `critical_values_single`, the function will perform a linear interpolation by using two `grid_size_scale` that are nearest below or above it and calculating the weighted average of the corresponding critical values. The computed result will be the final critical value for the SN test.

When called, `SNSeg_Uni_single_para` returns a list of objects containing the following entries:

- `type`: A string that records the type of parameter used for SN test in changes.

- `grid_size`: A numeric value that computes the window size h.

- `SN_sweep_result`: A list of matrices with length n, where each of the matrices consists of four columns: (1) SN test statistic T_n for each change-point location k (2) Location k (3) Lower bound of the window h (4) Upper bound of the window h.
- `est_cp`: A vector containing the locations of the estimated change points.
- `confidence`: Confidence level of the SN test as a numeric value.
- `critical_value`: Critical value of the SN test statistic given ϵ and the confidence level.

To illustrate this function, we discuss the time series provided as an example of the function `MAR_Variance` in Section 3.1. We store and names the time series as "`ts`" which is shown in the example code of `MAR_Variance`. We manage to detect change-points with the change in variance using $\epsilon = 0.067$ and 90% confidence level. We can visualize the input time series plot using `SNSeg_Uni_single_para` and the SN test statistics segmentation plot using `max_SNsweep` along with the estimated change-points. Note that it is necessary to set `grid_size = NULL` to make the argument `grid_size_scale = 0.067` play a role in the change-point detection process.

```
set.seed(7)
ts <- MAR_Variance(2, "V1")
ts <- ts[,2]
par(mfcol = c(2, 1), mar = c(4, 2.5, 2.5, 0.5))
result <- SNSeg_Uni_single_para(ts, type = "variance", confidence = 0.9,
                grid_size_scale = 0.067, grid_size = NULL, plot_SN = TRUE)
abline(v = result$est_cp, col = "red")
max_SNsweep(result$SN_sweep_result, plot_SN = TRUE)
abline(v = result$est_cp, col = "red")
abline(h = result$critical_value, col = "blue")
```

The result is plotted in Figure 3. It can be seen that the SN statistics of the detected change-points exceed the critical value and are the local maxima. The estimated change-point locations, the window size as well as the critical value can be called as below.

```
# Estimated change point locations
result$est_cp
# Window size
result$grid_size
# Critical value
result$critical_value

### output ###
> result$est_cp
[1] 401 738
> result$grid_size
[1] 68
> result$critical_value
[1] 129.1731
```

The estimated change-point locations are $\hat{k} = 401$ and 738 with a local window size of $h = 68$ and critical value of $K_n = 129.1731$ when we set parameters $\epsilon = 0.067$ and 90% confidence level. The result matches the original setting where the true change-points are $k = 400$ and 750.

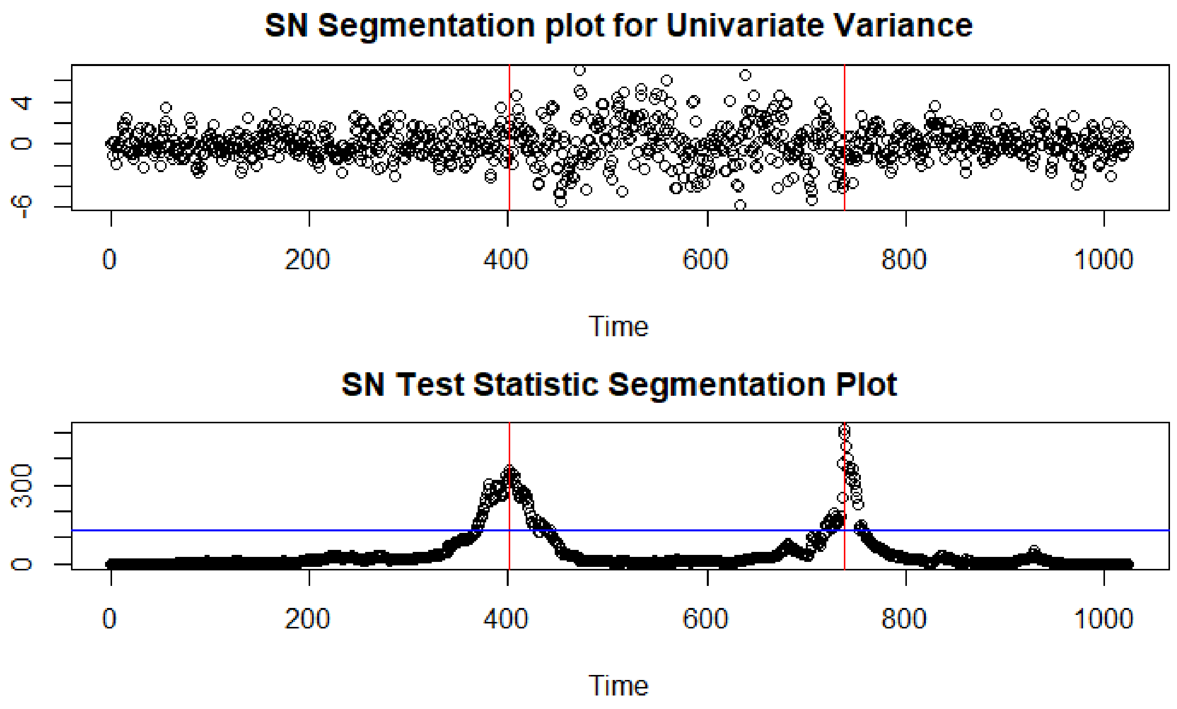

Figure 3: An example of simulated time series of `MAR_Variance` in Section 3.1. The upper plot detects change points for input time series and the lower one segments SN test statistics with estimated change points. The detected change-points locations are observed by a red vertical line and the critical value is observed by a blue horizontal line.

We then show another example where we use the argument `grid_size` to calculate the critical value and estimate change points. We apply the function `SNSeg_Uni_single_para` to the same time series as the previous example with the only difference that we set the window size h to be 65 instead of `NULL`. The estimated change-points and the critical value can be achieved as below.

```
result = SNSeg_Uni_single_para(ts, type = "variance", confidence = 0.9,
                grid_size_scale = 0.067, grid_size = 65, plot_SN = FALSE)
result$est_cp
result$critical_value

### Output ###
> result$est_cp
[1] 401 738
> result$critical_value
[1] 131.4857
```

The estimated change points still take place at $\hat{k} = 401$ and 738 but the critical value becomes 131.4857. Different critical values result from different window size h (or equivalently, different ϵs.)

3.5. SNCP Procedures for a Univariate Time Series with Change in Multiple Parameters

The SNCP change-points detection for a univariate time series based on the change in multiple parameters simultaneously can be implemented through the function `SNSeg_Uni_multi_para`.

```
    SNSeg_Uni_multi_para(ts, paras_to_test = c(0.9, 0.95), confidence = 0.9,
        grid_size_scale = 0.05, grid_size = NULL, d = 2, plot_SN = FALSE
)
```

Different from the function `SNSeg_Uni_single_para`, `SNSeg_Uni_multi_para` allows users to enter multiple parameters in the argument `paras_to_test` instead of a single parameter. It takes the following arguments besides those described by `SNSeg_Uni_single_para`:

- `ts`: Input time series $\{Y_t\}_{t=1}^n$, i.e., a univariate time series represented as a numeric vector with length n.
- `paras_to_test`: Parameters of which SN algorithms test on as a vector, which can be selected by users. Available choice of parameters to choose from:
  - `"mean"`: The function performs SN tests based on the change in mean of the time series.
  - `"variance"`: The function performs SN tests based on the change in variance.
  - `"acf"`: The function performs SN tests based on the change in autocorrelation.
  - A numeric value between [0,1] for SN tests based on the quantile change. Note that the function allows changes in multiple quantiles (e.g., `paras_to_test = c(0.6, 0.9)` refers to changes in 60% and 90% quantiles.)
- `d`: Dimension of the parameters (or equivalently, the length of `paras_to_test`.)

The function `SNSeg_Uni_multi_para` accepts a combination of the parameter types on the argument `paras_to_test`. For example, to test the change in mean, autocorrelation, 60% and 90% quantile of the input time series, users should set `paras_to_test = c("mean", "acf", 0.6, 0.9)`. It uses the critical value table `critical_values_multi` to compute critical values for SN-based tests. This function also applies the same linear interpolation rule on ϵ as `SNSeg_Uni_single_para` does in Section 3.4.

If called, `SNSeg_Uni_multi_para` returns a list of objects containing the following entries:

- `parameter`: A vector containing strings or numeric valuess that records the type of parameters used for SN tests.
- `grid_size`: A numeric value that computes the window size h.

- `SN_sweep_result`: A list of matrices with length n, where each of the matrices consists of four columns: (1) SN test statistic T_n for each change-point location k (2) Location k (3) Lower bound of the window h (4) Upper bound of the window h.
- `est_cp`: A vector containing the locations of the estimated change points as numeric values.
- `confidence`: Confidence level of the SN test as a numeric value.
- `critical_value`: Critical value of the SN test statistic given ϵ, the confidence level and the dimension d.

As an illustration, we consider the simulated univariate time series of ”MP1” setting in Section 4.4 of Zhao *et al.* (2021) in which the true change points are $k = 334, 667$. We aim to detect change points with the change in mean, variance, 90% and 95% quantiles simultaneously, using $\epsilon = 0.05$ and a 90% confidence level. We analyze this time series as below:

```
set.seed(7)
# simulation of time series data
n <- 1000
cp_sets <- c(0, 333, 667, 1000)
no_seg <- length(cp_sets) - 1
rho <- 0
ts <- MAR(n, 2, rho)*sqrt(1-rho^2)
no_seg <- length(cp_sets)-1
sd_shift <- c(1,1.6,1)
for(index in 1:no_seg){ # Mean shift
  tau1 <- cp_sets[index]+1
  tau2 <- cp_sets[index+1]
  ts[tau1:tau2,] <- ts[tau1:tau2,]*sd_shift[index]
}
d <- 2
ts <- ts[,2]

# change point estimation (4 parameters ---> d = 4)
result <- SNSeg_Uni_multi_para(ts, paras_to_test = c("mean","variance",0.9,
            0.95), confidence = 0.9, grid_size_scale = 0.05, grid_size = NULL,
            d = 4, plot_SN = FALSE)
result$est_cp
result$grid_size
result$critical_value

### Output ###
> result$est_cp
[1] 331 664
> result$grid_size
[1] 50
> result$critical_value
```

```
[1] 344.3622
```

As the example displays, the estimated change points take place at $\hat{k} = 331$ and 664. The locations match the simulation setting. This suggests testing the change in a broad of parameters is successful in detecting all change points.

3.6. SN Procedures for Multivariate Time Series

The SN change-point detection for multivariate time series can be realized via the function `SNSeg_Multi`. Changes in multivariate means or the covariance matrix of the input multivariate time series are available for this function.

```
SNSeg_Multi(ts, type = "mean", confidence = 0.9, grid_size = NULL,
            grid_size_scale = 0.05)
```

It takes the following arguments in addition to those accepted by `SNSeg_Uni_single_para` and `SNSeg_Uni_multi_para`:

- `ts`: Input time series $\{Y_t = (Y_{t1}, \cdots, Y_{td})\}_{t=1}^n$ as a matrix, i.e., a multivariate time series represented as a matrix with n rows and d columns, where each column is a univariate time series. The number of columns for `ts` should be integers within $[2, n]$.
- `type`: Type of the parameter as a string of which SN algorithms test for, which can be selected by users. Only one parameter can be selected. Available choice of `type` includes:
  - `"mean"`: The function performs SN tests based on the change in multivariate means of the time series.
  - `"covariance"`: The function performs SN tests based on the change in the covariance matrix of the time series.

Similar to the function `SNSeg_Uni_multi_para` in Section 3.5, `SNSeg_Multi` uses the table `critical_values_multi` and applies the same linear interpolation rule to find the critical value for SN tests. Note that `SNSeg_Multi` does not contain the `plot_SN` option to plot the input multivariate time series `ts`.

When called, `SNSeg_Uni_multi_para` returns a list containing `grid_size`, `SN_sweep_result`, `est_cp`, `confidence` and `critical_value` that are described by `SNSeg_Uni_single_para` and `SNSeg_Uni_multi_para`. The critical value is achieved by `critical_values_multi`.

As a simple example, we consider the simulation setting "M2" of Section 4.2 from Zhao *et al.* (2021). The multivariate time series is generated by the function `MAR_MTS_Covariance`, as described in Section 3.1 of this paper. The true change points are $k = 75, 376, 425, 525$ and 575. We analyze it based on the change in multivariate means using $\epsilon = 0.05$ under 90% confidence level as below:

```
# Generating Time Series
exchange_cor_matrix <- function(d, rho){
  tmp <- matrix(rho, d, d)
  diag(tmp) <- 1
  return(tmp)
}

set.seed(10)
d <- 5
n <- 1000
cp_sets <- round(n*c(0,cumsum(c(0.075,0.3,0.05,0.1,0.05)),1))
mean_shift <- c(-3,0,3,0,-3,0)/sqrt(d)
mean_shift <- sign(mean_shift)*ceiling(abs(mean_shift)*10)/10
rho_sets <- 0.5
sigma_cross <- list(exchange_cor_matrix(d,0))
ts <- MAR_MTS_Covariance(n, 2, rho_sets, cp_sets=c(0,n), sigma_cross)
noCP <- length(cp_sets)-2
no_seg <- length(cp_sets)-1
for(rep_index in 1:2){
  for(index in 1:no_seg){ # Mean shift
    tau1 <- cp_sets[index]+1
    tau2 <- cp_sets[index+1]
    ts[[rep_index]][,tau1:tau2] <- ts[[rep_index]][,tau1:tau2]
                                   + mean_shift[index]
  }
}
ts <- ts[1][[1]]

# Change-point detection
result <- SNSeg_Multi(ts, type = "mean", confidence = 0.9,
                      grid_size_scale = 0.05, grid_size = NULL)
### Output ###
> result$est_cp
[1]  73 376 525 575
> result$grid_size
[1] 50
> result$critical_value
[1] 415.8649
```

It is easy to observe that the estimated change points are $\hat{k} = 73, 376, 525$ and 575 with a window size $h = 50$ and critical value 415.8649. This result matches the true change-point locations except that it misses the change point at $k = 425$. To figure out why it is missed, we check the test statistics segmentation plot using the function `max_SNsweep`, as described by Section 3.3:

```
max_SNsweep(result$SN_sweep_result, plot_SN = TRUE)
abline(v = result$est_cp, col = "red")
abline(h = result$critical_value, col = "blue", lty = 2)
abline(v = 425, col = "red", lty = 2)
```

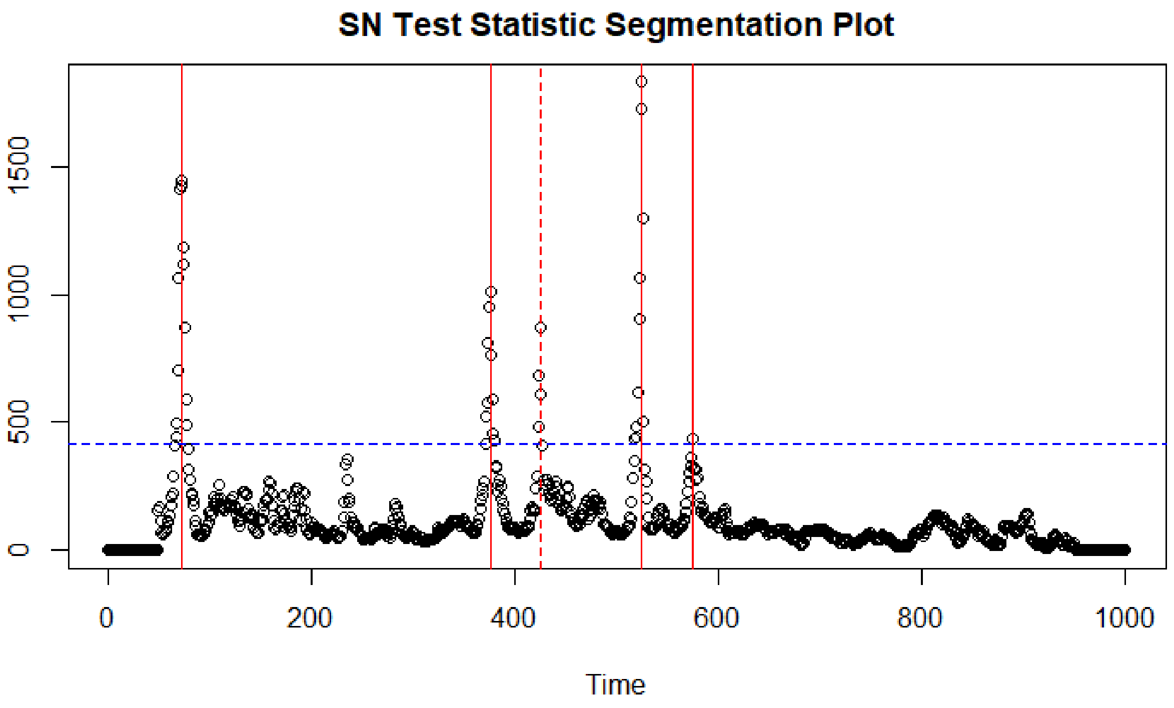

Figure 4: SN test statistics segmentation plot for multivariate time series based on the change in multivariate means. The blue line represents the critical value, the red solid lines are detected change-point locations, and the red dashed line refers to the missed change point at $k = 425$.

As shown in Figure 4, the four detected change points as well as the missed change point $k = 425$ all surpass the SN critical value. The reason that this change point is ignored is that all locally nested windows contain this point and the previous change point $k = 376$, but the SN algorithms tend to choose the point with the greatest test statistic as the change point if it is above the critical value. Thus, the SN procedures are successful in correctly estimating all change points though missing one of them.

3.7. SNHD Procedures for High-Dimensional Time Series

The function `SNSeg_HD` is dedicated to estimating change points for high-dimensional time series with the change in high-dimensional means.

```
SNSeg_HD(ts, confidence = 0.9, grid_size_scale = 0.05, grid_size = NULL)
```

It accepts these arguments that are incorporated in `SNSeg_Multi` with the difference that the input time series `ts` is a matrix whose dimension d is greater than ten.

When called, `SNSeg_HD` returns a list containing `grid_size`, `SN_sweep_result`, `est_cp`, `confidence` and `critical_value` that are described by `SNSeg_Uni_single_para` , `SNSeg_Uni_multi_para` and `SNSeg_Multi`. The critical value is derived from `critical_values_HD` introduced in Section 3.2.

As an example, we adopt the simulation setting "H4" from S.2.9 of supplementary texts of

Zhao *et al.* (2021). The true change points are stated as $k = 100, 200, 300, 400$ and 500. We apply `SNSeg_HD` to analyze this time series with $\epsilon = 0.05$ under 90% confidence level.

```
# Data Simulation
n <- 600
p <- 100
nocp <- 5
cp_sets <- round(seq(0,nocp+1,1)/(nocp+1)*n)
num_entry <- 5
kappa <- sqrt(4/5) # Wang et al(2020)
mean_shift <- rep(c(0,kappa),100)[1:(length(cp_sets)-1)]
set.seed(1)
ts <- matrix(rnorm(n*p,0,1),n,p)
no_seg <- length(cp_sets)-1
for(index in 1:no_seg){ # Mean shift
  tau1 <- cp_sets[index]+1
  tau2 <- cp_sets[index+1]
  ts[tau1:tau2,1:num_entry] <- ts[tau1:tau2,1:num_entry] + mean_shift[index]
}

# Change points detection
result <- SNSeg_HD(ts, confidence = 0.9, grid_size_scale = 0.05,
                   grid_size = NULL)

### Output ###
result$est_cp
[1]  94 199 301 406 500
```

We see that `SNSeg_HD` successfully detects all change-points. This shows the effectiveness and feasibility of SN algorithms for change point detection on high-dimensional time series.

4. More Usage Examples

In this section, we provide more usage examples on `SNSeg_UNi_single_para`, `SNSeg_UNi_multi_para` and `SNSeg_Multi` on cases with testing the change in different types of parameters. For examples of `SNSeg_HD`, please refer to Section 3.7.

4.1. Examples of Function "SNSeg_UNi_single_para"

For `SNSeg_UNi_single_para`, an example for testing the change in variance is given in Section 3.4. We offer examples of testing the change in mean, autocorrelation, bivariate correlation and quantile.

Change in Univariate Mean

To test the change in a single mean, we simply name the argument `type` with the string `"mean"` in `SNSeg_UNi_single_para`. As an example, we apply this to the example time series of MAR, which is described in Section 3.1. The change-points for this simulation setting are $k = 1000$ and 1500, and we check the performance of SNCP using $\epsilon = 0.05$ and 90% confidence level.

```
# Data Simulation
set.seed(7)
ts <- MAR(n = 2000, reptime = 2, rho = -0.7)
mean_shift <- c(0.4,0,0.4)
cp_sets <- round(n*c(0,cumsum(c(0.5,0.25)),1))
no_seg <- length(cp_sets)-1
for(index in 1:no_seg){ # Mean shift
  tau1 <- cp_sets[index]+1
  tau2 <- cp_sets[index+1]
  ts[tau1:tau2,] <- ts[tau1:tau2,] + mean_shift[index]
}
ts <- ts[,2]

# Change points detection
result <- SNSeg_Uni_single_para(ts,type = "mean", confidence = 0.9,
                grid_size_scale = 0.05, grid_size = NULL, plot_SN = FALSE)

### Output ###
> result$est_cp
[1] 1001 1485
```

The SNCP procedures correctly detect the correct number and the locations of change points.

Change in Univariate Variance

An example of testing the change in univariate variance is presented in Section 3.4. Please check Section 3.4 for more details.

Change in Univariate Autocorrelation

We rely on the argument `type = "acf"` to test the change in autocorrelation. As a simple example, we generate A3 time series described in Section 3.1 using MAR_Variance. The true change points are $k = 512$ and 768. We then observe the performance of SNCP algorithms with $\epsilon = 0.05$ and 90% confidence level.

```
# Data Simulation
set.seed(7)
ts <- MAR_Variance(2, "A3")
ts <- ts[,2]

# Change points detection
result <- SNSeg_Uni_single_para(ts,type = "acf", confidence = 0.9,
                grid_size_scale = 0.05, grid_size = NULL, plot_SN = FALSE)
```

```
### Output ###
> result$est_cp
[1] 510 767
```

The locations of detected change points match those of the true change points.

Change in Bivariate Correlation

For testing in bivariate correlation, we initially change `type` into `"bivcor"`. In addition, the dimension of the input time series must be two. As an example, we adopt the ”R1” simulation setting from supplementary materials of Zhao *et al.* (2021). The true change points take place at $k = 333$ and 667. We analyze this bivariate time series using a window size `grid_size=77` under 90% confidence level.

```
# Data Simulation
library(mvtnorm)
set.seed(7)
sigma_cross <- list(4*matrix(c(1,0.8,0.8,1), nrow=2),
      matrix(c(1,0.2,0.2,1), nrow=2), matrix(c(1,0.8,0.8,1), nrow=2))
cp_sets <- round(c(0,n/3,2*n/3,n))
noCP <- length(cp_sets)-2
rho_sets <- rep(0.5, noCP+1)
ts <- MAR_MTS_Covariance(n = 1000, reptime = 2, rho_sets = rho_sets,
                         cp_sets = cp_sets, sigma_cross = sigma_cross)
ts <- ts[1][[1]]

# Change points detection
result <- SNSeg_Uni_single_para(ts,type = "bivcor", confidence = 0.9,
                grid_size_scale = 0.05, grid_size = 77, plot_SN = FALSE)

### Output ###
> result$est_cp
[1] 336 667
```

The SN procedures successfully detect the locations of the true change points.

Change in Univariate Quantile

Finally, to test the change in a single quantile level, users need to set `type = "quantile"` and a specific quantile level within [0,1] for the argument `quantile_level`.

As an example, we utilize the ”Q1” simulation setting of Section S.2.8 in the supplementary texts of Zhao *et al.* (2021). The only change point is $k = 500$ for this time series. We apply SNCP algorithms to estimate change points as below, based on the change in 95% quantile, using a window size $h = 80$ and 95% confidence level:

```
# Data Simulation
library(truncnorm)
library(evd)
mix_GauGPD <- function(u,p,trunc_r,gpd_scale,gpd_shape){
  indicator <- u<p
  rv <- rep(0, length(u))
  rv[indicator>0] <- qtruncnorm(u[indicator>0]/p,a=-Inf,b=trunc_r)
  rv[indicator<=0] <- qgpd((u[indicator<=0]-p)/(1-p),
                      loc=trunc_r,scale=gpd_scale,shape=gpd_shape)
  return(rv)
}
set.seed(7)
cp_sets <- c(0,n/2,n)
noCP <- length(cp_sets)-2
rho <- 0.2
ts <- MAR(n = 1000, reptime = 2, rho = rho)*sqrt(1-rho^2)
trunc_r <- 0
p <- pnorm(trunc_r)
gpd_scale <- 2
gpd_shape <- 0.125
for(ts_index in 1:reptime){
  ts[(cp_sets[2]+1):n, ts_index] <- mix_GauGPD(pnorm(ts[(cp_sets[2]+1):n,
                           ts_index]),p,trunc_r,gpd_scale,gpd_shape)
}
ts <- ts[,2]

# Change points detection
result <- SNSeg_Uni_single_para(ts,type = "quantile", confidence = 0.9,
                quantile_level = 0.95, grid_size_scale = 0.05,
                grid_size = 80, plot_SN = FALSE)

### Output ###
> result$est_cp
[1] 493
```

The estimated change point location is 493, which is very close to the true location 500. These results show that SNCP procedures are successful in finding the correct number and the location of this change-point with different types of parameters.

4.2. Examples of Function ”SNSeg_UNi_multi_para”

In section 3.5, an example to test the change simultaneously in mean, variance, 90% and 95% quantiles is given, using the argument `paras_to_test = c("mean", "variance", 0.9, 0.95)`. For an input time series, any combination of `"mean"`, `"variance"`, `"acf"` and quantile values within [0,1] can work for `paras_to_test`. Users need to set the correct input value for the argument `d`, the dimension of the parameters. Please check Section 3.5 on the usage of `SNSeg_UNi_multi_para`.

4.3. Examples of Function "SNSeg_Multi"

4.4. Change in Multivariate Means

For the function `SNSeg_Multi`, an example of testing the change in multivariate means for a multivariate time series is given in Section 3.6. Please check Section 3.6 for usage details.

4.5. Change in Covariance Matrix

To test the change in covariance, one is obliged to set `type = "mean"` for the function `SNSeg_Multi`. As an example, we adopt the "C3" simulation setting of Section S.4.3 of the supplementary texts of Zhao *et al.* (2021). The true change-point locations are $k = 333$ and 667. We apply `SNSeg_Multi` to detect change points with window size $h = 70$ and 90% confidence level. The code to generate time series using `MAR_MTS_Covariance` is provided in Section 3.1, and we offer the code for change-points analysis as below:

```
# Change points detection
result <- SNSeg_Multi(ts, type = "covariance", confidence = 0.9,
                      grid_size_scale = 0.05, grid_size = 70)

### Output ###
> result$est_cp
[1] 330 650
```

The SNCP procedures correctly detect the number of change points and the location of the first change point. The second detected change point is $\hat{k} = 650$, and it is close to the true location $k = 667$.

5. Conclusion

In this paper, we have presented the R package **SNSeg** (need citation here). Implementations of SN procedures for both univariate, regular multivariate ($d \leq 10$) and high-dimensional ($d > 10$) time series, as well as testing based on the change in single or multiple parameters for change-points detection have been introduced. The main functions for multiple change-points detection, `SNSeg_Uni_single_para`, `SNSeg_Uni_multi_para`, `SNSeg_Multi` and `SNSeg_HD` have been described with examples to generate time series data, analyze time series and visualize both time series and SN test statistics segmentation plots.

References

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

Betken A, Wendler M (2018). "Subsampling for general statistics under long range dependence with application to change point analysis." *Statistica Sinica*, **28**(3), 1199–1224.

Chan NH, Ng WL, Yau CY (2021). "A self-normalized approach to sequential change-point detection for time series." *Statistica Sinica*, **31**(1), 491–517.

Cho H, Fryzlewicz P (2012). "Multiscale and multilevel technique for consistent segmentation of nonstationary time series." *Statistica Sinica*, pp. 207–229.

Dette H, Gösmann J (2020). "A likelihood ratio approach to sequential change point detection." *Journal of the American Statistical Association*, **115**(531), 1361–1377.

Dette H, Kokot K, Volgushev S (2020). "Testing relevant hypotheses in functional time series via self-normalization." *Journal of Royal Statistical Society: Series B*, **82**(3), 629–660.

Eichinger B, Kirch C (2018). "A MOSUM procedure for the estimation of multiple random change points." *Bernoulli*, **24**(1), 526–564.

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. URL `http://www.jstatsoft.org/v23/i03/`.

Hawkins DM, Qiu P, Kang CW (2003). "The changepoint model for statistical process control." *Journal of quality technology*, **35**(4), 355–366.

Hawkins DM, Zamba K (2005). "A change-point model for a shift in variance." *Journal of Quality Technology*, **37**(1), 21–31.

Hoga Y (2018). "A structural break test for extremal dependence in β-mixing random vectors." *Biometrika*, **105**(3), 627–643.

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. URL `https://www.jstatsoft.org/v62/i07/`.

Killick R, Eckley IA (2014). "changepoint: An R Package for Changepoint Analysis." *Journal of Statistical Software*, **58**(3), 1–19. URL `https://www.jstatsoft.org/article/view/v058i03`.

Killick R, Fearnhead P, Eckley IA (2012). "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`. URL `https://doi.org/10.1080%2F01621459.2012.737745`.

Korkas KK, PryzlewiczV P (2017). "Multiple change-point detection for non-stationary time series using wild binary segmentation." *Statistica Sinica*, pp. 287–311.

Lavitas L, Zhang T (2018). "A time-symmetric self-normalization approach for inference of time series." *Journal of Time Series Analysis*, **39**(5), 748–762.

Lobato IN (2001). "Testing that a dependent process is uncorrelated." *Journal of the American Statistical Association*, **96**(455), 1066–1076.

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.

Meier A, Kirch C, Cho H (2021). "mosum: A Package for Moving Sums in Change-Point Analysis." *Journal of Statistical Software*, **97**(8), 1–42. `doi:10.18637/jss.v097.i08`.

Phillips PC (1987). "Time series regression with a unit root." *Econometrica: Journal of the Econometric Society*, pp. 277–301.

Ross GJ (2014). "Sequential change detection in the presence of unknown parameters." *Statistics and Computing*, **24**(6), 1017–1030.

Ross GJ (2015). "Parametric and Nonparametric Sequential Change Detection in R: The cpm Package." *Journal of Statistical Software*, **66**(3), 1–20. URL `https://www.jstatsoft.org/v66/i03/`.

Shao X (2010). "A self-normalized approach to confidence interval construction in time series." *Journal of the Royal Statistical Society: Series B*, **72**(3), 343–366.

Shao X, Zhang X (2010). "Testing for change points in time series." *Journal of the American Statistical Association*, **105**(491), 1228–1240.

Wang G, Zou C (2022). "cpss: an package for change-point detection by sample-splitting methods." *Journal of Quality Technology*, pp. 1–14.

Zeileis A (2006). "Implementing a Class of Structural Change Tests: An Econometric Computing Approach." *Computational Statistics & Data Analysis*, **50**(11), 2987–3008. `doi:10.1016/j.csda.2005.07.001`.

Zeileis A, Kleiber C, Krämer W, Hornik K (2003). "Testing and Dating of Structural Changes in Practice." *Computational Statistics & Data Analysis*, **44**(1–2), 109–123. `doi:10.1016/S0167-9473(03)00030-6`.

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`.

Zhao Z, Jiang F, Shao X (2021). "Segmenting Time Series via Self-Normalization." *arXiv preprint arXiv:2112.05331*.

Zou C, Wang G, Li R (2020). "Consistent selection of the number of change-points via sample-splitting." *Annals of statistics*, **48**(1), 413.

Affiliation:

Zifeng Zhao
Assistant Professor
Dual Faculty of Business Analytics and Statistics
Mendoza College of Business, Notre Dame, IN 46556
University of Notre Dame
E-mail: `zzhao2@nd.edu`
URL: `https://mendoza.nd.edu/mendoza-directory/profile/zifeng-zhao/`

Shubo Sun
PhD Student
Department of Management Science
Miami Herbert Business School, Coral Gables, Miami, FL 33146
University of Miami
E-mail: `sxs3935@miami.edu`
URL: `https://sites.google.com/view/shubosun/home?authuser=0`

Xiaofeng Shao
Professor
Faculty of Statistics
College of Liberal Arts Sciences, Urbana, IL 61801
University of Illinois Urbana-Champaign E-mail: `xshao2@illinois.edu`
URL: `http://publish.illinois.edu/xshao/`

Feiyu Jiang
Assistant Professor
Faculty of Statistics and Data Science
School of Management
Fudan University
E-mail: `jiangfy2@fudan.edu.cn`
URL: `https://sites.google.com/view/feiyujiang/`

Journal of Statistical Software `http://www.jstatsoft.org/`
published by the American Statistical Association `http://www.amstat.org/`
Volume VV, Issue II *Submitted:* yyyy-mm-dd
MMMMMM YYYY *Accepted:* yyyy-mm-dd