Early classification of spatio-temporal events using time-varying models

Sevvandi Kandanaarachchi, Rob J Hyndman, Kate Smith-Miles

19 December 2018

Abstract

This paper investigates early event classification in spatio-temporal data streams. We propose a framework for early classification that considers the relationship between the features of an event and its age. The framework incorporates an event extraction algorithm as well as two early event classification algorithms, which use a series of logistic regression classifiers with penalty terms and state space models. We apply this framework to synthetic and real world problems and demonstrate its reliability and broad applicability. The algorithms 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 characterized by measaurable properties or features, including the "age" of the event – the difference between the current time and the start time of the event – with other event properties varying according to its age.

Given that interesting events in data streams arise in a variety of applications and are studied from the perspective of different disciplines, the field suffers from a plenthora of discipline-specific techniques that have been developed, leading to little consensus on associated terminology and methodology across disciplines. On the one hand, similar terms are used to describe different phenomena — this is called the synonym problem [1]. On the other hand, different terms are used to describe similar phenomena — known as the homonym problem [1]. This makes it difficult to objectively compare results and methods across disciplines. For example, the term "event detection" is used in video surveillance and video applications [2, 3, 4], social media [5, 6, 7], broadcast news stories [8, 9] and wireless sensor networks [10, 11]. However, it is difficult to apply an event detection method used in one application to another domain as many contextual properties depend on the problem domain, resulting in a proliferation of custom-made algorithms. As a result, event detection has become a discipline-specific study with little overlap between different research problems. Similarly spatio-temporal event detection, which is a popular sub-topic, has emerged from different applications such as river sensor networks [11], traffic data [12], wireless sensor networks [13], social networks [14] and epidemiology [15]. While spatio-temporal event detection has been studied extensively, early classification of spatio-temporal events has not received much attention. Mostly standard classification techniques are used on detected events where the focus has been on event detection [16].

For events with age-varying behaviour, standard classification techniques may not be suitable, especially if early detection is important. This is because the event features may change as the event progresses, such that an accurate feature vector can only be computed after the event is finished. For applications where early classification is important, this is not satisfactory as one cannot wait until the event finishes before classifying it. While the event is still progressing, we have access to incomplete or partial information, giving rise to a incomplete or premature feature vector. Our focus is on developing new methods to reliably classify events using this premature feature vector.

Fundamentally, this problem can be tackled by embedding age-varying coefficients in a learned model [17]. 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. 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 [18, 19, 20, 21] or non-stationarity of data streams [22, 23, 24] is different from age-varying events. As we will show in Section 6.1, age-varying events can occur in stationary data streams or data streams without any concept drift. This is because age-varying behaviour comprises change within the event as opposed to a change within the data stream. Even though non-stationarity and time-varying models have been studied in different contexts [25, 26, 27], they have not been explored in the context of premature event classification to the best of our knowledge.

We explore early event classification using two approaches that are based on different principles: 1) logistic regression; and 2) state-space models. These two approaches complement each other as logistic regression by itself is a static model more suited for stationary distributions, and a state space model coupled with a Kalman filter can update easily, which is advantageous for non-stationary data distributions. On the other hand, logistic regression may give better results than state space models for a stable data distribution.

Our proposed framework, depicted in Figure 1, can be used in applications that give rise to spatio-temporal events with contiguous spatial dimensions. Typically, pre-processing is also a common step done before event extraction. However we do not focus on pre-processing data as techniques differ depending on the application. As event extraction is needed before classification, we also propose a simple method for event extraction. We make this work available in the R package *eventstream* [28].

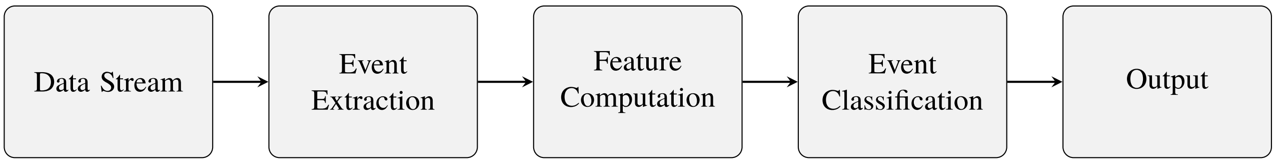
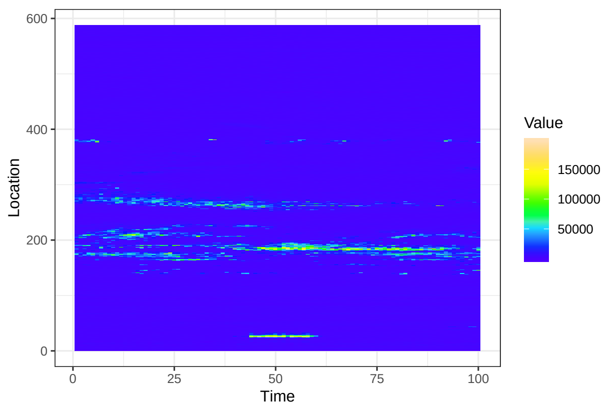


Figure 1: *Framework for event extraction and classification for spatio-temporal data.*

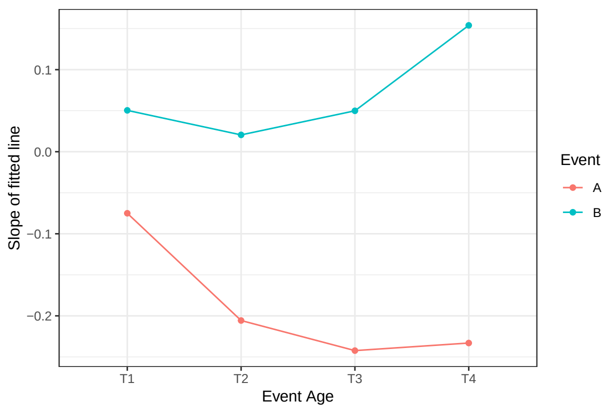
Figure 2 shows the heatmap of a dataset produced from a fibre optic cable. A pulse is periodically sent through the cable and this results in a data matrix where each horizontal row gives the strength of the signal at a fixed location x_0, and each vertical column gives the strength of the signal along 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 [29], detecting oil leakages [30], detecting intruders on secured premises [31], monitoring health of infra-structure such as bridges and pipe-lines [32], to name a few. 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 2, events are seen in the lighter-coloured parts. The event approximately at location 30 between the time interval 45 to 60 is of class A while other events that appear between locations 150 and 400



(a)



(b)

Figure 2: *Figure 2a shows data from a fibre optic cable. We extract events from this window and compute event features. We consider two events belonging to two different classes and an event feature that changes with event-age. Figure 2b shows this event feature, which is the slope of a line fitted to the event signal values, and how it changes with event-age.*

are of class B. Due to the commercially sensitive nature of the dataset, we refrain from giving details about the actual application.

We will investigate early classification of spatio-temporal events using the framework in Figure 1. The remainder of the paper is organised as follows. Section 2 discusses event extraction and feature computation, while Section 3 introduces the notation for classification of age-varying events. In Sections 4 and 5 we introduce the age-varying events classifier and the cascaded non-Gaussian state space models respectively. In Section 6 we discuss results of our framework using synthetic data and two real applications. The first application is associated with the dataset in Figure 2 and the second application is on Nitrogen dioxide (NO_2) data from NASA's NEO [33] website. Finally, in Section 7, we present our conclusions and discuss future work.

2 Event extraction and feature computation

2.1 Event extraction

We extract events from data streams of 2 or 3 dimensions; i.e. 1 time dimension and either 1 or 2 spatial dimensions. We employ a simple method for event extraction using DBSCAN [34], which is a density based clustering algorithm. Our event extraction algorithm is explained in Algorithm 1.

```
input  : a 2 or 3-dimensional array $A_{n\times m}$ or $A_{n\times m\times k}$ , and parameters $\alpha$, $\epsilon$ and $minPts$.
output : events $A|_S \subset A$ and the event ids for each $s_{ij} \in A|_S$ (or $s_{ijk}$ if $A$ is 3-dimensional).
1 Let $a_{ij}$ be the signal value at $(i, j)$ position of $A$, if $A$ is 2-dimensional and $a_{ijk}$ be the signal value at $(i, j, k)$ position of $A$ if $A$ is 3−
    dimensional.
2 Let $q$ denote the $\alpha$-percentile of the signal values of $A$.
3 $S = \{(i, j) \mid a_{ij} > q\}$ for 2-dimensional $A$ and $S = \{(i, j, k) \mid a_{ijk} > q\}$ for 3-dimensional $A$.
4 $B = A|_S$, i.e., signal values of $A$ in $S$ locations.
5 Using DBSCAN cluster B using $\epsilon$ and $minPts$.
6 This clustering gives each $s \in A|_S$ a cluster id.
7 Consider each cluster as an event.
```

Algorithm 1: *Extract events from a dataset or window.*

Figure 3 shows a synthetic dataset generated from the package *eventstream*, along with the events extracted using Algorithm 1.

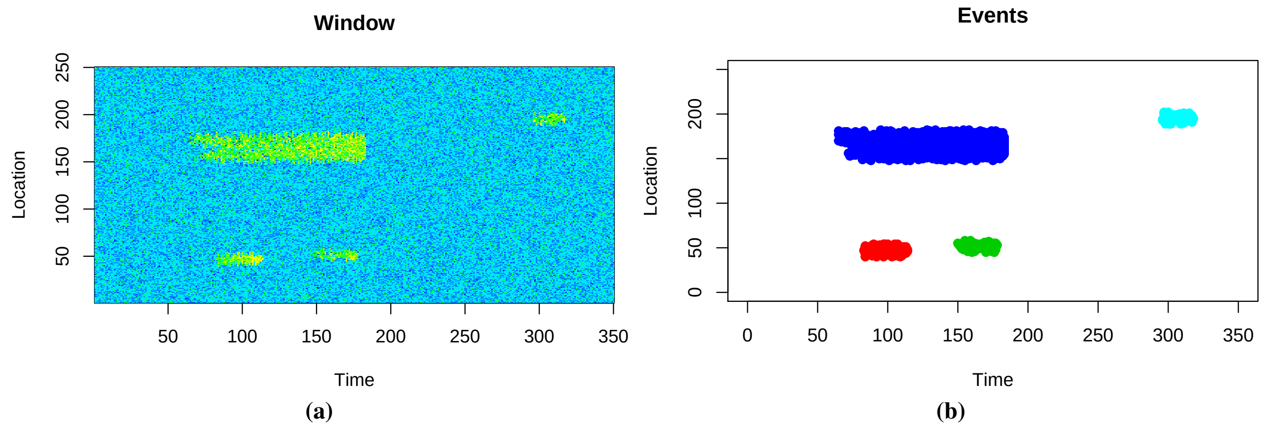


Figure 3: *A synthetic dataset from eventstream and the events extracted from it using* $\alpha = 0.95$*,* $\epsilon = 5$ *and* $minPts = 10$*.*

2.2 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 features and are motivated from the fibre optic application (see Figure 1).

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, 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 4a are shown in Figures 4b and 4c.

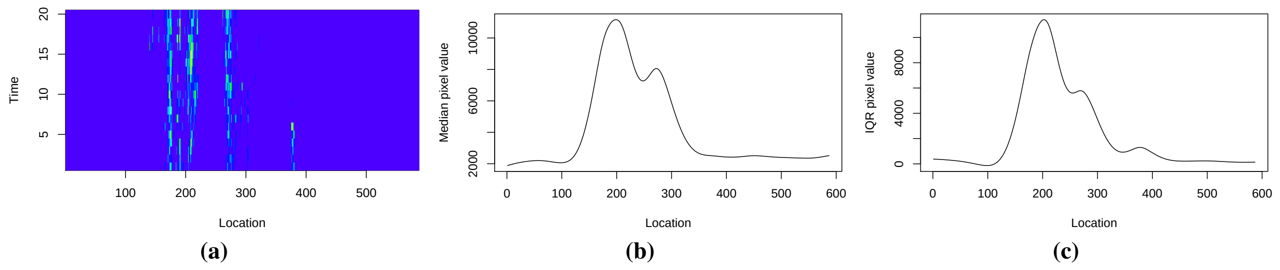


Figure 4: *The initial portion of a window and the resulting median and IQR splines.*

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 $n \in \{5, \dots, 8\}$
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 1–8 from the above list and an equivalent of feature 14 using the global standard deviation and the global mean. 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.

3 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 [35, 36, 37, 38, 39, 40, 41]. 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 2. 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 $\{\boldsymbol{p}_{t_1}^k, \boldsymbol{p}_{t_2}^k, \boldsymbol{p}_{t_3}^k, \ldots, \boldsymbol{p}_{t_n}^k\}$, the classifier C_{t_i} can be used on $\boldsymbol{p}_{t_i}^k$. Thus, as the partial observation grows, we have a growing prediction $\{y_{t_1}, y_{t_2}, \ldots, y_{t_n}\}$ 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. The age-varying events classifier described in the next section addresses this limitation, while giving a growing prediction.

4 Age-varying events classifier

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

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

where $\beta = (\beta_0, \beta_1, \ldots, \beta_l)$ and $(\boldsymbol{x}_i, y_i)$ are observations for $i \in \{1, \ldots, N\}$. By extending a classifier to take in partial observations, we initially minimize the following loss function,

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

where $\tilde{\beta} = \{\tilde{\beta}_{jk}\}$ for $j \in \{1, \ldots, n\}$, $k \in \{0, 1, \ldots, l\}$ and $\boldsymbol{x}_j$ gradually becomes available as $\{\boldsymbol{p}_{t_1}^j, \boldsymbol{p}_{t_2}^j, \boldsymbol{p}_{t_3}^j, \ldots, \boldsymbol{p}_{t_n}^j\}$ with $\boldsymbol{x}_j = \boldsymbol{p}_{t_n}^j$. For a given j, y_j is the class label of $\boldsymbol{x}_j$, and so for all k, $\boldsymbol{p}_{t_k}^j$ have the same y_j. Let us call this initial extended classifier $\mathscr{E}$.

We note that $\mathscr{E}$ is equivalent to a series of n classifiers $\{C_{t_j}\}_{j=1}^n$. To show this, first note that C_{t_j} minimizes the loss function

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

with $\tilde{\beta}_j = (\tilde{\beta}_{j0}, \tilde{\beta}_{j1}, \tilde{\beta}_{j2}, \ldots, \tilde{\beta}_{jl})$. Also,

$$\arg\min_{\tilde{\beta}} \sum_{i=1}^{N} \sum_{j=1}^{n} \mathscr{L}(\boldsymbol{p}_{t_j}^i, y_i; \tilde{\beta}) = \arg\min_{\tilde{\beta}} \sum_{j=1}^{n} \sum_{i=1}^{N} \mathscr{L}(\boldsymbol{p}_{t_j}^i, y_i; \tilde{\beta}). \tag{3}$$

In addition $\{\boldsymbol{p}_{t_j}^i\}_{i=1}^N$ 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

$$\arg\min_{\tilde{\beta}} \sum_{j=1}^{n} \sum_{i=1}^{N} \mathscr{L}\left(\boldsymbol{p}_{t_j}^i, y_i; \tilde{\beta}\right) = \left[\arg\min_{\tilde{\beta}_1} \sum_{i=1}^{N} \mathscr{L}\left(\boldsymbol{p}_{t_1}^i, y_i; \tilde{\beta}_1\right) \cdots \arg\min_{\tilde{\beta}_n} \sum_{i=1}^{N} \mathscr{L}\left(\boldsymbol{p}_{t_n}^i, y_i; \tilde{\beta}_n\right)\right]^T. \tag{4}$$

From equations (3) and (4) we see that the initial extended classifier $\mathscr{E}$ is equivalent to a series of n classifiers $\{C_{t_j}\}_{j=1}^n$.

Having n independent classifiers $\left\{C_{t_j}\right\}_{j=1}^n$ will make the class boundaries for each classifier independent of each other. As $\boldsymbol{x}_i \in \mathbb{R}^b$, the class boundary of C_{t_j} lives in the ambient space $\mathbb{R}^b$. Let us denote the class boundary of C_{t_j} by B_j and the class boundary of $\mathscr{E}$ by $B_{\mathscr{E}}$. As n independent classifiers $\left\{C_{t_j}\right\}_{j=1}^n$ in totality are equivalent to

$\mathscr{E}$, it follows that the class boundary $B_{\mathscr{E}}$ comprises the set $\{B_j\}_{j=1}^{n}$. By stacking B_j in the order of increasing j, i.e. by considering the set $\{(t_j, B_j)\}_{j=1}^{n}$, we can visualize a continuous class boundary that connects the set $\{B_j\}_{j=1}^{n}$ with increasing t_j. This continuous class boundary lives in $\mathbb{R}^{b+1}$. Thus, the set $\{(t_j, B_j)\}_{j=1}^{n}$ can be viewed as a discrete approximation of a continuous class boundary. Therefore, $B_{\mathscr{E}}$ can be viewed as living in $\mathbb{R}^{b+1}$.

In addition, it is desirable if $B_{\mathscr{E}}$ has a certain level of smoothness, because it aids convergence of early classification based on partial observations to full classification based on full observations. To incorporate smoothness in $B_{\mathscr{E}}$, we modify the original loss function by including an L_2 penalty term as follows:

$$\varphi\left(\tilde{\beta}, \lambda\right) = \frac{1}{nN} \sum_{i=1}^{N} \sum_{j=1}^{n} \mathscr{L}\left(\boldsymbol{p}_{t_j}^{i}, y_i; \tilde{\beta}\right) + \lambda \sum_{k=1}^{l} \sum_{j=1}^{n-1} \left(\tilde{\beta}_{j+1,k} - \tilde{\beta}_{j,k}\right)^2 , \tag{5}$$

for some $\lambda > 0$. The constant λ is a parameter that can be specified. Recall that $\tilde{\beta}_j = \left(\tilde{\beta}_{j1}, \tilde{\beta}_{j2}, \ldots, \tilde{\beta}_{jl}\right)$ relates to partial observations $\{\boldsymbol{p}_{t_j}^{i}\}_{i=1}^{N}$ for a fixed t_j, i.e. $\tilde{\beta}_{j1}$ is the coefficient of the first covariate at age t_j and $\tilde{\beta}_{j2}$ the coefficient of the second covariate at age t_j. Thus the penalty 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. We minimize the loss function described by equation (5) in our proposed classifier. As we use this classifier on premature events, we call this the *age-varying events classifier*.

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

This is a general formulation for an age-varying events classifier, i.e. any loss function $\mathscr{L}$ can be used in equation (6). Without loss of generality, we implement the age-varying events classifier for logistic regression in our R package *eventstream*. For logistic regression [42] the loss function $\mathscr{L}$ is given by

$$\mathscr{L}(\boldsymbol{p}_{t_j}^{i}, y_i; \tilde{\beta}) = -y_i \left([1\ (\boldsymbol{p}_{t_j}^{i})^T] \tilde{\beta}_j \right) + \log\left(1 + \exp\left\{ [1\ (\boldsymbol{p}_{t_j}^{i})^T] \tilde{\beta}_j \right\}\right) . \tag{7}$$

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

Next we explore a non-Gaussian state space model for partial observations.

5 Cascaded non-Gaussian state space models

We use non-Gaussian state space models as described in [43] and [44] to model partial observations, and the implementation in R package *dma* [45]. We start with the linear Gaussian state space models which are given by the following equations :

$$y_t = Z_t \alpha_t + \varepsilon_t , \tag{8}$$

$$\alpha_{t+1} = T_t \alpha_t + R_t \eta_t , \tag{9}$$

where $\varepsilon_t \in \mathcal{N}(0, H_t)$ and $\eta_t \in \mathcal{N}(0, Q_t)$. Here y_t is an N-dimensional vector of observations and α_t is a $(l+1)$-dimensional state vector. The matrices Z_t, T_t, R_t, H_t and Q_t are known at the beginning. Equation (8) is called the "observation equation" and equation (9) the "state equation". When using state space models for dynamic regression, α_t denotes the coefficients which update dynamically.

If y_t is a binary response such that

$$y_t \sim \text{Bernoulli}(p_t) ,$$

equation (8) changes as

$$\text{logit}(p_t) = Z_t \alpha_t , \tag{10}$$

with logit function $f(t) = \log(t/(1-t))$, while keeping equation (9) unchanged.

5.1 Modelling partial observations

We are interested in a growing prediction $\{\theta^j_{t_1}, \theta^j_{t_2}, \ldots, \theta^j_{t_n}\}$ for partial observations $\{\boldsymbol{p}^j_{t_1}, \boldsymbol{p}^j_{t_2}, \boldsymbol{p}^j_{t_3}, \ldots \boldsymbol{p}^j_{t_n}\}$ belonging to the same complete observation. Again, we note that for partial observations $\{\boldsymbol{p}^j_{t_1}, \boldsymbol{p}^j_{t_2}, \boldsymbol{p}^j_{t_3}, \ldots \boldsymbol{p}^j_{t_n}\}$, the time stamps $(t_1, t_2, \ldots, t_n)$ denote the ages of the partial observations, not the time of occurrence.

As state space models generally deal with time, for each age t_k we use a separate state space model. However, for some cases it may benefit if there is communication between the models regarding the same observation. To add communication between the models, we bundle the partial observation $\boldsymbol{p}^j_{t_k}$ with the prediction output $\theta^j_{t_{k-1}}$ of the lower age model and use it as input for the age t_k model.

To explain this further, let us define $\boldsymbol{P}_{t_k} = \left[\boldsymbol{p}^1_{t_k}\ \boldsymbol{p}^2_{t_k}\ \cdots\ \boldsymbol{p}^N_{t_k}\right]^T$ - an $l \times N$ matrix containing all partial observations of age t_k, $\mathbf{1} = [1, 1, \ldots, 1]^T$ and $\theta_{t_k} = \left[\theta^1_{t_k}, \theta^2_{t_k}, \ldots, \theta^N_{t_k}\right]^T$ - the logit predictions of the partial observations in $\boldsymbol{P}_{t_k}$, where $\boldsymbol{p}^j_{t_k}$ is an l-vector and N denotes the number of complete observations. Then the equation corresponding to equation (10) for the initial model for age t_1 can be written as

$$\theta_{t_1} = Z_{t_1}\alpha_{t_1}\,. \tag{11}$$

$$\text{with} \quad Z_{t_1} = [\mathbf{1}\ \boldsymbol{P}_{t_1}], \tag{12}$$

Using the output θ_{t_1}, we write the model for age t_2 as

$$Z_{t_2} = [\mathbf{1}\ \boldsymbol{P}_{t_2}\ \theta_{t_1}], \tag{13}$$

$$\theta_{t_2} = Z_{t_2}\alpha_{t_2}\,. \tag{14}$$

Building in this way we obtain

$$Z_{t_k} = [\mathbf{1}\ \boldsymbol{P}_{t_k}\ \theta_{k_{t-1}}], \tag{15}$$

$$\theta_{t_k} = Z_{t_k}\alpha_{t_k}\,. \tag{16}$$

The equations (15), (16) and (9) describe the model for the partial observations at age t_k. Let us denote the model which describes the partial observations of age t_k by Λ_k and the predicted output using the inverse link function by $\hat{\boldsymbol{y}}_{t_k}$. Then Figure 5 gives the structure of our partial observations model.

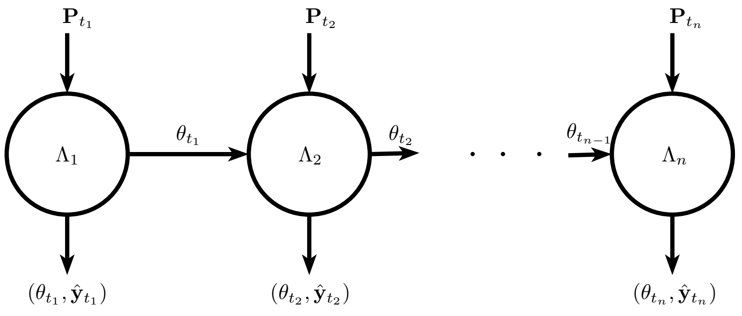


Figure 5: *Structure of the partial observations model with communication.*

However, communication between models Λ_{k-1} and Λ_k may not improve the overall accuracy for every partial observations problem. For example, for a class imbalanced problem, a small number of incorrect predictions in $\theta_{t_{k-1}}$ can have an adverse effect on the model Λ_k. Therefore, for some instances such as class imbalanced datasets, it is desirable to have models Λ_k without communication between the models, as shown in Figure 6.

6 Applications

We use age-varying events classifiers and cascaded state space models to classify events in synthetic and real data streams. The R code applicable to this section is available in the supplementary material.

6.1 Synthetic data

The synthetic data used in this section can be generated from 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. The use of three different shapes for class B events and one shape for class A events (with that shape being similar to some of those of class B) was motivated from the real application.

Figure 7a contains two events of class A, and Figure 7b contains 3 events of class B. The shapes are labelled as 1, 2 or 3 in both Figures 7a and 7b, with shape 1 being the common shape.

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.

We classify the extracted events from synthetic data using: 1) the age-varying events classifier; 2) cascaded state space models; and 3) logistic regression. We use the same feature vector to summarise the extracted events. Figure 8 shows snapshots of the data and the extracted events for two time windows.

We repeat each experiment 5 times with data streams generated with different seeds. For each experiment we generate a data stream of dimension 3500×250 of which 80% (2800×250) is used for training and the remaining

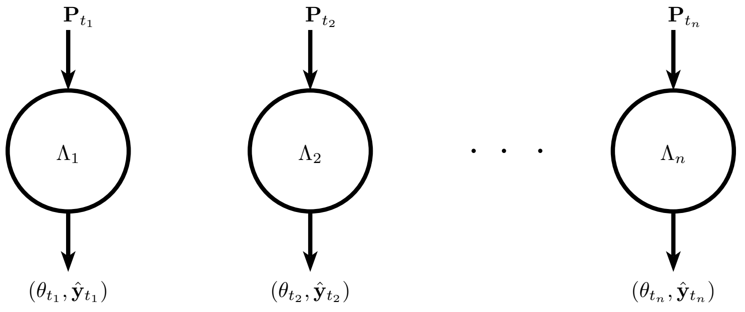

$\mathbf{P}_{t_1}$
$\mathbf{P}_{t_2}$
$\mathbf{P}_{t_n}$
Λ_1
Λ_2
Λ_n
$(\theta_{t_1}, \hat{\mathbf{y}}_{t_1})$
$(\theta_{t_2}, \hat{\mathbf{y}}_{t_2})$
$(\theta_{t_n}, \hat{\mathbf{y}}_{t_n})$

Figure 6: *Structure of the partial observations model without communication.*

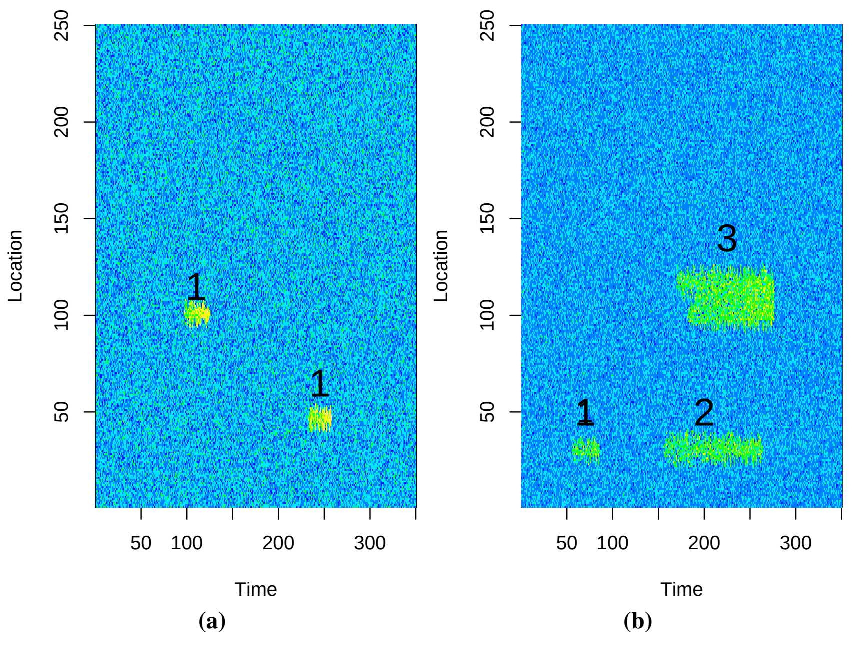
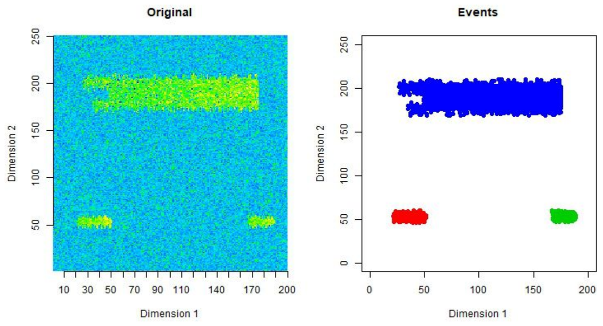


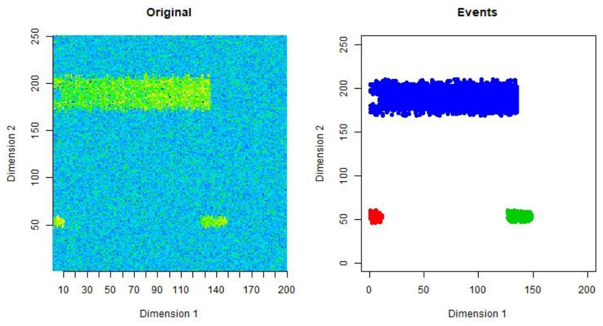
Figure 7: *Class A events in Figure 7a and Class B events in Figure 7b.*

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.*

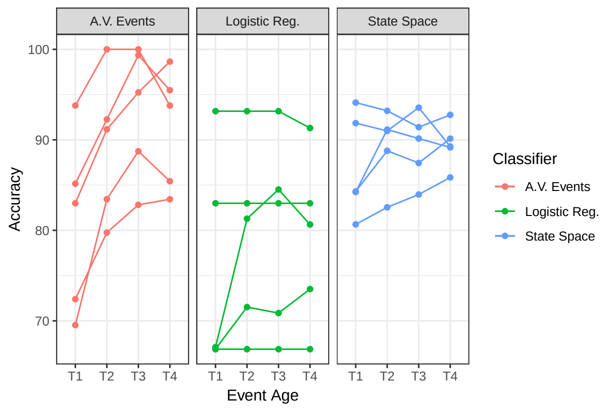


(a)

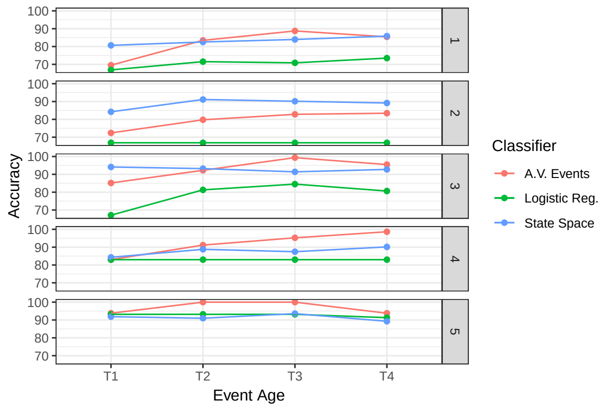


(b)

Figure 8: *Two windows of data and extracted events.*



(a)



(b)

Figure 9: *Accuracy of the 3 classifiers over 5 repeats grouped by classifier in Figure 9a and by repetition in Figure 9b.*

20% for testing. For synthetic data classification, we do not use features which were motivated from the real example, i.e. we do not use features 11 and 12 from the list in Section 2.2, for computational efficiency. In addition, the centroid is not used in any classification task. 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. That is, event features are calculated at these ages. We use a moving window model of dimension 200×250 which moves by a step of 8×250.

Figure 9 shows the accuracy of the age-varying events classifier, state space model and logistic regression classifier over 5 repetitions. From Figure 9b we see that for each repetition either the age-varying events classifier or the state space model surpasses the logistic regression classifier. From Figure 9a we see that the age-varying events classifier performs better on average than the other two classifiers for synthetic data. Table 2 gives the average test set accuracy and standard deviation results for the 3 classifiers, which confirm these observations. Also we see that all 3 classifiers improve their average accuracy levels with the age of the events.

Classifier	Accuracy				Standard deviation			
	t_1	t_2	t_3	t_4	t_1	t_2	t_3	t_4
Age-varying events classifier	80	89	93	91	9	7	7	6
Cascaded state space models	87	89	89	89	5	4	3	2
Logistic regression	75	79	79	79	12	10	10	9

Table 2: *Average test set accuracy and standard deviation* (%) *over* 5 *repetitions.*

6.2 Fibre optic cable data

The data for the first real application is from a fibre optic cable, and is shown in Figure 10. 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**. The extracted events have a maximum age of 40 time units, so we use a window model with a window size 40×587 and a step size 10×587 to extract events and compute features.

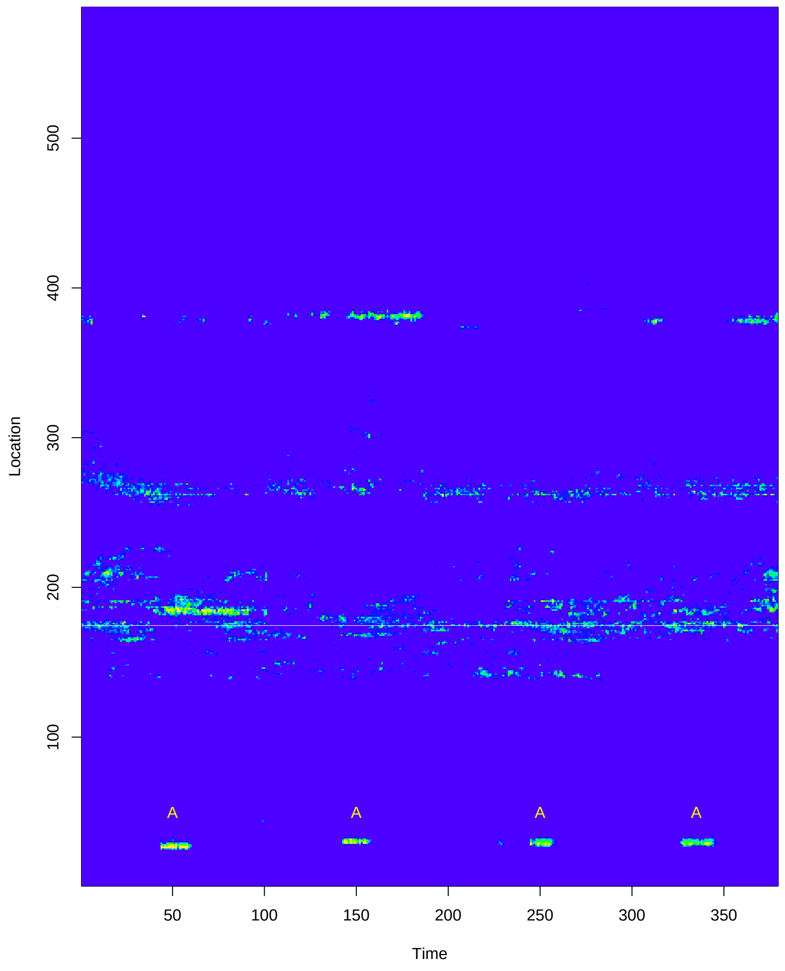


Figure 10: *Data stream from a fibre optic cable.*

Similar to the synthetic data stream, we classify the extracted event features using 3 classifiers. The main difference is that the real data stream has a much smaller number of class A events compared to class B events. Due to this class imbalance, we configure the state space models for real data differently from those used for the synthetic data. First, we use the state space models structure depicted in Figure 6. Second, as there are only 4 class A events, there is not enough class A data to update the state space models regularly. Consequently, we use the state space models for training and prediction without the updating step given in equation (9).

We use 4-fold cross validation because there are 4 class A events, with event ages $t = 10, 20, 30$ and 40, because the maximum event-age is 40 time units. The age-varying events, logistic regression and the state space models classifiers are trained on a data stream comprising 3 training folds, and tested on the remaining fold.

We report additional accuracy measures that are designed for imbalanced datasets. These metrics are positive predictive value (PPV), 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 measures. 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.

In contrast, the area under the receiver operator characteristic (ROC) curve is a single measure that captures the effectiveness of a classifier. The 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 can be interpreted as the probability that a positive observation is ranked higher than a negative observation. 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.

Accuracy Measure	Classifier	Mean				Standard deviation			
		t_1	t_2	t_3	t_4	t_1	t_2	t_3	t_4
PPV	Age-varying events classifier	95	95	100	95	8	8	0	8
	Cascaded state space models	74	69	68	68	18	15	12	23
	Logistic regression	91	75	75	70	16	31	31	34
NPV	Age-varying events classifier	91	92	93	92	6	3	4	5
	Cascaded state space models	98	100	99	100	2	0	1	0
	Logistic regression	95	95	95	92	1	1	3	5
AUC	Age-varying events classifier	93	94	96	94	5	5	2	5
	Cascaded state space models	94	97	94	97	7	0	5	1
	Logistic regression	93	85	85	81	7	15	15	14

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. From Table 3, we see that the age-varying events classifier has the highest average PPV and the cascaded state space models have the highest average NPV for all event ages. Of the two classifiers, state space models are more conservative in predicting class A events compared with the age-varying events classifier. That is, if the state space models predict an event as belonging to class A, it is more likely that the actual event is of class A compared with the age-varying classifier. This is supported by the high mean NPV values and the associated low standard deviations of the state space models. On the other hand, the age-varying events classifier is much more likely to catch all class A events compared with the state space models as seen by the higher PPV values. Whether it is better to be conservative and miss some class A events while being very accurate in terms of the predicted class A events, or to identify all class A events while giving some false positives, depends on the application. Often the cost of false positives and false negatives are not the same; for example consider a medical test for cancer. Thus, we see that our proposed two models have different strengths. In addition, both these models outperform the logistic regression classifier as seen by the AUC values.

6.3 Nitrogen dioxide monitoring

The second application focuses on event extraction from Nitrogen Dioxide (NO_2) data from NASA's NEO website [33]. The Ozone Monitoring Instrument (OMI) [46] aboard the Aura satellite records a variety of air quality measures including NO_2 concentrations around the world. We use some of this data to demonstrate the event extraction and feature computation process for a 3-dimensional data stream.

We use OMI NO_2 monthly data from March to June in the years 2008 and 2018 for a comparison study. For each month the data comes in a matrix of 1799×3600 dimensions. For ease of calculation we reduce the dimension of this matrix to 180×360, by computing the average of each 10×10 block of data in the original matrix. The OMI NO_2 data for two months (March 2008 and March 2018) are shown in Figure 11.

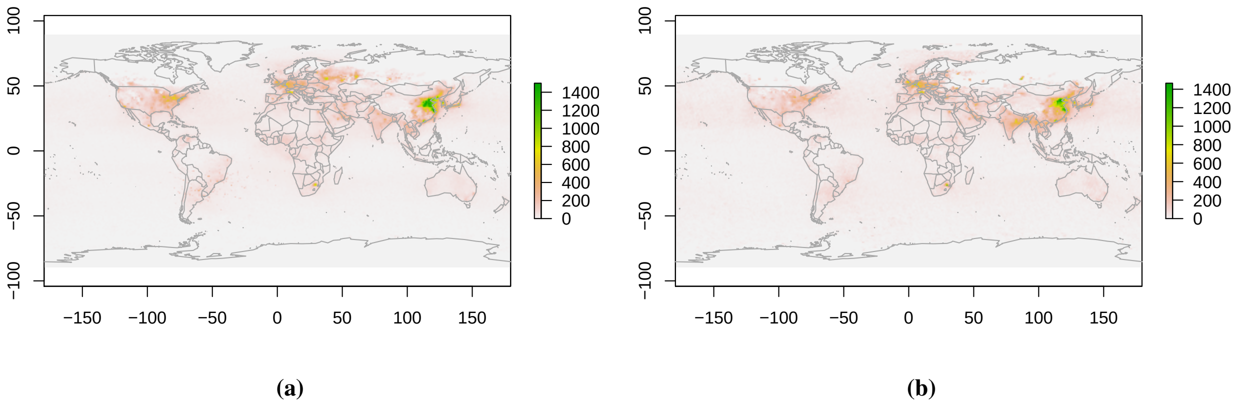


Figure 11: *NO_2 data for March 2008 (Figure 11a) and March 2018 (Figure 11b).*

Using Algorithm 1, we extract events from this data set, which are clusters of high NO_2 levels spanning space and time. Events are extracted from a 3-dimensional data stream of $4 \times 180 \times 360$ array, where each 180×360 matrix corresponds to the NO_2 levels of a given month. The parameters used in Algorithm 1 are $\alpha = 0.97$, $\epsilon = 2$ and minPts = 20. There are 23 events/clusters in the 2008 data and 13 events/clusters in 2018 data. Figure 12 shows these clusters for March 2008 and March 2018 (corresponding to the data in Figure 11) with each cluster/event depicted by a single colour.

For each event we compute features 1–8 and 14 from the list of features in Section 2.2. These features are chosen for ease of computation. Specially as we are not focusing on event classification in this application, we do not compute the other features which involve fitting splines for 3-dimensional data.

We analyse the two events that had the highest average NO_2 levels for each of 2008 and 2018. The maps in Figure 13 show the spatial locations and the NO_2 levels of these two events for 2008 and 2018. As these two events are geographically at the same location, we compare the average NO_2 levels for these two events for the months from March to June in 2008 with those of 2018. The graph in Figure 14 shows that the average NO_2 levels have decreased in 2018 compared to 2008 for this geographical location.

Next we match the events in 2008 and 2018 by their spatial location. We do a simple analysis of these NO_2 events. We want to know if these NO_2 events have increased or decreased in severity during this 10 year time gap. For each of these matched events we find the average NO_2 level difference between 2018 and 2008. Figure 15 shows the graph of these differences. We see that 4 events have positive NO_2 differences for each month, i.e., the average NO_2 levels have increased from 2008 to 2018 for these 4 events. In addition, event 15 is different from

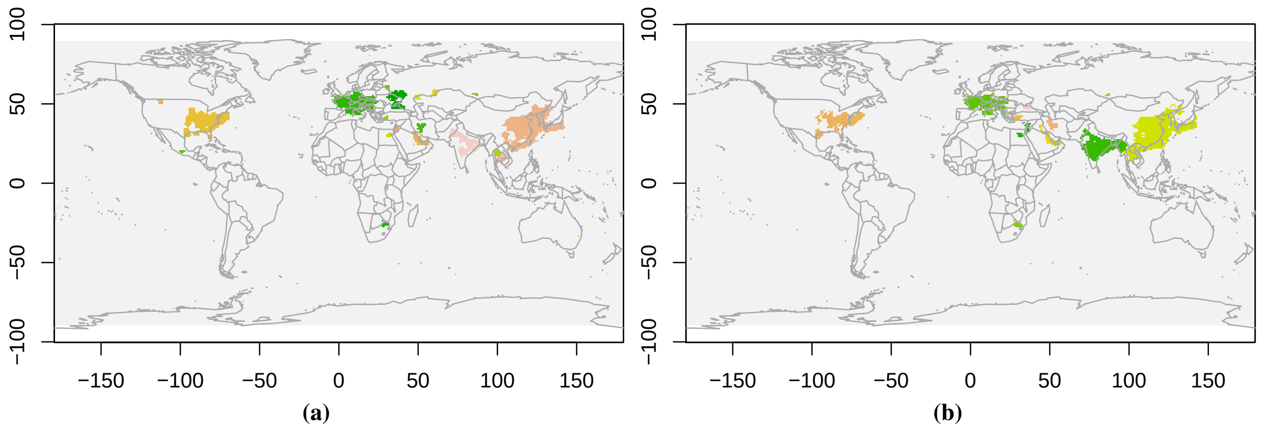


Figure 12: *Events extracted from NO_2 data for March 2008 (Figure 12a) and March 2018 (Figure 12b). Colours do not reflect NO_2 levels. Each event is depicted by a single colour.*

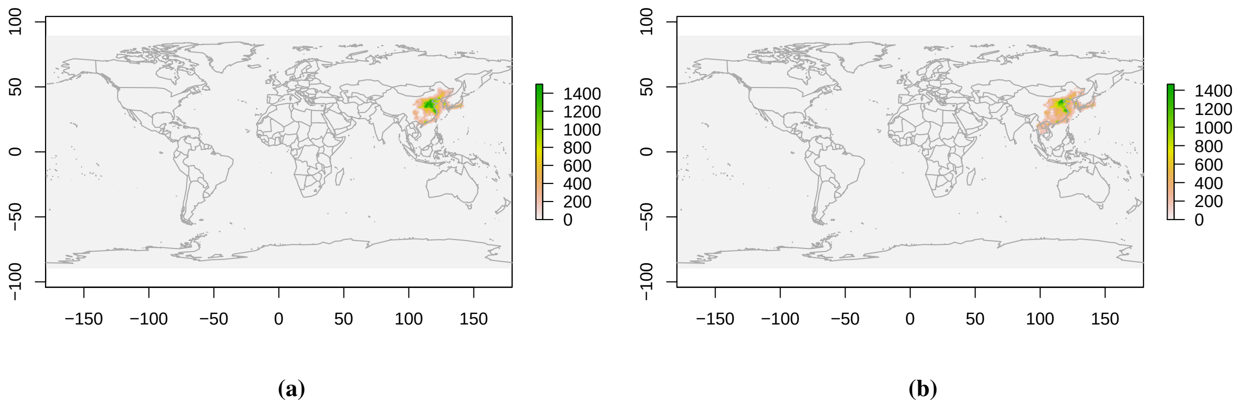


Figure 13: *The two events with the highest average NO_2 levels for March 2008 (Figure 13a) and March 2018 (Figure 13b).*

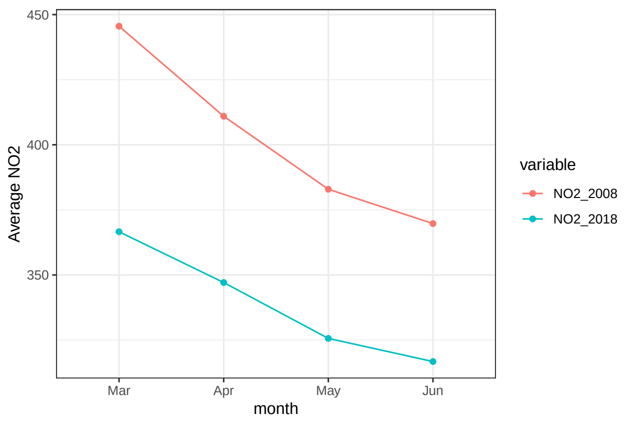
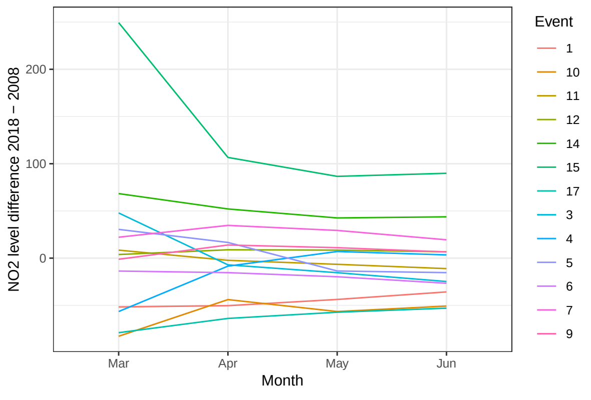
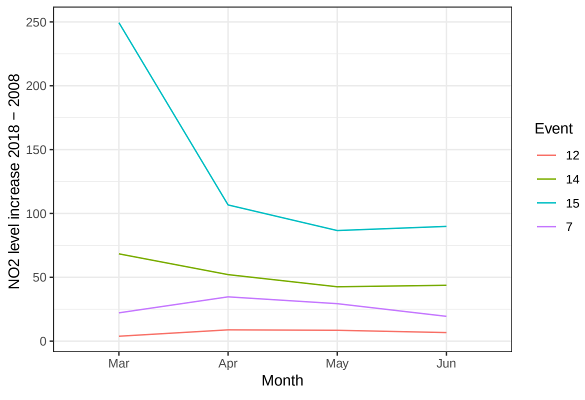


Figure 14: *The average NO_2 levels for the two events in Figure 13.*

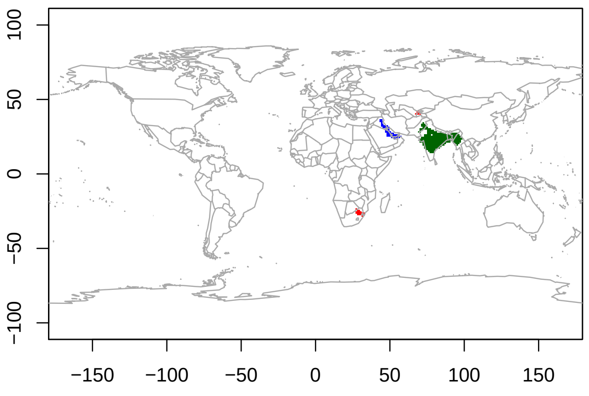


(a)

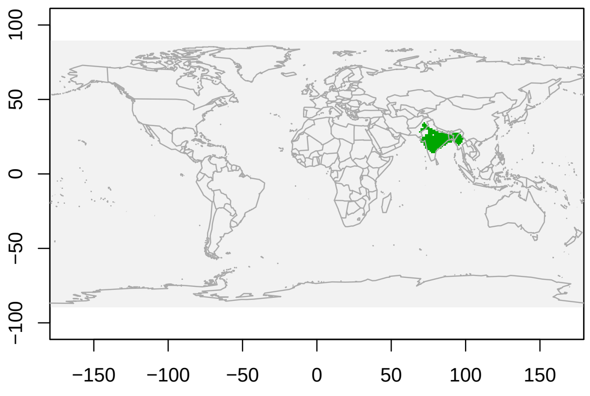


(b)

Figure 15: *(a) Difference in average NO_2 levels (2018 – 2008) for all matched events. (b) The events for which average NO_2 level difference is positive (average NO_2 levels have increased from 2008 to 2018).*



(a)



(b)

Figure 16: *The location in March 2018 of events with increased average NO_2 levels in Figure 16a, and the outlying NO_2 event in Figure 16b.*

other events. Figure 16 shows the locations of the increased NO_2 events in March 2018 along with the location of the outlying event 15.

While the increased industrialization in India and the ongoing war in the Middle-East may be contributors, an investigation of reasons behind the increased NO_2 levels is beyond the scope of this study. Our aim is to demonstrate the applicability of our event extraction and event classification algorithms for different applications.

7 Conclusions

This paper has proposed a framework for event extraction and event classification in data streams with a focus on early event classification. We have introduced time-varying coefficient models for event classification, and we have proposed two classifiers: a static and an updating classifier, both of which take developing event features/partial observations as input. We have tested our framework using 3 applications, one synthetic data application and two real data applications. We have shown that we obtain better accuracy results compared to logistic regression for these applications.

In addition, we have proposed a simple algorithm for event extraction, which can be used on two or three dimensional data streams. The applicability of our event extraction algorithm has been demonstrated using the NO_2 data from NASA's NEO website.

Future directions for this research include extending the age-varying events classifier to automatically update coefficients, and including a greater variety of event extraction processes.

8 Supplementary material

R-package eventstream: This package contains Algorithm 1 for event extraction, the age-varying event classifier for early event classification, functions for synthetic data generation, the fibre-optic data stream and NO_2 data for 2008 and 2018. It is available from Github at `https://github.com/sevvandi/eventstream`.
Scripts: There are three files containing the R code used in Section 6. The file `Supp_Mat_1.R`, `Supp_Mat_2.R` and `Supp_Mat_3.R` contain the code applicable for synthetic data in Section 6.1, fibre optic data in Section 6.2 and NO_2 data in Section 6.3 respectively. The file `Supp_Mat_4.R` contains the code used to produce some other graphs and images in the paper.
Other R-packages: We have used the following R-packages either in this paper or within the package *eventstream* : *dma* [45], *abind* [47], *AtmRay* [48], *pROC* [49], *reshape2* [50], *ggplot2* [51], *raster* [52], *maps* [53], *tensorA* [54], *glmnet* [55], *dbscan* [56] and *MASS* [57].

Acknowledgements

Funding was provided by the Australian Research Council through the Linkage Project LP160101885. The first author would like to thank Dr Andres Ramirez Hassan for useful discussions.

References

[1] X. Zhou, S. Shekhar, and R. Y. Ali, "Spatiotemporal change footprint pattern discovery: an inter-disciplinary survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 1, pp. 1–23, 2014.

[2] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz, "Robust real-time unusual event detection using multiple fixed-location monitors," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 3, pp. 555–560, 2008.

[3] Y. Ke, R. Sukthankar, and M. Hebert, "Efficient visual event detection using volumetric features," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1. IEEE, 2005, pp. 166–173.

[4] G. Medioni, I. Cohen, F. Brémond, S. Hongeng, and R. Nevatia, "Event detection and analysis from video streams," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23, no. 8, pp. 873–889, 2001.

[5] J. Weng and B.-S. Lee, "Event detection in twitter." *ICWSM*, vol. 11, pp. 401–408, 2011.

[6] R. Li, K. H. Lei, R. Khadiwala, and K. C.-C. Chang, "Tedas: A twitter-based event detection and analysis system," in *Data engineering (icde), 2012 ieee 28th international conference on*. IEEE, 2012, pp. 1273–1276.

[7] H. Abdelhaq, C. Sengstock, and M. Gertz, "Eventweet: Online localized event detection from twitter," *Proceedings of the VLDB Endowment*, vol. 6, no. 12, pp. 1326–1329, 2013.

[8] J. Allan, R. Papka, and V. Lavrenko, "On-line new event detection and tracking," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 1998, pp. 37–45.

[9] Z. Li, B. Wang, M. Li, and W.-Y. Ma, "A probabilistic model for retrospective news event detection," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2005, pp. 106–113.

[10] J. Yin, D. H. Hu, and Q. Yang, "Spatio-temporal event detection using dynamic conditional random fields." in *Ijcai*, vol. 9, 2009, pp. 1321–1327.

[11] Y. Mao, Q. Jie, B. Jia, P. Ping, and X. Li, "Online event detection based on the spatio-temporal analysis in the river sensor networks," in *Information and Automation, 2015 IEEE International Conference on*. IEEE, 2015, pp. 2320–2325.

[12] G. Souto and T. Liebig, "On event detection from spatial time series for urban traffic applications," in *Solving large scale learning tasks. Challenges and algorithms*. Springer, 2016, pp. 221–233.

[13] A. Mousavi, M. Duckham, R. Kotagiri, and A. Rajabifard, "Spatio-temporal event detection using probabilistic graphical models (pgms)," in *Computational Intelligence and Data Mining (CIDM), 2013 IEEE Symposium on*. IEEE, 2013, pp. 81–88.

[14] T. Cheng and T. Wicks, "Event detection using twitter: a spatio-temporal approach," *PloS one*, vol. 9, no. 6, p. e97807, 2014.

[15] J. Gomide, A. Veloso, W. Meira Jr, V. Almeida, F. Benevenuto, F. Ferraz, and M. Teixeira, "Dengue surveillance based on a computational model of spatio-temporal locality of twitter," in *Proceedings of the 3rd international web science conference*. ACM, 2011, p. 3.

[16] Y. Kang, D. Belušić, and K. Smith-Miles, “Detecting and classifying events in noisy time series,” *Journal of the Atmospheric Sciences*, vol. 71, no. 3, pp. 1090–1104, 2014.

[17] T. Hastie and R. Tibshirani, “Varying-coefficient models,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 757–796, 1993.

[18] G. Widmer and M. Kubat, “Learning in the presence of concept drift and hidden contexts,” *Machine learning*, vol. 23, no. 1, pp. 69–101, 1996.

[19] A. Tsymbal, “The problem of concept drift: definitions and related work,” *Computer Science Department, Trinity College Dublin*, vol. 106, no. 2, 2004.

[20] R. Klinkenberg and T. Joachims, “Detecting concept drift with support vector machines.” in *ICML*, 2000, pp. 487–494.

[21] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM computing surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[22] G. Hulten, L. Spencer, and P. Domingos, “Mining time-changing data streams,” in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 97–106.

[23] J. Gama, *Knowledge discovery from data streams*. Chapman and Hall/CRC, 2010.

[24] M. M. Gaber, A. Zaslavsky, and S. Krishnaswamy, “Mining data streams: a review,” *ACM Sigmod Record*, vol. 34, no. 2, pp. 18–26, 2005.

[25] C. Harvey, “Time-varying conditional covariances in tests of asset pricing models,” 1989.

[26] L. Wang and P. J. Steinhardt, “Cluster abundance constraints for cosmological models with a time-varying, spatially inhomogeneous energy component with negative pressure,” *The Astrophysical Journal*, vol. 508, no. 2, p. 483, 1998.

[27] D. R. Hoover, J. A. Rice, C. O. Wu, and L.-P. Yang, “Nonparametric smoothing estimates of time-varying coefficient models with longitudinal data,” *Biometrika*, vol. 85, no. 4, pp. 809–822, 1998.

[28] S. Kandanaarachchi, *eventstream: An implementation of streaming events and their classification*, r package version 0.0.9000.

[29] Q. Jiang and Q. Sui, “Technological study on distributed fiber sensor monitoring of high voltage power cable in seafloor,” in *Automation and Logistics, 2009. ICAL’09. IEEE International Conference on*. IEEE, 2009, pp. 1154–1157.

[30] M. Niklès, B. H. Vogel, F. Briffod, S. Grosswig, F. Sauser, S. Luebbecke, A. Bals, and T. Pfeiffer, “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, 2004, pp. 18–26.

[31] B. Griffiths, “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, 1995, pp. 325–330.

[32] H.-N. Li, D.-S. Li, and G.-B. Song, “Recent applications of fiber optic sensors to health monitoring in civil engineering,” *Engineering structures*, vol. 26, no. 11, pp. 1647–1657, 2004.

[33] “Omi science team (2012), omi/aura level 2 nitrogen dioxide (no2) trace gas column data,” ftp://neoftp.sci.gsfc.nasa.gov/csv/AURA_NO2_M/, accessed: 2018-11-07.

[34] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, “A density-based algorithm for discovering clusters in large spatial databases with noise.” in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[35] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[36] N. Dabbagh and B. Bannan-Ritland, *Online learning: Concepts, strategies, and application*. Pearson/Merrill/Prentice Hall Upper Saddle River, NJ, 2005.

[37] P. W. Frey and D. J. Slate, “Letter recognition using holland-style adaptive classifiers,” *Machine learning*, vol. 6, no. 2, pp. 161–182, 1991.

[38] G. Giacinto and F. Roli, “Adaptive selection of image classifiers,” in *International Conference on Image Analysis and Processing*. Springer, 1997, pp. 38–45.

[39] K. Nishida, K. Yamauchi, and T. Omori, “Ace: Adaptive classifiers-ensemble system for concept-drifting environments,” in *International Workshop on Multiple Classifier Systems*. Springer, 2005, pp. 176–185.

[40] C. Alippi and M. Roveri, “Just-in-time adaptive classifiers—part i: Detecting nonstationary changes,” *IEEE Transactions on Neural Networks*, vol. 19, no. 7, pp. 1145–1153, 2008.

[41] ——, “Just-in-time adaptive classifiers—part ii: Designing the classifier,” *IEEE Transactions on Neural Networks*, vol. 19, no. 12, pp. 2053–2064, 2008.

[42] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, NY, USA:, 2001, vol. 1, no. 10.

[43] J. Durbin, S.-J. Koopman *et al.*, *Time series analysis of non-Gaussian observations based on state space models from both classical and Bayesian perspectives*. Tilburg University, 1998.

[44] T. H. McCormick, A. E. Raftery, D. Madigan, and R. S. Burd, “Dynamic logistic regression and dynamic model averaging for binary classification,” *Biometrics*, vol. 68, no. 1, pp. 23–30, 2012.

[45] T. H. McCormick, A. Raftery, D. Madigan, S. Kandanaarachchi, and H. Sevcikova, *dma: Dynamic Model Averaging*, 2018, r package version 1.4-0. [Online]. Available: https://CRAN.R-project.org/package=dma

[46] P. F. Levelt, G. H. van den Oord, M. R. Dobber, A. Malkki, H. Visser, J. de Vries, P. Stammes, J. O. Lundell, and H. Saari, "The ozone monitoring instrument," *IEEE Transactions on geoscience and remote sensing*, vol. 44, no. 5, pp. 1093–1101, 2006.

[47] T. Plate and R. Heiberger, *abind: Combine Multidimensional Arrays*, 2016, r package version 1.4-5. [Online]. Available: https://CRAN.R-project.org/package=abind

[48] J. Anderson, *AtmRay: Acoustic Traveltime Calculations for 1-D Atmospheric Models*, 2013, r package version 1.31. [Online]. Available: https://CRAN.R-project.org/package=AtmRay

[49] X. Robin, N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J.-C. Sanchez, and M. Müller, "proc: an open-source package for r and s+ to analyze and compare roc curves," *BMC Bioinformatics*, vol. 12, p. 77, 2011.

[50] H. Wickham, "Reshaping data with the reshape package," *Journal of Statistical Software*, vol. 21, no. 12, pp. 1–20, 2007. [Online]. Available: http://www.jstatsoft.org/v21/i12/

[51] ——, *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. [Online]. Available: http://ggplot2.org

[52] R. J. Hijmans, *raster: Geographic Data Analysis and Modeling*, 2018, r package version 2.8-4. [Online]. Available: https://CRAN.R-project.org/package=raster

[53] O. S. code by Richard A. Becker, A. R. W. R. version by Ray Brownrigg. Enhancements by Thomas P Minka, and A. Deckmyn., *maps: Draw Geographical Maps*, 2018, r package version 3.3.0. [Online]. Available: https://CRAN.R-project.org/package=maps

[54] K. G. van den Boogaart, *tensorA: Advanced Tensor Arithmetic with Named Indices*, 2018, r package version 0.36.1. [Online]. Available: https://CRAN.R-project.org/package=tensorA

[55] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22, 2010. [Online]. Available: http://www.jstatsoft.org/v33/i01/

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

[57] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*, 4th ed. New York: Springer, 2002, iSBN 0-387-95457-0. [Online]. Available: http://www.stats.ox.ac.uk/pub/MASS4