Early classification of spatio-temporal events using partial information

Sevvandi Kandanaarachchi[1]
Rob J Hyndman[1]
and
Kate Smith-Miles[2]

[1]Department of Econometrics and Business Statistics, Monash University, Clayton VIC 3800, Australia.
[2]School of Mathematics and Statistics, The University of Melbourne, Parkville, VIC 3010, Australia

24 October 2019

Abstract

This paper investigates event extraction and early event classification in contiguous spatio-temporal data streams, where events need to be classified using partial information, i.e. while the event is ongoing. The framework incorporates an event extraction algorithm and an early event classification algorithm. We apply this framework to synthetic and real problems and demonstrate its reliability and broad applicability. The algorithms and data are available in the R package *eventstream*, and other code in the supplementary material.

1 Introduction

Early detection and classification of emerging events in data streams is an important challenge in our data-rich world. Data streams may arise from many different applications including social media, Internet of Things, video surveillance, epidemiology and wireless sensors, to name a few. In each of these diverse applications, there are typically events that occur and are of interest because of their disruptive behaviour to the system.

In particular, we are interested in events that start, develop for some time, and stop at a certain time. Such events can be characterised by measurable properties or features, including the "age" of the event. It is a challenge to classify these events while they are still developing because only partial information is available at this stage. Once the events have stopped developing — when the events are finished — it is easier to classify them as the complete event features are now available. For example, it is easier to differentiate a daffodil from a tulip when both are in full bloom, but more difficult to differentiate a daffodil bud from a tulip bud without resorting to other information such as characteristics of leaves. Another example is identifying a network intrusion attack in its early stages. While it may be easier to identify a breach after it has happened, it is more difficult to identify which bits of network traffic is causing the breach while it is happening (Suthaharan 2014).

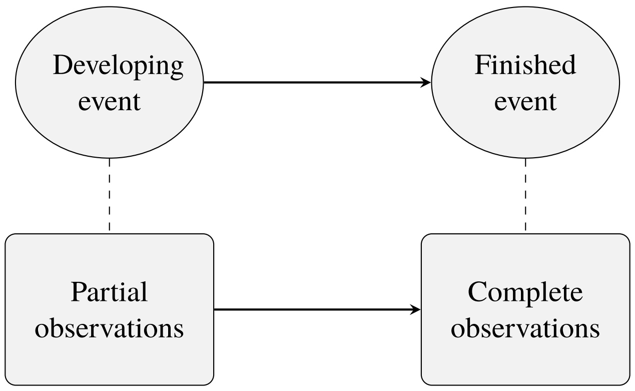


Figure 1: *Event states and partial observations.*

In this regard, we can think of these events as having two states: developing and finished (Figure 1). The partial information contained in the developing events give rise to partial or premature observations, while the finished events give rise to complete observations. As the event develops, it gives rise to a series of partial observations — each partial observation encapsulating more information than its predecessor — culminating with the complete observation (Figure 2). Thus partial observations vary with the age of the event, the difference between the current time and the start time of the event. If early classification is important, one needs to take partial observations into account in the classification process. While event detection in data streams has received much attention from different disciplines ranging from video surveillance to social media (Medioni et al. 2001, Li et al. 2012), there has been little exploration on developing/premature event classification to the best of our knowledge.

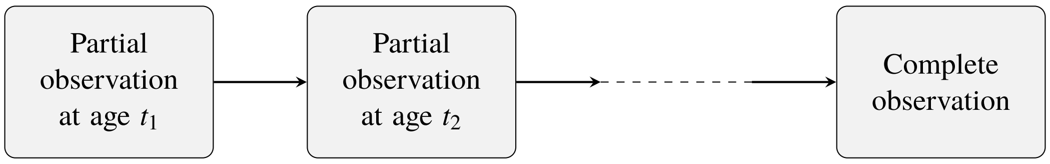

Partial observation at age t_1
Partial observation at age t_2
Complete observation

Figure 2: *Partial observations growing with event-age.*

A general framework for event classification in data streams comprises different stages: 1. data pre-processing; 2. event detection and extraction; 3. feature computation; and 4. event classification. This framework, augmented with partial observations, gives the additional functionality of early event classification as depicted in Figure 3. In our framework we do not explicitly consider data pre-processing as a separate stage as this is highly dependent on the application.

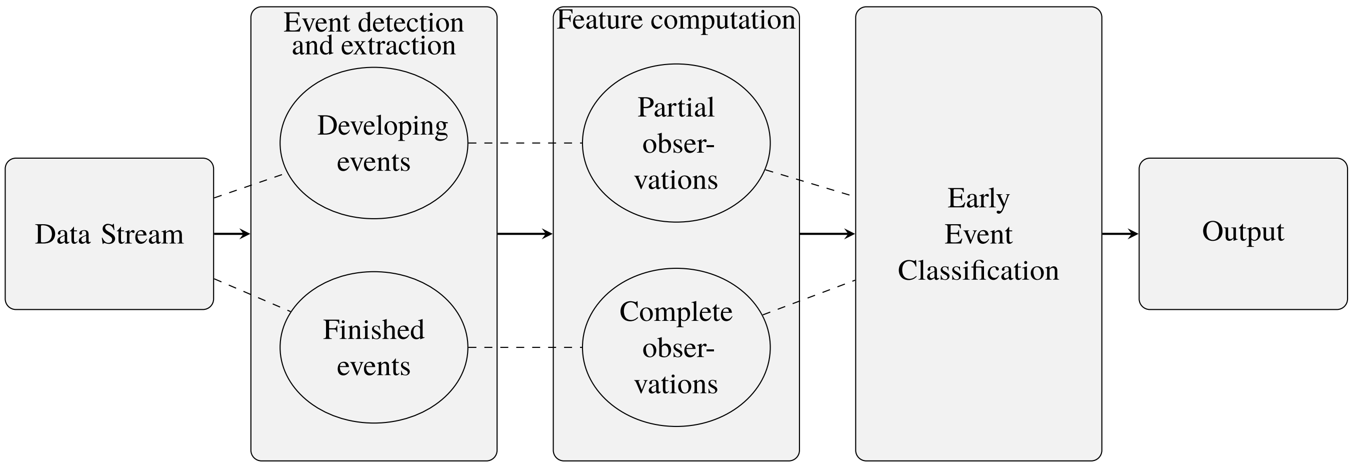


Figure 3: *Framework for event extraction and classification for spatio-temporal data.*

Fundamentally, early event classification can be tackled by embedding age-varying coefficients in a learned model (Hastie & Tibshirani 1993). A linear model with age-varying coefficients is given by

$$y_t = a_0(t) + a_1(t)x_1(t) + \cdots + a_b(t)x_b(t) + \varepsilon_t, \tag{1}$$

where y_t is the output at age t, $a_i(t)$ are the age-varying coefficients, and $x_i(t)$ are the attributes of the event at age t; i.e. the partial/premature observation. A logistic model with age-varying coefficients is given by Equations (1) and (2):

$$z_t = e^{y_t}/(1 + e^{y_t}), \tag{2}$$

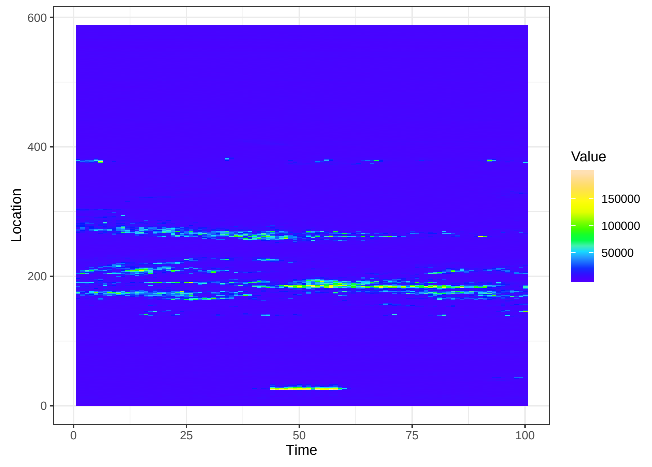
where z_t is the probability of the event being of a given class. As an event develops, the features $x_i(t)$ change with the age of the event, while keeping the class label constant. Thus, it is clear that the coefficients $a_i(t)$ need to change with the age of the event.

At this point, we note that concept drift (Gama et al. 2014) or non-stationarity of data streams (Hulten et al. 2001) is different from age-varying events. For non-stationary data streams the distribution of data changes with time. For example consider a fixed part of a river, which is monitored for fluctuations in water volume and for animals. In months of heavy rains, the water volume increases changing the distribution compared to previous months. This is an example of non-stationarity. In contrast, age-varying events are about the extracted events and not the data-stream. To continue with the same example, consider a log appearing on this portion of the river. When the log comes closer and the image becomes clearer, suppose it becomes apparent that it is not a log, but a crocodile. This is an example of an age-varying event. Clearly, from the time when the log appeared to the time when it was detected that it was a crocodile, no significant changes in water volume or the animal distributions took place. The volume of water and the average number of crocodiles in the river does not need to change when the perception of the log changed to that of a crocodile as a result of more information. Thus age-varying events comprise change within the event as a result of maturing partial observations, while non-stationarity concerns change within the data stream.

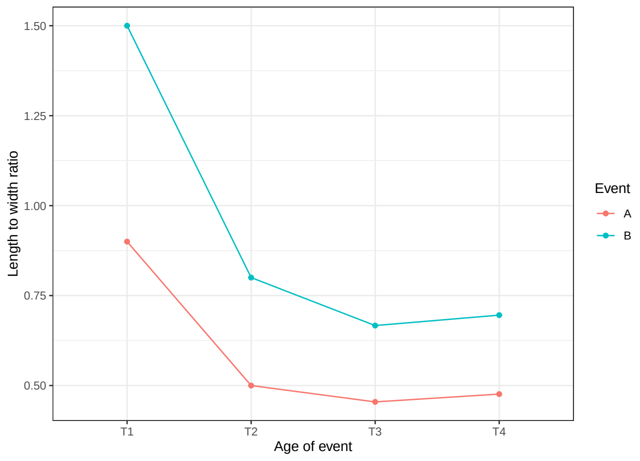
1.1 Fibre optic cable example

We turn to an application where age-varying events occur. Figure 4a shows the heatmap of a dataset produced from a fibre optic cable. A pulse is periodically sent through the cable and this results in a data matrix where each horizontal row gives the strength of the signal at a fixed location x_0, and each vertical column gives the strength of the signal along the cable at a fixed time t_0. In this dataset the yellow parts represent high intensity values and the blue parts represent low intensity values.

Fibre optic sensor cables are used in many applications including optical communications, detecting undersea cable faults (Jiang & Sui 2009), detecting oil leakages (Niklès et al. 2004), detecting intruders on secured premises (Griffiths 1995), and monitoring health of infra-structure such as bridges and pipe-lines (Li et al. 2004).



(a)



(b)

Figure 4: *Figure 4a shows data from a fibre optic cable. We extract events from this dataset and compute event features. We consider two events belonging to two different classes and an event feature that changes with event-age. Figure 4b shows this event feature, which is the length to width ratio of the event, and how it changes with event-age.*

Events in these applications can often be grouped into two classes. For example, a cable lying on the sea bed can produce spatio-temporal events that are either cable faults (A), or non-fault events due to the activity in the ocean (B). Due to its sensitivity, fibre optic cables are also prone to noise. In a setting where early classification is important, we need to classify these events quickly, preferably while they are still ongoing.

In the dataset in Figure 4a, events are seen in the lighter-coloured parts. The event at approximately location 30 between the time interval 45 to 60 is of class A while other events that appear between locations 150 and 400 are of class B. Figure 4b shows an event feature, which is the length to width ratio of the event, computed on two events belonging to each class. As the event matures, we see this feature change with event-age and also that no single threshold can differentiate between the two events for all event ages. Due to the commercially sensitive nature of the dataset, we refrain from giving details about the actual application.

1.2 Contributions

We propose the framework depicted in Figure 3, which is summarized in Algorithm 1, for early event detection, extraction and classification in contiguous spatio-temporal data streams using the partial observation structure.

Specifically, our contributions in this paper are:

1. We introduce an algorithm for event detection and extraction from contiguous spatio-temporal data. We use change point analysis and density based clustering to detect events. We call this algorithm Change-Point Density-Based Event Extraction (CPDBEE).
2. We introduce a partial observations classifier suitable for early event classification. This classifier comprises multiple base classifiers, which are connected together using L_2 penalty terms. We refer to this Connected Classifier as CC.
3. We demonstrate the validity of these algorithms on synthetic and real data and make the algorithms and data available in the R package *eventstream* (Kandanaarachchi 2018)

The remainder of the paper is organised as follows. Section 2 discusses related work in event detection, extraction and classification. In Section 3 we introduce the datasets: synthetic data, fibre optic cable data, of which a portion is shown in Figure 4, and NO_2 data from NASA's NEO (*NASA Earth Observations* 2004) website. We use all these datasets to demonstrate the effectiveness of the proposed event extraction and classification algorithms in subsequent Sections. We introduce our event detection and extraction algorithm CPDBEE in Section 4 and

```
input  : a 2 or 3-dimensional array denoting contiguous spatio-temporal data.
output : events, event features and early classification results
1 Detect and extract developing and complete events from the data stream using CPDBEE (Sections 4 and 5).
2 Compute event features. These are either partial or complete features computed from the extracted events (Section 6.1).
3 Use the Connected Classifier CC for early classification of events (Sections 6.3–7).
```

Algorithm 1: *Early event extraction and classification framework.*

discuss event extraction results in Section 5. Section 6 presents the early classification framework by starting with event features in Section 6.1, followed by an explanation of partial observations in Section 6.2, and culminating with the connected classifier CC in Section 6.3. We discuss the early classification results in Section 7 and present our conclusions and discuss future work in Section 8. Section 9 gives details on supplementary materials, which can be used to reproduce the results and Appendix A gives additional graphs of CPDBEE results.

2 Related work and their applicability

Spatio-temporal event detection is studied in many application related research areas such as epidemiology (Kulldorff 1997), deforestation (Verbesselt, Hyndman, Zeileis & Culvenor 2010), video streaming (Ke et al. 2007) and social media research (Weng & Lee 2011). In these applications the focus is on detecting "events of interest". For some applications events of interest are rare events, while for others they are specific events, which match certain criteria (Ke et al. 2007). Typically these events form a subgroup of data rather than a single data-point and their early detection has a strong societal impact (Earle et al. 2012).

2.1 Change-point detection

Univariate event detection has much overlap with change-point detection methods in time series (Guralnik & Srivastava 1999). Killick et al. (2012) introduces change-point analysis as "the identification of points within a dataset where statistical properties change". They formally consider a time series $y_{1:n} = (y_1, \dots, y_n)$ with m change-points $\tau_{1:m} = (\tau_1, \dots, \tau_m)$, with $\tau_i < \tau_j$ for $i < j$, resulting in $m + 1$ segments of the time series with the ith segment containing $y_{(\tau_{i-1}+1):\tau_i}$. They identify change-points by minimizing

$$\sum_{i=1}^{m+1} C(y_{(\tau_{i-1}+1):\tau_i}) + \beta m,$$

where C is the cost function for a segment and βm is the penalty term for having m segments. An example cost function is the negative log-likelihood. Their method PELT identifies change-points with linear computational time.

Multivariate change-point detection extends this framework to multiple time series measuring different quantities. Bardwell et al. (2019) consider multivariate change-point detection in a panel data setting. They define $\mathcal{G}_i(r)$ as the cost of segmenting time series i with the most recent change point r and minimize a penalized version of

$$\mathscr{C}_K = \min_{I_1,\dots,I_K \;\; r_1,\dots,r_K} \sum_{k=1}^{K} \sum_{i \in I_k} \mathcal{G}_i(r_k),$$

where K denotes the number of change-points, $I_k \subset \{1, 2, \dots, N\}$ and N the number of time series, such that for all time series $i \in I_k$ the most recent change-point is located at r_k.

Even though change-point detection methods detect changes, they do not generally identify a subset of changed observations, i.e. they do not perform event extraction. For our applications we need event detection as well as event extraction.

2.2 Scan Statistics

In epidemiology, the scan statistic introduced by Kulldorff (1997) and its later versions (Kulldorff 2001, Neill 2012) have gained much popularity. Using patient counts for each zip-code or similar region, the scan statistics approach detects events or clusters of interest, which may correspond to regions affected by a disease outbreak. The underlying assumption is that a true event will significantly increase patient counts, which is not accounted for by seasonality effects or random noise. Thus, events detected by the scan statistic approach are candidate regions for disease outbreaks.

The spatial scan statistic model (Kulldorff 1997) considers the null hypothesis H_0 representing no events and alternative hypotheses $H_1(S)$ representing an event in a region S for some S. They compute the score function

$$F(S) = \frac{\Pr[\text{Data}|H_1(S)]}{\Pr[\text{Data}|H_0(S)]}$$

using Bernoulli and Poisson models, for different regions S, with circular scanning windows of varying radii centred at each spatial location. To account for multiple hypotheses testing, they conduct Monte Carlo simulations. They perform 9999 replications of the dataset under the null hypothesis and compute the test statistic for each replicated dataset and region S. Then they rank the actual test statistic for region S with the replicated test statistics and consider the actual to be significant if it is within the top 5% of replicated test statistic values. This is a time intensive algorithm.

The focus here is mainly on event detection and extraction and not on event classification. For example every event detected may not correspond to a disease outbreak. There may be some other explanation for an event.

2.3 Deforestation studies

A popular use of Landsat images is the study of deforestation and changes in land cover (Verbesselt, Hyndman, Newnham & Culvenor 2010). Verbesselt et al. (2012) discusses changes in land cover caused by 1. seasonal effects driven by annual temperature and rainfall patterns, 2. gradual changes such as forest regrowth after fire, or 3. abrupt changes caused by deforestation, bushfires or urbanisation. Detecting abrupt changes, while accounting for seasonal variations is an important research problem in this domain.

However, all detected changes may not be due to deforestation. In order to detect only deforestation, Hamunyela et al. (2016) calibrate their change detection algorithm using training data. In their paper, they tune the detection algorithm to capture certain activities of interest, i.e. detection and classification are performed as a single task.

The study conducted by Zhu & Woodcock (2014) considers detection and classification as two separate tasks. After detecting changes, they classify the land cover using a Random Forest classifier on the time series model coefficients. We note that they do not classify the event, but the land cover, which is again different to our focus. Furthermore, these studies do not consider event extraction; they treat each pixel separately and report results at a pixel level.

In addition to these research areas, social media research (Atefeh & Khreich 2015) also investigates event detection. However, their focus is on text analysis and related techniques, which is quite different from ours.

2.4 A note on comparison

In our framework, event detection and extraction is a different task from event classification. Events of interest — class A events in Section 1.1 — may not necessarily have higher signal values compared to class B events as in disease outbreak scenarios. Furthermore, it is not desirable to improve the accuracy of the event extraction algorithm at the expense of missing class A events. Missing a class A event has a much higher cost than detecting a non-event for applications such as intrusion detection. Moreover, some applications require faster response times than is feasible by scan statistics methods.

Unlike in deforestation studies, analysis at a pixel level is not beneficial for the fibre optic application discussed in Section 1.1. A contiguous block of space-time pixels comprising an event needs to be considered for effective classification. Furthermore, even applications that consider event classification as well as extraction do not modify the original classifier to suit partial observations. This may be partly because they do not classify the event while it is taking place.

3 Applications and datasets used

We use three sets of datasets to evaluate the event extraction and classification algorithms: synthetic data, fibre optic cable data and NO_2 data.

3.1 Synthetic data

The synthetic data was motivated from the fibre optic application and can be generated using the R package *eventstream*. The synthetic data contains events of two classes: A and B. All events belonging to class A look similar, that is they have one single non-standard shape or visual pattern. In contrast, events belonging to class B can have one of three different non-standard shapes, including the shape of events of class A. This is a characteristic of the fibre-optic application data, which prevents effective early classification of events based on shape alone.

Figure 5a contains two events of class A, and Figure 5b contains 3 events of class B. The shapes are labelled as 1, 2 or 3 in both Figures 5a and 5b, with shape 1 being the common shape.

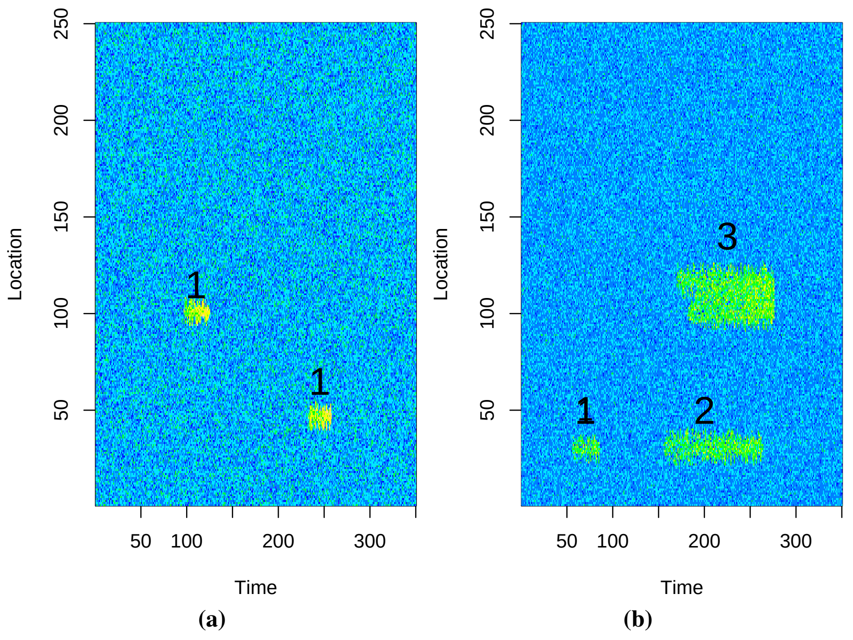


Figure 5: *Class A events in Figure 5a and Class B events in Figure 5b.*

The number of events of class A and B, and their positions, are randomly generated. The other difference between the events of class A and B, apart from the shape, is that values of the pixels belonging to events of class A and B come from different probability distributions. For both classes the intensity of pixel values increase linearly with the age of the event. We list the differences between class A and B events in Table 1.

Feature	Class A value distribution	Class B value distribution
Starting cell/pixel values	$\mathcal{N}(4, 3)$	$\mathcal{N}(2, 3)$
Ending cell/pixel values	$\mathcal{N}(8, 3)$	$\mathcal{N}(4, 3)$
Maximum age of event: shape 1	$\mathcal{U}(20, 30)$	$\mathcal{U}(20, 30)$
Maximum age of event: shape 2	–	$\mathcal{U}(100, 150)$
Maximum age of event: shape 3	–	$\mathcal{U}(100, 150)$
Maximum location width of event: shape 1	$\mathcal{U}(20, 26)$	$\mathcal{U}(20, 26)$
Maximum location width of event: shape 2	–	$\mathcal{U}(30, 38)$
Maximum location width of event: shape 3	–	$\mathcal{U}(50, 58)$

Table 1: *Differences in class A and class B events.*

3.2 Fibre optic cable data

The data for the first real application is from a fibre optic cable, and is shown in Figure 6. The data set is available in the R package *eventstream*. Again, for commercially sensitive reasons, we cannot provide more information about the application. The data set has dimensions 379×587, with class A events labelled with letter **A**. All other events belong to class B.

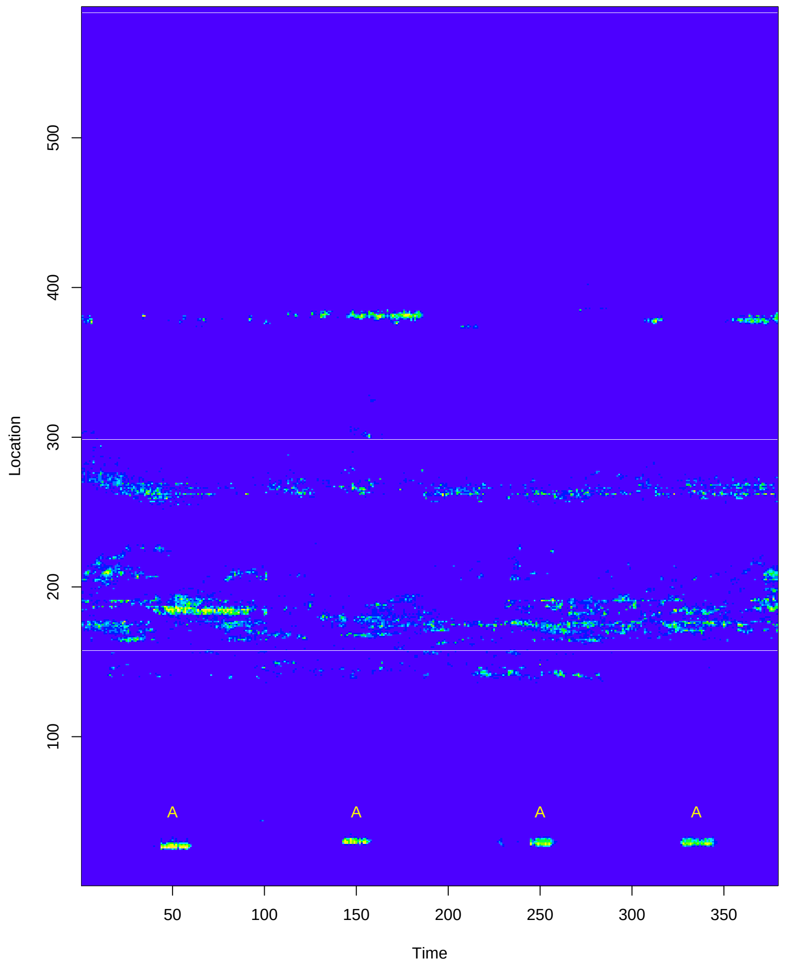


Figure 6: *Data stream from a fibre optic cable.*

3.3 Nitrogen dioxide monitoring

The second real world application uses Nitrogen Dioxide (NO_2) data obtained from NASA's NEO website (*NASA Earth Observations* 2004). Nitrogen Dioxide is a major factor of air pollution (Geddes et al. 2015), which causes approximately 7 million deaths per year according to the World Health Organisation (*WHO air pollution* 2019).

The Ozone Monitoring Instrument (OMI) (Levelt et al. 2006) aboard the Aura satellite records a variety of air quality measures including NO_2 concentrations around the world. This is a 3-dimensional data stream with two spatial and one time dimension.

We use OMI NO_2 monthly data from March to June for 10 years from 2010 to 2019 to detect and classify NO_2 clusters. For each month the data comes in a matrix of 180×360 dimensions. The OMI NO_2 data for March 2018 is shown in Figure 7.

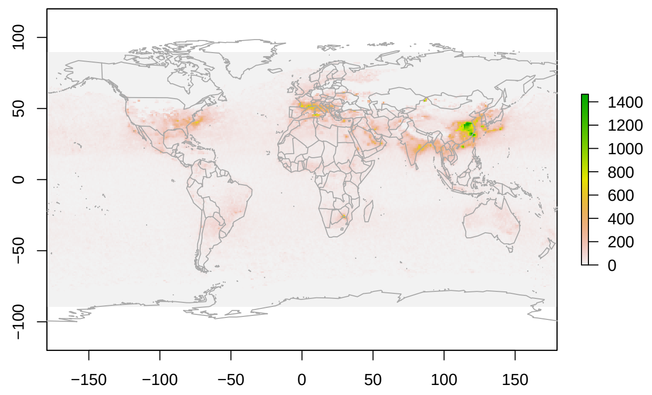


Figure 7: *NO_2 data for March 2018.*

4 Event detection and extraction

We extract events from data streams of two or three dimensions having one time dimension and one or two spatial dimensions. We employ a method for event extraction using change point detection (Killick & Eckley 2014) and DBSCAN (Ester et al. 1996), which is a density based clustering algorithm. Change point detection is used for event detection purposes and clustering for event extraction purposes.

4.1 Event detection

Change point detection in time series is a well studied topic as seen from the survey by Aminikhanghahi & Cook (2017). Killick & Eckley (2014) discuss the R package *changepoint*, which includes three change point detection algorithms: a binary segmentation algorithm (Scott & Knott 1974, Sen & Srivastava 1975), a segment neighborhood algorithm (Auger & Lawrence 1989, Bai & Perron 1998) and PELT (Killick et al. 2012). These algorithms are capable of detecting structural changes in time series based on mean and/or variance.

As we work with two or three-dimensional data streams we transform the data to suit univariate change point detection methods described in the R package *changepoint*. For a two-dimensional dataset we perform Principal Component Analysis (PCA) twice on the data similar to Sadia et al. (2019). Consider, a dataset $X_{n \times t}$ having n contiguous spatial points and t equi-distant time points. First we consider each location as an observation and perform PCA on $X_{n \times t}$. We are interested in the first set of PC scores of this analysis. Second, we consider each time point as an observation and perform PCA on X^T. For each analysis, we consider the first set of PC scores and find change points using PELT as it is faster.

For a three-dimensional dataset $X_{n \times m \times l}$ we compute averages $\tilde{X}_{n \times m}$ and $\tilde{X}_{m \times l}$ and perform PCA twice on each of these averaged matrices as in the two-dimensional case.

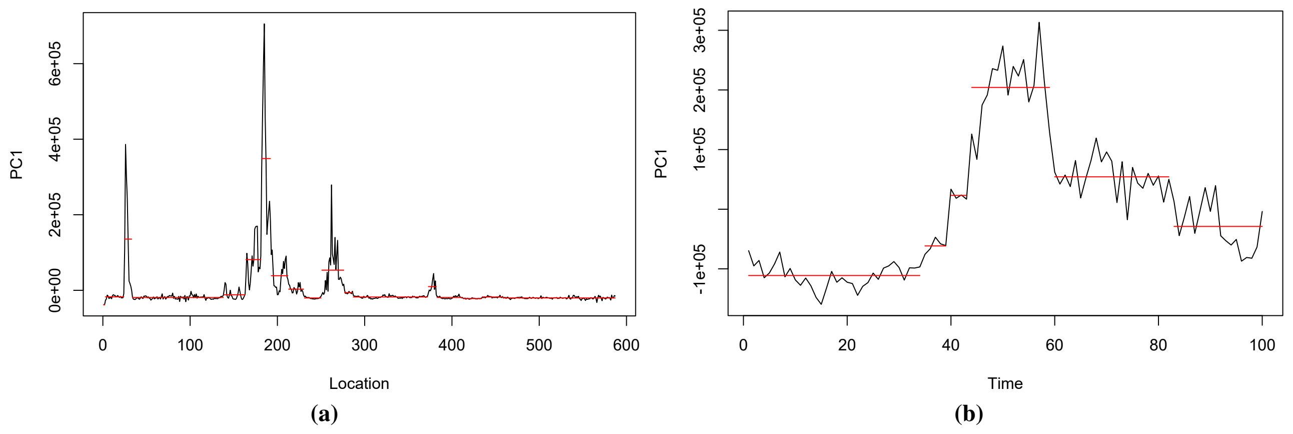


Figure 8: *Change points of the first PC scores of the dataset in Figure 4. Figure 8a shows the first PC scores when taking each location as an observation. The horizontal red lines denote the levels and the change points correspond to the breaks or discontinuities of levels. Figure 8b shows the first PC scores when taking each time point as an observation and the associated change points.*

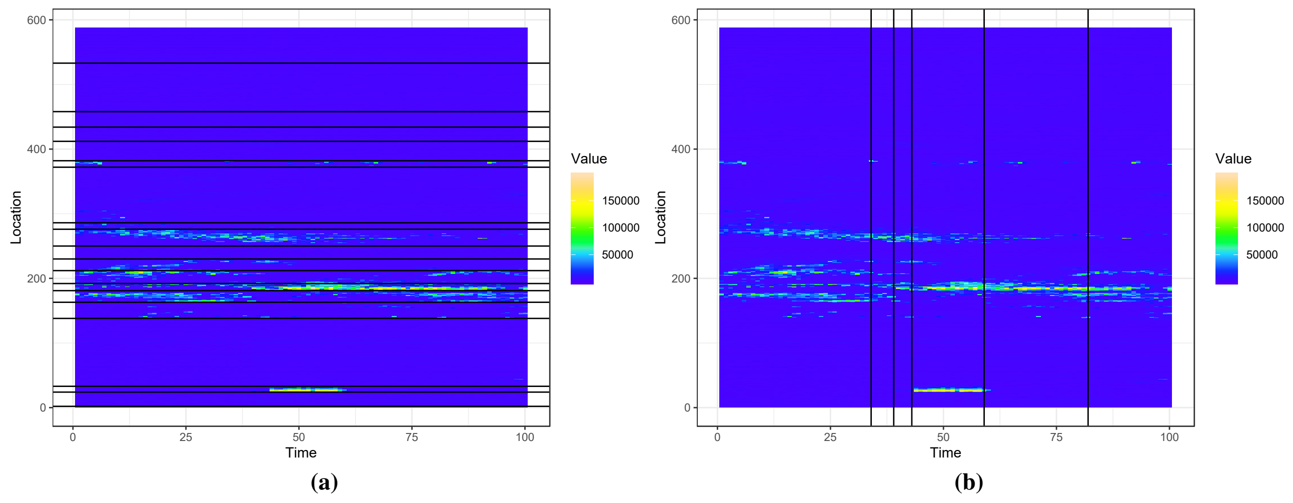


Figure 9: *Location change points in Figure 9a and time change points in Figure 9b.*

Figure 8 shows the coordinates of the first PC vector of the dataset illustrated in Figure 4. This dataset has 587 contiguous location points and 100 time points, and can be denoted as $X_{587\times100}$. By performing PCA on X, we obtain 100 PC vectors for 587 observations, where each observation denotes a location. We consider the coordinates of these observations in the direction of first PC vector, i.e. the first set of PC scores, and find their change points. PELT detects the following location change points: 2, 24, 33, 138, 163, 181, 192, 212, 230, 250, 276, 286, 372, 382, 412, 434, 458 and 533. Figure 8a illustrates the first set of PC scores and the location change points. Similarly, performing PCA on X^T considers each time point as an observation. PELT detects time change points at 34, 39, 43, 59 and 82 using the first set of PC scores of X^T. Figure 8b illustrates the first set of PC scores and the associated time change points. Figure 9 shows the time and location change points as vertical and horizontal lines drawn on the heatmap of this dataset.

We see that the class A event at location 30 between time intervals 45 and 60 is detected by PELT in time and location using the first set of PC scores. In addition, the events denoted by lighter-coloured parts between locations 150 and 300 are also detected by location change points. However, the location change points that are greater than 400 do not correspond to any lighter-coloured parts in Figure 9a. In our framework summarized in Algorithm 1, event extraction precedes event classification. Thus, it is preferred to detect and extract candidate events which may not correspond to real events, rather than employ stringent event extraction methods and miss real events, i.e. type 1 errors are preferred at the event extraction stage.

4.2 Event extraction

Once the events are detected the next task is to extract them. For the dataset illustrated in Figure 4, the true events are light-coloured contiguous parts, which have higher signal values than the background. The change points computed in Section 4.1 alone are not sufficient to extract these events as seen in Figure 9. Clustering is a tool that is often used in event extraction (Ruocco & Ramampiaro 2014). We use DBSCAN clustering in our event extraction process.

To extract events we consider pixels which have high signal values, defined by a percentile α. That is, if x_{ij} is the signal value at (i, j) position of X, then we denote by q a signal value corresponding to the percentile α. The default value of α is 95%. We consider pixels x_{ij} greater than q and cluster these in time and location using DBSCAN. DBSCAN allocates pixels that are close to each other to the same cluster. These clusters are our candidate events. However, some candidate events may not have contributed to the change points discussed in Section 4.1. We are interested in candidate events that are detected by change points. Thus, if a time or location change point is detected within a candidate event or at the boundary of a candidate event, we consider that candidate event as a legitimate event. We discard candidate events which do not meet this criterion.

We summarize the event detection and extraction algorithm CPDBEE for two dimensional datasets in Algorithm 2.

```
   input  : a 2 dimensional matrix $X_{n\times m}$, and parameters $\alpha$, $\epsilon$ and $minPts$.
   output : events and event ids
1  Compute PCA on $X_{n\times m}$.
2  Let $z_1$ denote the first set of PC scores of $X$.
3  Let $C_1$ be the set of change points of $z_1$ using PELT.
4  Compute PCA on $X^T$.
5  Let $z_2$ denote the first set of PC scores of $X^T$.
6  Let $C_2$ be the set of change points of $z_2$ using PELT.
7  Let $q$ denote the $\alpha$-percentile of the signal values of $X$.
8  $S = \{(i, j) \mid x_{ij} > q\}$. $S$ is a matrix of 2 columns, which gives locations of $X$, which have signal values greater than the $\alpha^{\text{th}}$
     percentile.
9  Let $X(S)$ be signal values of $X$ in $S$ locations.
10 Using DBSCAN cluster $S$ using $\epsilon$ and $minPts$.
11 This clustering gives each $(i, j) \in S$ a cluster id. Noise points are given cluster id 0.
12 Let $T$ be the vector of cluster ids for each $(i, j)$ pair in $S$, i.e. the $k^{\text{th}}$ row of $S$ denotes a pixel location in cluster $T(k)$.
13 Consider each cluster as a candidate event and the cluster id as the candidate event id.
14 Let $S_1$ be the first column of $S$, i.e. $S_1$ has the first coordinate of each pair $(i, j)$ in $S$. Similarly let $S_2$ denote the second column
     of $S$.
15 Let $I_1 = S_1 \cap C_1$. These are $x_1$ positions of candidate events that are change points.
16 Let $I_2 = S_2 \cap C_2$. These are $x_2$ positions of candidate events that are change points.
17 Let $E = \{(i, j) \in S \mid i \in I_1 \text{ or } j \in I_2\}$. These are $x_1$ or $x_2$ positions of candidate events that are also change points.
18 Find candidate event ids $G$ which do not have any change points in $E$.
19 /* For example the $k^{\text{th}}$ candidate event may not have any pixels that are change points.   */
20 Remove these candidate events from $T$ and $S$. The remaining clusters $(S, X(S))$ are considered events.
```

Algorithm 2: *Algorithm CPDBEE for 2D datasets.*

For a three dimensional dataset $X_{n\times m\times l}$ with two spatial and one time dimension, DBSCAN clustering is performed on the three dimensional matrix to find the candidate events, and PCA is performed on two dimensional averaged matrices $\bar{X}_{n\times m}$ and $\bar{X}_{m\times l}$ to find the change points in each dimension. Candidate events which contribute to change points are considered events of the three dimensional dataset.

CPDBEE considers pixels which have high signal values for event extraction. For a different application such as deforestation, true event pixels may have lower values compared to the rest. For such applications, CPDBEE can be adapted to consider pixels that have signal values less than the percentile α, or alternatively, used in its current form by multiplying the dataset by -1.

4.2.1 The parameters of CPDBEE

The algorithm CPDBEE has three parameters α, ϵ and $minPts$ with the following defaults:

$$\alpha = 0.95, \qquad \epsilon = 5, \qquad \text{and} \qquad minPts = 10. \tag{3}$$

The parameter α depends on the application. It can be roughly described as the proportion of data contributing to events. In our fibre optic example, events are rare and correspond to high signal values in the data matrix. As such we set α to a high percentile.

The parameters ϵ and *minPts* are DBSCAN parameters. The parameter ϵ describes the size of the ϵ-neighbourhood and *minPts* denotes the the minimum number of points in the ϵ-neighbourhood that are needed to make a cluster. DBSCAN has a default value of 5 for *minPts*, which we have increased to 10 as we are not interested in very small events. The value of ϵ is set to 5 because we would like to consider two high signal valued pixels that are 5 pixels apart as belonging to the same event.

5 Event extraction results

We extract events using CPDBEE and compare with events extracted using Kulldorff's Scan Statistic. We use the R implementation by Kim & Wakefield (2018) to extract events using the Scan Statistic. The scan statistic was originally computed using population counts and the number of patient visits of each geo-spatial region. For our data we analogize the signal value in each cell to the number of patient visits in an epidemiology context. In addition, the formulation needs the population of each cell to compute significant clusters. As we do not have an underlying population for the fibre optic cable, each cell is equally likely to belong to a significant cluster. Therefore we assign the same population value to all cells in our data. We take the maximum signal value of the window multiplied by 20 as the population value of every cell. Thus each cell has a maximum of 5% of population "sick" at a given time. We use a significance level of 5% in our experiments.

Appendix A contains the complete comparison results for fibre optic, synthetic and NO_2 data. This section contains only two figures for each dataset due to space constraints.

5.1 Events extracted from fibre optic data

Figure 10 shows the fibre optic data and the extracted events using CPDBEE and Kulldorff's Scan Statistic.

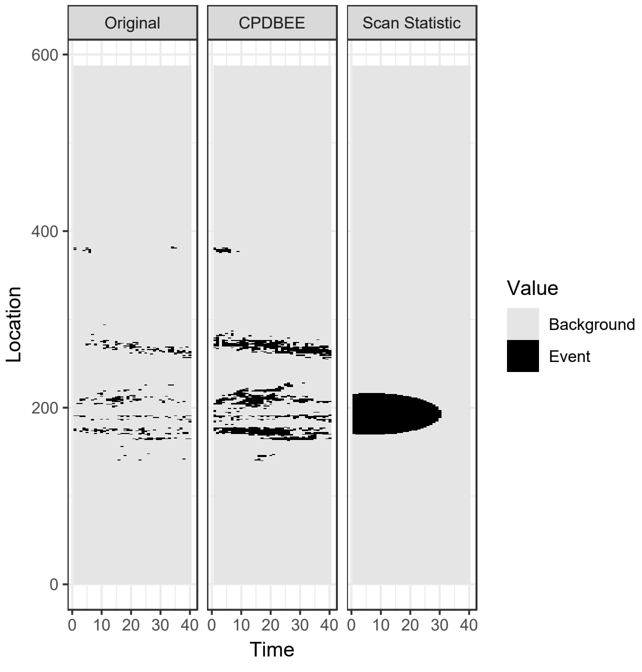
We used a window model and chose a window size of 40 as the Scan Statistic implementation could not handle a larger window size. Even with a window size of 40, Scan Statistic computation took much longer than CPDBEE.

The first tile of each graph shows a simplified version of the original data, i.e. pixels having signal values greater than $40,000$ are depicted in black while other pixels are depicted in grey. Even though the cut-off value of $40,000$ is completely arbitrary, it is purely used for visualisation purposes and is not an input parameter for the event extraction algorithms. The second tile shows the events extracted using CPDBEE and the third tile shows the events extracted using the Scan Statistic algorithm. Pixels belonging to extracted events are depicted in black, while other pixels are depicted in grey.

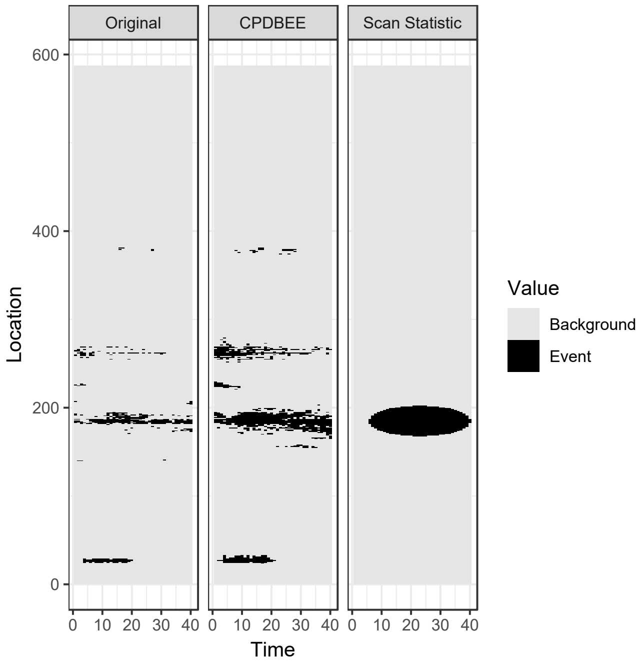
Class A events are present in the original data in Figure 10b and Figures 24b, 24d, 25c and 25e in Appendix A. As discussed previously we do not want to miss Class A events for this particular application. We see that CPDBEE extracts all four class A events, while the Scan Statistic algorithm only extracts the class A event in Figure 24d. Furthermore, CPDBEE extracts events in their original shape, while the Scan Statistic algorithm extracts events more in the shape of an ellipse in these examples. As such, CPDBEE is more efficient and effective that the Scan Statistic for this application.

5.2 Events extracted from synthetic data

Using the R package *eventstream*, we generate a 350×250 matrix of synthetic data, where 350 denotes the time units and 250 denotes location units. Figure 11 shows two 50×250 windows of synthetic data and the events extracted using CPDBEE and Scan Statistic algorithms. The full comparison is illustrated in Figure 26. The choice of the window size is because the Scan Statistic algorithm could not work with bigger window sizes.



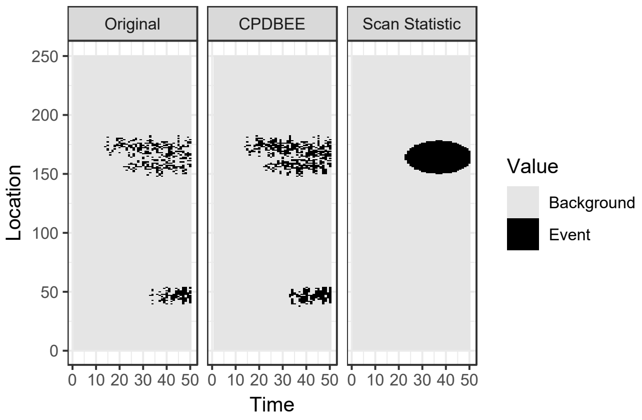
(a)



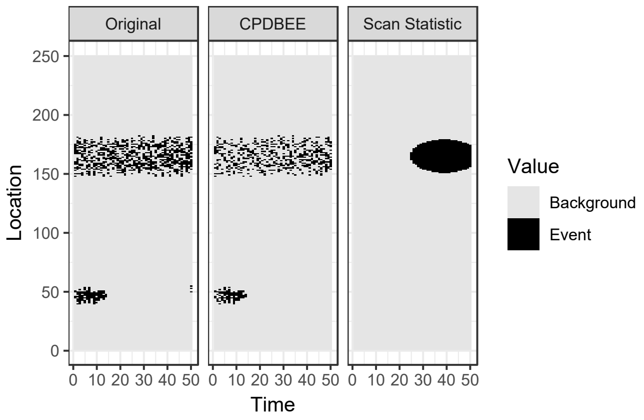
(b)

Figure 10: *Event extraction comparison for fibre optic data.*

The first tile of each graph shows a simplified version of the original window, with pixel values greater than 10 coloured in black and the rest in grey. The second and the third tiles show the events extracted using CPDBEE and Scan Statistic algorithms. Again we see that the events extracted by CPDBEE are more accurate than those extracted using the Scan Statistic algorithm. In addition, the Scan Statistic algorithm misses events in Figures 11a, 11b, 26b, 26c and 26d which is detrimental to certain applications.



(a)

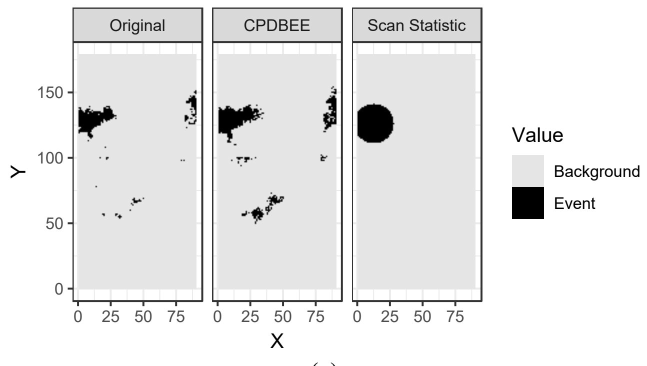


(b)

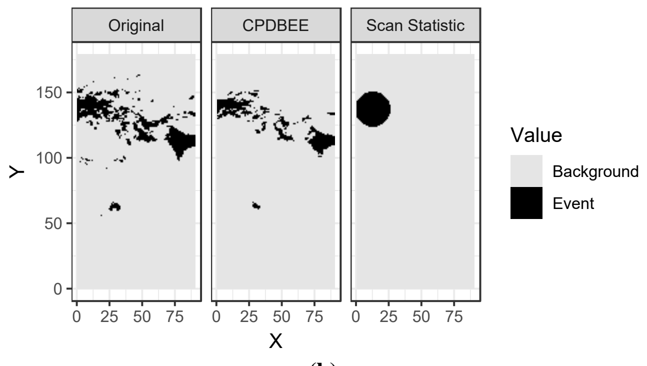
Figure 11: *Event extraction comparison for synthetic data.*

5.3 Events extracted from NO_2 data

We chose NO_2 data for March 2018 to evaluate the event extraction algorithms CPDBEE an Scan Statistic. Figure 12 shows the original data and the events extracted by each algorithm for two spatial windows. The first panel shows a simplified version of the original data with NO_2 values greater than 100 depicted in black and the rest in grey. The second and the third panels show the events extracted using CPDBEE and the Scan Statistic algorithm.



(a)



(b)

Figure 12: *Event extraction comparison for NO_2 data.*

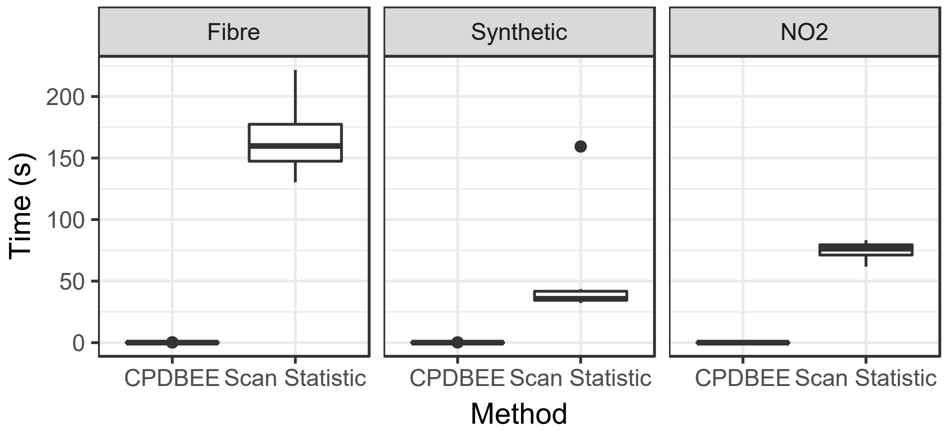


Figure 13: *Time comparison of CPDBEE with Scan Statistic for fibre optic data, synthetic data and NO_2 data.*

Figure 13 shows the time taken by CPDBEE and the Scan Statistic algorithm for these three applications using an Intel®Core™i7-6700, 3.4 GHz processor with 16 GB RAM. We see that CPDBEE extracts events much faster than the Scan Statistic algorithm for all three applications. The Monte Carlo simulations, which is an integral part of the Scan Statistic algorithm contributes to its time intensiveness. Furthermore, CPDBEE does not miss any important events and extracts better shaped events compared to the Scan Statistic algorithm.

6 Early event classification framework

6.1 Event features

As we work with a data stream, we use a moving window model in our experiments. We extract events from data in the current window and compute features for these events. The feature set comprises some basic features such as length and width of each event, and some other features that compute the intensity of each event relative to the background. The “relative to the background” features are equivalent to a family of signal to noise ratio (SNR) features and are motivated from the fibre optic application (see Figure 4a).

To compute the SNR family of features we use smoothing splines and thus they are only computed for two-dimensional data streams due to ease of computation. Using a small portion from the beginning of each window, which correspond to the recent past, we compute the mean, median, interquartile range (IQR) and standard deviation for each location. Using these values at each location, we compute four smoothing splines. The objective is to have the background mean, median, IQR and standard deviation pixel value for each location. The median and IQR splines from a small window in Figure 14a are shown in Figures 14b and 14c.

For two-dimensional events we compute the following features:

1. Number of cells/pixels in event
2. Length of event
3. Width of event
4. Length to width ratio of event
5. Centroid
   The centroid is used to compute other features which are relative to the event. It is not used in event classification.
6. Sum of signal-values of cells in event
7. Mean signal-value of event
8. Standard deviation of signal-values of event
9. Slope of the fitted line ζ
   The average signal value at each time of the event is computed and a line ζ is fitted to the average values. The slope of the fitted line ζ is a feature of interest .
10. Linear and quadratic coefficients of a fitted parabola p
    The average signal value at each time of the event is computed and a parabola p is fitted to the average values. The linear and quadratic coefficients of the fitted parabola p are features of interest.
11. n standard deviations from the mean
    The proportion of event cells/pixels that has signal-values greater than n global standard deviations from the global mean for $n \in \{2, 3, 4\}$.
12. n local IQR from local median
    The value of the median smoothing spline at each event centroid is used as the local median for that event. Similarly, the value of the IQR smoothing spline at each event centroid is used as the local IQR for that event. This feature gives the proportion of event pixels/cells that has signal-values greater than n local IQRs from the local median for $n \in \{5, \ldots, 8\}$
13. Local IQRs from local median
    Let us denote the 75th percentile of the event signal value by x. This feature gives the number of local IQRs for which x is greater than the local median. Both local IQR and local median are computed using splines described above.
14. Local standard deviation from local mean
    Similar to the previous feature, our x is the 80th percentile of the event signal value. Here we compute the number of local standard deviations for which x is greater than the local mean.

For three-dimensional data streams we compute a subset of the above features. In particular, we compute features 1–10 from the above list and an equivalent of feature 14 using the global standard deviation and the global mean. In addition, we use the squared value of these features in applications with enough data, i.e. the synthetic and NO_2 datasets. These features now provide a compact way to represent a data stream and the embedded events, summarising salient properties of the time window in terms of event signal strength and shape. This summary becomes input to a classifier to identify types of events.

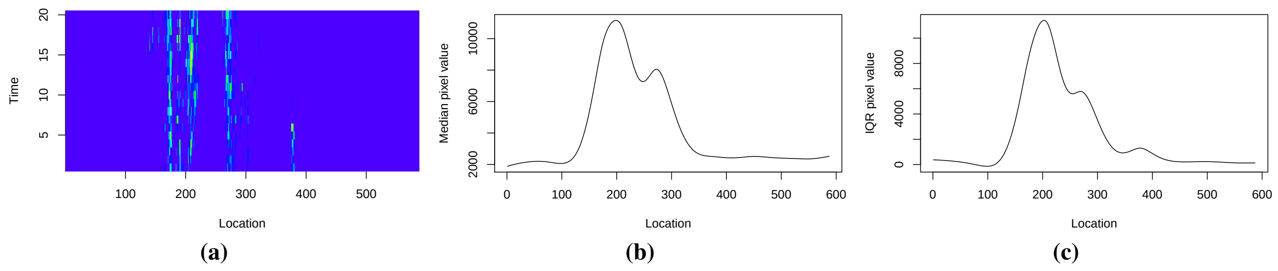


Figure 14: *The initial portion of a window and the resulting median and IQR splines.*

6.2 Partial/incomplete observations

In the classical setting, a classification problem comprises observations $(\boldsymbol{x}_i, y_i)$ for $i \in 1, \dots, N$ where $\boldsymbol{x}_i \in \mathbb{R}^b$ is the attribute vector of the ith observation and y_i is its class label. The task of the classifier is to learn the class boundary by using the given set of observations. Then for any new observation $\boldsymbol{x}_j$, the classifier can predict its class label using the learned class boundaries. Let us call this a **standard classifier**.

Standard classifiers have been widely popular in diverse fields of study and practice. However, they are not without limitations. One of the limitations is that once a classifier is trained, it has fixed class boundaries. If the new data is different from the data learned by the classifier, the output of the classifier is of little use. This is particularly the case in data-streaming scenarios, where data distributions are non-stationary (sometimes also referred to as concept drift). It is necessary for a classifier to re-adjust its class boundaries when faced with non-stationarity. The literature on adapting or evolving classifiers is significant (Ditzler et al. 2015). Let us call these classifiers **evolving classifiers**.

Now, consider the case when a new observation is not made available at once but gradually, where we get partial information about the new observation and the amount of partial information increases with time. This is the case for events described in Section 1.1. Let $\boldsymbol{x}_j$ be a new observation which becomes available partially via the following finite sequence of partial observations $\{\boldsymbol{p}^j_{t_1}, \boldsymbol{p}^j_{t_2}, \boldsymbol{p}^j_{t_3}, \dots, \boldsymbol{p}^j_{t_n}\}$. Here the partial observation of $\boldsymbol{x}_j$ at age t_k is denoted by $\boldsymbol{p}^j_{t_k}$ and $\boldsymbol{p}^j_{t_n} = \boldsymbol{x}_j$ with $t_1 < t_2 < \cdots < t_n$. We differentiate between the time and the age of a partial observation. A partial observation that begins at time $t = t_1$ has age 0 at time t_1, and at time $t = t_2$ it has age $t_2 - t_1$.

We consider the question "how can we classify partial observations?" If one trains a single standard classifier on all partial observations, it may be optimal for a certain set of partial observations p_{t_k} at a given age t_k, but not all partial observations, because partial observations change with time. If one waits until the partial observation has formed into a full observation $\boldsymbol{x}_j$, then a standard classifier can be used. However, for some applications such as intrusion detection it might be too late to wait until the full observation has formed. One option is to have a series of standard classifiers $\{C_{t_i}\}_{i=1}^n$ each trained on partial observations p_{t_i}. When a new observation gradually arrives in the form of a sequence of partial observations $\{\boldsymbol{p}^k_{t_1}, \boldsymbol{p}^k_{t_2}, \boldsymbol{p}^k_{t_3}, \dots, \boldsymbol{p}^k_{t_n}\}$, the classifier C_{t_i} can be used on $\boldsymbol{p}^k_{t_i}$. Thus, as the partial observation grows, we have a growing prediction $\{y_{t_1}, y_{t_2}, \dots, y_{t_n}\}$ of the class label. More importantly, we do not need to wait until the partial observation matures to a full observation before making a prediction.

However, having a series of classifiers independent of each other is sub-optimal because each classifier is only trained on a portion of the data, i.e. it is trained on individual snapshots of events at different ages. By linking event snapshots of different event-ages in an appropriate way, better predictions can be achieved.

In addition, event extraction algorithms may miss events of interest when the events are very young. As such there may be more events extracted at age t_2 compared to age t_1. Similarly, all events may not continue until age t_n. Consequently there may be more events at age t_{n-1} compared to t_n. For example consider 20 events which are extracted at ages $\{t_1, t_2, t_3, t_4, t_5\}$ such that only 5 are extracted at age t_1, 15 at t_2, 20 at t_3, 15 at t_4 and 5 at t_5. In such a scenario, if we have a set of 5 independent classifiers $\{C_{t_1}, C_{t_2}, C_{t_3}, C_{t_4}, C_{t_5}\}$, C_{t_1} and C_{t_5} have only a quarter of the observations for training. In contrast, a classifier that links all partial event observations has access to a bigger pool of training data.

Furthermore, classifying young events is generally harder than classifying matured events because often there is no clear separability of classes when events are young. If we continue with the previous example, obtaining the correct class boundary at t_1 is harder than at t_2, making it difficult for C_{t_1} to independently ascertain the class boundary. However, a linked classifier which sees all partial event observations can come up with a realistic class boundary for age t_1 because it sees the partial observations at ages t_2, t_3 and t_4, which helps it to form the class boundary at t_1. Thus, we expect linking partial event observations at different ages to aid early classification.

The Connected Classifier CC described in the next section links partial event observations of all ages to give a growing prediction.

6.3 CC: Connected Classifier

Let the standard classifier minimize the loss function given by $\mathscr{L}$, i.e.

$$\underset{\beta}{\arg\min} \frac{1}{N} \sum_{i=1}^{N} \mathscr{L}(\boldsymbol{x}_i, y_i; \beta),$$

where $\beta = (\beta_0, \beta_1, \ldots, \beta_l)$ and $(\boldsymbol{x}_i, y_i)$ are observations for $i \in \{1, \ldots, N\}$. Now consider a set of independent classifiers $\{C_{t_j}\}_{j=1}^{n}$ each trained and tested on partial observations of age t_j as denoted in Figure 15.

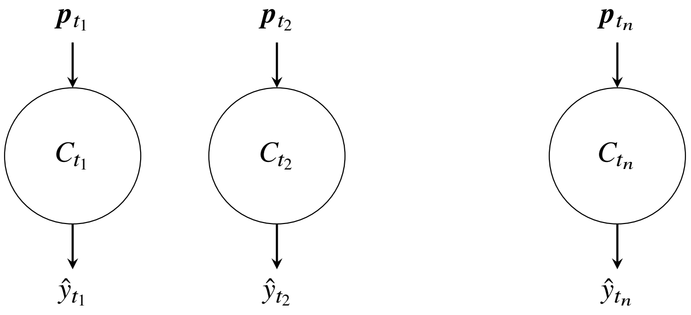


Figure 15: *n independent classifiers*

Note that each C_{t_j} minimizes the loss function

$$\frac{1}{N} \sum_{i=1}^{N} \mathscr{L}(\boldsymbol{p}_{t_j}^{i}, y_i; \tilde{\beta}_j),$$

with $\tilde{\beta}_j = (\tilde{\beta}_{j0}, \tilde{\beta}_{j1}, \tilde{\beta}_{j2}, \ldots, \tilde{\beta}_{jl})^T$. If we stack $\tilde{\beta}_j$ in rows we get a resulting matrix $\tilde{\beta}$ such that

$$\tilde{\beta} = \{\tilde{\beta}_{jk}\} = \left[\tilde{\beta}_1, \tilde{\beta}_2, \ldots, \tilde{\beta}_n\right]^T .$$

The matrix $\tilde{\beta}$ can also be obtained by minimizing the loss function

$$\frac{1}{nN} \sum_{j=1}^{n} \sum_{i=1}^{N} \mathscr{L}(\boldsymbol{p}_{t_j}^{i}, y_i; \tilde{\beta}), \tag{4}$$

as $\{\boldsymbol{p}_{t_j}^{i}\}_{i=1}^{N}$ for a fixed age t_j only affects $\tilde{\beta}_j$. Therefore the matrix $\tilde{\beta}$ can be computed row by row by minimizing $\sum_{i=1}^{N} \mathscr{L}\left(\boldsymbol{p}_{t_j}^{i}, y_i; \tilde{\beta}_j\right)$ for each j. Thus, we can write

$$\underset{\tilde{\beta}}{\arg\min} \sum_{j=1}^{n} \sum_{i=1}^{N} \mathscr{L}(\boldsymbol{p}_{t_j}^{i}, y_i; \tilde{\beta}) = \left[\underset{\tilde{\beta}_1}{\arg\min} \sum_{i=1}^{N} \mathscr{L}(\boldsymbol{p}_{t_1}^{i}, y_i; \tilde{\beta}_1), \ldots, \underset{\tilde{\beta}_n}{\arg\min} \sum_{i=1}^{N} \mathscr{L}(\boldsymbol{p}_{t_n}^{i}, y_i; \tilde{\beta}_n) \right]^T. \tag{5}$$

Thus, n independent classifiers $\{C_{t_j}\}_{j=1}^{n}$ minimize the loss function given in equation (4).

Having n independent classifiers $\left\{C_{t_j}\right\}_{j=1}^{n}$ is sub-optimal because the partial observations at ages t_j are not independent from those aged t_{j-1} and t_{j+1}. Thus the classifier C_{t_j} can benefit from the knowledge of $C_{t_{j+1}}$ and vice-versa. Furthermore, the partial observations of an event change little from t_j to t_{j+1}. Taking these into account, we modify the original loss function given in equation (4) by including an L_2 penalty term as follows:

$$\varphi(\tilde{\beta}, \lambda) = \frac{1}{nN} \sum_{j=1}^{n} \sum_{i=1}^{N} \mathscr{L}(\boldsymbol{p}_{t_j}^{i}, y_i; \tilde{\beta}) + \lambda \sum_{j=1}^{n-1} \left\|\tilde{\beta}_{j+1} - \tilde{\beta}_j\right\|^2 \tag{6}$$

for some $\lambda > 0$, where $\|\cdot\|$ denotes the L_2 norm. The constant λ is a parameter that can be specified. Recall that $\tilde{\beta}_j = \left(\tilde{\beta}_{j0}, \tilde{\beta}_{j1}, \tilde{\beta}_{j2}, \ldots, \tilde{\beta}_{jl}\right)$ relates to partial observations $\{\boldsymbol{p}_{t_j}^{i}\}_{i=1}^{N}$ for a fixed t_j, i.e. $\tilde{\beta}_{j0}$ is the coefficient of the intercept at age t_j and $\tilde{\beta}_{j1}$ is the coefficient of the first covariate at age t_j. Thus the penalty term

$$\left\|\tilde{\beta}_{j+1} - \tilde{\beta}_j\right\|^2 = \sum_{k=0}^{l} (\tilde{\beta}_{j+1,k} - \tilde{\beta}_{j,k})^2,$$

and each term $(\tilde{\beta}_{j+1,k} - \tilde{\beta}_{j,k})^2$ takes coefficients for the kth covariate at ages t_j and t_{j+1} and penalizes the difference, enforcing a certain smoothness in event-age. The connected classifier CC minimizes this loss function. As a result of the L_2 penalty term, the individual classifiers are connected to form a single classifier. Thus CC finds $\tilde{\beta}_*$ such that,

$$\tilde{\beta}_* = \arg\min_{\tilde{\beta}} \left(\frac{1}{nN} \sum_{i=1}^{N} \sum_{j=1}^{n} \mathscr{L}(\boldsymbol{p}^i_{t_j}, y_i; \tilde{\beta}) + \lambda \sum_{j=1}^{n-1} \left\| \tilde{\beta}_{j+1} - \tilde{\beta}_j \right\|^2 \right). \tag{7}$$

When $\lambda = 0$, CC is equivalent to n independent classifiers as the cost function is reduced to that of equation (4). When $\lambda \to \infty$ the coefficients $\tilde{\beta}_{j+1} \to \tilde{\beta}_j$ to minimize the cost function. We recall that $\tilde{\beta}_j$ is the vector of coefficients of C_{t_j}. This gives rise to the same classifier for all event ages. Thus, CC is a connected classifier that is in between a single classifier and n independent classifiers. A schematic diagram of CC is shown in Figure 16.

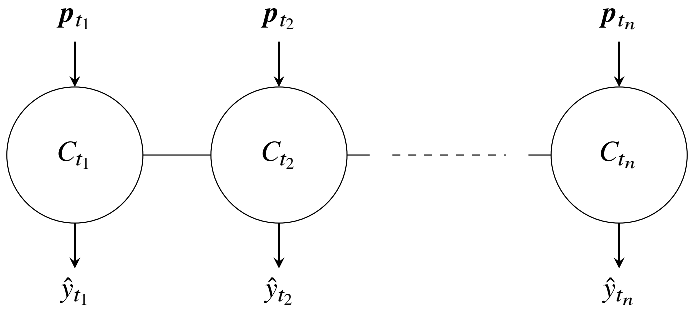


Figure 16: *A connected classifier.*

As any loss function $\mathscr{L}$ can be used in equation (7), CC is a general formulation. Our implementation of CC in *eventstream* (Kandanaarachchi 2018) uses logistic regression as the base classifier and the associated loss function. However, CC can be implemented with any loss function. For logistic regression (Friedman et al. 2001) the loss function $\mathscr{L}$ is given by

$$\mathscr{L}(\boldsymbol{p}^i_{t_j}, y_i; \tilde{\beta}) = -y_i\left([1 \;\; (\boldsymbol{p}^i_{t_j})^T]\tilde{\beta}_j\right) + \log\left(1 + \exp\left\{[1 \;\; (\boldsymbol{p}^i_{t_j})^T]\tilde{\beta}_j\right\}\right). \tag{8}$$

Here the vector $[1 \;\; (\boldsymbol{p}^i_{t_j})^T]$ denotes the concatenation of the vector $\boldsymbol{p}^i_{t_j}$ with the constant 1 to account for the intercept.

6.4 Comparison with unlinked classifiers

We refer to CC with logistic regression as CC-Log in the following sections and compare its performance with two configurations of logistic regression classifiers. The first configuration comprises a single classifier, which is trained on all partial observations and their ages $(t_i, \mathbf{p}_{t_i})$ as shown in Figure 17. We refer to this configuration as 1-Log in the following sections. The second configuration comprises n independent classifiers as shown in Figure 15. We refer to this configuration of n independent classifiers as n-Log. The n-Log classifier comprises of $\{C_{t_j}\}_{j=1}^n$, where each C_{t_j} is trained on partial observations of age t_j, i.e. $\{\boldsymbol{p}^i_{t_j}\}_{i=1}^N$.

The difference between 1-Log and n-Log is the number of independent classifiers. The only difference between n-Log and CC-Log is the connections between the independent classifiers, brought about by the L_2 penalty term $\sum_{j=1}^{n-1} \left\|\tilde{\beta}_{j+1} - \tilde{\beta}_j\right\|^2$. We compare CC-Log with 1-Log and n-Log because we want to understand whether linking independent classifiers benefits early classification.

7 Event classification results

We explore three groups of datasets in this Section: synthetic, fibre optic and NO_2 data. For each application, we use CC-Log, 1-Log and n-Log to classify events extracted by CPDBEE algorithm. For all three applications we

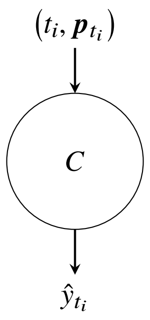
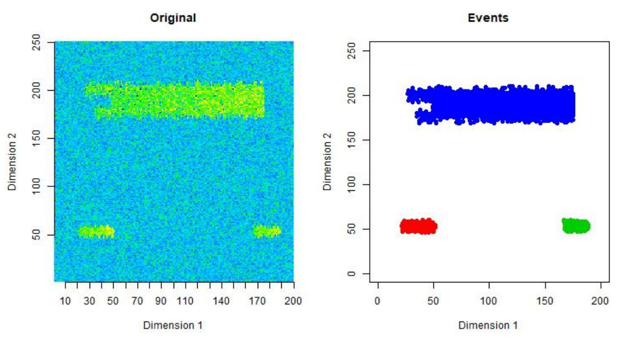
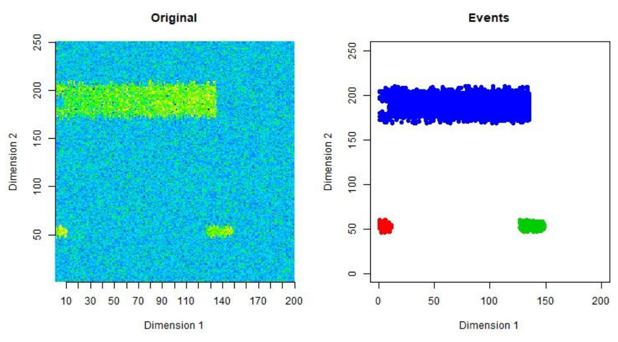


Figure 17: *A Single classifier*



(a)



(b)

Figure 18: *Two windows of data and extracted events.*

use $\lambda = 0.05$ for consistency. The R code applicable to this section is available in the supplementary material. The data is either included in the R package *eventstream* or can be generated using its functionality.

7.1 Synthetic data

We generate a data stream of dimension 3500×250, of which 80% (2800×250) is used for training and the remaining 20% for testing. We use a moving window of dimension 200×250 which moves by a step of 8×250. For each window we extract events using CPDBEE algorithm as shown in Figure 18. As class A events can have a maximum age of 30, we use 4 event ages for the classification tasks at $t = 8, 16, 24$ and 32 time units, i.e. event features are calculated at these ages. For synthetic data classification, we do not use features which were motivated from the fibre optic example, i.e. we do not use features 11 and 12 from the list in Section 6.1, for computational efficiency. Furthermore, the centroid is not used in any classification task.

To obtain unbiased estimates, we repeat this experiment 5 times using different seeds for data generation. We measure the classification accuracy, which is defined as $1-$ misclassification error. Table 2 gives the mean and standard deviation of test set classification accuracy for the 3 classifiers, which shows that CC-Log surpasses 1-Log and n-Log classifiers. Also we see that all 3 classifiers improve their average accuracy levels with the age of the events with the exception of n-Log at t_4.

Accuracy Measure	Classifier	Accuracy				Standard deviation			
		t_1	t_2	t_3	t_4	t_1	t_2	t_3	t_4
Classification Accuracy	CC-Log	0.79	0.88	0.91	0.91	0.13	0.11	0.10	0.08
	1-Log	0.71	0.85	0.88	0.87	0.20	0.13	0.13	0.11
	n-Log	0.76	0.84	0.89	0.51	0.12	0.11	0.07	0.28

Table 2: *Mean and standard deviation of classification accuracy over 5 repetitions.*

7.1.1 Significance results

To determine if there is a significant difference between the three classifiers, we conduct a Friedman test on classification accuracy results. The Friedman test on classification accuracy results gave a p-value of 0.0156, showing that the classifiers are different at 5% level of significance. To ascertain which methods perform better we conduct a Nemenyi test on classification accuracy results.

Figure 19 shows the resulting Nemenyi plot of ranks with a 90% level of confidence. Lower rank values indicate better performing methods. Blue coloured boxes indicate methods which do not differ significantly from each other. From Figure 19 we see that CC-Log is best suited for this data followed by 1-Log and n-Log, with no significant difference between the two latter methods.

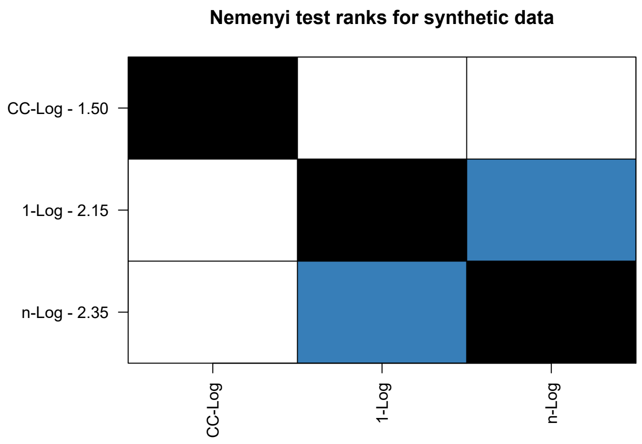


Figure 19: *Nemenyi plot for synthetic data using classification accuracy results.*

7.2 Fibre optic cable data results

The fibre optic dataset is a 379×587 matrix as shown in Figure 6. We use a moving window model with a window size 40×587 and a step size 10×587 to extract events and compute features. For each window we extract events using CPDBEE and use CC-Log, 1-Log and n-Log to classify them. We use event ages $t = 10, 20, 30$ and 40, because the maximum event-age is 40 time units.

As the fibre optic dataset has 4 class A events, we use 4-fold cross validation. The events extracted from the data stream is divided into four folds with each fold containing one class A event resembling Figure 20.

This modified form of cross validation is commonly used for time dependent observations (Leigh et al. 2019) as in a streaming data scenario. The classifiers CC-Log, 1-Log and n-Log are trained on events comprising of 3 training folds, and tested on the remaining fold.

As this dataset has a smaller number of class A events compared to class B events, we report additional accuracy measures that are designed for imbalanced datasets. We compute the positive predictive value (PPV) and the negative predictive value (NPV) and area under the receiver operator characteristic curve (AUC). We give the definitions of these metrics below:

$$\text{Positive predictive value (PPV)} = \frac{\text{Number of true positives}}{\text{Number of predicted positives}},$$

$$\text{Negative predictive value (NPV)} = \frac{\text{Number of true negatives}}{\text{Number of predicted negatives}}.$$

The number of predicted positives in PPV is the sum of true positives and false positives, and the number of predicted negatives in NPV is the sum of true negatives and the false negatives. Considering PPV and NPV together gives a two-sided accuracy measure. For example, a classifier that predicts all observations as negative except for one correct positive observation achieves a PPV of 100% but a small NPV. The combination of PPV and NPV gives the overall accuracy of the model.

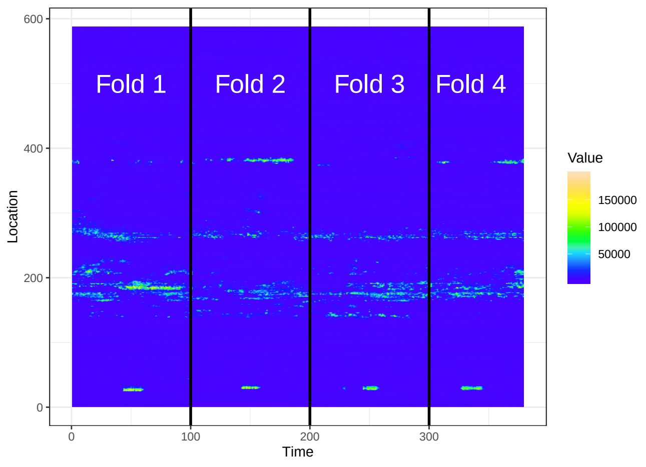


Figure 20: *Data chunks for 4-fold cross validation.*

In contrast, AUC is a single measure that captures the effectiveness of a classifier. The receiver operator characteristic (ROC) curve is a plot of the true positive rate against the false positive rate for different classification thresholds. The area under the curve (AUC) provides a measure of discrimination between positive and negative classes. The AUC does not depend on the classification threshold as it is an aggregate measure. An AUC closer to 1 is reflective of a good model, while a random predictor will give an AUC closer to 0.5. The AUC can be interpreted as the probability that a positive observation is ranked higher than a negative observation.

Accuracy Measure	Classifier	Mean				Standard deviation			
		t_1	t_2	t_3	t_4	t_1	t_2	t_3	t_4
PPV	CC-Log	1.00	1.00	1.00	0.95	0.0	0.0	0.00	0.1
	1-Log	0.81	0.73	0.73	0.73	0.21	0.32	0.32	0.32
	n-Log	0.90	1.00	0.93	1.00	0.20	0.0	0.12	0.0
NPV	CC-Log	0.92	0.94	0.95	0.95	0.03	0.03	0.03	0.03
	1-Log	0.95	0.96	0.95	0.93	0.01	0.02	0.05	0.06
	n-Log	0.97	0.92	0.96	0.96	0.02	0.03	0.02	0.03
AUC	CC-Log	0.96	0.97	0.97	0.95	0.01	0.01	0.01	0.06
	1-Log	0.88	0.84	0.84	0.83	0.09	0.15	0.15	0.13
	n-Log	0.93	0.96	0.94	0.98	0.09	0.01	0.05	0.01

Table 3: *Mean and standard deviation of PPV, NPV and AUC* (%) *over* 4 *folds.*

Table 3 gives the average PPV, NPV and AUC values with their standard deviations over the 4-folds for the fibre-optic data stream. For PPV and NPV, we use a probability threshold of 0.5; i.e. if the output probability is greater than 0.5, it is deemed class A, and class B otherwise.

7.2.1 Significance results

Friedman tests on AUC, PPV and NPV results gave p-values of 0.0048, 0.0045 and 0.7583 respectively. This shows that while AUC and PPV results differ significantly across classifiers, NPV results are similar. As such, we conduct Nemenyi tests on AUC and PPV values. Figure 21 shows Nemenyi test ranks for PPV and AUC, with lower ranks denoting better performance.

From Figure 21a we see that for PPV CC-Log outperforms n-Log and n-Log outperforms 1-Log with a 95% level of confidence. For AUC results, both CC-Log and n-Log significantly outperform 1-Log. Even though CC-Log outperforms n-Log, the two classifiers are not significantly different from each other as depicted by the blue squares in Figure 21b.

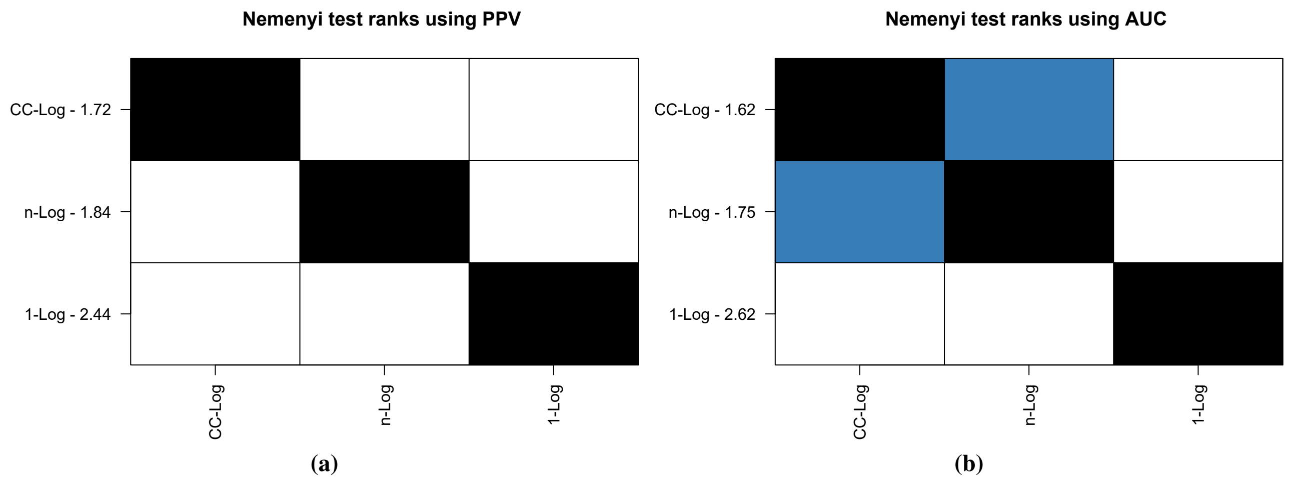


Figure 21: *Nemenyi plots for fibre optic cable data using PPV and AUC results.*

As NPV results across the classifiers are not significantly different, we turn our attention to PPV and AUC results in Table 3. We see that CC-Log performs better at earlier event ages compared to *n*-Log. Noting that the only difference between *n*-Log and CC-Log is the connections between the independent classifiers, this demonstrates the importance of the connections for early event classification. That is, the knowledge of "older" events aids classification of "younger" events.

7.3 Nitrogen dioxide monitoring

We consider monthly NO_2 data from March to June for a 10 year period from 2010 to 2019. These are three dimensional datasets with two spatial and one time dimension. First we extract events using CPDBEE for each year separately. Then we perform 10-fold cross validation by training CC-Log, 1-Log and *n*-Log classifiers on event data of 9 years and testing it on the remaining year's data.

The extracted events are three dimensional clusters of high NO_2 levels spanning space and time. Events are extracted from a data stream of $4 \times 180 \times 360$ array, where each 180×360 matrix corresponds to the NO_2 levels of a given month. We use CPDBEE parameters $\alpha = 0.97$, $\epsilon = 2$ and minPts = 20. Figure 22 shows two dimensional cross sections of the three dimensional NO_2 clusters in March and June 2018.

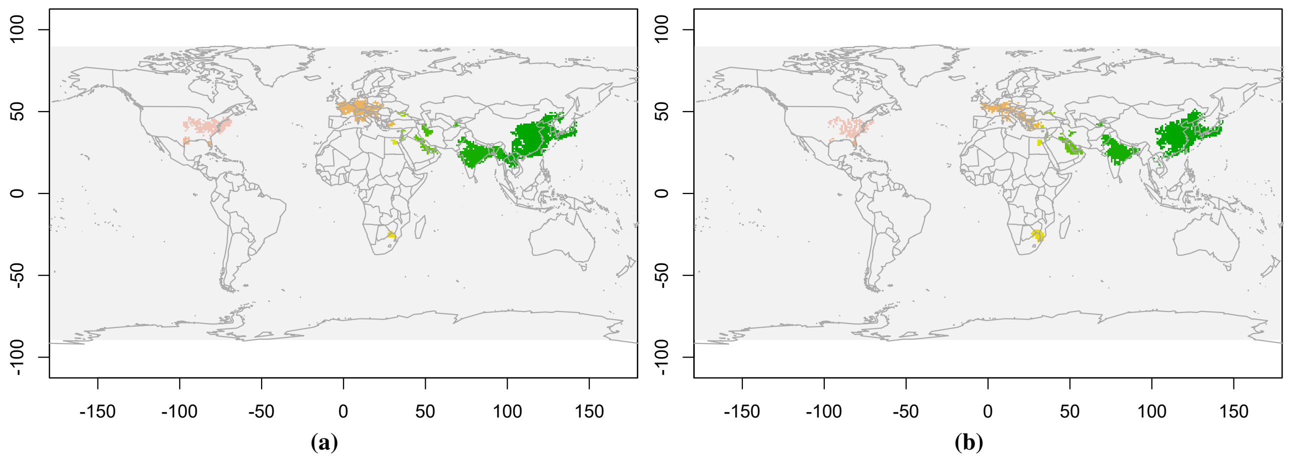


Figure 22: *Events extracted from NO_2 data from March to June 2018. Figures 22a and 22b show two dimensional cross sections of the three dimensional NO_2 clusters in March and June 2018. Colours do not reflect NO_2 levels. Each event is depicted by a single colour.*

For each 3-dimensional event we compute features 1–10 and 14 from the list of features in Section 6.1. These features are chosen for ease of computation. NO_2 clusters which have grown in average intensity during this time period are assigned the class label 1 and others 0. As some NO_2 clusters only start in April we designate the value in April as the starting value for class label computation. Thus, the task is to detect if NO_2 clusters grow in intensity as soon as possible.

Table 4 gives the 10-fold cross validation test results on NO_2 clusters. We see that CC-Log achieves better accuracy results compared to n-Log and 1-Log, with the only exception that 1-Log achieves slightly better results at t_4.

Accuracy Measure	Classifier	Mean				Standard deviation			
		t_1	t_2	t_3	t_4	t_1	t_2	t_3	t_4
Classification Accuracy	CC-Log	0.84	0.81	0.90	0.88	0.08	0.12	0.07	0.07
	1-Log	0.80	0.80	0.88	0.89	0.09	0.11	0.07	0.07
	n-Log	0.80	0.80	0.87	0.83	0.09	0.12	0.06	0.11

Table 4: *10-Fold Cross validation accuracy comparison on NO_2 clusters.*

7.3.1 Significance results

Similar to the previous applications, we perform a Friedman test on the classification accuracy results. The Friedman test gave a p-value of 0.01752, showing that there is a significant difference between the classifiers.

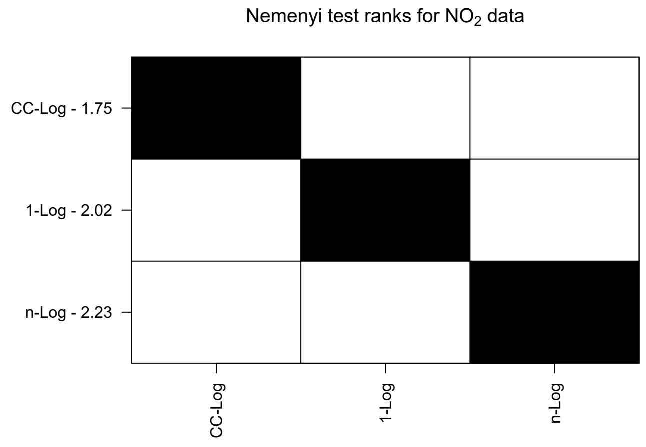


Figure 23: *Nemenyi plot for NO_2 data using classification accuracy results.*

Figure 23 shows the resulting Nemenyi plot, which shows that CC-Log outperforms 1-Log and n-Log with a 5% level of significance. The success of CC-Log demonstrates the benefit of linking classifiers that are trained on similar but slightly different data, on early event classification.

8 Conclusions

This paper has proposed a framework for event extraction and early event classification in contiguous spatio-temporal data streams. We proposed an event detection and extraction algorithm as well as an early event classification algorithm. We tested our event detection and classification framework using 3 applications, one synthetic and two real.

The event extraction algorithm CPDBEE uses change point detection and clustering techniques to detect and extract events. We compared CPDBEE with Kuldorff's Scan Statistic and achieved better results for all three applications in a much shorter time period.

The early event classification algorithm comprises of a set of base classifiers connected using an L_2 penalty term, inducing a certain level of smoothness in event age. We compared the connected classifier CC-Log, with two configurations of unlinked classifiers 1-Log and n-Log and achieved better results for all three applications. Furthermore, for all three applications CC-Log achieved better results for early event ages. As the only difference between n-Log and CC-Log was the connections between the base classifiers, this reveals that classification of early events benefits from knowledge of more mature events. Future directions for this research include extending CPDBEE for non-contiguous spatio-temporal data as well as extending CC to use other base classifiers such as decision trees.

9 Supplementary material

R-package eventstream: This package contains CPDBEE algorithm for event extraction, the connected classifier CC-Log for early event classification, functions for synthetic data generation, the fibre-optic data stream and NO_2 data. It is available from GitHub at `https://github.com/sevvandi/eventstream`.

Scripts: The file `Supp_Mat_CPDBEE` contains the code used in Section 4. There are three files containing the R code used in Section 7. The files `Supp_Mat_1.R`, `Supp_Mat_2.R` and `Supp_Mat_3.R` contain the code applicable for synthetic data, fibre optic data and NO_2 data respectively.

Other R-packages: We have used the following R-packages either in this paper or within the package *eventstream*: *changepoint* (Killick & Eckley 2014), *abind* (Plate & Heiberger 2016), *AtmRay* (Anderson 2013), *pROC* (Robin et al. 2011), *ggplot2* (Wickham 2016), *raster* (Hijmans 2018), *maps* (Becker et al. 2018), *tensorA* (van den Boogaart 2018), *glmnet* (Friedman et al. 2010), *dbscan* (Hahsler & Piekenbrock 2018) and *MASS* (Venables & Ripley 2002).

Acknowledgements

Funding was provided by the Australian Research Council through the Linkage Project LP160101885.

References

Aminikhanghahi, S. & Cook, D. J. (2017), 'A survey of methods for time series change point detection', *Knowledge and Information Systems* **51**(2), 339–367.

Anderson, J. (2013), *AtmRay: Acoustic Traveltime Calculations for 1-D Atmospheric Models*. R package version 1.31.
URL: *https://CRAN.R-project.org/package=AtmRay*

Atefeh, F. & Khreich, W. (2015), 'A survey of techniques for event detection in Twitter', *Computational Intelligence* **31**(1), 132–164.

Auger, I. E. & Lawrence, C. E. (1989), 'Algorithms for the optimal identification of segment neighborhoods', *Bulletin of Mathematical Biology* **51**(1), 39–54.

Bai, J. & Perron, P. (1998), 'Estimating and testing linear models with multiple structural changes', *Econometrica* **66**(1), 47–78.

Bardwell, L., Fearnhead, P., Eckley, I. A., Smith, S. & Spott, M. (2019), 'Most recent changepoint detection in panel data', *Technometrics* **61**(1), 88–98.

Becker, R. A., Wilks, A. R., Brownrigg, R., Minka, T. P. & Deckmyn., A. (2018), *maps: Draw Geographical Maps*. R package version 3.3.0.
URL: *https://CRAN.R-project.org/package=maps*

Ditzler, G., Roveri, M., Alippi, C. & Polikar, R. (2015), 'Learning in nonstationary environments: A survey', *IEEE Computational Intelligence Magazine* **10**(4), 12–25.

Earle, P., Bowden, D. & Guy, M. (2012), 'Twitter earthquake detection: earthquake monitoring in a social world', *Annals of Geophysics* **54**(6).

Ester, M., Kriegel, H.-P., Sander, J., Xu, X. et al. (1996), A density-based algorithm for discovering clusters in large spatial databases with noise., *in* 'KDD', Vol. 96, pp. 226–231.

Friedman, J., Hastie, T. & Tibshirani, R. (2001), *The elements of statistical learning*, Vol. 1, Springer Series in Statistics New York, NY, USA.

Friedman, J., Hastie, T. & Tibshirani, R. (2010), 'Regularization paths for generalized linear models via coordinate descent', *Journal of Statistical Software* **33**(1), 1–22.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M. & Bouchachia, A. (2014), 'A survey on concept drift adaptation', *ACM Computing Surveys (CSUR)* **46**(4), 44.

Geddes, J. A., Martin, R. V., Boys, B. L. & van Donkelaar, A. (2015), 'Long-term trends worldwide in ambient NO_2 concentrations inferred from satellite observations', *Environmental Health Perspectives* **124**(3), 281–289.

Griffiths, B. (1995), Developments in and applications of fibre optic intrusion detection sensors, *in* 'Security Technology, 1995. Proceedings. Institute of Electrical and Electronics Engineers 29th Annual 1995 International Carnahan Conference on', IEEE, pp. 325–330.

Guralnik, V. & Srivastava, J. (1999), Event detection from time series data, *in* 'Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', ACM, pp. 33–42.

Hahsler, M. & Piekenbrock, M. (2018), *dbscan: Density Based Clustering of Applications with Noise (DBSCAN) and Related Algorithms*. R package version 1.1-2.
URL: *https://CRAN.R-project.org/package=dbscan*

Hamunyela, E., Verbesselt, J. & Herold, M. (2016), 'Using spatial context to improve early detection of deforestation from Landsat time series', *Remote Sensing of Environment* **172**, 126–138.

Hastie, T. & Tibshirani, R. (1993), 'Varying-coefficient models', *Journal of the Royal Statistical Society. Series B (Methodological)* **55**(4), 757–796.

Hijmans, R. J. (2018), *raster: Geographic Data Analysis and Modeling*. R package version 2.8-4.
URL: *https://CRAN.R-project.org/package=raster*

Hulten, G., Spencer, L. & Domingos, P. (2001), Mining time-changing data streams, *in* 'Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', ACM, pp. 97–106.

Jiang, Q. & Sui, Q. (2009), Technological study on distributed fiber sensor monitoring of high voltage power cable in seafloor, *in* '2009 IEEE International Conference on Automation and Logistics', pp. 1154–1157.

Kandanaarachchi, S. (2018), *eventstream: An implementation of streaming events and their classification*. R package version 0.0.9000.
URL: *https://github.com/sevvandi/eventstream*

Ke, Y., Sukthankar, R. & Hebert, M. (2007), Event detection in crowded videos, *in* '2007 IEEE 11th International Conference on Computer Vision', pp. 1–8.

Killick, R. & Eckley, I. (2014), 'changepoint: An R package for changepoint analysis', *Journal of Statistical Software* **58**(3), 1–19.

Killick, R., Fearnhead, P. & Eckley, I. A. (2012), 'Optimal detection of changepoints with a linear computational cost', *Journal of the American Statistical Association* **107**(500), 1590–1598.

Kim, A. Y. & Wakefield, J. (2018), *SpatialEpi: Methods and Data for Spatial Epidemiology*. R package version 1.2.3.
URL: *https://CRAN.R-project.org/package=SpatialEpi*

Kulldorff, M. (1997), 'A spatial scan statistic', *Communications in Statistics-Theory and Methods* **26**(6), 1481–1496.

Kulldorff, M. (2001), 'Prospective time periodic geographical disease surveillance using a scan statistic', *Journal of the Royal Statistical Society: Series A (Statistics in Society)* **164**(1), 61–72.

Leigh, C., Kandanaarachchi, S., McGree, J. M., Hyndman, R. J., Alsibai, O., Mengersen, K. & Peterson, E. E. (2019), 'Predicting sediment and nutrient concentrations in rivers using high frequency water quality surrogates', *PLOS ONE* . Accepted.
URL: *https://www.biorxiv.org/content/10.1101/599712v1*

Levelt, P. F., van den Oord, G. H., Dobber, M. R., Malkki, A., Visser, H., de Vries, J., Stammes, P., Lundell, J. O. & Saari, H. (2006), 'The Ozone monitoring instrument', *IEEE Transactions on Geoscience and Remote Sensing* **44**(5), 1093–1101.

Li, H.-N., Li, D.-S. & Song, G.-B. (2004), 'Recent applications of fiber optic sensors to health monitoring in civil engineering', *Engineering Structures* **26**(11), 1647–1657.

Li, R., Lei, K. H., Khadiwala, R. & Chang, K. C. (2012), TEDAS: A twitter-based event detection and analysis system, *in* '2012 IEEE 28th International Conference on Data Engineering', pp. 1273–1276.

Medioni, G., Cohen, I., Brémond, F., Hongeng, S. & Nevatia, R. (2001), 'Event detection and analysis from video streams', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**(8), 873–889.

NASA Earth Observations (2004), `https://neo.sci.gsfc.nasa.gov/view.php?datasetId=AURA_NO2_M`.

Neill, D. B. (2012), 'Fast subset scan for spatial pattern detection', *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **74**(2), 337–360.

Niklès, M., Vogel, B. H., Briffod, F., Grosswig, S., Sauser, F., Luebbecke, S., Bals, A. & Pfeiffer, T. (2004), Leakage detection using fiber optics distributed temperature monitoring, *in* 'Smart Structures and Materials 2004: Smart Sensor Technology and Measurement Systems', Vol. 5384, International Society for Optics and Photonics, pp. 18–26.

Plate, T. & Heiberger, R. (2016), *abind: Combine Multidimensional Arrays*. R package version 1.4-5.
URL: *https://CRAN.R-project.org/package=abind*

Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J.-C. & Müller, M. (2011), 'pROC: an open-source package for R and S+ to analyze and compare ROC curves', *BMC Bioinformatics* **12**, 77.

Ruocco, M. & Ramampiaro, H. (2014), 'A scalable algorithm for extraction and clustering of event-related pictures', *Multimedia Tools and Applications* **70**(1), 55–88.

Sadia, F., Kandanaarachchi, S., Smith-Miles, K. & Keith, J. (2019), Event detection in spatio-temporal data using a bayesian segmented arma change-point model, Working paper.

Scott, A. J. & Knott, M. (1974), 'A cluster analysis method for grouping means in the analysis of variance', *Biometrics* **30**(3), 507–512.

Sen, A. & Srivastava, M. S. (1975), 'On tests for detecting change in mean', *The Annals of Statistics* **3**(1), 98–108.

Suthaharan, S. (2014), 'Big data classification: Problems and challenges in network intrusion prediction with machine learning', *ACM SIGMETRICS Performance Evaluation Review* **41**(4), 70–73.

van den Boogaart, K. G. (2018), *tensorA: Advanced Tensor Arithmetic with Named Indices*. R package version 0.36.1.
URL: *https://CRAN.R-project.org/package=tensorA*

Venables, W. N. & Ripley, B. D. (2002), *Modern Applied Statistics with S*, 4th edn, Springer, New York. ISBN 0-387-95457-0.
URL: *http://www.stats.ox.ac.uk/pub/MASS4*

Verbesselt, J., Hyndman, R., Newnham, G. & Culvenor, D. (2010), 'Detecting trend and seasonal changes in satellite image time series', *Remote Sensing of Environment* **114**(1), 106–115.

Verbesselt, J., Hyndman, R., Zeileis, A. & Culvenor, D. (2010), 'Phenological change detection while accounting for abrupt and gradual trends in satellite image time series', *Remote Sensing of Environment* **114**(12), 2970–2980.

Verbesselt, J., Zeileis, A. & Herold, M. (2012), 'Near real-time disturbance detection using satellite image time series', *Remote Sensing of Environment* **123**, 98–108.

Weng, J. & Lee, B.-S. (2011), Event detection in twitter, *in* 'Fifth International AAAI Conference on Weblogs and Social Media'.
URL: *https://www.aaai.org/ocs/index.php/ICWSM/ICWSM11/paper/viewPaper/2767*

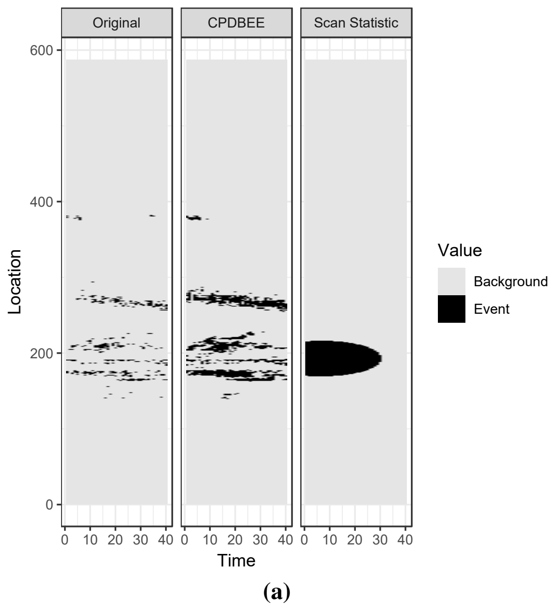
WHO air pollution (2019).
URL: *https://www.who.int/airpollution/en/*

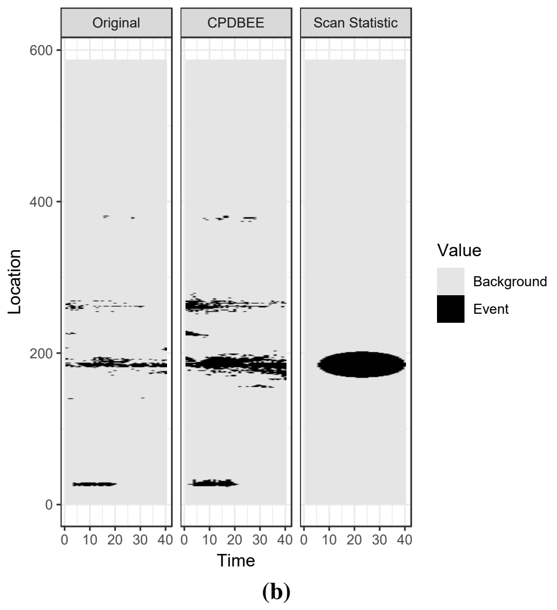
Wickham, H. (2016), *ggplot2: Elegant Graphics for Data Analysis*, Springer-Verlag New York.
URL: *http://ggplot2.org*

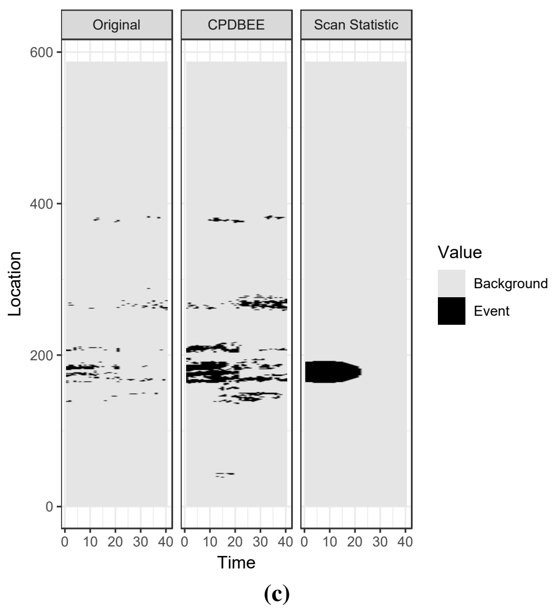
Zhu, Z. & Woodcock, C. E. (2014), 'Continuous change detection and classification of land cover using all available Landsat data', *Remote Sensing of Environment* **144**, 152–171.

A Appendix

A.1 Event extraction results for fibre optic cable data







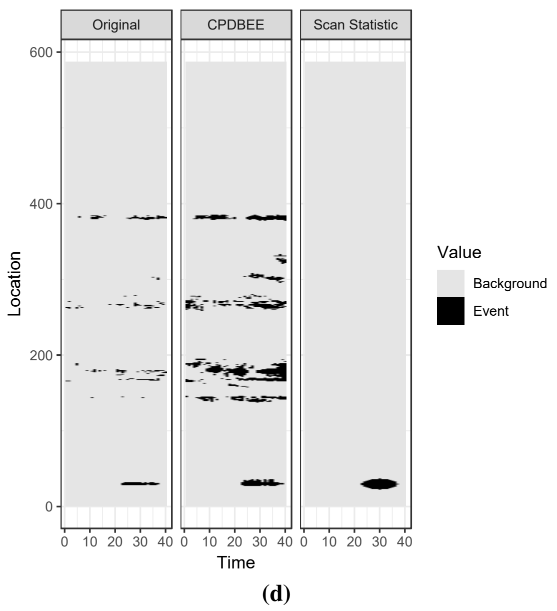
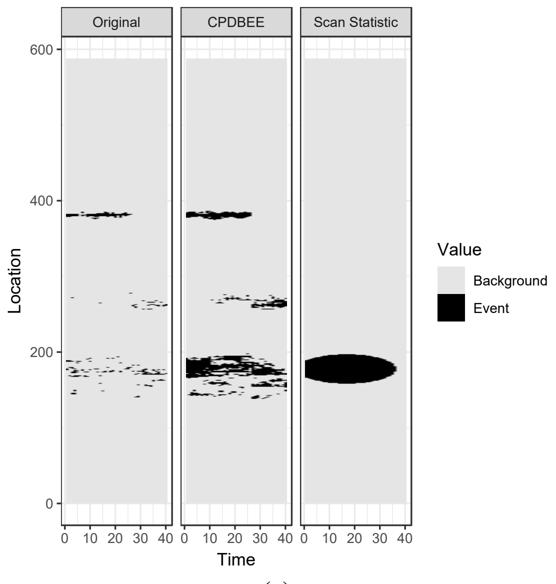
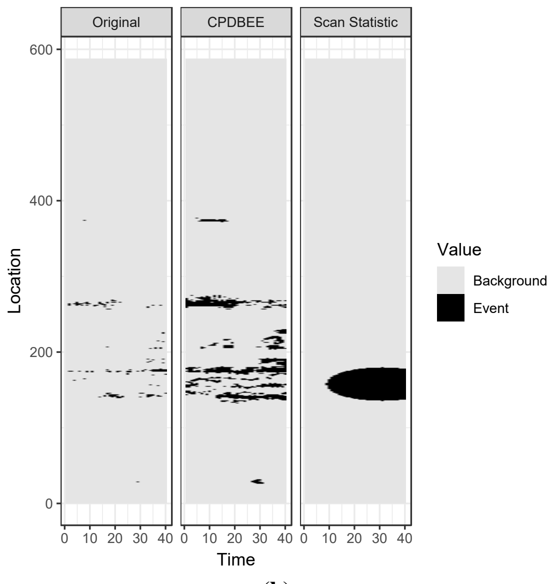


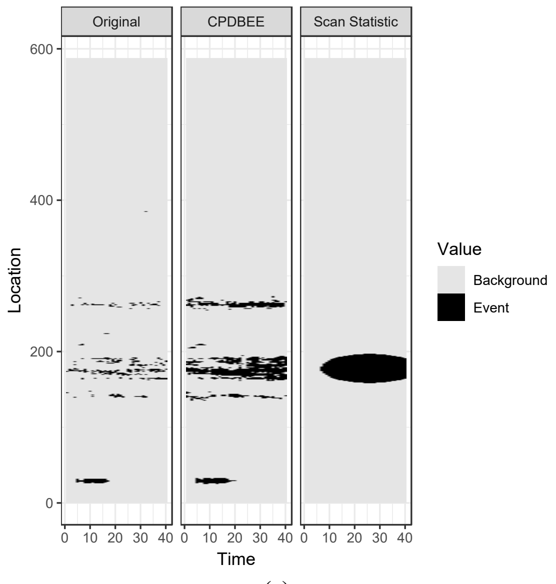
Figure 24: *Event extraction comparison for fibre optic data windows 1–4.*



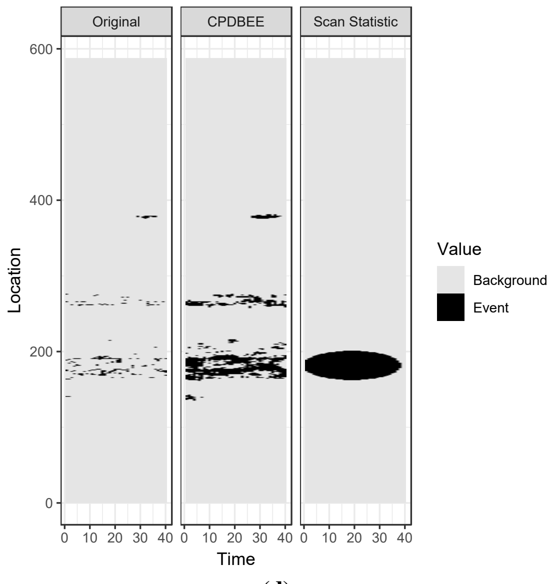
(a)



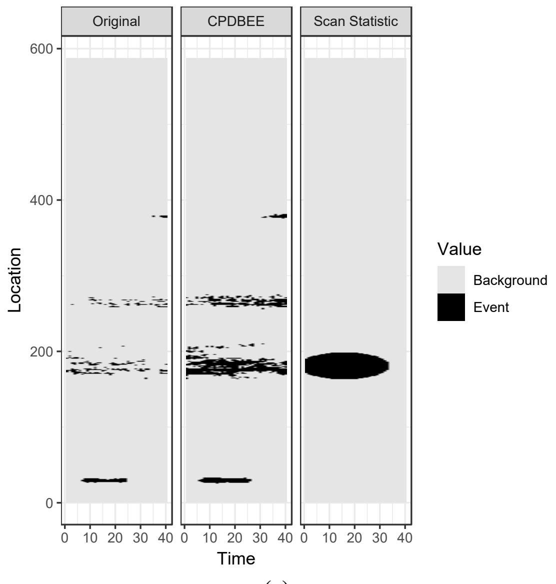
(b)



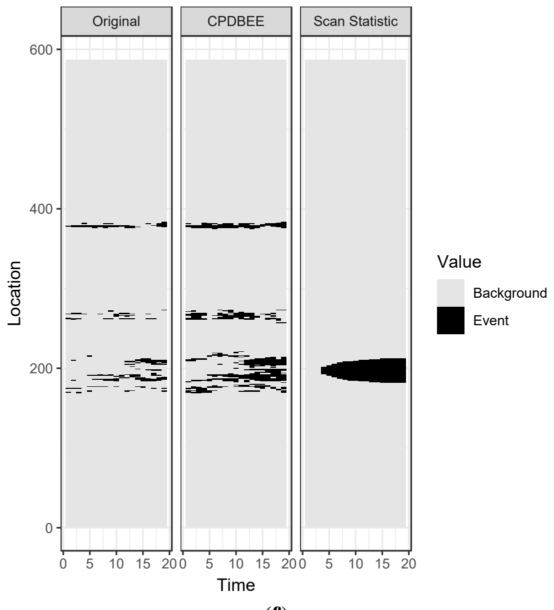
(c)



(d)



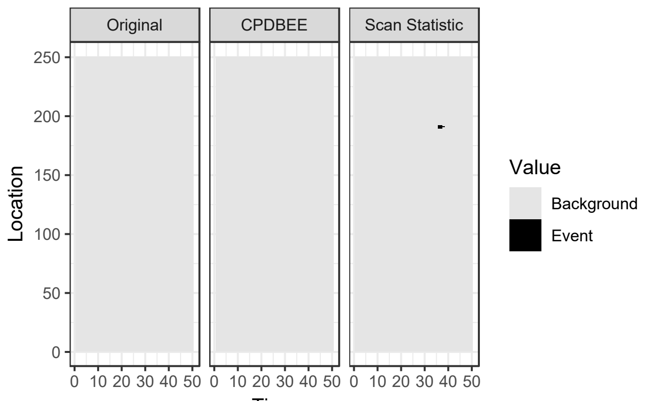
(e)



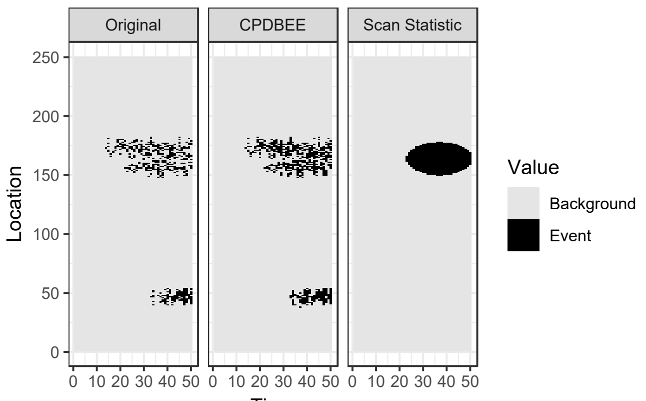
(f)

Figure 25: *Event extraction comparison for fibre optic data windows 5–10.*

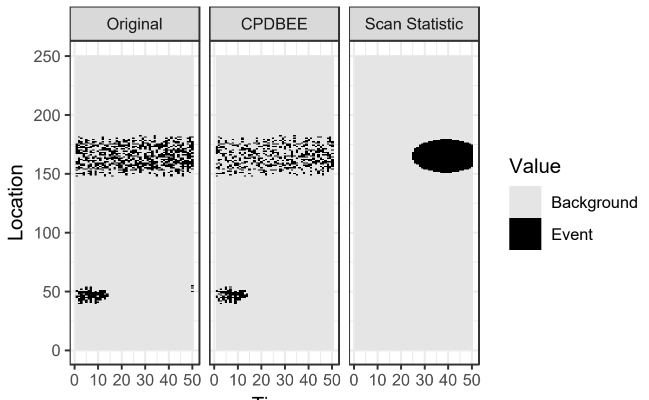
A.2 Event extraction results for synthetic data



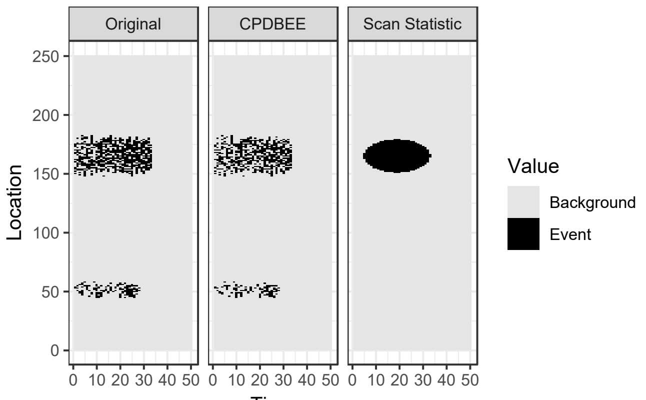
(a)



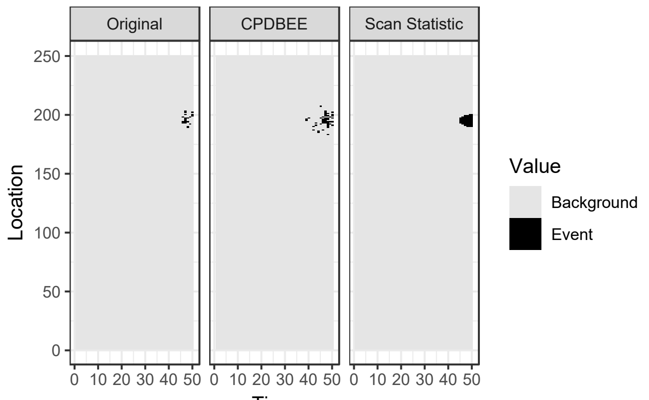
(b)



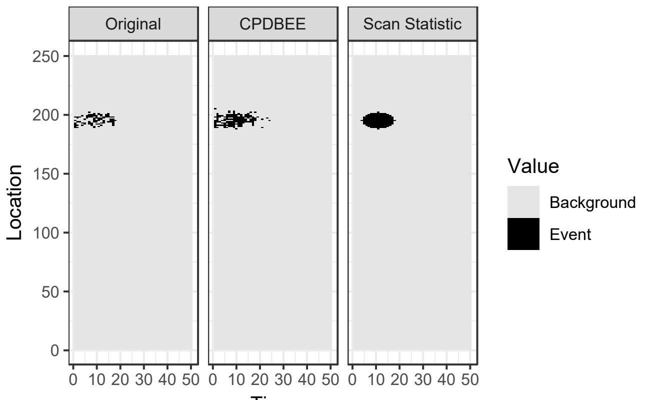
(c)



(d)



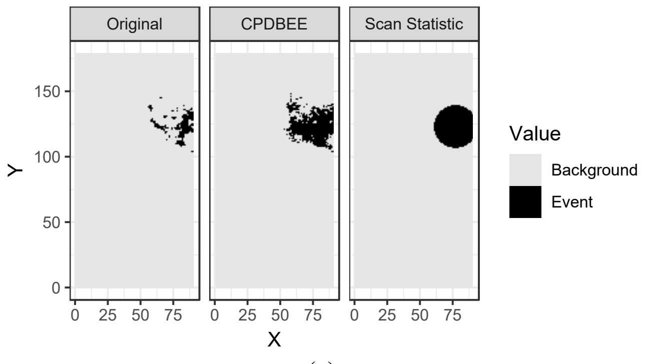
(e)



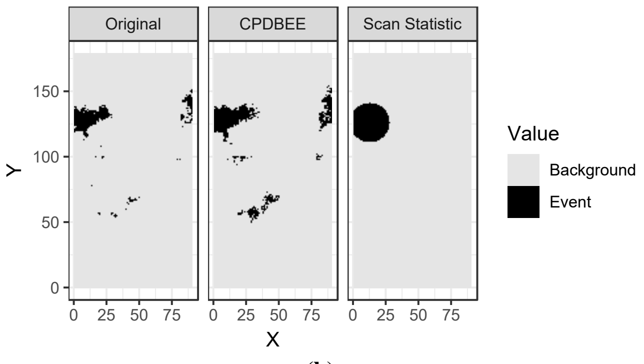
(f)

Figure 26: *Event extraction comparison for synthetic data.*

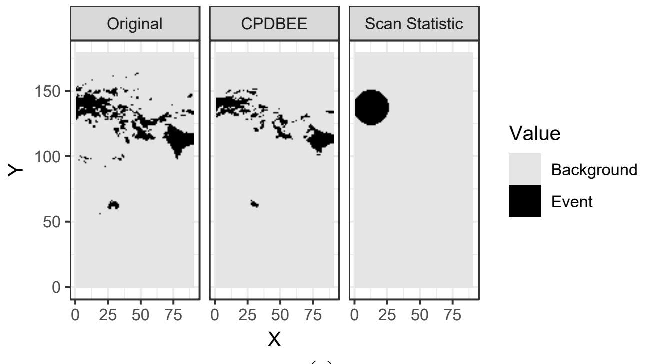
A.3 Event extraction results for NO_2 data



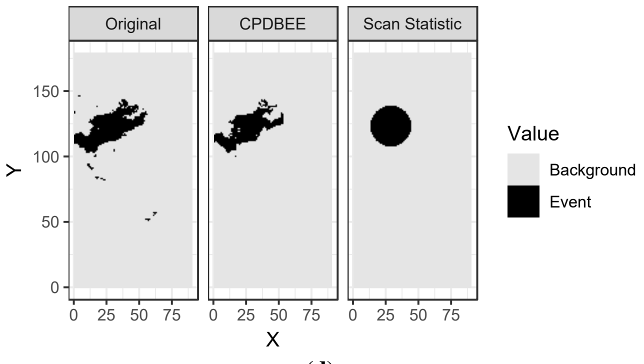
(a)



(b)



(c)



(d)

Figure 27: *Event extraction comparison for NO_2 data.*