An overview of self-excited point process models of crime

Matthew Daws

June 7, 2021

Abstract

We provide a detailed study of self-exciting point process models of crime. We study a range of models, starting with very simple grid based models where all interactions happen within grid cells, and work towards a fully spatial model where interactions are arbitrary, and KDE techniques are used to estimate background and trigger density functions in a data-driven manner. A detailed and mathematically rigourous derivation (in the appendix) of the fitting algorithms for each model is given. We show explicitly how the general expectation maximisation (EM) algorithm leads in each case to a specific fitting algorithm for each model. By showing explicitly how the EM algorithm is used, we aim to make it easy for future researchers to derive fitting algorithms for new models. We are also able to use the (slow) EM algorithm with complex models, instead of just relying upon the Stochastic EM algorithm: on real data, we have found huge differences in the results, which might explain problems previous studies have found with these complex models. All of our models are implemented in an open source Python library, to allow for reproducibility of research, and to allow future researchers to quickly build upon our work. We demonstrate each model by using burglary crime data from Chicago, and finish the paper by evaluating the models for the ability to predict future crime, in the short (one day) term. We find that the simple models perform just as well as more complex (and slow) models.

1 Introduction

Crime can spread through spacial locations and across time in a contagion-like way, [13], and that after a (for example, property) crime is committed, there is a spatially- and time-limited increased risk of further crime, [15]. At the same time, there is an underlying spatial variation that may not change much with time, [13, 2]. There has long been an interest in mapping variations in crime rate to discern patterns, [12]. More recently, there has been much interest in using the repeat, or near-repeat, pattern of crime to predict future crime events, [12, 14, 27]. A broad class of prediction techniques come under the heading of “prospective hot-spotting”, [3, 16], where we estimate risk at the current time by looking at recent, nearby crime events. More formally, this is a kernel density estimation (KDE) method, with an additional weighting by time.

In such techniques, it is necessary to have an *a priori* choice of the kernels, and bandwidths, to use in space and time. Furthermore, there is no formal statistical model in evidence. Building on work in the theory of seismology, [19, 36, 38], there has been much recent interest in “self-exicited point processes”, [22, 23, 24, 33]. Here we have an explicit statistical model, either parametric or non-parametric, and we proceed in a standard way to fit this model to data, and then to use the model to make predictions. We note that there has been surprisingly little discussion in the literature about how *exactly* to make predictions based on this class of model; see Section 8 below for further discussion. It is widely believed that, in particular, [24] is the basis of the algorithm employed by the commercial crime prediction package PredPol, [28].

These techniques are attractive as they combine explicit statistical models with techniques for fitting these models to data. This allows us to both make inferences based upon the data, [23, Section 4], and to make predictions, [23, Section 5], [24, 33]. However, the relatively sophisticated mathematics, and in our view the slightly opaque presentation of the algorithms involved, may pose a barrier to entry into this field.

We have three aims in this paper. While [24] mentions the EM algorithm by name, [23] does not, and it is far from clear from the majority of papers in this area that all of the algorithms are actually variants of the Expectation Maximisation (EM) algorithm, [21], and that it is possible to rigorously derive all of the specific algorithms from this one technique. We carefully show have the technique of the EM algorithm can be applied to each specific model we develop.

This work came about as part of our efforts to build an open source, Python based implementation of different crime prediction algorithms, which forms the `open_cp` package, [29]. The current paper is backed by Python code, [9], which implements each model. We use the framework provided by `open_cp`, and base classes which abstract typical EM algorithm code. It is our hope that this open source framework, together with our development of the theory, will make it easier for other researchers to both reuse and reproduce our work, and also allow others to develop, more easily, their own models.

Finally, we have found that while the implementation of existing algorithms in `open_cp` work well with artificial data, generated from the statistical model, running such algorithms on real-world data is fraught with problems. This is alluded to in [33] as well. In this paper we discuss theoretical reasons for the problems we have observed, and suggest new models which perform more robustly. Having a well-developed understanding of the probability theory behind these algorithms has been crucial in gaining these insights.

1.1 Acknowledgements

This work was partly funded by the UK Home Office Police Innovation Fund, through the project "More with Less: Authentic Implementation of Evidence-Based Predictive Patrol Plans". The author wishes to thank Gabriel Rosser, Kristian Lum, William Issac, and Martin Short for useful electronic correspondance.

2 Point process models

For general background on point processes, see the mathematical [6], the introductory lecture notes [30], the gentle introduction in [32, Chapter 8], or the encyclopedic [10].

We shall view crime events as point events in time and space, say N events which occur at times $t_1 < t_2 < \cdots < t_N$ and at locations $(x_i, y_i)_{i=1}^N$ which are all distinct. The "distinct" condition is important because of the mathematical abstraction we use, but we could model exact repeats using *marks*; see also the discussion in Section 4.3 below. Alternatively, we can assign points to cells in a spatial grid ([24]) and/or bin timestamps to intervals of time (see again Section 4.3 below). As noted in [10, Part V] while we can treat time as just another dimension, and work with "spatial" point processes in $\mathbb{R}^3$, for motivation, and interpretting results, it makes more intuitive sense to work with time as being different. We shall continue to assume that time is ordered, flowing from the past to the future. A more sophisticated way to treat time is via "aoristic analysis", [31], treating times as being probabilistic *time ranges* instead of exact times; however, we shall not consider this added complication here.

We shall only consider point processes which admit an *intensity function*, Appendix A.1. Informally, we consider a function $\lambda^*(t, x, y)$ which gives the probability of an event occurring in an infinitesimal volume around the point (t, x, y). A useful interpretation is that for any region V of space time, if

$$\lambda(V) = \int_V \lambda^*(t, x, y) \; dt \; dx \; dy, \tag{1}$$

and we define $N(V)$ to be the number of events we observe in V, then $N(V)$ will be a random variable distributed as a Poisson distribution with mean $\lambda(V)$. If U is another space time region disjoint from V, then $N(V)$ and $N(U)$ are independent random variables.

For example, a homogeneous Poisson process has λ^* being constant. An inhomogeneous Poisson process has λ^* being a genuine function (compare the discussion below). We might, for example, model the time dependence as $f(t)$, perhaps using tools from time series analysis, and model space dependence as $g(x,y)$, maybe using a KDE method, by setting $\lambda^*(t,x,y) = f(t)g(x,y)$.

The models we shall be interested in are "self-excited", which loosely means that the present behaviour of the process will depend upon its history. This explains our use of the $*$ in the notation λ^* which reminds us on the dependence on the history of the process. Indeed, our models will always be of the form

$$\underbrace{\lambda^*(t,x,y)}_{\text{Total intensity}} = \underbrace{\mu(t,x,y)}_{\text{Background rate}} + \underbrace{\sum_{t_i<t}}_{\substack{\text{Sum over events } \textit{be-}\\ \textit{fore} \text{ the current time}}} \underbrace{\mu_1(t-t_i, x-x_i, y-y_i)}_{\text{Triggering intensity}}.$$

Thus λ^* depends upon the *history* of the process, the events (t_i, x_i, y_i) with $t_i < t$, where t is the current time. μ is a genuine function, specifying the background rate or intensity, which is the rate at which events occur "just by chance". Each time an event at time t_i occurs, it adds (linearly) extra intensity governed by the "triggering" intensity μ_1. Notice that μ_1 only depends upon the distance, in space and time, between the event and position of interest (t,x,y).

We could consider more general μ_1, which also depended upon (t,x,y), and indeed there might be good reason for doing so (perhaps to model a more spatially concentrated triggering in a densely populated area of a city, vs a spatially disperse triggering in a suburban setting). However, we shall find that fitting the simpler model is currently enough of a challenge, and leave this extension for future research.

The classical prospective hot-spotting technique, [3], can be thought of as a model of this form, where $\mu = 0$ identically, and μ_1 is chosen ahead of time. An advantage of the models considered in [23, 24] is that they allow μ to be non-zero, and that they use *past data* to inform the shape of μ and μ_1. See also the discussion in Section 7 below.

The EM algorithm is an iterative algorithm which seeks to find Maximum Likelihood estimates of parameters (there are extensions to other applications) for models which admit "hidden variables" or "unobserved data". A classical example is a *censored* experiment: perhaps we know how long patients lived who were treated with a variety of treatments, except that if a patient lived for the entire study duration then we only know that they lived *at least* that long. We would typically model this as letting Z_i be the random variable which is survival time, but we only observe $Y_i = \min(Z_i, T)$ for some fixed T. The EM algorithm can also be applied to models where the hidden variables are merely useful mathematical constructs added to simplify analysis.

A useful way to *simulate* a process (compare the introduction of [25, 26]) of our form is to exploit the hidden *branching structure* of the process, which exists because the triggering kernel is linear. We first simulate an inhomogeneous Poisson process with intensity μ, giving the "background events". Then for each background event i, we start at time t_i and simulate a new inhomogeneous Poisson process with intensity $\mu_1(t-t_i, x-x_i, y-y_i)$. We then translate this process by space time position (t,x,y), and add the points into our collection. We consider these to be "triggered events" which have been "triggered by i". We now look at all the new points, and continue the process. As each new point is created into the future, and we only wish to simulate a finite interval of time, our simulation will eventually end. (We note that we might also somehow "simulate into the past", as events before time 0 might trigger events after time 0. Such considerations lead to the concept of a "perfect simulation", see [25]).

Let z_i be the event which "triggered" event i, with the convention that if $z_i = i$ then i is a background event. Thus $z_i \leq i$. The (z_i) form our unobserved data. The EM algorithm then works by using the current estimate of the parameters to assign a probability to the event $z_i = j$ for all i and j. The E step computes the conditional expectation of the log likelihood, conditional over the hidden data (z_i). The M step then maximises the parameters given this conditional expectation. The idea is that if we *knew* the values of (z_i), we could replace the E step by simple evaluation, and then the M step would be the usual maximum likelihood estimation. The EM algorithm can be proved to converge to a *local maximum* under mild conditions, and the beauty of it is that in many practical applications, the resulting procedure is computationally tractable.

We give further details in Appendix A.1 and A.2, but summarise the resulting family of algorithms here. Let $p_{j,i}$ be the probability that event j triggered event i, for $j \leq i$. We have

$$p_{j,i} = \mathbb{P}(z_i = j) = \begin{cases} \mu(t_i, x_i, y_i)/\lambda^*(t_i, x_i, y_i) & : j = i \\ \mu_1(t_i - t_j, x_i - x_j, y_i - y_j)/\lambda^*(t_i, x_i, y_i) & : j < i \end{cases}$$

Notice that then $\sum_j p_{j,i} = 1$ for each i.

Let $\mu(t, x, y) = \mu(t)f(x, y; t)$ be a factorisation, where we suppose that always f is a probability density, so $\int_{\mathbb{R}^2} f(x, y; t)\ dx\ dy = 1$ for all t. For many models, f will not depend upon time and so $f(x, y; t) = f(x, y)$ will simply be a function of x, y. Similarly let $\mu_1(t, x, y) = \mu_1(t)g(x, y; t)$. Then μ_1 governs the rate of triggering, and to stop a runaway process, we must have that $\int_0^\infty \mu_1(t)\ dt < 1$.

The log likelihood, given (z_j), is

$$\begin{aligned} \log L = \sum_j \log \Big(& \mu(t_j)f(x_j, y_j; t_j))[z_j = j] \\ & + \mu_1(t_j - t_{z_j})g(x_j - x_{z_j}, y_j - y_{z_j}; t_j - t_{z_j})[z_j < j]\Big) \\ & - \int_0^T \mu(t)\ dt - \sum_{j=1}^n \int_0^{T-t_j} \mu_1(t)\ dt, \end{aligned}$$

where $[\cdot]$ is the Iverson bracket, so for example $[z_j = j]$ is a random variable, equal to 1 if z_j is j, and 0 otherwise. The integrals are essentially normalisation factors.

This sum may look formidable, but we can use two tricks. Firstly, the E step computes the conditional expectation, and we can use the *linearity* of the expectation. See equation (5) in the appendix, for example. Secondly, at the maximisation step, we have parameters we wish to maximise, and typically will take partial derivatives and solve for when the derivates are zero. Differentiation is also linear, and so commutes with the expectation. Unfortunately, we have not been able to find a simple recipe that translates the EM algorithm into a general procedure, at least at this level of generality. However, for any given model, it is typically rather simple to follow the EM algorithm.

We also note that in the point process (see [24, page 1402] for just one example) literature, one often finds algorithms with steps labelled the "E step" and "M step", but, comparing with our derivations in the appendix, these really do not have much to do with the "E" and "M" steps of the (abstract) EM algorithm. We will avoid such terminology.

2.1 Hawkes processes

Suppose we ignore any spatial component, so the intensity is

$$\lambda^*(t) = \mu(t) + \sum_{t_i < t} \mu_1(t - t_i).$$

The original Hawkes process takes $\mu(t) = \mu$ to be constant, and sets μ_1 to be exponential decay, say $\mu_1(s) = \theta\omega e^{-\omega s}$. Such models have been long studied by the financial Mathematics community, see the nice overview [17], for example. It is worth noting that [17, Section 4] warns of the complexity and difficulty of fitting such models to real data: we of course are trying to fit much more complicated models to data! We have found that thinking about Hawkes processes is a great way to understand the more complicated, spatial models this paper develops. The lecture notes [30] are a gentle and useful introduction.

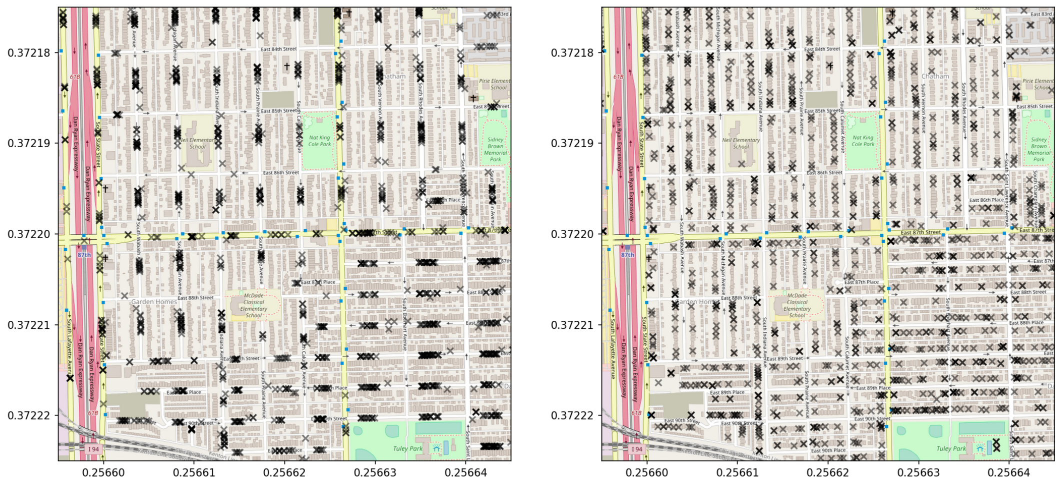

Figure 1: A section of the Chicago data: notice the clustering of events along city streets in the left-hand plot. The right-hand plot shows artificially "redistributed" points, each aligned to a building. Here and in the rest of the paper, we look only at Burglary crime, and mostly focus on the North side of Chicago.

The paper [18] develops the EM algorithm as applied to a Hawkes process; we repeat this derivation, with further details, in Appendix A.5. In `open_cp` we implement this model, and recreate the work leading to [18, Figure 1]. This demonstrates one of the problems with parameter estimation here: when the "clusters" coming from the trigger function are dispersed (so ω^{-1} is large, relative to μ^{-1}) it can be very hard to distinguish background events from triggered events. Indeed, our recreation of [18, Figure 1] typically fails[1] when $\omega^{-1} = 10$.

3 Trial data

We illustrate the models by using open crime data available from the city of Chicago, [4], as used in [33]. See Figure 1 for example plots of the data. We explore this dataset in detail in [7] where we argue that the coordinates are only approximate, and are deliberately clustered to anonymise location. We downloaded this dataset in 2017, while [33] used an earlier version of the data which displays slightly different location clustering, see the plot in [33] and the extended discussion in [7].

Given that this data is artificially clustered, we also use "redistributed" data from [7]. The original points have been moved a small distance around the street network, and then assigned to the nearest building (see [7] for a fuller discussion). Police data in the UK, especially for Burglary events, is often geocoded to buildings, and so we hope this second dataset will be more representative of (at least certain sorts of) real-world data.

4 Grid based models

The model in [24] is a grid-based, parametric model, for which the EM algorithm is easier to motivate. The model considers a grid laid over the study area, and in each grid cell k we assume that we have a Hawkes-like model, which is independent of other grid cells. Thus, if events in

[1]See `https://github.com/QuantCrimAtLeeds/PredictCode/blob/master/notebooks/sepp_2_testbed.ipynb`

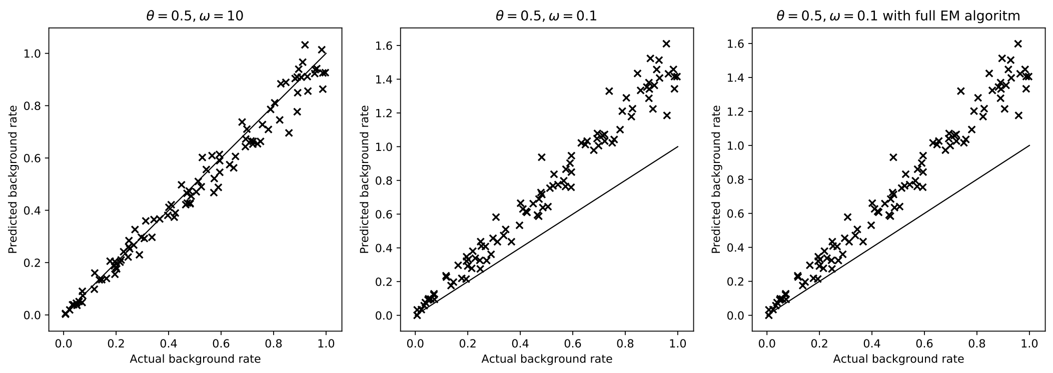

Figure 2: Fitting the grid based model to simulated data, Section 4.

cell k occur at times $t_1^{(k)} < t_2^{(k)} < \cdots < t_{n(k)}^{(k)}$ then the intensity for cell k is

$$\lambda_k^*(t) = \mu_k + \sum_{t_i^{(k)} < t} \theta \omega e^{-\omega(t - t_i^{(k)})}.$$

Notice that the background rate μ_k do not vary in time, but does depend on the cell k, while the triggering parameters θ and ω are assumed to be constant in both time and grid cell k. Appendix A.6 presents a careful derivation of the full EM algorithm in this case, and explains the assumptions which lead to the simplified algorithm presented in [24, Section 2.2].

This algorithm is implemented as the `seppexp` module in `open_cp`. The `sources.sepp` module in `open_cp` allows simulating (using a "non-perfect" simulation method) data from these types of model. We simulated data from this specific model with $\theta = 0.5, \omega = 10$ and with μ_k randomly chosen uniformly from $[0, 1]$. Using either the full EM algorithm, or the simplified algorithm, we estimate $\theta = 0.51, \omega = 8.9$ and obtain a good match with μ_k, see Figure 2. However, with $\theta = 0.5, \omega = 0.1$, we estimate $\theta = 0.2$ and $\omega = 0.65$, and from Figure 2 we see that the background rate is over-estimated. That is, a maximum likelihood approach over-estimates the number of background events. We see no difference between the simplified EM algorithm (which assumes no "edge effects") and the full algorithm.

Unfortunately, we have not been able to obtain sensible results from this algorithm as applied to real data. The algorithm converges without issue, but seems to always report completely unrealistic triggering parameters. We use Burglary crime events from 2016 for the North side of Chicago, with a grid size of 150m, as in [24]. The model predicts $\theta = 2.13 \times 10^{-2}$ and $\omega = 18.32$, with a time unit of one day, so that $\omega^{-1} \approx 79$ minutes. The background rate has a maximum of 2.73×10^{-2}, so that θ is comparable to the background rate. This means that the triggering kernel gives an elevated risk above background for only a matter of hours. This seems unrealistic, as previous studies, [14], considered an increased risk of (near-)repeat burglary on the order of days or weeks.

Similar behaviour (with only a slightly longer value for ω^{-1}) is shown for the second dataset. We have repeated the study using the different regions of Chicago, and have obtained similar results for all, with the exception of the "Far Southwest" which has ω^{-1} of about 10 hours, and the "Southwest", where the algorithm fails to converge. Running the algorithm on other year's worth of data shows similar behaviour.

The failure of convergence just mentioned is due to a problem which we will meet again below, so it is worth exploring now. The log likelihood of the total data is

$$\sum_{k,i} \log \Big(\mu_k + \sum_{j<i} \theta \omega e^{-\omega(t_i^{(k)} - t_j^{(k)})} \Big) - \sum_k T \mu_k - \sum_{k,i} \theta (1 - e^{-\omega(T - t_i^{(k)})}),$$

(compare with Appendix A.1). Suppose that there exists k and $j < i$ with $t_i^{(k)} = t_j^{(k)}$. Then we have a term which is

$$\geq \log\left(\mu_k + \theta\omega e^{-\omega(t_i^{(k)} - t_j^{(k)})}\right) \geq \log(\theta\omega),$$

as $e^0 = 1$. We have a further summand which is negative and scales with θ, but no such "correction" for ω, and so as $\omega \to \infty$ the likelihood increases without bound. We find that the EM algorithm does exactly this: it estimates ω to be larger and larger until a numerical error occurs.

If we change the data by adding small random numbers to the timestamps (while maintaining the ordering) then the problem disappears. Examining the input data, we do indeed find some exact repeats: these might be data errors, or they could be due to the fact that timestamps are only approximate, and that the geocoding is partly anonymised.

This occurs because, throughout, we work with *simple* point processes throughout, see the discussion in [5, Section 8.2.7] for example; as [5] notes, we could deal with this through *marked* point processes, the mark giving how many repeats occurred at that time and place. This seems like an unnecessary complication which we have not seen explored elsewhere.

There are two possible problems with this model. Firstly, as we saw in the simulation study, when the actual value of ω is small, we can significantly over-estimate both ω and the background rate. Secondly, the model does not model interaction between grid cells, while clearly there is every chance of a burglar moving between different cells. We deal with the second objection in later sections; the first issue seems harder to address.

4.1 Understanding the repeat rate

Above, we found a small value of θ, and compared it to the background rate. An arguably better way to interpret θ is as follows. Each event adds a new trigger kernel at its location, and the total intensity of the trigger, which is θ, is exactly the expected number of new events the trigger will give rise to. As each of these events will give rise to further triggers, the total number of expected events, in total, which one event will subsequently trigger is

$$\theta + \theta^2 + \theta^3 + \cdots = \frac{\theta}{1-\theta}.$$

See Appendix A.4 for a rigorous argument. For example, with $\theta = 2.13 \times 10^{-2}$ found above, we expect each event to trigger 2.18×10^{-2} further events (that is, essentially none!)

4.2 With a truncated exponential kernel

If exact repeats in times can cause the estimation of ω to diverge to ∞, then we might worry that different, but rather close, timestamps might cause ω to be estimated larger than is reasonable. A small change we can make to the model is replace the "triggering" kernel $\omega e^{-\omega t}$ with

$$g(t) = \begin{cases} 0 & : t < t_0, \\ \omega e^{-\omega(t-t_0)} & : t \geq t_0. \end{cases}$$

This corresponds to "inhibiting" triggering from happening between events which are $< t_0$ in time apart. The details are worked out in Appendix A.6.1.

Again, on real data, this technique does not appear to work well. While we sometimes see $1/\omega$ get larger (e.g. the order of days) this is accompanied by θ being estimated as being vanishly small, suggesting almost no triggering effect. The second dataset behaves in the same way.

4.3 Allowing exact repeats

While our formal probability model does not allow exact repeats ($t_i = t_j$ with $i < j$), as we briefly explain in Appendix A.3, we can adapt the EM algorithm to deal with such data.

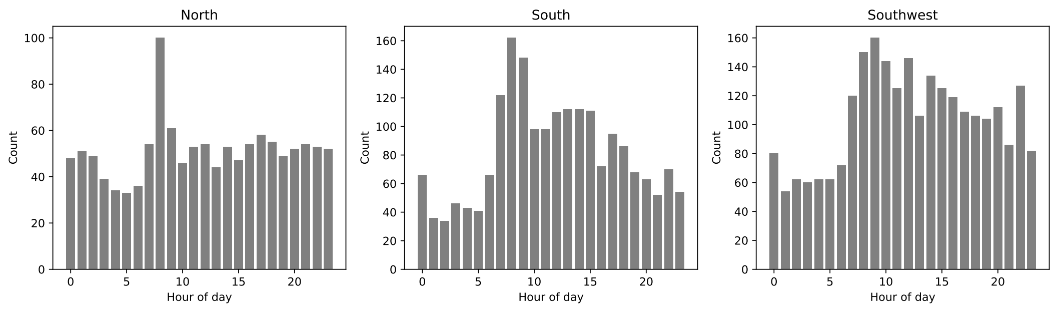

Figure 3: For three sides of Chicago, data from 2016, for Burglary event, the number of events per hour of the day.

This opens the door to “binning timestamps”, which from informal conversations with other researchers, is seemingly a data preparation task which is often carried out. There are perhaps good reasons for this, two of which are that: firstly, if we are making predictions for, say, the following day, then we perhaps should not be interested in exactly when a previous event occurred, only which day it occurred on; and secondly, we perhaps do not quite believe the accuracy of timestamps in our data (which are likely not really accurate to the nearest minute, for example).

Yet again, on real data, this technique does not appear to work well. There is a lot of sensitivity to exactly how we bin the timestamps (the length of the bins, and the offset from which bins start). Again, we often see that when ω^{-1} becomes large, θ is estimated as being vanishly small, suggesting almost no triggering effect. It is possible to “tune” the binning, on some datasets, but this begs the question of how we implement a *robust* method. Much the same behaviour is seen with the second dataset, which is perhaps not surprising, as we have not altered the timestamps, only the spatial locations.

From looking at the input data, we do not notice anything in particular which would obviously cause such a dependence on the offset of binning. The number of events per day is distributed much as a Poisson random variable, and the number of events per weekday is fairly constant across the working week, and a little lower at the weekend. The only slight surprise is shown in Figure 3, where we see that for the North side of Chicago, there are an unexpectedly large number of Burglary events timestamped to between 8am and 9am. Otherwise the number of events is fairly constant across the day, with a slight dip from 3am to 7am. We do not know if, perhaps, people awake up and discover an overnight burglary; or if events are logged at this time but actually happen early, perhaps due to some sort of “overnight working pattern” effect on behalf of the police (to give just two plausible reasons). Notice that we see somewhat different patterns for the South and Southwest sides.

Repeating the sort of experiment which lead to Figure 2, with $\omega = 10$, we find that binning the timestamps to the nearest hour works well, but that if we bin to the nearest 6 hours interval, then we again tend to systematically underestimate the background rate (that is, the effect is the same as making ω small).

There is no “cost” to allowing exact repeats, except for the tiny computational effort of adjusting the p-matrix. Henceforth, we shall allow exact repeats in time, for simplicity.

5 Non-parametric grid based models

Rather than assume a parametric form of the triggering kernel, we could instead take a more “data driven” approach, and use a non-parametric form for the triggering kernel. We explore this first for our grid based models, before later extending it to give a non-parametric form for the background as well, and allowing interaction between different grid cells.

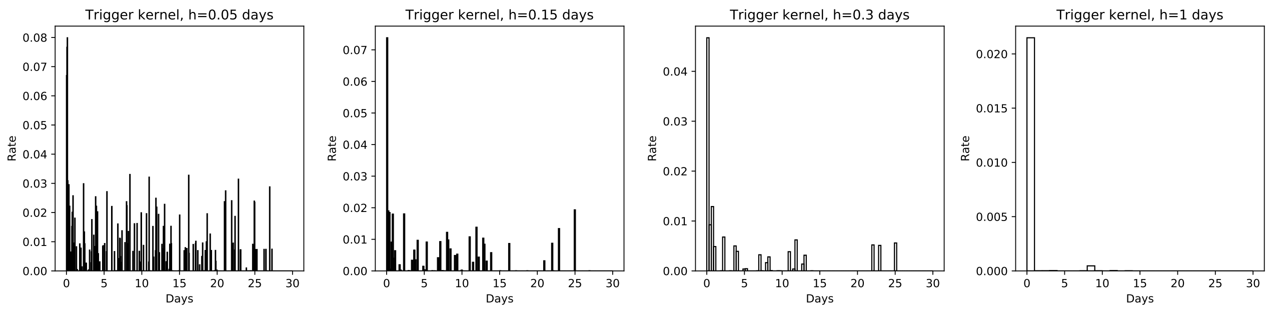

Figure 4: Histogram estimated trigger as the bandwidth varies.

5.1 A histogram estimator

A first step in this direction is to consider a “histogram estimator”, compare [35, Section 2.2]. Already here, the details of putting this into the EM algorithm are slightly non-trivial, see Appendix B. The particular model we use is detailed in Appendix C. We replace the exponential triggering kernel $\omega e^{-\omega t}$ by the “histogram”

$$g(t) = \alpha_r \quad \text{if} \quad r \in \mathbb{Z}, rh \leq t < (r+1)h,$$

where $h > 0$ is the *bandwidth* and (α_r) are positive values; we assume that $\alpha_r = 0$ for $r < 0$.

We have fitted this model to the simulated data we used in Figure 2; it works very well at a variety of different bandwidths, the histogram estimator well approximating the integral of the actual kernel over each interval.

For our Chicago case study, we again find quite some dependence on the bandwidth: with a small bandwidth the histogram is very noisy, but we do see repeating patterns, see Figure 4. At a large bandwidth, almost all the mass is estimated in the first bin (so an unrealistic almost exact repeat). Furthermore, the estimate of θ decreases notably as the bandwidth increases. It is interesting to note that the patterns are similar to those we find in the next section when using a KDE trigger, compare Figure 5.

The results for the 2nd dataset are similar qualitatively, but differ in exact form. In particular, with $h = 0.3$ days, we get probability mass only a two isolated points near 7 and 9 days; and with $h = 1$ we get a much more pronounced spike at around 9 days.

5.2 A kernel density estimator

Having introduced the histogram estimator, we now look at using a full-blown kernel density estimator, see Appendix C.1. In our analysis of the EM algorithm, we have ignored “edge correction issues” as it seems computationally infeasible to do so. However, we typically have a large interval of data (say, a year) to fit the model to, and from the above work, we expect that the trigger kernel will be short-lived by comparison.

We initialise the model in the same way as in Section 4, so initially the trigger kernel is exponential decay. However, for all subsequent iterations of the EM algorithm, we use a kernel density estimator. Experiments suggest that the initial condition is not terribly important, the end result of the iterative algorithm being the same.

When using a KDE method, we have to choose the bandwidth. As [35, Section 3.4] notes, there is no completely satisfactory way of proceeding. We can simply specify a bandwidth up front. There are various estimation procedures, from “plug-in” guesses (just looking at the sample variance of the data, and the size of the data) to various cross validation techniques. As Silverman argues, cross validation can be difficult in the presence of outliers or binned / discretised data. A further complication for us is the unusual use of variable “weights”, which we see arising naturally from the EM algorithm, Appendix B. It is far from clear how to adequately take account of these weights in either cross-validation or even a “plug-in” estimate.

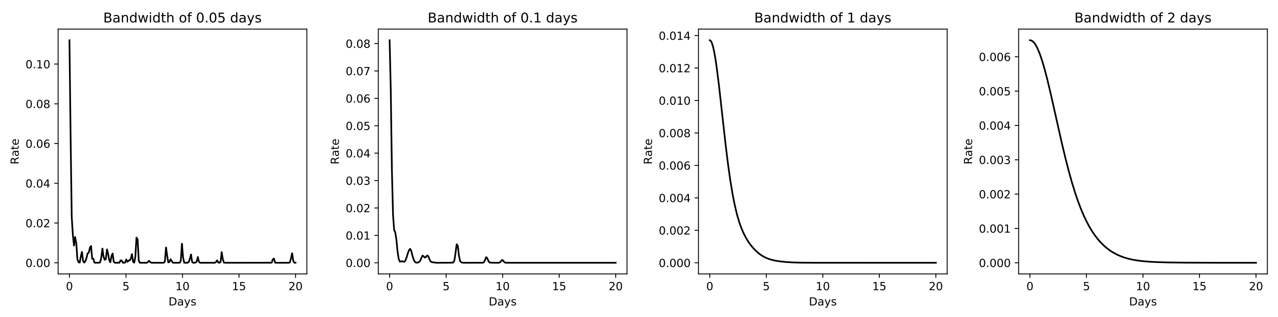

Figure 5: South side of Chicago, KDE estimated trigger kernels for differing bandwidths.

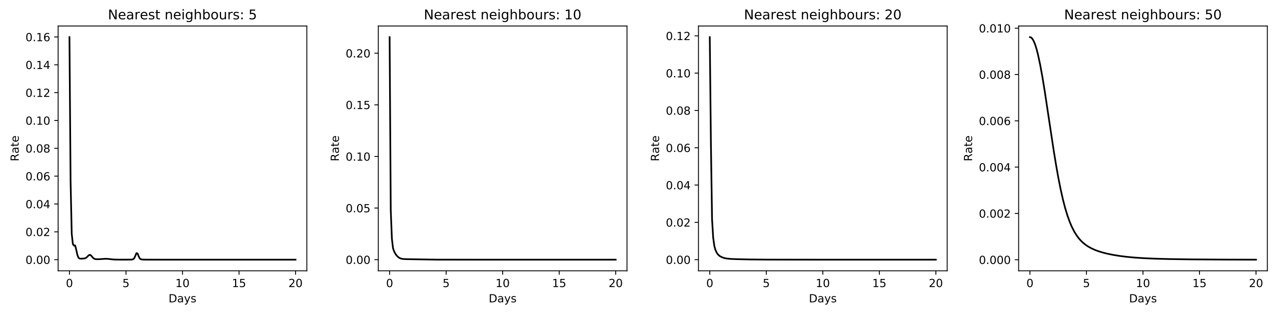

Figure 6: South side of Chicago, KDE estimated trigger kernels for different nearest neighbour choices.

For the moment, we shall simply choose the bandwidth by hand, with a view to attempting to learn something about the data. Given a reproducible pattern, we might then try to find a parametric form which captures it. For completeness, we also follow the idea of [23] and use a variable bandwidth, the local bandwidth chosen by the distance to the kth nearest neighbour, for various values of k.

For our Chicago case study, the results are interesting, in that for small bandwidths, we see temporally localised peaks in the trigger kernel, at 1, 2 and 4 days, and then further peaks at 7, 8 and 11, 12 and 13 days. As the bandwidth increases, these are smoothed away until we get a curve which smoothly decays in time; see Figure 5. The nearest neighbour variable bandwidth pattern is similar, Figure 6. It is also worth noting that the estimated value of θ decreases with increasing bandwidth, but stabilises for larger bandwidths.

With the second dataset, we get exactly the same results with a bandwidth of 1 or 2 days, but slightly different kernels for smaller bandwidths; in particular, for 0.1 days, we get a big spike at 9 days.

We have also fitted this model to the simulated data we used in Figure 2; it does not perform terribly well, tending the estimate a trigger distribution with a much longer tail than the real distribution, and being very noisy for smaller bandwidths.

6 Allowing spatial interaction

To move beyond this class of grid based model, we shall keep the grid "histogram" based estimator of the background risk, but will consider the possibility of events from one grid cell triggering events in another cell. We will work with a very general triggering kernel, but will assume that there is no link between space and time. Thus the intensity is

$$\lambda^*(t,x,y) = \mu_k + \sum_{t_i<t} \theta f(t-t_i) g(x-x_i, y-y_i),$$

where k is the cell which (x,y) falls in, but we now consider *all* events before time t, not just those in cell k. Here f governs the time triggering, and g the spatial triggering.

We showed previously that "exact repeats" in time can cause a maximum likelihood estimation of parameters to estimate a parameter at ∞. This will now also occur for exact repeats in space, if g is parameterised in such a way that it can approximate a delta function. This is more troubling, as we expect exact repeats in space (because geocoding is likely to assign the exact same coordinates to a building which is burgled twice) and because artificial "near repeats" occur in our data: see the clustering we observed in Figure 1. Notice that ignoring exact repeats in space (by adjusting the p matrix) seems unreasonable: we actually do want to model the exact repeats we have just described. It hence becomes critical to ensure that exact repeats do not cause parameter estimates for g to diverge.

We shall start with exponential decay in time, $f(t) = \omega e^{-\omega t}$ and a modified Gaussian in space,

$$g(x,y) = \begin{cases} \alpha & : x^2+y^2 \leq r_0^2, \\ \beta e^{-(x^2+y^2)/2\sigma^2} & : x^2+y^2 > r_0^2. \end{cases}$$

So near the origin, g is equal to α, and further away is a Gaussian with shape σ. The parameter r_0 controls the change in behaviour, and will be chosen and fixed. β is a normalisation factor, see Appendix D.

The EM algorithm becomes a little more complicated here, for the details see Appendix D. The important thing to note is that in optimising the trigger kernel, we now need to consider all order N^2 relations between the different event times, as opposed to partly aggregating by grid cell. Thus these algorithms take longer to run.

A similar model was considered in [22] (but a *marked point process* in order to model different crime types) where the background was estimated by a fixed bandwidth KDE method. The issue of exact repeats is discussed in this paper, and the offered solution is to set $p_{i,j} = 0$ if $t_i = t_j$ or the location of events i and j agree. As argued above, we find this unrealistic.

We performed a similar experiment to that which lead to Figure 2. Here we simulate background events as being distributed uniformly in each grid cell, according to the background rate in that cell, and then simulate triggered events from the actual distribution we are fitting to, but with $\alpha = 0$. The EM algorithm again fits well to this model. We estimate the parameters fairly precisely, and the background rate is estimated well, up to a similar error as we saw in Figure 2.

For real data, we perform an initial experiment with $r_0 = 5m$, and estimate $\alpha \approx 8.19 \times 10^{-3}, \sigma \approx 6.66$ and $\theta \approx 0.278$. Thus there is certainly repeat-like behaviour, but it is very tightly grouped in space (almost all the probability of the trigger kernel is within 20m of the triggering event, which seems unrealistically small). The estimated value of ω is now small, with $1/\omega \approx 153$ days! With the second dataset, we obtain a smaller value of σ, a somewhat smaller $\theta \approx 6.67 \times 10^{-2}$ and now $1/\omega \approx 254$ days. Needless to say, this is an unreasonably long decay time!

6.1 Interpreting fitted parameters

As we now have a completely spatial model, some care is required in interpreting values of the intensity. If we think back to equation (1) then we see that if we integrate λ^* over a space-time region, we obtain a value which is the expected number of events we'd observe in that region. Suppose we care about the number of events we'd see in a single grid cell in a single day, and let's suppose that the space trigger component has support well contained in a grid cell (for this model, for real data, this seems plausible). Background events would contribute $\mu_k A$ events where A is the area of the grid cell, while an event which occurred t time in the past, and close enough to the grid to matter, would contribute around

$$\theta \int_t^{t+1} f(t)\ dt = \theta e^{-\omega t}\big(1 - e^{-\omega}\big)$$

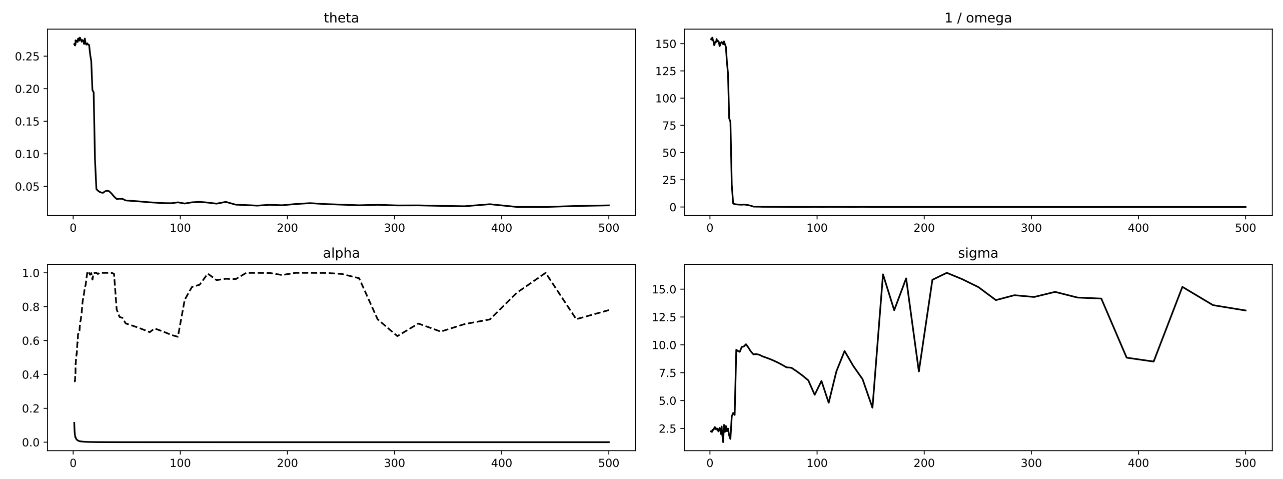

Figure 7: North side of Chicago, varying r_0 in the model described in Section 6. For the plot for α, in dashed, we plot $\alpha\pi r_0^2$ which is the total probability of the "cap" of g.

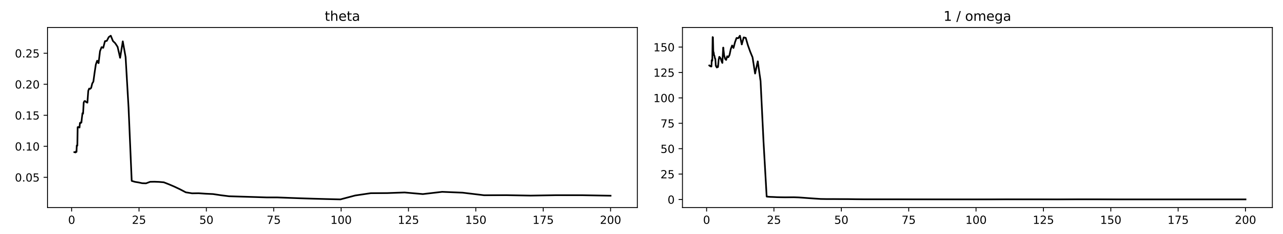

Figure 8: North side of Chicago, varying r_0 in the model described in Section 6 without a Gaussian decay term.

extra events, on average. However, to make sensible use of this information requires knowing the distribution of past events.

As we are now seeing a trigger with a long tail, we have found it most useful to actually form *predictions* based upon the model. This is explored more in Section 8 below; but in the most simple form, it simply involves evaluating the intensity function at a point in time, and different points in space. How much the resulting risk surface differs from just the background rate gives a good indication of how much the triggering is making a real difference, as applied to the actual data.

For the model we fitted, there is considerable variation in the resulting predictions, both as the prediction time changes, and against the background rate. We explore below whether these predictions have much to say about reality.

Figure 7 shows how the estimated parameters vary as we change r_0. Both θ and $1/\omega$ decrease rapidly once r_0 is greater than around 20m; this is too small a value to be realistic, and the change is so sudden, that we conjecture that it is caused by the artificial clustering seen in the input data. The dashed line in the plot for α is the ratio of α to $\alpha_{\max}$. When this is close to 1, we are assigning almost no probability to the gaussian decay part. We see that σ remains small, even for rather large r_0.

For the second dataset we see similar behaviour, but θ and ω^{-1} decay more quickly. (This removes evidence for the behaviour being caused by the artificial clustering.) A qualitative difference is that $\alpha\pi r_0^2$ increases to 1 more slowly, between 0 and $r_0 = 200$m; similarly $\sigma \approx 10$m in this region.

In summary, we seem to actually gain rather little from the Gaussian decay term. It is easy to remove the Gaussian term, and simply let the triggering be uniform in a disc. We then need estimate only θ and ω. Figure 8 shows the result. We see the same cliff-edge like behaviour at around 20m.

For the second dataset, we see slightly different behaviour: ω^{-1} decays between $r_0 = 0$ and

$r_0 = 15$m, with a spike around 7m. θ behaves in a rather similar way, and we do not see the increase and sudden cliff-edge of Figure 8.

6.2 Histogram in time

We implemented a histogram estimator for the time trigger f, using g constant on a disc of radius r_0, as before. For small values of r_0, we get interesting repeat behaviour from the histogram, but such behaviour seems very dependent on the bandwidth. For larger values of r_0, such at 100m or 250m (which seem realistic) we obtain repeat behaviour only of the order of two days. We note that, again, this model fits simulated data drawn from the model well. Technical details are in the short Appendix D.2.

For the second dataset, with our initial investigation with $r_0 = 20$m, we find that the overall triggering risk is about 10 times smaller, regardless of the bandwidth used for the histogram. With $r_0 = 100m$ however we see rather similar trigger behaviour, with a longer lasting trigger. Similar behaviour occurs for $r_0 = 250m$. This change is likely due to the less "clustered" nature of the data, meaning that a large r_0 value leads to more repeat behaviour being captured.

6.3 KDE in time and space

It hence seems profitable to try to use a full KDE estimator for both f and g. We now run into computational challenges. If we have N points then we have order N^2 possible differences $t_j - t_i$ for $i < j$ (in the case of f), and the KDE method requires forming a sum over all these points, for each function evaluation. As the next iteration of the EM algorithm requires computing the p-matrix, order N^2 terms, we end up with an order N^4 algorithm. This is too slow. There are two possible speed-ups:

- We form a "weighted" KDE, with weights $p_{i,j}$. If some of these numbers are very small (or zero) they can be safely ignored with little (or no) impact on the estimated function.
- We might *a priori* limit the support of f and g (e.g. to a few months and 500m, or some similar values). We could hence ignore $t_j - t_i$ larger than this value (and similarly for g). In practise, this could be achieved by setting the relevant $p_{i,j}$ value to 0, and then applying the previous idea.

We implement an adaptive, data-driven approach. For each column of the p-matrix, we list the entries in decreasing order, and form the cumulative sum. Any entry which is beyond a cutoff (we use 99.9%) we set to 0. This gives an automatic algorithm which leads to considerable speedup. However, the resulting algorithm is still very slow (e.g. on large simulated datasets). On the Chicago dataset it is slow, but acceptable; the first issue above appears to be the critical one as far as speed is concerned.

The basis of the EM algorithm is to introduce "hidden variables" which simplify the likelihood calculation, and then to iterate taking the expectation over the hidden variables, and then maximising the parameters. As is common in computational statistics, if we cannot compute the expectation, then it may be possible to *sample* from the distribution at hand. This leads to the "monte carlo E-step", see [21, Chapter 6]. An extreme form of this is to take one sample from the distribution of the hidden variables, the "stochastic EM algorithm". In our setting, this corresponds to, at each iteration, using the p-matrix to take a sample of which events are "background", and which are "triggered". We then use these identified events to maximise our parameters, and/or form a KDE etc. This is the approach taken in [24], following [38] (although they do not explicitly make connections with the EM algorithm).

This stochastic EM algorithm has the disadvantage of not using all the information we have about the branching process: we assign each point as either background or triggered, even if the current p-matrix suggests that both are equally likely. However, a large advantage is that we simplify and speed up the KDE, especially for the triggering, as we will now only have order N

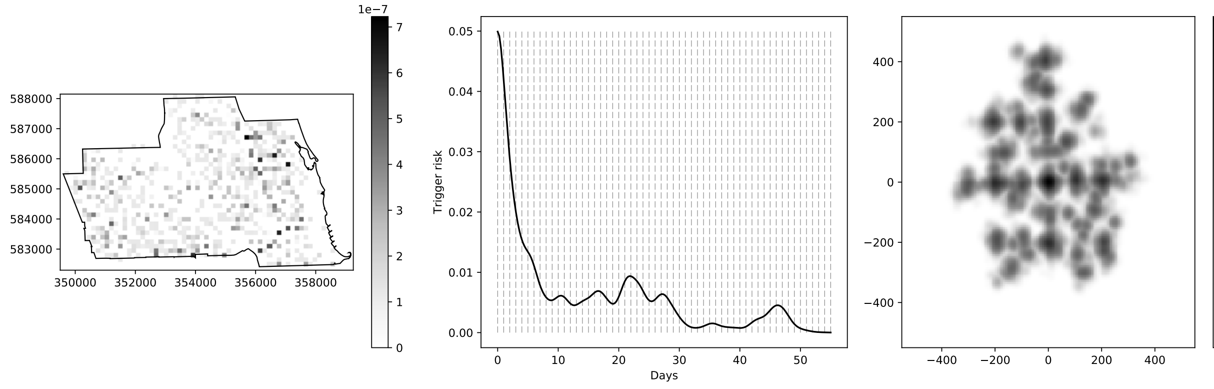
Figure 9: North side of Chicago, for the model described in Section 6.3, with a time bandwidth of 1 day, and a space bandwidth of 10 metres. From left to right we show the background rate, the time kernel f, and the space kernel g. For g we actually plot the surface $\log(e^{-20} + g)$.

points to consider, instead of N^2. However, as we shall shortly see, on real data, the stochastic EM algorithm can lead to different (and, frankly, incorrect) results, as compared to the "full" EM algorithm.

We fit simulation data (drawn from the same distribution as the model) and obtain a good match. As observed in previous studies (see [24] or [33, Figure 3]) the time component of the trigger tends to underestimate near 0. This is because the Gaussian based KDE method, even with reflection about $t = 0$, does not well capture the exponential decay which we use in the simulation. Both the full EM algorithm and the stochastic variant perform equally well.

We observe some very interesting behaviour from real data. Figure 9 shows an initial result, with f estimated with a bandwidth of 1 day, and g estimated with a bandwidth of 10 metres. We obtain $\theta \approx 0.34$. The vertical lines in the plot of f indicate each day; for example, we see a peak somewhere between 21 and 22 days. The estimate for g involves very small numbers, so to better visualise, we have shown the surface for $\log(e^{-20} + g)$.

As argued in Section 6.1, it can be very hard to interpret these plots, except in general terms. One possible visualisation technique is motivated by the stochastic EM algorithm: we can use the fitted model to form a p-matrix, and then take a sample of which events are background, and which are triggered. Figure 10 shows the result: we plot both the trigger "delta" (the vector $(t_j - t_i, x_j - x_i, y_j - y_i)$) and a (plug-in bandwidth) KDE, using the same visualisation strategy of plotting $\log(\epsilon + k)$ for some small ϵ, to visualise the density. In both the plot of g, and the sample, we see a clear grid-like pattern. This is a reflection of the north/south and east/west grid layout of streets in Chicago, together we the observed "clustering" from Figure 1. That we see such a pronounced grid pattern should not surprise us, given the geography of Chicago, but whether the artifical clustering in the input data exaggerates this is unknown.

We note that if we continue to run the EM algorithm for further iterations (which is time consuming!) then the shape of the triggers does not qualitatively change, but θ (the overall trigger risk) does tend to increase; after 100 iterations we have $\theta \approx 0.41$.

With the second dataset, Figure 11, we get a smaller value for θ, namely 0.17, and a rather different behaviour for the time trigger, with nearly linear decay on the order of 8 or 9 days. The space trigger does look similar, suggesting the "grid" pattern we observe here is due to the geography of Chicago, and not just the artificial clustering in the initial input data. A plot of the estimated trigger points, not shown, is similar to Figure 10 but shows less "clumpy-ness" in space. If we run the EM algorithm for further iterations, then again θ increases (to 0.20 after 100 iterations) and the time trigger develops a slight bump at 4 days.

We fitted the same model using the stochastic EM algorithm, and obtained quite different results, see Figure 12 (the sample of trigger points plot is similarly different). The estimated value of θ is 0.12, about half that we saw before. The time kernel is similar is shape, but with

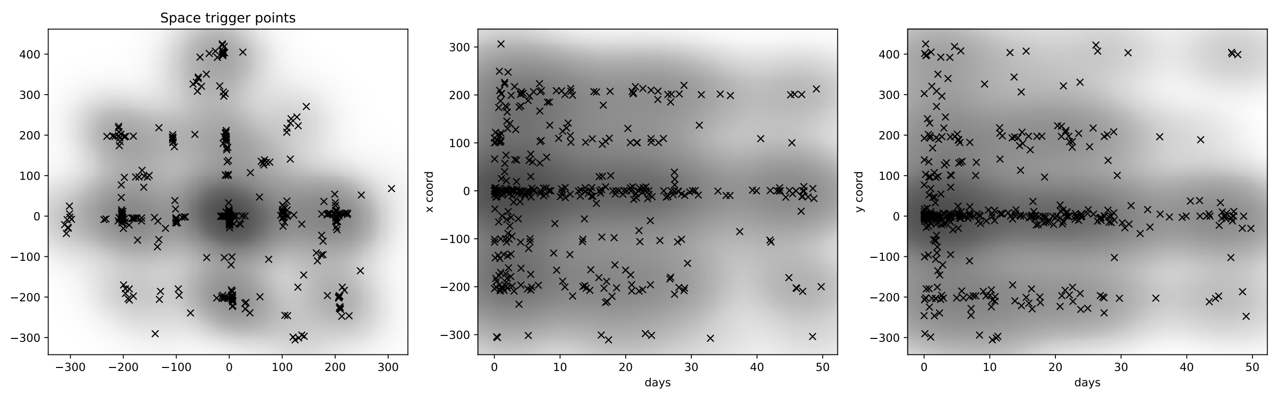

Figure 10: For the model from Figure 9, sampled trigger points.

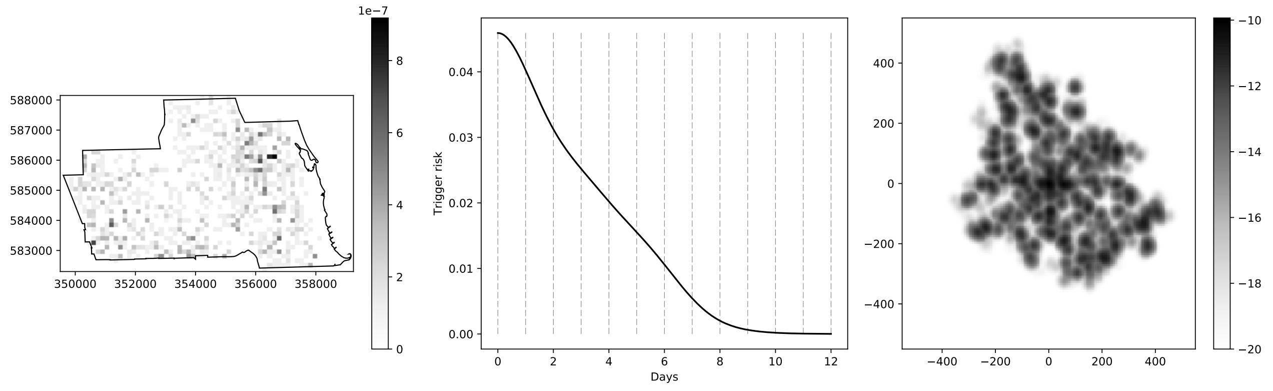

Figure 11: As Figure 9 but with the second dataset.

different quantitative properties. The space kernel is also similar in shape, but with a much smaller support, and is much less symmetric. We believe that these differences are caused by exactly the problem we identified above, namely that by sampling, we assign for definite that each point is either background or triggered, instead of allowing a mixture. Taking the full expectation allows us to take account of the probability that an event is triggered, and hence allow that event to contribute some weight to the KDE. The second dataset exhibits similar behaviour: again θ is smaller, at 0.052, the time trigger is similar, but the space trigger is much more localised in space.

Unfortunately, these results seem to (yet again) depend a great deal on the bandwidths chosen. Figure 13 shows the same model as Figure 9 but now g uses a bandwidth of 50 metres. The resulting estimate for g is much smoother (of course), but also θ becomes 0.058, while f

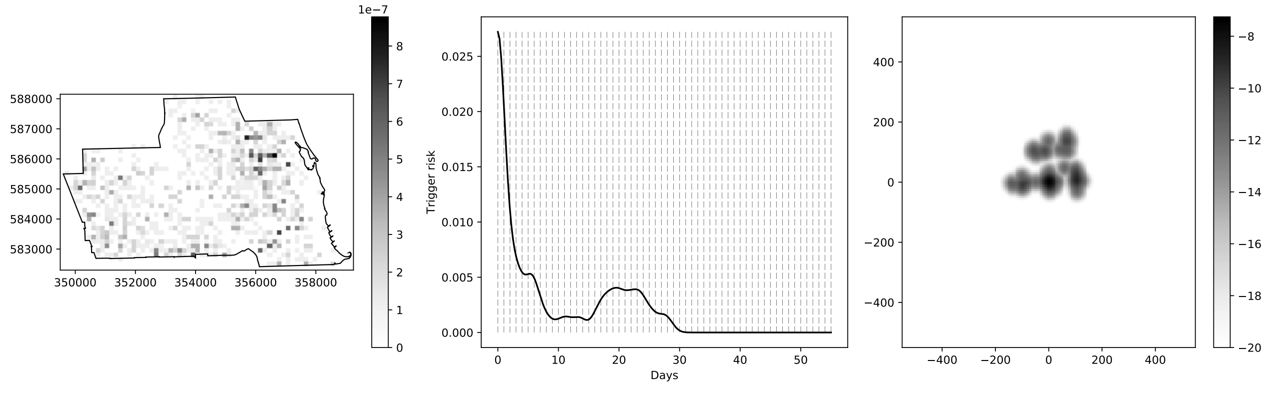

Figure 12: North side of Chicago, as in Figure 9, but model fitted using the stochastic EM algorithm.

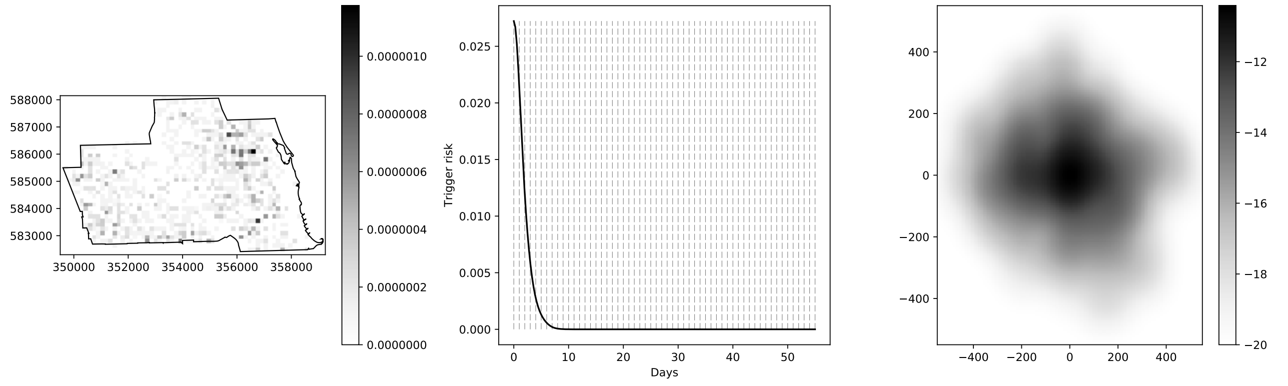
Figure 13: North side of Chicago, for the model described in Section 6.3, with a time bandwidth of 1 day, and a space bandwidth of 50 metres.

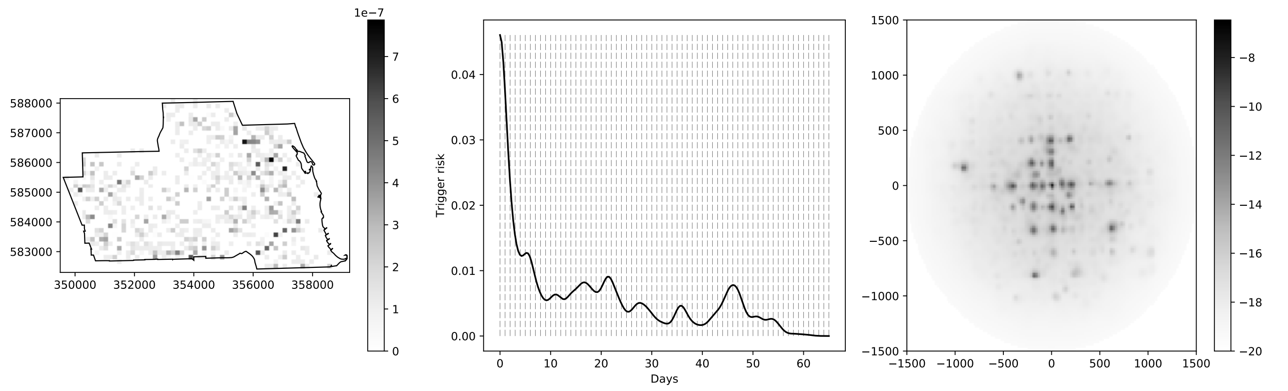
Figure 14: North side of Chicago, for the model described in Section 6.3, with a time bandwidth of 1 day, and the space kernel estimated with a nearest neighbour variable bandwidth, $k = 15$.

now smoothly and quickly decays in the order of a week. With our second dataset, we get $\theta = 0.036$ but rather similar kernel shapes. By adjusting the bandwidths, we obtain a rather wide range of behaviours.

The authors of [23] and [38] suggest using a variable bandwidth KDE with the bandwidth for each point chosen by the distance to the kth nearest neighbour; [23] uses a value of $k = 15$. See Appendix C.2 for details. We use the same idea for the space kernel g, keeping the time kernel as being a fixed bandwidth of 1 day. The results are shown in Figure 14; here $\theta \approx 0.38$. Of note is the very long fall-off in time, and the large support for the space kernel g. We artificially forced the support of g to be within a disk of radius 1500m, to speed up the algorithm (which is still extremely slow). Notice that the support of g really does extend to the edge of this disk, and that it has a strong pattern. The sampled trigger points, Figure 15, similarly shows a very strong pattern; notice that the spatial extent is somewhat larger than that seen in Figure 10, though the pattern is rather similar.

Again such behaviour seems hugely dependent on the chosen bandwidth. By varying the value of k, or the fixed bandwidth used for f, we observe a large range of different behaviours and, especially, a large range in the value of θ. For the second dataset, Figure 16, again shows very different behaviour, again being much closer to Figure 11. The sampled trigger points, not shown, shows that relatively few events are identified as being triggered; indeed, with $\theta \approx 0.04$, this is to be expected.

Furthermore, and of particular interest given the algorithm used in [23], we ran the same model as lead to Figure 14, but using the stochastic EM algorithm. The speed of execution is far quicker, but the result, Figure 17, is radically different, with a huge east/west bias in the triggering kernel. The sampled trigger points (not shown) show a similar bias. It is extremely

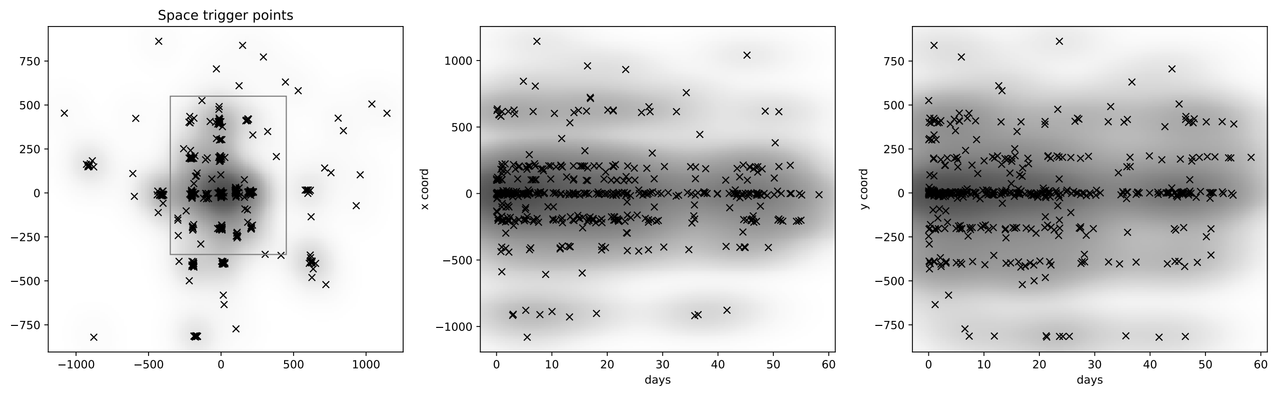

Figure 15: Sampled trigger points from the model fitted as in Figure 14. In the left plot, outlined is the region we show in the corresponding plot of Figure 10.

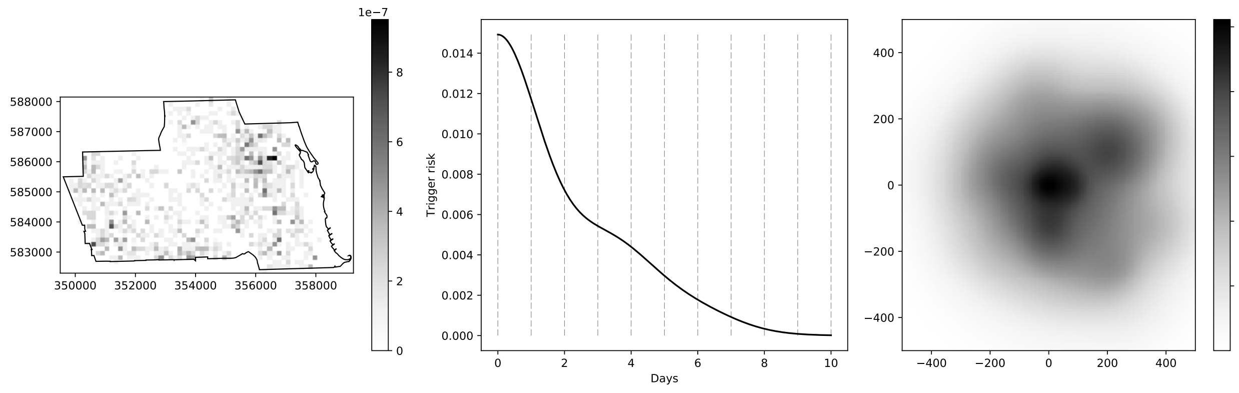

Figure 16: As Figure 14 but for the second dataset.

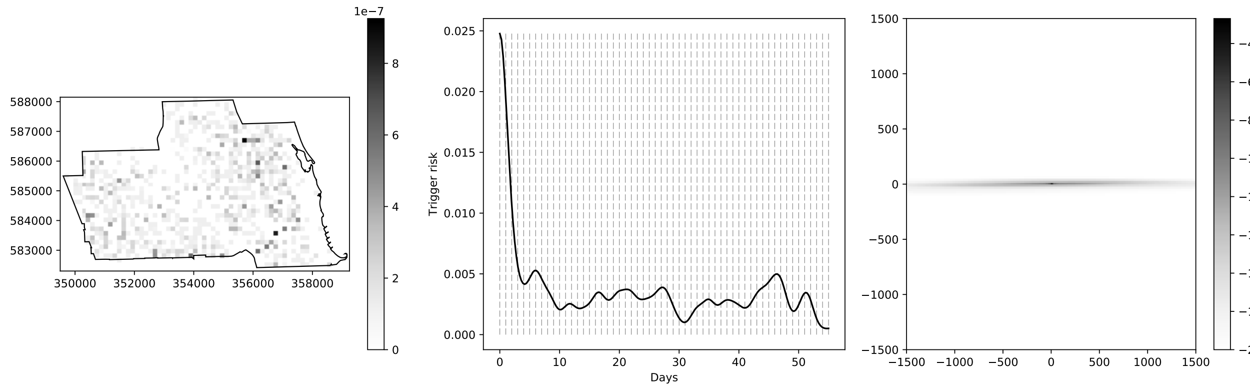

Figure 17: North side of Chicago, for the model described in Section 6.3, with a time bandwidth of 1 day, and the space kernel estimated with a nearest neighbour variable bandwidth, $k = 15$. Uses the stochastic EM algorithm.

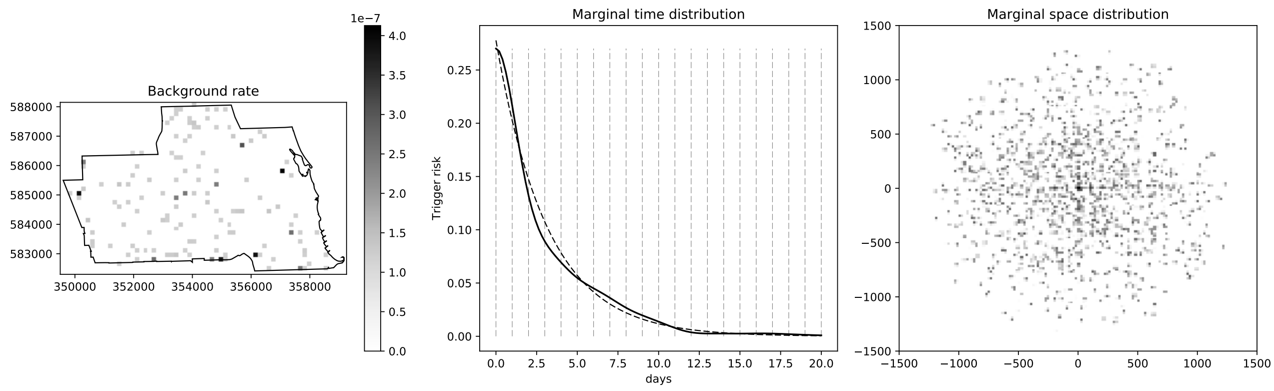

Figure 18: North side of Chicago, for the model described in Section 6.4. We divide the spatial coordinates by 20, and then estimate the combined time/space kernel with a bandwidth of 1.

interesting that a similiar "bias" was found in [33]; there, the authors proposed the solution of enforcing radial symmetry on the triggering kernel. Our results here suggest that the bias is actually caused by using the approximation of the stochastic EM algorithm instead of the "full" EM algorithm.

We cannot present the same findings for the second dataset because the stochastic EM algorithm refuses to run for many iterations before stopping with numerical errors. This occurs because the sampled number of triggered points becomes too small.

6.4 Joint dependence in trigger

Instead of supposing that the trigger density factorises as $f(t)g(x, y)$ we can instead use a KDE to estimate the joint time/space density. This approach is close in spirit to [23]. As time and space have different dimensions, we first scale the x and y coordinates by a factor of 20, and then treat the data as being three dimensional, with a bandwidth of 1. Figure 18 shows the result, showing marginal distributions (that is, integrate out the other variables). For the marginal time kernel, we fit an exponential (dashed line) which shows very good agreement. The marginal space kernel is visualised in the same way as before (plot $\log(e^{-20} + g)$) and shows the by now familiar pattern.

Figure 19 shows slices of the triggering kernel at varying time points, each normalised by the marginal time kernel at that time. We see that the space kernel changes in time, becoming more concentrated as time increases. This seems counter-intuitive, as we would expect trigger events to become more dispersed as time went on (to be consistent with the criminological basis

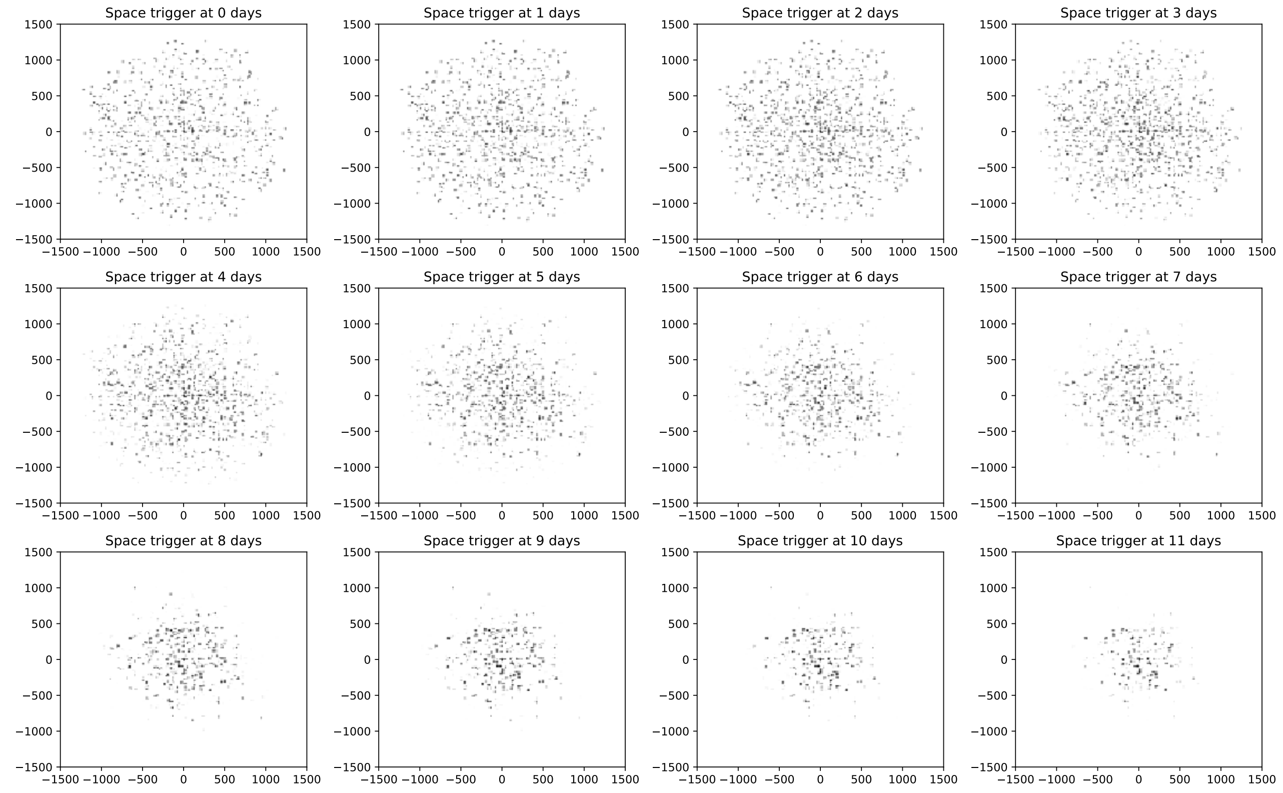

Figure 19: As Figure 18, showing slices of the triggering kernel at varying time points.

of repeat and near-repeat behaviour).

Finally Figure 20 shows the sampled triggered events. This is similar to Figure 10, but the space locations are a little more concentrated (within 300m, instead of 400m) but the grid pattern is a little less evident. The plots against time show the increasing spatial concentration with time that we observed before.

For the second dataset, the results are rather similar. The fit of an exponential to the time kernel is a little worse, and the marginal space kernel looks very similar, but, again, with a less pronounced "grid-like" pattern, and with a smaller spatial distribution. The version of Figure 19 for the second dataset is also much the same, but again with a smaller spatial distribution, and without the long "tail" from 15 to 30 days seen in the right two plots of Figure 19.

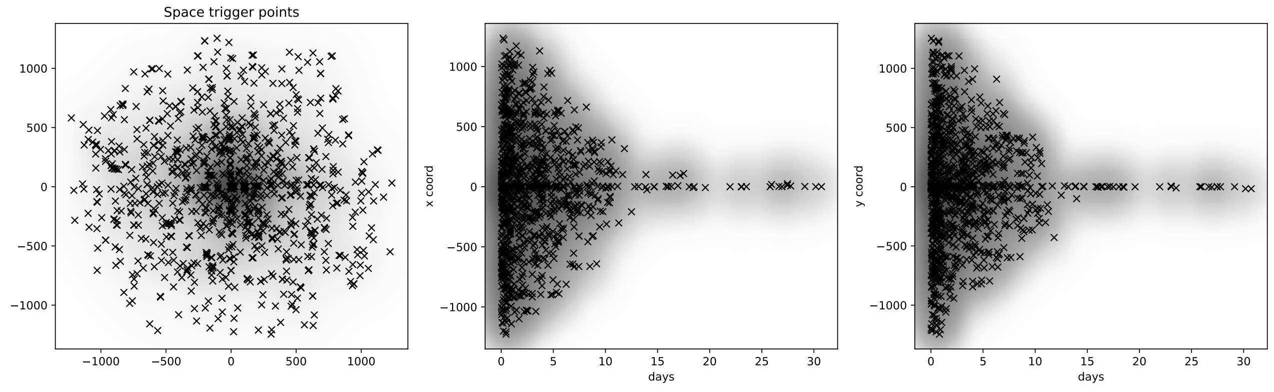

Figure 20: As Figure 18, showing the sampled trigger events

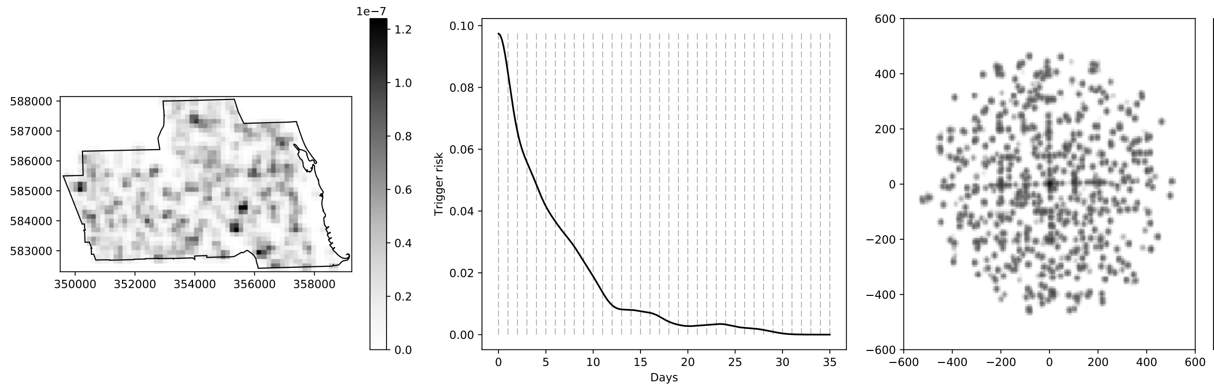

Figure 21: For the model described in Section 6.5, showing the background rate and the marginal time and space kernels, fitted using the same KDE method as in Figure 18, and with a bandwidth of 100m for the background.

6.5 Non-grid based model

The final step towards the model used in [24] is to stop using a grid to estimate the background rate. So our general model will be

$$\lambda^*(t,x,y) = \mu(t,x,y) + \sum_{t_i<t} \mu_1(t-t_i, x-x_i, y-y_i).$$

In [24] μ is set to to $f(t)g(x,y)$ and f,g are estimated via KDE methods. A problem here, as explained before, is that KDE methods are susceptible to "edge-effect", which will in particular badly effect f near the start and end of the time period we are interested in. As our ultimate aim is prediction, we actually need to project f forwards (a little) in time. There are no details in [24] as to how prediction was actually carried out. We instead prefer to assume a constant background rate in time. A more sophisticated analysis, which we do not explore here, would perhaps use time-series techniques to identify periodic and overall trend levels in the background rate.

So we work with

$$\lambda^*(t,x,y) = \mu f(x,y) + \theta \sum_{t_i<t} g(t-t_i, x-x_i, y-y_i),$$

where μ is the overall background rate, θ is the triggering rate, and f,g are probability densities. We will again estimate f and g by KDE methods, and we reflect g about 0 in the first variable to take account of time always being positive.

Applied to synthetic data, as in Figure 2, the model fits well, with the exception that the time trigger is rather underestimated near 0 (judging from the plots give in [23] and [33] those authors encountered the same issue).

We used the same space/time KDE method as in Section 6.4, and for the background used a KDE with a fixed bandwidth of 100m. The results are shown in Figure 21 and Figure 22. Here we estimate $\theta \approx 0.58$. These results come from 20 iterations of the EM algorithm; unfortunately, with another 30 iterations, we find that θ decays, and the trigger kernel becomes more concentrated. Here the results of using the stochastic EM algorithm are nothing like as extreme as seen in Figure 17, being rather similar to the results of the full EM algorithm after 50 iterations.

As ever, by changing the bandwidths, we obtain different results. However, by contrast with previous sections, the results using the second dataset are rather similar. The main difference is in the analogue of Figure 22 where, again, the sample space trigger pattern is less clustered, and the right-hand plots only extend to around 18 days instead of closer to 30 days.

Following [24] more closely, we also used a nearest neighbour KDE method. Figure 23 shows the result, which is notably different to Figure 21, especially in the lack of a strong pattern in

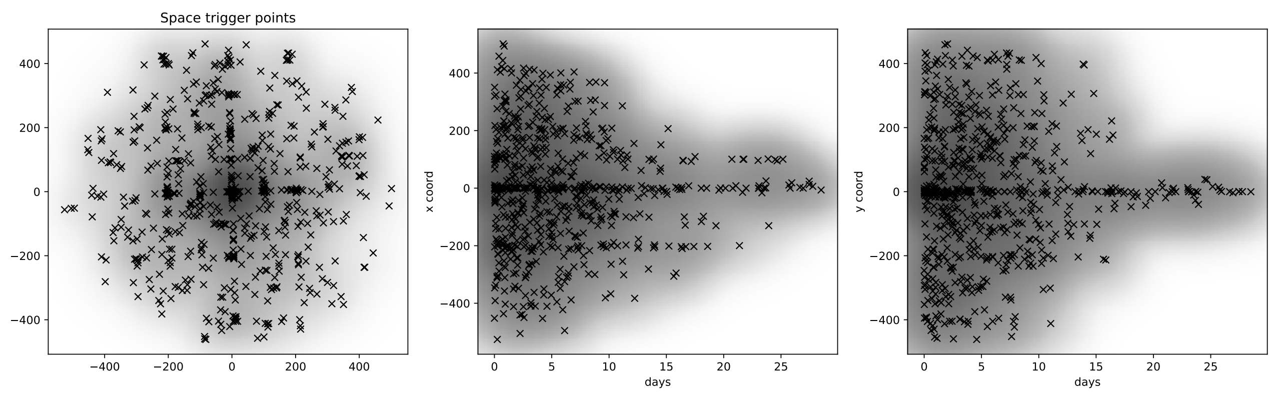

Figure 22: Sampled trigger points from the fitted model displayed in Figure 21.

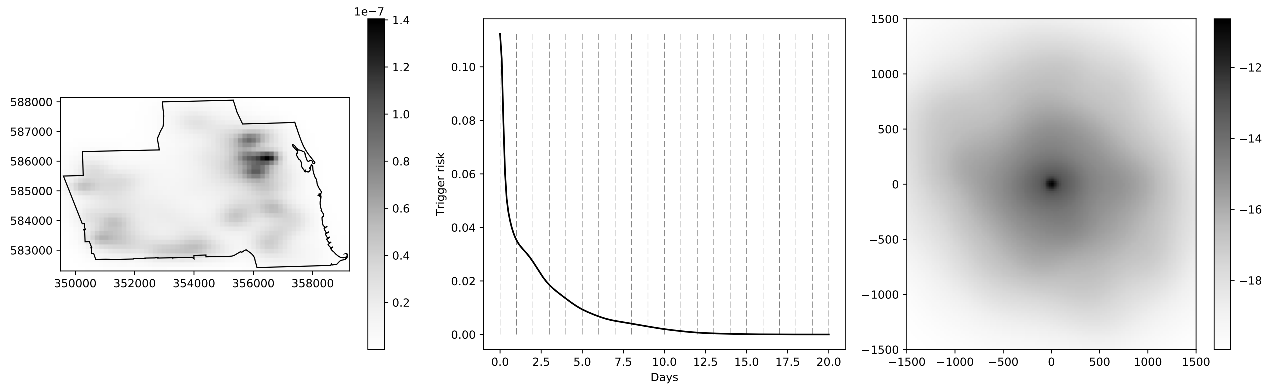

Figure 23: For the model described in Section 6.5, showing the background rate and the marginal time and space kernels, fitted with a nearest neighbour KDE method, with $k = 15$.

the marginal space trigger kernel. Figure 24 shows the sampled trigger points, which again are much less extreme than that seen in Figure 22.

We again found that the stochastic EM algorithm did not work well. With the default initial conditions we use, the algorithm tends to diverge quickly, predicting all events as being triggered. We also tried the initial conditions suggested by [33]. This leads to an initially more stable result, but after more iterations (say, 15) the fitting again diverges.

The second dataset shows similar behaviour, with the only difference really being that the estimated marginal spatial kernels are less concentrated near the origin (and similarly in the sampled trigger plots) which we expect, given the artificial clustering seen in the first dataset. In particular, the stochastic EM algorithm still diverges to estimate nearly every event as having been triggered.

6.6 With the trigger split in space and time

We now look at

$$\lambda^*(t,x,y) = \mu f(x,y) + \theta \sum_{t_i<t} g_1(t-t_i) g_2(x-x_i, y-y_i),$$

that is, assume that the trigger factors in space and time. We have two motivations for this: speed (as the full three dimensional KDE method is slow to fit, and slow to make predictions from) and Figure 19 which showed unexpected (and somewhat counter-intuitive) changes in the space kernel through time.

Figure 25 shows the result, with the background fit with a KDE of fixed bandwidth 100m, the time component of the trigger with a bandwidth of 1 day, and the space component with

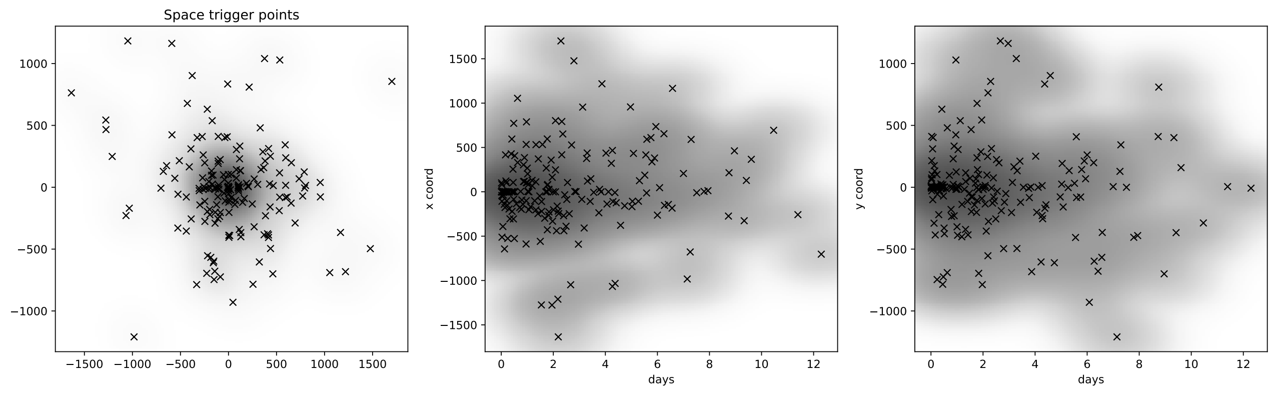

Figure 24: Sampled trigger points from the fitted model displayed in Figure 23.

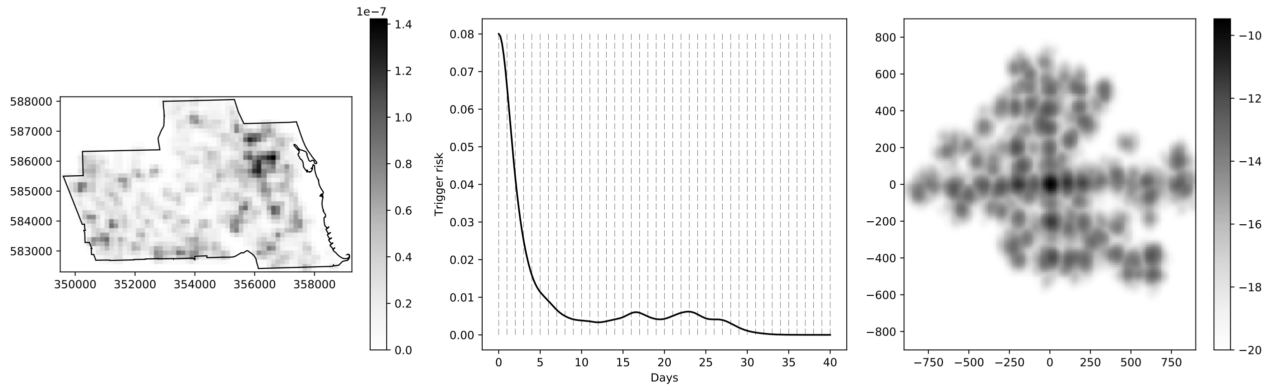

Figure 25: As Figure 21 but with the trigger kernel “split” in space and time. The background uses a KDE of bandwidth 100m, the time component of the trigger a bandwidth of 1 day, and the space component a bandwidth of 20m.

a bandwidth of 20m. Figure 26 shows the sampled triggers, which show less symmetry than Figure 22, for example.

Again, these results are rather dependent on the bandwidths chosen. With a background KDE of 30m, and a space component KDE of 30m, we obtain a fitted model which has almost no repeat behaviour. The stochastic EM algorithm, for the same settings as Figure 25 leads to a similar result, but with further iterations converges to a case where almost all events are estimated as having been triggered. The second dataset shows similar behaviour, though with a trigger kernel more tightly supported in time, and more widely supported in space. The stochastic EM algorithm behaves in the same way, as does changing the bandwidths.

Finally, we use a nearest neighbour estimator. We initially experimented with using $k = 30$ for time and space for the trigger estimator, and $k = 20$ for the background. This was very slow to compute, and lead to a very noisy time trigger kernel. The stochastic EM algorithm again diverged to predicting almost all triggered events.

We then looked at $k = 15$ for the spatial components, and $k = 30$ for the time trigger estimator (so more closely following [24, 33]). After 15 iterations of the EM algorithm, this lead to predicting almost all events as being triggered, and having a very noisy time trigger. The stochastic EM algorithm is similar. It is interesting to note that for the second dataset, for the stochastic EM algorithm, the time kernel for the trigger shows strong peaks at exactly 1 day and 2 days, and then at a little over 3 days, at about 5.5 days, and at 7 days. However, given that almost all events end up estimated as triggered events, it is hard to suggest that this pattern really “means” anything.

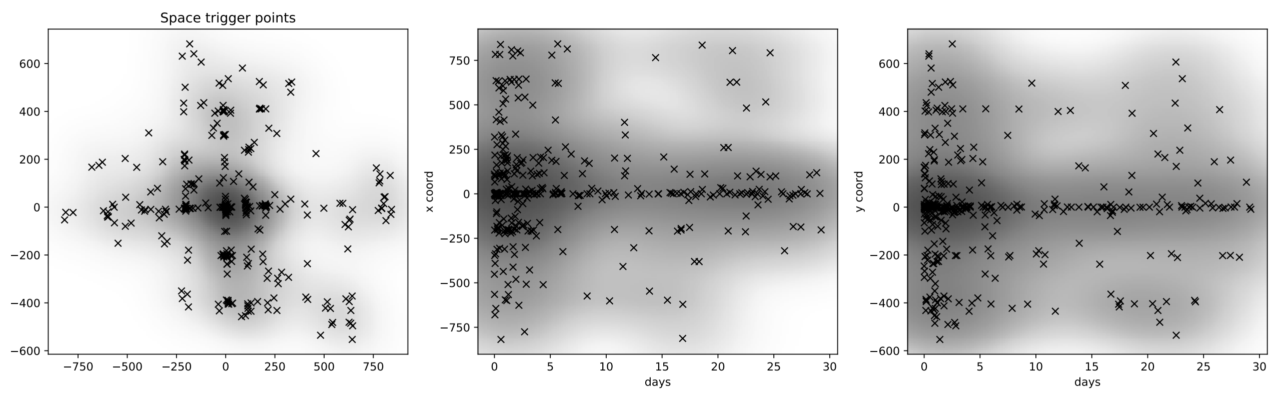

Figure 26: Sampled trigger points from the fitted model displayed in Figure 25.

7 Fixed kernels

At the other extreme from using a KDE method to "choose" a trigger kernel is to actually fix a trigger kernel. That is, we consider an intensity function of the form

$$\lambda^*(t,x,y) = \mu f(x,y) + \theta \sum_{t_i<t} g(t-t_i, x-x_i, y-y_i),$$

where now g is chosen and fixed, and we seek to fit θ, μ and f to the data.

We implement models where f is given by a grid-based histogram (that is, exactly as in Sections 4, 5 and the first part of 6) and is given by a KDE method (as in Section 6.5). Notice that the EM algorithm copes with such models seamlessly: we simply have no parameters to estimate for g. Note also that even for, say, a grid based f, there is no closed-form which gives the maximum likelihood estimate for f, μ and θ.

Yet again, we check that our algorithm fits well to synthetic data. For real data, we again split g into a time component, which is exponential decay, and a space kernel, which is Gaussian, namely

$$g(t,x,y) = \omega e^{-\omega t} \frac{1}{2\pi\sigma^2} \exp\Big(\frac{-(x^2+y^2)}{2\sigma^2}\Big).$$

We choose and fix ω, σ^2 before fitting.

We can compare this model to the classical "prospective hotspotting" technique, [3]. Following the more modern approach in, for example, [34], we estimate the risk intensity at a point (t,x,y) as

$$\sum_{t_i<t} g_1(\|(x,y)-(x_i,y_i)\|) g_2(t-t_i),$$

where g_1 and g_2 are kernels on $[0,\infty)$. (We could also, in principle, consider spatial kernels which had no radial symmetry, but we shall not do this here). [34] uses a linear decay for g_1 and exponential decay for g_2, while the original [3] uses an explicitly grid-based variant of this idea to compute g_1.

For our purposes, we note that this is exactly the same as our model, if $f=0$ (we may ignore θ as it is now just a normalisation factor). That is, our model can be thought of as prospective hotspotting with the addition of a background rate which is derived from the data.

When the background rate f is estimated from a grid, we have just two parameters to fix, σ^2 and ω. Figure 27 shows how θ varies with σ and ω^{-1}. We notice that to obtain noticeable repeat behaviour, we need to choose a very small spatial bandwidth ($\sigma = 10$ or so) and a long decay in time (100 days or so). Figure 28 shows the same for the second dataset. We see that the value of θ is somewhat smaller, but that the maximum value occurs for the somewhat more realistic values of $\sigma = 100m$ and an ω^{-1} on the order of days. That is, some of the repeat behaviour seems to be due to the artificially clustered nature of the first dataset.

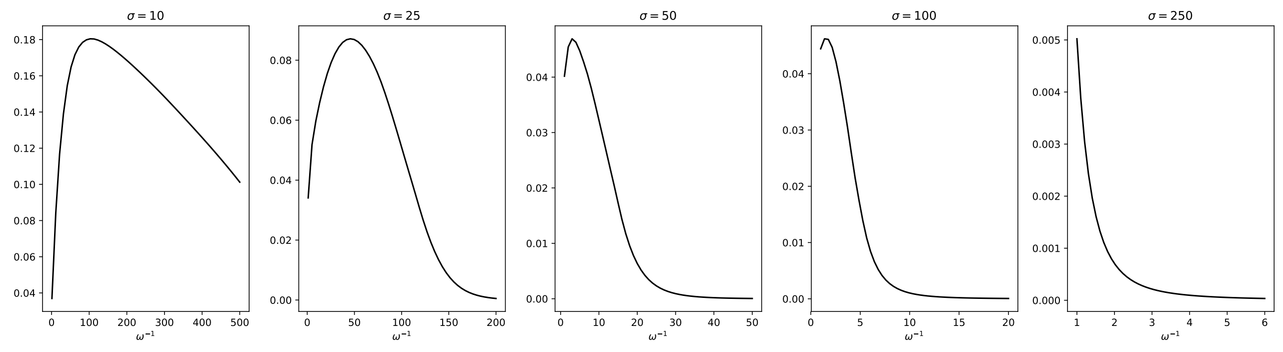
Figure 27: Results of fitting the model from Section 7 with a grid-based background. In each plot, the vertical axis shows θ and the horizontal axis shows ω^{-1}, the expectation of the time decay. We plot for varying values of σ.

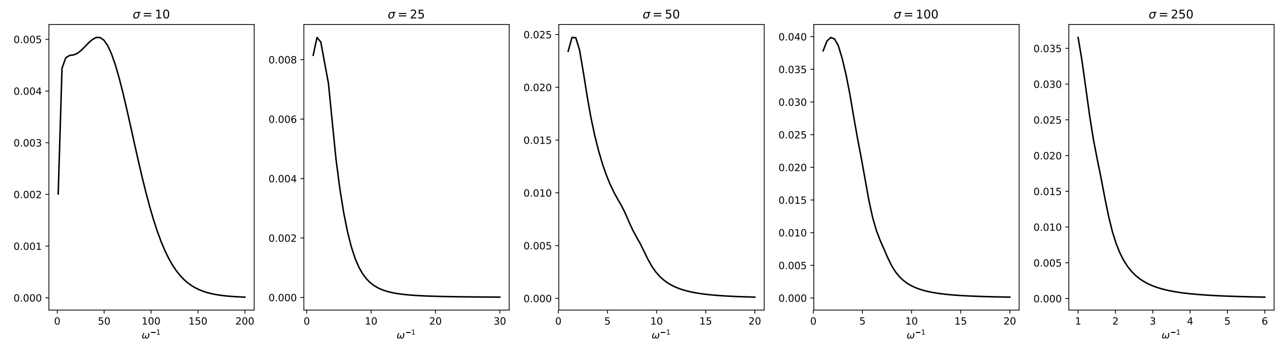
Figure 28: As Figure 27 but for the second dataset.

8 Making predictions

There is surprisingly little in the crime prediction literature ([23, 24, 33]) relating to the actual process of translating a model into a prediction. We have found the discussion in [37, Section 1] to be very helpful in this regard, and we summarise it now. We firstly assume that we have a process evolving in time only, with conditional intensity function λ^*. Given data up to a time T, we assume we have fitted a parametric model, or otherwise, and so have knowledge of $\lambda^*(t)$ for $t \leq T$. We also assume that it makes sense to extrapolate to values $\lambda^*(T+x)$ for $x > 0$ small. To give a concrete example, if we have a Hawkes process with trigger function $\omega e^{-\omega t}$, compare Appendix A.5, then we assume we have settled on values for μ, θ, ω, and so given past events (t_i) we have that

$$\lambda^*(t) = \mu + \sum_i \theta\omega e^{-\omega(t-t_i)} \qquad (t > T).$$

The assumption here is that there are no further events in the time interval $[T, T+x)$ which would add further triggering intensity.

We now use a standard procedure to simulate a point process starting at T with intensity $\lambda^*(t)$. In doing this, we must take account of the self-exicited nature of the process: so each new simulated event will change λ^* for times after that event. However, we do not change the parameters (μ, θ, ω in our example) or non-parametric functions. We simulate over the time range we wish to make a prediction for; perhaps the next day, for example.

We perform such a simulation many times, and, for example, may take an average of the number of events which occur to get a prediction of the total count of expected events. Notice that the self-excited nature of the process means that we cannot simply look at the value of $\lambda^*(T)$: the history (values of $\lambda^*(T-t)$ for at least small $t > 0$) will be important, as will the triggering process.

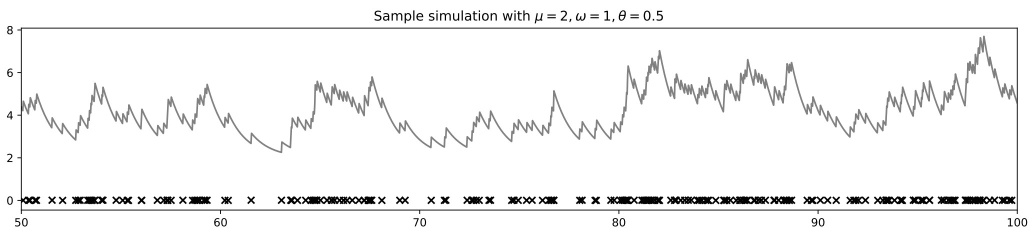

Figure 29: A simulated Hawkes process, showing the event times and the conditional intensity function.

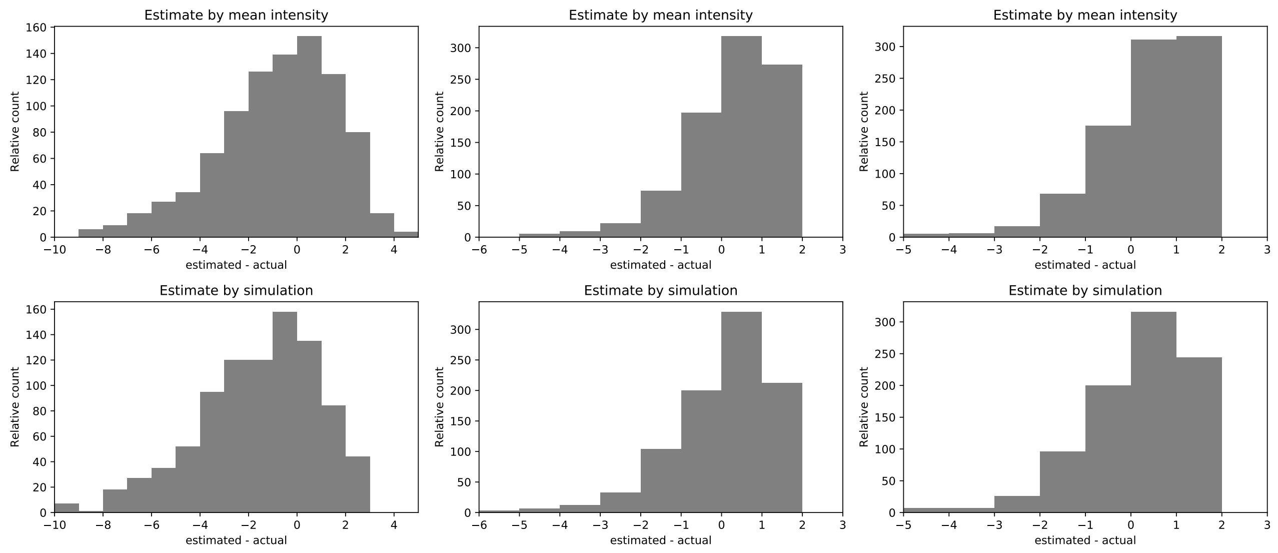

Figure 30: Predicting the number of events from a Hawkes process. Left to right, we have $\mu = 2, \omega = 1, \theta = 0.5$; $\mu = 1, \omega = 1, \theta = 0.2$; and $\mu = 1, \omega = 50, \theta = 0.5$.

We have performed this computer simulation for a number of different parameter combinations. Figure 29 shows a window on a simulation with $\mu = 2$ and $\omega = 1, \theta = 0.5$.

We also performed simulations with $\mu = 1, \omega = 1, \theta = 0.2$ (so less triggering) and with $\mu = 1, \omega = 50, \theta = 0.5$ (so a rather tighter "trigger interval"). In each case we simulated 1000 days of events, and then for $T = 100$ through 999 performed our prediction technique for the next day, the time interval $[T, T + 1)$. We performed 100 such trials for each day, and take the mean as our prediction. We also predicted the number of events simply by integrating the intensity function across this time interval. For the parameters we have chosen, we lose very little accuracy by simply evaluating λ^* at $T + 1/2$, instead of integrating it between T and $T + 1$. Figure 30 shows the results of the difference between the predicted number of events, and the actual number of events. In all cases, there is a large amount of error, with a longer tail on the negative side. This means we often under-estimate, but more rarely over estimate, which seems intuitively plausible, as it is hard to predict a large but rare number of triggered events. The simulation prediction method seems to perform better, but not by a huge amount, as compared to simply using the mean value of λ^*.

8.1 Spatial models

Extending this to models with a spacial component, there is no difficulty in extending the simulation. Suppose we are not interested in the exact occurance times, but rather wish to obtain a prediction for the likely location of crime events (again, over the next day, for example), this being the typical output of a crime prediction algorithm. What our simulation gives us is many samples of *points*.

We come then to issues discussed in [8]. In this paper, we have been concerned with specifying probability models and then fitting data to those models. For a "prediction" it is not quite clear what the (implicit) model is. In [8] we decided upon looking only at grid-based predictions, and essentially a multinomial distribution, with unknown total intensity. Thus our study region is divided into K cells, say $(A_k)_{k=1}^K$, and the "prediction" provides a probability (p_k) of crime occurring in cell A_k. We assume that crimes are independent, leading to a multinomial model. This paper assumes that crimes are *not* independent, of course, but it is hard to see how to include such information into a prediction– it is plainly the case that current prediction methods make no attempt to model this.

From each simulation i, we can calculate the number of events which occurred in cell k, say $n_k^{(i)}$. Notice that the simulation also tells us the total event count, $n_i = \sum_k n_k^{(i)}$, but current prediction systems do not concern themselves with trying to predict the total event count. If we make the simplifying assumption that each sample is from a multinomial, then the maximum likelihood estimation of p_k is simply

$$\hat{p}_k = \sum_i n_k^{(i)} / \sum_i n_i.$$

If we instead use λ^* directly, for example integrating it over both the time interval and each grid cell, then we obtain expected counts (n_k) say, and the maximum likelihood estimation is now $\hat{p}_k = n_k / \sum_k n_k$.

9 Prediction results

We used the `open_cp.scripted` Python module, which allows us to produce predictions, and to "score" them, in an automated way, using a simple Python script. We adapted the major classes of models discussed in previous sections to give predictions, following the paradigm above. Given the findings above, we have decided to integrate (approximately) λ^* over time rather than run a simulation. To be precise, we fit the model using the EM algorithm, and then produce grid based predictions:

- for purely grid based models, we need to integrate over the time kernel for each grid cell. We approximate this by sampling at 100, equally spaced, places over the time range which interests us (the next day, in this case);
- for more general models, we sample at 20 equally spaced places over time, and at 20 randomly chosen places in space. For speed, we only support trigger kernels which factor into a space component and a time component.
- finally, for the KDE based models, we further speed things up by converting the kernels into histograms, for the purposes of prediction. We typically use a bandwidth of 1/10 days and 10m, for time and space, respectively. From numerical experiments on our data, replacing continuous kernels with histograms in this way leads to a huge increase in speed at very little cost in accuracy for the final prediction.

We use the first dataset of Chicago data, for the year 2016. We make a prediction for each day in October, November and December, using all the data before midnight on that day to inform the prediction, and "scoring" the prediction against the reality of which events occurred the next day. For fitting the models, we use the data from January through September 2016 inclusive.

To "score" a prediction, we use the usual procedure (see [1] or [23, Section 5] for example) of fitting a "coverage level" (say, 10%) and select that fraction of grid cells, using the prediction to rank the cells from most risky to least risky. We then compare the number of crime events on the next day with the number of events we have "captured": events which have fallen in the selected grid cells. This is termed the "hit-rate" in the literature. Following our suggestion in

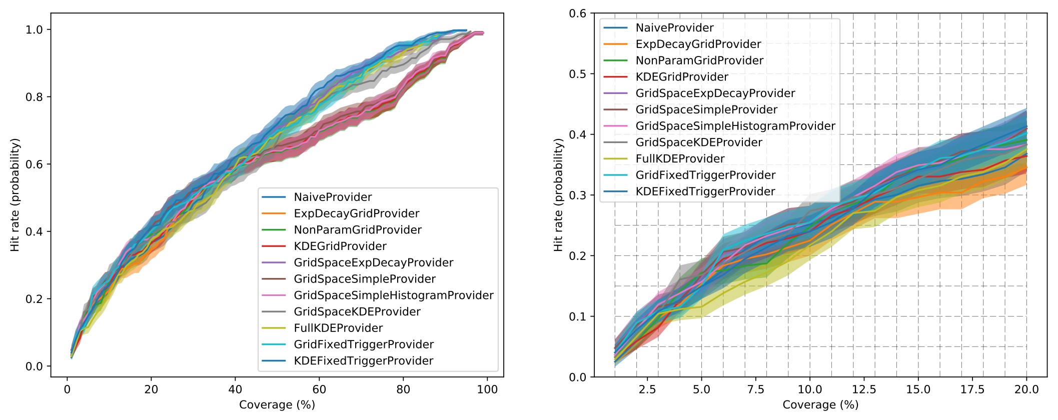

Figure 31: For a variety of models giving rise to predictions as described in Section 9, we select the best parameters, and here show the hit rate.

[8], we use a simple Bayesian approach, fitting a Binomial to the hit rates, with parameter p, the probability of capturing the event, with a Beta distribution prior on p. We then plot the median value of p from the posterior, together with a Bayesian confidence interval of $\pm 34\%$.

By way of comparison, we also use the "Naive Provider" prediction method from `open_cp`. This can be thought of as a grid-based SEPP model with *zero* trigger intensity. That is, for each grid cell independently, we estimate the risk as simply being the proportion events before the current day which have occurred in that grid cell. We would certainly hope that a "good" prediction method, perhaps taking account of spatial interactions, and/or repeat behaviour, would out-perform this method.

For those models which have fixed parameters, we fit the models with a variety of parameters. We then pick the "best" parameter(s) to use by looking at the hit-rate over coverage levels up to 20% (we use smaller coverage levels, as these seem more realistic, from the perspective of supposing that selected grid cells will somehow receive extra police attention, for example).

Figure 31 shows the result. We notice that all the methods perform similarly, with the expected variation between different days for any given model certainly being comparable to the difference between different models. To compare better the different models, we compare the hit-rate against that given by the "Naive Provider", as shown in Figure 32. For the North side of Chicago, the model with the best average increase over the Naive Provider is the `GridSpaceExpDecayProvider` which is the first model considered in Section 6. The other models in Section 6 also perform well.

We also ran the same analysis on the South side of Chicago, see the right-hand side plot in Figure 32. The overall results are similar, with again the models from Section 6 performing well. Here, we actually found that a grid-based fixed trigger kernel method (see Section 7) performed overall the strongest. However, the "best" parameters turned out to have $\sigma = 20m$ and $\omega^{-1} = 20$ days. While this time decay seems reasonable, given previous research, the space depedence is tiny.

We also ran the same experiment on the second dataset, see Figure 33. The results are somewhat surprising, and differ markedly between the North and South sides of Chicago. For the North side, by far the best performer is the KDE based fixed trigger model from Section 7, followed by the KDE model from Section 6.5. The optimal settings for the fixed trigger model are $\omega^{-1} = 100$ days (which is on the long side) and $\sigma = 100m$ which is fairly reasonable. Comparing betwee the two datasets, the KDE fixed trigger model performs about the same, in terms of absolute hit-rate, while the grid based fixed trigger model, and the "naive" predictor, both perform somewhat worse. Thus the radical change in the plot shown in Figure 33 is mostly caused by this decrease in the quality of the "naive" predictor. An alternative interpretation is

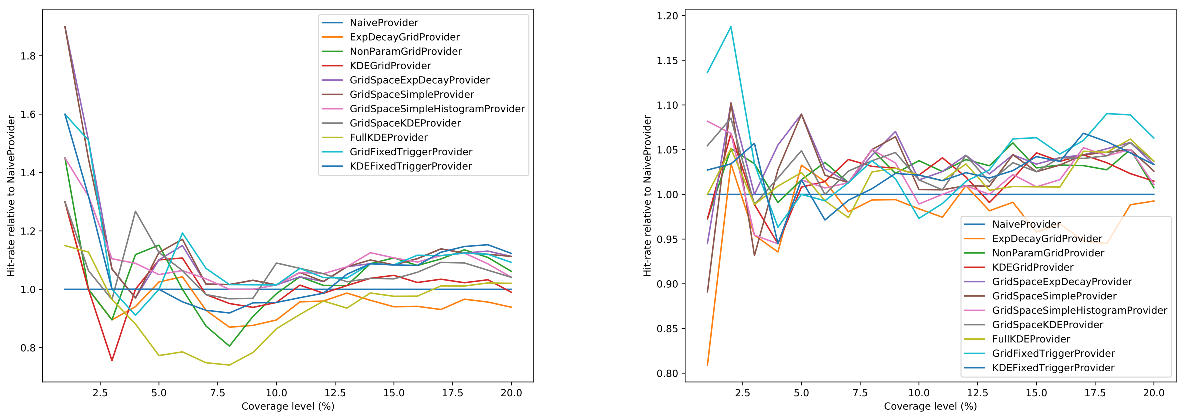

Figure 32: The median hit-rate at varying coverage levels, compared to the hit-rate given by the “Naive Provider”. To the left, for the North side of Chicago, and on the right, the South side of Chicago.

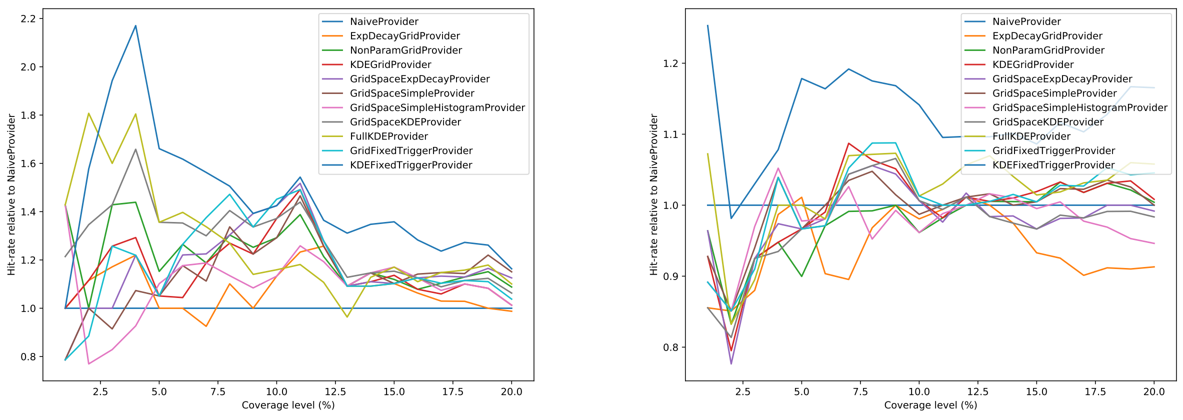

Figure 33: As Figure 32 but for the second dataset.

that for the more "realistic" dataset, the KDE method(s) show their flexibility. For the South side, the story is similar, but the decrease in quality of the "naive" prediction is less marked, which explains why the plot in Figure 33 is less radically different than before. Again, the KDE based methods now perform the best (which is to say, compared with the first dataset, show the least decrease is hit-rate).

These results again stress the importance of thinking about the input data, and its particular characteristics, instead of just focusing on the prediction algorithm. We explore this idea more fully in [7].

Finally, let us compare our slightly disappointing findings with other papers. To our knowledge, there is no published work which uses openly available source code and openly available datasets; this was a significant motivation for the writing of the present paper. [24] does not discuss results except in the most aggregated form, and only compares against "hotspots" produced by a human operator. [23, Figure 6] displays hit-rates and standard errors, and shows a significant improvement from the SEPP model compared to the classical prospective hotspot method. However, once bandwidths for the classical model are adjusted (see the second plot of [23, Figure 6]) this advantage becomes somewhat less, and almost non-existent at some coverage levels. [33, Figure 12] shows the SEPP method out-performing the STKDE method on some areas of Chicago (in particular, for the South Side). The STKDE method is, to quote [33], "a variable bandwidth KDE with bandwidths computed in the same way as they are for the triggering function." That is, presumably a nearest neighbour KDE method. A later paper by some of the same authors, [1], also compares the SEPP method against the classical prospective hotspot method. For the South Side of Chicago, [1, Figure 5] finds results consistent with ours, namely that at smaller coverage levels, a simple (non-time dependent) KDE method (so rather similar to our "naive" method) out-performs both the SEPP and prospective hotspot methods.

A Appendix

In the appendix, we give a brief, but relatively self-contained, guide to the probability theory behind the point process models we consider, and outline the Expectation Maximisation (EM) algorithm we apply to fit models.

A.1 Point processes

We shall treat point processes via their *conditional intensity functions*. Let us first consider a point process evolving only in time, where events occur at times $0 < t_1 < t_2 < \cdots$ and carry no further information. In probability theory, a point process is a "random" collection of times, satisfying certain mild conditions. Let $N(t)$ be the *counting process* of the point process, that is, $N(t)$ is the random variable giving the number of events which have occurred before time t. The conditional intensity function (or sometimes, *hazard function*) is, intuitively speaking,

$$\lambda^*(t) = \lim_{h\to 0} \frac{\mathbb{E}[N(t+h) - N(t) | \mathcal{H}_t]}{h}.$$

Here $(\mathcal{H}_t)_{t\geq 0}$ is the filtration forming the history of the process; we write λ^* and not λ to stress this dependence on the past. That is, conditional on what has occurred before time t, we think of $\lambda^*(t)\ \mathrm{d}t$ as being the expected number of events which will occur near time t times an infinitesimal length of time. Under reasonable conditions, the conditional intensity function exists, and when it exists, it uniquely characterises the point process.

For example, if $\lambda^*(t) = \mu$ independent of time, we have a homogeneous Poisson process with intensity μ (the expected number of events in any unit interval of time is μ). If $\lambda^*(t) = \mu(t)$ is a function of time, but independent of the past, then we have an inhomogeneous Poisson process. A more interesting example is the *Hawkes process* where

$$\lambda^*(t) = \mu + \alpha \sum_{t_i<t} e^{-\beta(t-t_i)}.$$

We think of this as a combination of a fixed "background" rate μ and then a "self-exciting" component, one for each past event, of the form $\alpha e^{-\beta(t-t_i)}$. Thus, immediately after an event occurs, the intensity increases (which makes it more likely that further events will occur) but this decays exponentially in time (meaning that, almost surely, the self-exciting "cascade" will end and we will not have an infinite number of events occurring in finite time, so long as $\alpha < \beta$).

For treating point processes in many dimensions, we find it useful to work with "marked processes" which we think of as a point process evolving in time, where each event carries extra information: the "mark" (here 2D coordinates (x, y)). We shall only consider models with conditional intensity functions of the form

$$\lambda^*(t,x,y) = \mu(t,x,y) + \sum_{t_j<t} \mu_1(t_j,x_j,y_j;t-t_j,x-x_j,y-y_j). \tag{2}$$

Here we have an inhomogeneous background process, and each event in the past adds, independently and linearly, to the total with a function μ_1 which depends on the location of the historical event, and the vector from the event to the location of interest. As we discussed in the introduction, in this paper, we shall actually only consider the case when μ_1 only depends on $(t - t_i, x - x_i, y - y_i)$ and not the absolute location.

Suppose $(t_i, x_i, y_i)_{i=1}^n$ is a realisation of a point process in the time interval $[0, T]$. The likelihood L is

$$L = \Big(\prod_{i=1}^n \lambda^*(t_i,x_i,y_i)\Big) \exp\Big(-\int_0^T \int_{\mathbb{R}^2} \lambda^*(t,x,y)\ \mathrm{d}x\mathrm{d}y\ \mathrm{d}t\Big).$$

With λ^* of the above form, each term in the product is the sum of a μ term and $i-1$ terms involving μ_1. Let Z be the collection of all choices of integers $(z_i)_{i=1}^n$ with $1 \leq z_i \leq i$ for each i. If we fully multiply out the product, then we end with summands indexed by Z, where the choice $z_i = i$ corresponds to choosing the μ term for index i, and the choice $j = z_i < i$ corresponds to choosing the term $\mu_1(t_j, x_j, y_j, t_i - t_j, x_i - x_j, y_i - y_j)$. For example, with $n = 2$, and dropping the x and y parts for clarity, we find

$$\lambda^*(t_1)\lambda^*(t_2) = \mu(t_1)\mu(t_2) + \mu(t_1)\mu_1(t_1, t_2 - t_1)$$

where the summands correspond to the choices $z_1 = 1, z_2 = 1$ and $z_1 = 1, z_2 = 2$.

The integral in the exponential term may be computed in an iterative way. Again for clarify we shall ignore the x and y parts for the moment, to find that

$$\int_0^T \lambda^*(t)\ \mathrm{d}t = \int_0^{t_1} \mu(t)\ \mathrm{d}t + \int_{t_1}^{t_2} \mu(t) + \mu_1(t_1, t-t_1)\ \mathrm{d}t + \cdots + \int_{t_n}^T \mu(t) + \sum_{j=1}^n \mu_1(t_j, t-t_j)\ \mathrm{d}t.$$

This may be re-written as

$$\int_0^T \mu(t)\ \mathrm{d}t + \sum_{j=1}^n \int_{t_j}^T \mu_1(t_j, t-t_j)\ \mathrm{d}t = \int_0^T \mu(t)\ \mathrm{d}t + \sum_{j=1}^n \int_0^{T-t_j} \mu_1(t_j, t)\ \mathrm{d}t.$$

Thus the likelihood is equal to

$$\sum_{z\in Z} \prod_{j=1}^n \big(\mu(t_j)[z_j = j] + \mu_1(t_{z_j}, t_j - t_{z_j})[z_j < j]\big)$$

$$\exp\Big(-\int_0^T \mu(t)\ \mathrm{d}t\Big) \prod_{j=1}^n \exp\Big(-\int_0^{T-t_j} \mu_1(t_j, t)\ \mathrm{d}t\Big), \tag{3}$$

where $[z_j = j]$ is the random variable (the Iverson bracket) taking the value 1 when $z_j = j$ and 0 otherwise, and similarly for $[z_j < j]$.

This form of the likelihood has an interesting interpretation, which has obvious links with how to *simulate* such a point process, see the discussion in [25, 26]. Given z let $Z_0 = \{j : z_j = j\}$ so we have the term

$$\prod_{j\in Z_0} \mu(t_j) \exp\Big(- \int_0^T \mu(t)\ \mathrm{d}t\Big).$$

This is nothing but the likelihood of an inhomogeneous Poisson process– these are the "background" events. For i set $Z_i = \{j > i : z_j = i\}$ so we have the term

$$\prod_{j\in Z_i} \mu_1(t_i, t_j - t_i) \exp\Big(- \int_0^{T-t_i} \mu_1(t_i, t)\ \mathrm{d}t\Big).$$

This is the likelihood of an inhomogeneous Poisson process with intensity $\mu_1(t_i, \cdot)$ starting at time t_i. These are the events "triggered" by event t_i.

We have hence simplified the form of the likelihood, given the "branching structure" of the process, encoded in the random variable z. We cannot observe z directly, but this form allows us to apply the *Expectation Maximisation* algorithm, [21].

A.2 The Expectation Maximisation algorithm

The Expectation Maximisation algorithm is a general technique giving iterative algorithms for parameter estimation for problems which have "unobserved data". Each iterative step has two stages:

- The *expectation* (or "E") step. Given the observed data, we compute the conditional expectation of the log likelihood over the hidden data z. When computing the conditional expectation, we use the *current* estimate of the parameters.
- The *maximisation* (or "M") step. We now maximise the resulting function in the parameters, giving new estimates for the parameters.

Under mild conditions, each step is guaranteed to increase the likelihood, but convergence can be slow, and it is possible to converge only to a local maximum.

In our case, given the above remarks, we find that

$$\mathbb{P}((z_i)|(t_i), (x_i), (y_i)) = \prod_{i=1}^{n} p_{j,i}$$

where

$$p_{j,i} = \mathbb{P}(z_i = j \leq i) = \begin{cases} \mu(t_i, x_i, y_i)/\lambda^*(t_i, x_i, y_i) & : j = i, \\ \mu_1(t_j, x_j, y_j; t_i - t_j, x_i - x_j, y_i - y_j)/\lambda^*(t_i, x_i, y_i) & : j < i \end{cases}$$

Here $\lambda^*(t_i, x_i, y_i)$ is the normalisation factor ensuring that $\sum_{i=1}^n p_{i,j} = 1$ for each j. For a careful justification of this, see Appendix F below.

The (general) E step is hence to compute

$$\Psi = \mathbb{E}_z(\log L) = \sum_z \log L((t_i), (x_i), (y_i), (z_i)) \prod_i p_{z_i,i}, \tag{4}$$

and then the M step is to optimise Ψ. Here μ and μ_1 will depend on parameters which will be fixed for the purposes of computing the terms $p_{j,i}$, but will be variable when maximising Ψ.

It will be convenient to factor μ and μ_1 as

$$\mu(t, x, y) = \mu(t) f(x, y|t), \quad \mu_1(t, x, y; t', x', y') = \mu_1(t, x, y; t') g(x', y'; t, x, y, t')$$

where $\mu(t) = \int_{\mathbb{R}^2} \mu(t,x,y)\ \mathrm{d}x\mathrm{d}y$ and $f(x,y|t) = \mu(t,x,y)/\mu(t)$ for $\mu(t) > 0$; and similarly for μ_1 and g. Notice that then f and g are *probability* densities.

From (3) above (with dependence on x, y now included) we see that

$$\begin{aligned}\Psi = \mathbb{E}_z\Big(\sum_{j=1}^{n} \log \big(&\mu(t_j) f(x_j, y_j|t_j)[z_j = j] \\ &+ \mu_1(t_{z_j}, x_{z_j}, y_{z_j}; t_j - t_{z_j}) g(x_j - x_{z_j}, y_j - y_{z_j}; t_{z_j}, x_{z_j}, y_{z_j}, t_j - t_{z_j})[z_j < j]\big) \\ &\qquad - \int_0^T \mu(t)\ \mathrm{d}t - \sum_{j=1}^{n} \int_0^{T-t_j} \mu_1(t_j, t)\ \mathrm{d}t\Big).\end{aligned}$$

Using linearity of the expectation, this is equal to

$$\begin{aligned}\Psi =& \sum_{j=1}^{n} p_{j,j} \log \big(\mu(t_j) f(x_j, y_j|t_j)\big) - \int_0^T \mu(t)\ \mathrm{d}t \\ &+ \sum_{j=1}^{n} \sum_{i<j} p_{i,j} \log \big(\mu_1(t_i, x_i, y_i; t_j - t_i) g(x_j - x_i, y_j - y_i; t_i, x_i, y_i, t_j - t_i)\big) \\ &\qquad - \sum_{j=1}^{n} \int_0^{T-t_j} \mu_1(t_j, t)\ \mathrm{d}t. \qquad (5)\end{aligned}$$

As we shall see shortly in examples, in practice, this form allows us to optimise the parameters for, say, μ, without needing to consider f, μ_1 or g.

To our knowledge, [36] was the first article to explore the EM algorithm in the setting of self-excited point processes. Unfortunately, [36] uses different notation to later works, and treats only the "ETAS" model. We prefer to present a rather general construction which can be specialsed to many models.

A.3 Allowing exact repeats, in time

Looking again at equation (2) defining our model, we sum over only those events $t_j < t$. We can consequently make sense, at least in a formal way, of realisations (t_i) where merely $t_1 \leq t_2 \leq \cdots \leq t_n$. In the EM algorithm, this corresponds to summing over not just $i < j$ but also $t_i < t_j$. Equivalently, we may set $p_{i,j} = 0$ for $i < j$ when $t_i = t_j$.

A.4 Expected number of triggered events

Let $\theta = \int \mu_1$ be the total intensity of the trigger. The expected number of events that the trigger kernel will give rise to is thus θ. Let $Z_0 = 1$ be the initial event, which triggers $Z_1 \geq 0$ further events. Each of these events triggers further events, independently, say giving a total of $Z_2 \geq 0$ further events, and so forth. We thus have a *branching process*, [11, Section 5.4], and so by using generating functions, we can prove that $\mathbb{E}(Z_n) = \mu^n$, and hence that the expected total number of triggered events is $\mu + \mu^2 + \mu^3 + \cdots = \mu(1-\mu)^{-1}$ as claimed in the main text.

A.5 Fully parametric example

This example is the model used in [18, Section 2]. We shall again ignore spatial components. Here we set μ to be a constant, and let the triggering kernel have the form

$$\mu_1(t_i; t - t_i) = \theta\omega e^{-\omega(t-t_i)},$$

where θ, ω are parameters (along with μ) to be estimated. We find that

$$\begin{aligned}\Psi &= \sum_{j=1}^{n} p_{j,j} \log\mu - \int_0^T \mu\ dt + \sum_{i<j} p_{i,j}\big(\log\theta + \log\omega - \omega(t_j - t_i)\big) - \sum_{j=1}^{n}\int_0^{T-t_j} \theta\omega e^{-\omega t}\ dt \\ &= \sum_{j=1}^{n} p_{j,j}\log\mu - \mu T + \sum_{i<j} p_{i,j}\big(\log\theta + \log\omega - \omega(t_j - t_i)\big) - \theta\sum_{j=1}^{n}\big(1 - e^{-\omega(T-t_j)}\big).\end{aligned}$$

Maximising in μ gives $\mu = \frac{1}{T}\sum_j p_{j,j}$, and similarly we find

$$\theta = \frac{\sum_{i<j} p_{i,j}}{\sum_j 1 - e^{-w(T-t_j)}}, \qquad \omega = \frac{\sum_{i<j} p_{i,j}}{\sum_{i<j} p_{i,j}(t_j - t_i) + \theta\sum_j (T - t_j)e^{-\omega(T-t_j)}}.$$

In the expression for θ we use the *previous* estimate for ω, and similarly for ω.

It is often assumed that $e^{-w(T-t_j)}$ is small for all j, giving the simplifications

$$\theta = \frac{\sum_{i<j} p_{i,j}}{n}, \qquad \omega = \frac{\sum_{i<j} p_{i,j}}{\sum_{i<j} p_{i,j}(t_j - t_i)}.$$

A.6 With a spatial component

Following [24, Section 2.2] we now introduce a spatial component. We assume that events occur in regions $k = 1, \cdots, K$, where region k has background rate μ_k, but that all regions share the same "triggering parameters" θ and ω. We assume that events in region k only trigger future events in the same region. Let the events (t_i, x_i, y_i) be such that (x_i, y_i) occurs in region $k(i)$. Thus

$$\lambda^*(t_i, x_i, y_i) = \mu_{k(i)} + \sum_{j<i, k(j)=k(i)} \theta\omega e^{-\omega(t_i - t_j)}.$$

We can similarly compute the integral

$$\begin{aligned}\int_0^T\int_{\mathbb{R}^2} \lambda^*(t,x,y)\ \mathrm{d}x\mathrm{d}y\ \mathrm{d}t &= \int_0^T \sum_{k=1}^{K}\int_{\text{region } k}\Big(\mu_k + \sum_{t_i<t, k(i)=k}\theta\omega e^{-\omega(t-t_i)}\ \mathrm{d}t\Big) \\ &= \sum_{k=1}^{K} A_k\Big(T\mu_k + \int_0^T \sum_{t_i<t,k(i)=k}\theta\omega e^{-\omega(t-t_i)}\ \mathrm{d}t\Big)\end{aligned}$$

where region k has area A_k. Let us group the data by region, say region k has events which occur at times $(t_i^{(k)})_{i=1}^{n(k)}$. Then the above integral becomes

$$\begin{aligned}&\sum_{k=1}^{K} TA_k\mu_k + \sum_{k=1}^{K} A_k \sum_{i=1}^{n(k)}\int_0^{T-t_i^{(k)}} \theta\omega e^{-\omega t}\ dt \\ &\qquad = \sum_{k=1}^{K} TA_k\mu_k + \sum_{k=1}^{K}\theta A_k\sum_{i=1}^{n(k)}\big(1 - e^{-\omega(T-t_i^{(k)})}\big).\end{aligned}$$

From (5), we find that

$$\begin{aligned}\Psi = &\sum_{k=1}^{K}\log(\mu_k)\sum_{j=1}^{n(k)} p_{j,j}^{(k)} - \sum_{k=1}^{K} TA_k\mu_k \\ &+ \sum_{k=1}^{K}\sum_{1\leq i<j\leq n(k)} p_{i,j}^{(k)}\log\big(\theta\omega e^{-\omega(t_j^{(k)} - t_i^{(k)})}\big) - \sum_{k=1}^{K}\theta A_k\sum_{i=1}^{n(k)}\big(1 - e^{-\omega(T-t_i^{(k)})}\big)\end{aligned}$$

Here

$$p_{j,j}^{(k)} = \mu_k / \lambda_k^*(t_j^{(k)}), \quad p_{i,j}^{(k)} = \theta\omega e^{-\omega(t_j^{(k)} - t_i^{(k)})} / \lambda_k^*(t_j^{(k)})$$

where $\lambda_k^*(t) = \mu_k + \sum_{t_i^{(k)} < t} \theta\omega e^{-\omega(t - t_i^{(k)})}$.

Maximising in μ_k and θ gives

$$\mu_k = \frac{1}{TA_k} \sum_{j=1}^{n(k)} p_{j,j}^{(k)}, \qquad \theta = \frac{\sum_k \sum_{i<j} p_{i,j}^{(k)}}{\sum_k A_k \sum_i (1 - e^{-\omega(T - t_i^{(k)})})},$$

and maximising in ω gives

$$\omega = \frac{\sum_k \sum_{i<j} p_{i,j}^{(k)}}{\sum_k \sum_{i<j} p_{i,j}^{(k)} (t_j^{(k)} - t_i^{(k)}) + \sum_k \theta A_k \sum_i (T - t_i^{(k)}) e^{-\omega(T - t_i^{(k)})}}.$$

If we assume $A_k = A$ for all k, then we can absorb the A constant into each μ_k and θ, and assume that $A = 1$. If we further assume that $e^{-\omega(T - t_i^{(k)})} = 0$ for each k, i then we obtain

$$\mu_k = \frac{1}{T} \sum_{j=1}^{n(k)} p_{j,j}^{(k)}, \quad \theta = \frac{\sum_k \sum_{i<j} p_{i,j}^{(k)}}{n}, \quad \omega = \frac{\sum_k \sum_{i<j} p_{i,j}^{(k)}}{\sum_k \sum_{i<j} p_{i,j}^{(k)} (t_j^{(k)} - t_i^{(k)})}.$$

Assuming there is an obvious typo in (6) of [24], this agrees exactly with the algorithm given in [24, Section 2.2].

A.6.1 With an inhibited trigger

We explained in Section 4 that exact repeats, cases when $t_i^{(k)} = t_j^{(k)}$ for some k and some $i < j$, are not only against the underlying probability model, but also cause problems for the EM algorithm, as maximum likelihood is found at $\omega = \infty$. One possible way around this problem is to modify the triggering function from $g(t) = \omega e^{-\omega t}$ to

$$g(t) = \begin{cases} 0 & : t < t_0, \\ \omega e^{-\omega(t - t_0)} & : t \geq t_0. \end{cases}$$

This corresponds to inhibiting triggering before a time of $t_0 \geq 0$ has elapsed.

We modify $p_{i,j}^{(k)}$ to use g in the obvious way; then $p_{i,j}^{(k)} = 0$ for $i < j$ if $t_j^{(k)} - t_i^{(k)} < t_0$. Only the parts of Ψ not involving (μ_k) change, and become

$$\sum_{k=1}^{K} \sum_{1 \leq i < j \leq n(k)} p_{i,j}^{(k)} \log\left(\theta\omega e^{-\omega(t_j^{(k)} - t_i^{(k)} - t_0)}\right) - \sum_{k=1}^{k} \theta A_k \sum_{i=1}^{n(k)} \begin{cases} 0 & : T - t_i^{(k)} < t_0, \\ 1 - e^{-\omega(T - t_i^{(k)} - t_0)} & : T - t_i^{(k)} \geq t_0. \end{cases}$$

where we use the convention that $0 \cdot \log(x) = 0$ for any $x \in \mathbb{R}$. Define $\alpha_i^{(k)} = 0$ if $T - t_i^{(k)} < t_0$ and $\alpha_i^{(k)} = 1 - e^{-\omega(T - t_i^{(k)} - t_0)}$ otherwise; similarly define $\beta_i^{(k)} = T - t_i^{(k)} - t_0$ if $T - t_i^{(k)} \geq t_0$ or 0 otherwise. Notice that $1 - e^{-\omega\beta_i^{(k)}} = \alpha_i^{(k)}$. Optimising in θ gives

$$\theta = \frac{\sum_k \sum_{i<j} p_{i,j}^{(k)}}{\sum_k A_k \sum_i \alpha_i^{(k)}}, \quad \omega = \frac{\sum_k \sum_{i<j} p_{i,j}^{(k)}}{\sum_k \sum_{i<j} p_{i,j}^{(k)} (t_j^{(k)} - t_i^{(k)} - t_0) + \sum_k \theta A_k \sum_i \beta_i^{(k)} e^{-\omega\beta_i^{(k)}}}.$$

If we assume that $e^{-\omega(T - t_i^{(k)} - t_0)} \approx 0$ for all $t_i^{(k)}$ with $T - t_i^{(k)} \geq t_0$, then we obtain

$$\theta = \frac{\sum_k \sum_{i<j} p_{i,j}^{(k)}}{\sum_k A_k m(k)}, \quad \omega = \frac{\sum_k \sum_{i<j} p_{i,j}^{(k)}}{\sum_k \sum_{i<j} p_{i,j}^{(k)} (t_j^{(k)} - t_i^{(k)} - t_0)},$$

where $m(k)$ is the number of i with $T - t_i^{(k)} \geq t_0$.

B Histogram estimators

If we consider equation (5) above, then the abstract problem we typically need to solve in the M step is to maximise a sum like

$$\sum_{i=1}^{n} p_i \log f(x_i)$$

where given are $(p_i) \subseteq [0,\infty)$ (usually with $\sum_i p_i = 1$), and (x_i) are, say, positive numbers, but f is unknown. If f belongs to a parametric family, then we can find the values of the parameters which maximise this sum, through partial differentiation, for example.

Rather than work with parametric models, we will often wish to work with non-parametric models. A step in this direction is to consider kernel density estimators which are *histograms*, that is, f of the form

$$f(x) = h^{-1}\alpha_r \quad \text{where } r \in \mathbb{Z} \text{ satisfies } hr \leq x < h(r+1).$$

Here $h > 0$ is the *bandwidth* and $(\alpha_r) \subseteq [0,\infty)$ with $\int_{\mathbb{R}} f(x)\ dx = 1$, equivalently, $\sum_r \alpha_r = 1$. As we assume $x_i \geq 0$, we may suppose that $\alpha_r = 0$ for $r < 0$.

For $r \geq 0$ set

$$\beta_r = \sum \big\{ p_i : hr \leq x_i < h(r+1) \big\}.$$

Thus our task is to

$$\text{maximise} \sum_{r\geq 0} \beta_r \log(h^{-1}\alpha_r) \quad \text{subject to} \quad \alpha_r \geq 0, \sum_r \alpha_r = 1.$$

For this to make sense, we shall insist that $\alpha_r > 0$ if $\beta_r > 0$; if $\beta_r = 0$ then we shall ignore the summand $\beta_r \log(h^{-1}\alpha_r)$ (thus allowing $\alpha_r = 0$).

This can be solved by standard Lagrangian techniques; the maximum occurs at

$$\alpha_r = \frac{\beta_r}{\sum_i \beta_i}. \tag{6}$$

We then find that actually

$$f(x) = \Big(\sum_{i=1}^{n} p_i \Big)^{-1} \sum_{i=1}^{n} h^{-1} p_i \chi_{[hk_i, h(k_i+1))}(x),$$

where $\chi_{[hk_i,h(k_i+1))}$ is the indicator function of the interval $[hk_i, h(k_i+1))$, and k_i is the unique integer with $hk_i \leq x_i < h(k_i+1)$.

The histogram estimator is often used (compare [35, Section 2.4]) to motivate more general *kernel density estimation* techniques. If we define $w(x) = 1$ for $-1/2 \leq x < 1/2$, and 0 otherwise, then f is

$$f(x) = \Big(\sum_{i=1}^{n} p_i \Big)^{-1} \sum_{i=1}^{n} \frac{p_i}{h} w\Big(\frac{x - hk_i - \frac{1}{2}h}{h} \Big).$$

Now, $h(k_i - 1/2)$ is approximately x_i, and so we can define the "naive estimator" as

$$f(x) = \Big(\sum_{i=1}^{n} p_i \Big)^{-1} \sum_{i=1}^{n} \frac{p_i}{h} w\Big(\frac{x - x_i}{h} \Big). \tag{7}$$

This is a very similar form to the usual kernel density estimator, where w is the "kernel", except that we now have the factors p_i instead of the usual $1/n$. That is, we use the "probabilities" (p_i) to "weight" the kernel estimator.

B.1 With "edge correction"

Again with reference to equation (5), the complete M step is actually to maximise something of the form

$$\sum_{i=1}^{n} p_i \log f(x_i) - \theta \sum_{i=1}^{m} \int_0^{T_i} f(x)\ dx,$$

where again f is given by h and (α_r) as before, and θ and (T_i) are known. Again form (β_r), and now let γ_r be

$$\gamma_r = \sum_{i=1}^{m} \int_0^{T_i} \chi_{[rh,(r+1)h)}(x)\ dx \geq 0.$$

Thus we wish to

$$\text{maximise} \sum_{r\geq 0} \beta_r \log(h^{-1}\alpha_r) - \theta h^{-1} \sum_{r\geq 0} \alpha_r \gamma_r \quad \text{subject to} \quad \alpha_r \geq 0, \sum_r \alpha_r = 1.$$

Lagrangian techniques can again be applied to find that the maximum occurs at

$$\alpha_r = \frac{\beta_r}{\lambda + \theta h^{-1}\gamma_r} \quad \text{where } \lambda \in \mathbb{R} \text{ satisfies} \quad \sum_r \frac{\beta_r}{\lambda + \theta h^{-1}\gamma_r} = 1.$$

More correctly, we need only consider $\lambda > \lambda_0$ where $\lambda_0 = \max\{-\theta h^{-1}\gamma_r : \beta_r > 0\}$.

Set $h(\lambda) = \sum_r \beta_r/(\lambda+\theta h^{-1}\gamma_r) - 1$ so that $h'(\lambda) = -\sum_r \beta_r/(\lambda+\theta h^{-1}\gamma_r)^2 < 0$, and $h(\lambda) \to \infty$ as λ decreases to λ_0. Hence h has a unique root. Numerically, this may be found easily using Newton-Raphson, for example.

As $\gamma_r \geq 0$ we see that $\sum_r \beta_r/(\lambda + \theta h^{-1}\gamma_r) \leq \lambda^{-1}\sum_r \beta_r$ and so $\lambda \leq \sum_r \beta_r$. Furthermore, (γ_r) is a decreasing sequence. Thus, compared with the non-edge-corrected case (equation (6)), we see that α_r will be smaller for small r, and larger for large r.

This gives a rough guide as to how we might "bias" a general KDE method to take account of edge-effects; but it is unclear how to do this is practise.

C Grid based model with histogram trigger

Continuing as in Section A.6 we replace the trigger function $\omega e^{-\omega s}$ with a histogram estimator with bandwidth $h > 0$ and parameters (α_r). A somewhat similar idea was explored in [20], though we are rather more explicit in our use of the EM algorithm; see also [19] which gives more details. We believe our use of the argument in Appendix B.1 above is new.

We have

$$\Psi = \sum_{k=1}^{K} \log(\mu_k) \sum_{j=1}^{n(k)} p_{j,j}^{(k)} - \sum_{k=1}^{K} T A_K \mu_k$$
$$+ \sum_{k=1}^{K} \sum_{1\leq i<j\leq n(k)} p_{i,j}^{(k)} \log(\theta f(t_j^{(k)} - t_i^{(k)})) - \sum_{k=1}^{K} \theta A_k \sum_{i=1}^{n(k)} \int_0^{T-t_i^{(k)}} f(t)\ dt.$$

Here f is the histogram, $f(x) = h^{-1}\alpha_r$ for $hr \leq x < h(r+1)$. The p matrix is

$$p_{j,j}^{(k)} = \mu_k/\lambda_k^*(t_j^{(k)}), \quad p_{i,j}^{(k)} = \theta f(t_j^{(k)} - t_i^{(k)})/\lambda_k^*(t_j^{(k)}),$$

with normalisation $\sum_i p_{i,j}^{(k)} = 1$ for each j, k.

We maximise μ_k as before. Following Appendix B.1, we define

$$\beta_{k,r} = \sum \left\{ p_{i,j}^{(k)} : hr \leq t_j^{(k)} - t_i^{(k)} < h(r+1) \right\}, \quad \gamma_{k,r} = \sum_{i=1}^{n(k)} \int_0^{T-t_i^{(k)}} \chi_{[hr,h(r+1))}(t)\ dt.$$

Then the part of Ψ involving f becomes

$$\sum_k \sum_{i<j} p_{i,j}^{(k)} \log\theta + \sum_{k,r} \beta_{k,r} \log(h^{-1}\alpha_r) - \sum_{k,r} \theta A_k h^{-1} \alpha_r \gamma_{k,r}.$$

Maximising in θ gives

$$\theta = \frac{\sum_k \sum_{i<j} p_{i,j}^{(k)}}{\sum_{k,r} A_k h^{-1} \alpha_r \gamma_{k,r}}.$$

Finally, maximising in (α_r) can be done as in Appendix B.1 with

$$\beta_r = \sum_{k=1}^K \beta_{k,r}, \qquad \gamma_r = \sum_{k=1}^K A_k \gamma_{k,r}.$$

As before, we typically assume A_k does not vary, and so set $A_k = 1$ for each k.

C.1 With a KDE trigger

We now follow through on the thoughts at the end of Appendix B and replace the histogram with a general, weighted, kernel density estimator. For simplicity (compare with the discussion in Section 5.2) we will ignore edge effects, and so

$$\begin{aligned}\Psi = &\sum_{k=1}^K \log(\mu_k) \sum_{j=1}^{n(k)} p_{j,j}^{(k)} - \sum_{k=1}^K T A_K \mu_k \\ &+ \sum_{k=1}^K \sum_{1\leq i<j\leq n(k)} p_{i,j}^{(k)} \log(\theta f(t_j^{(k)} - t_i^{(k)})) - \sum_{k=1}^K \theta A_k n(k)\ dt,\end{aligned}$$

where now f is as in equation (7). To be clear, the assumption of no edge effects is that $\int_0^{T-t^{(k)}i} f(t)\ dt \approx 1$.

We maximise μ_k in the by now usual way. Similarly,

$$\theta = \frac{\sum_k \sum_{i<j} p_{i,j}^{(k)}}{\sum_k A_k n(k)}.$$

Finally, to find f, we follow Appendix B and find

$$f(t) = \Big(\sum_k \sum_{i<j} p_{i,j}^{(k)}\Big)^{-1} h^{-1} \sum_k \sum_{i<j} p_{i,j}^{(k)} w\Big(\frac{t-(t_j^{(k)} - t_i^{(k)})}{h}\Big).$$

Here $h > 0$ is a bandwidth, and w is a kernel. We shall take w to be a Gaussian, and we shall either set h manually, or use a variable bandwidth estimator, and so allow h to vary, typically via a near-neighbour estimator (compare [23, Appendix], and see the next section). Finally, as f is only supported on the positive real line, we will reflect f about 0 (see [35, Section 2.10] for other possible options).

C.2 Nearest neighbour KDE method

Following [23] we use a k-th nearest neighbour KDE method. That is, and compare to the above, we now use

$$f(x) = \Big(\sum_i p_i\Big)^{-1} \sum_i h_i^{-1} p_i w\Big(\frac{x-x_i}{h_i}\Big),$$

where (p_i) are the *weights*, w is a kernel, for us always Gaussian, x is a vector, and (x_i) are vectors. Notice here we allow the bandwidths (h_i) to vary.

Following [23] we first scale each coordinate of the input vectors (x_i) so that each coordinate has unit sample variance. Note that we do this for each coordinate individually, which differs from most "plug-in" KDE methods which use the full covariance matrix. We then compute the distance from x_i to its nearest kth neighbour to give us h_i. If this happens to be 0, then we set $h_i = 1$. We then scale back to the original sizes.

D Grid based model with interaction

We keep the grid based prediction of the background rate, but now allow all events to trigger each other. We now have

$$\lambda^*(t,x,y) = \mu_k + \sum_{t_i<t} \theta f(t-t_i) g(x-x_i, y-y_i),$$

where k is the cell which (x,y) falls in, but we now consider *all* events before time t, not just those in cell k.

In the EM algorithm, once again from equation (5), we now have

$$\begin{aligned}\Psi &= \sum_j p_{j,j} \log \mu_{k(j)} - \sum_k T A_k \mu_k \\ &\quad + \sum_{i<j} p_{i,j} \log \big(\theta f(t_j - t_i) g(x_j - x_i, y_j - y_i)\big) - \sum_j \int_0^{T-t_j} \theta f(t-t_j)\, dt,\end{aligned} \tag{8}$$

where $k(j)$ is the grid cell which contains event j. Thus the M stage estimates

$$\mu_k = \frac{1}{TA_k} \sum_{j \text{ in cell } k} p_{j,j},$$

for each k, and also

$$\theta = \frac{\sum_{i<j} p_{i,j}}{\sum_j \int_0^{T-t_j} f(t)\, dt}.$$

If we ignore edge effects, this becomes $\theta = \frac{1}{n}\sum_{i<j} p_{i,j}$.

As the trigger kernel now takes account of space, we need to work in real units. That is, while it is find to assume $A_k = A$ for all A, we need to choose A realistically, and not just set it equal to 1.

D.1 Exponential in time, Gaussian in space

As in Section 6, we set

$$f(t) = \omega e^{-\omega t}, \qquad g(x,y) = \begin{cases} \alpha & : x^2+y^2 \leq r_0^2, \\ \beta e^{-(x^2+y^2)/2\sigma^2} & : x^2+y^2 > r_0^2. \end{cases}$$

Noting that g is radially symmetric, if we work in polar coordinates, then

$$\int_{\mathbb{R}^2} g = 2\pi \int_0^\infty r g(r)\, dr = \pi\alpha r_0^2 + 2\pi\beta \int_{r_0}^\infty r e^{-r^2/2\sigma^2} = \pi\alpha r_0^2 + 2\pi\beta\sigma^2 e^{-r_0^2/2\sigma^2}.$$

As g is meant to be a probability density, we need $\alpha \leq 1/\pi r_0^2$ and set

$$\beta = \frac{1-\pi\alpha r_0^2}{2\pi\sigma^2 e^{-r_0^2/2\sigma^2}}.$$

So we regard β as a function of α and σ.

The EM algorithm estimates ω in the by now usual form,

$$\omega = \frac{\sum_{i<j} p_{i,j}}{\sum_{i<j} p_{i,j}(t_j - t_i) + \theta \sum_j (T - t_j) e^{-\omega(T-t_j)}},$$

and if we ignore edge effects, the 2nd term in the denominator disappears.

To estimate α, σ^2 we look at the following part of the expression for Ψ (see equation (8) above),

$$\sum_{i<j} p_{i,j} \log(g(x_j - x_i, y_j - y_i)).$$

Let us write $r_{i,j}$ for the distance between (x_i, y_i) and (x_j, y_j), so the sum is

$$\sum_{i<j} p_{i,j} \begin{cases} \log(\alpha) & : r_{i,j} \leq r_0, \\ \log(\beta) - r_{i,j}^2/2\sigma^2 & : r_{i,j} > r_0, \end{cases} = a\log(\alpha) + b\log(\beta) - \frac{c}{2\sigma^2},$$

if

$$a = \sum \{p_{i,j} : r_{i,j} \leq r_0\}, \quad b = \sum \{p_{i,j} : r_{i,j} > r_0\}, \quad c = \sum \{p_{i,j} r_{i,j}^2 : r_{i,j} > r_0\}.$$

Optimising in α involves solving

$$0 = \frac{a}{\alpha} + \frac{b}{\beta}\frac{\partial\beta}{\partial\alpha} = \frac{a}{\alpha} - \frac{b}{\beta}\frac{\pi r_0^2}{2\pi\sigma^2 e^{-r_0^2/2\sigma^2}} = \frac{a}{\alpha} - \frac{b\pi r_0^2}{1 - \pi\alpha r_0^2},$$

that is,

$$\alpha = \frac{a}{a+b} \times \frac{1}{\pi r_0^2},$$

in particular, always $0 \leq \alpha \leq 1/\pi r_0^2$ which was our original constraint. Having fixed α to this value, we have that $\beta = \frac{b}{a+b}(2\pi\sigma^2 e^{-r_0^2/2\sigma^2})^{-1}$, and so the dependance on σ^2 becomes

$$b \log \frac{e^{r_0^2/2\sigma^2}}{\sigma^2} - \frac{c}{2\sigma^2} = \frac{br_0^2}{2\sigma^2} - 2b\log(\sigma) - \frac{c}{2\sigma^2}.$$

Optimising this yields

$$\sigma^2 = \frac{c - br_0^2}{2b}.$$

Notice that $c > r_0^2 b$ by definition, and so $\sigma^2 > 0$ as we might hope.

D.2 Histogram time trigger

We now replace f by a histogram with bandwidth h and values (α_r). The part of Ψ which depends upon f, see equation (8), is

$$\sum_{i<j} p_{i,j} \log f(t_j - t_i) - \theta \sum_j \int_0^{T-t_j} f(t - t_j)\, dt.$$

This is again exactly the setup of Appendix B.1, with $(p_{i,j})_{i<j}$, $(t_j - t_i)_{i<j}$ and $T_j = T - t_j$. It is worth noting then that the EM algorithm finds

$$\theta = \frac{\sum_{i<j} p_{i,j}}{\sum_j \int_0^{T-t_j} f(t)\, dt} = \frac{\sum_r \beta_r}{h^{-1}\sum_r \alpha_r \gamma_r},$$

using (β_r) and (γ_r) from Appendix B.1

D.3 KDE time and space triggers

We now replace both f and g by kernel density estimators, following exactly Appendix C.1, and Appendix C.1.

We also consider a KDE estimate where the trigger does not factor as f times g. In this case, we reflect the kernel about $t = 0$ in time (as in Appendix C.1).

E Fully parametric

We finally come to, essentially, the model from [24]. As explained in Section 6.5, we set the background time rate to be constant, but otherwise follow [24]. We will estimate both the background and triggered components using KDE methods. The general model is

$$\lambda^*(t,x,y) = \mu f(x,y) + \sum_{t_i<t} \theta g(t-t_i, x-x_i, y-y_i).$$

Here f, g are probability densities, θ is again the overall trigger rate, and μ is now the background rate per unit time.

In the EM algorithm, we have

$$\begin{aligned}\Psi &= \sum_j p_{j,j} \log\big(\mu f(x_j,y_j)\big) - T\mu \\ &\quad + \sum_{i<j} p_{i,j} \log\big(\theta g(t_j-t_i, x_j-x_i, y_j-y_i)\big) - n\theta.\end{aligned}$$

Here we have again ignored edge effects. Maximising in μ and θ gives

$$\mu = \frac{1}{T}\sum_j p_{j,j}, \qquad \theta = \frac{1}{n}\sum_{i<j} p_{i,j}.$$

Again motivated by the arguments in Appendix B, we estimate f as a "weighted KDE" using the weights $(p_{j,j})$, and similarly estimate g using the weights $(p_{i,j})_{i<j}$.

F Derivation of the p-matrix

The abstract setting which the EM algorithm works in is the following. We have a probability space Y ($\subseteq \mathbb{R}^n$ say) and data $y \in Y$. Typically we have a parametric model, so a parameter space Θ and given $\theta \in \Theta$ we have a density $f(y;\theta) : Y \to [0,\infty)$.

We also have hidden data, a probability space Z and $z \in Z$. The full data is the pair $x = (y,z)$. We assume that given θ we know the full probability density $g(y,z;\theta)$. The relation between f and g is that $f(y;\theta) = \int_Z g(y,z;\theta)\ dz$.

We have observed data y and so form the conditional distribution of z, which is

$$h : Z \to [0,\infty); \quad h(z;\theta) = \frac{g(y,z;\theta)}{\int_Z g(y,z';\theta)\ dz'}.$$

The EM algorithm is iterative. Suppose we have a current estimate θ_n for the parameters, and aim to form θ_{n+1}. We consider the log-likelihood $\log g(y,z;\theta)$. As we do not know z we take the expectation with regards to the current estimate θ_n, namely

$$\Psi(\theta;\theta_n) = \int_Z h(z;\theta_n) \log g(y,z;\theta)\ dz.$$

This is the E step. Then the M step is to find the value θ_{n+1} of θ which maximises Ψ, so

$$\Psi(\theta_{n+1};\theta_n) \geq \Psi(\theta;\theta_n) \qquad (\theta \in \Theta).$$

F.1 For the branching structure of a point process

We take $Z = \{(z_j)_{j=1}^n : z_j \in \{1, 2, \cdots, j\}\}$ a finite set, so we may replace integrals by sums. From equation (3) we have an expression of the form

$$f(t = (t_i); \theta) = \sum_{z \in Z} \prod_j a(t, z_j, j; \theta) C(t; \theta)$$

where θ encodes the details of the background μ and trigger μ_1. Here $a(t, z_j, j; \theta) \in [0, \infty)$ is a number, and $C(t; \theta)$ is a normalising constant which is independent of z. Hence we immediately see that

$$g(t, z; \theta) = \prod_j a(t, z_j, j; \theta) C(t; \theta)$$

Notice that (essentially reversing the argument which lead to equation (3)),

$$\sum_{z \in Z} \prod_j a(t, z, j; \theta) = \big(a(t, 1, 1; \theta)\big)\big(a(t, 1, 2; \theta) + a(t, 2, 2; \theta)\big) \cdots \Big(\sum_{z_n=1}^{n} a(t, z_n, n; \theta)\Big).$$

Thus when we compute $h(z; \theta)$ a large amount of cancellation occurs, and

$$h(z; \theta) = \prod_{j=1}^{n} \frac{a(t, z_j, j; \theta)}{\sum_{i=1}^{j} a(t, i, j; \theta)}$$

which gives exactly the "p-matrix" we had before.

Forming Q is now a sum,

$$\Psi(\theta; \theta_n) = \sum_z \Big(\prod_j p_{z_j, j}\Big) \log(g(t, z; \theta)) = \sum_z \Big(\prod_j p_{z_j, j}\Big)\Big(\log C(t; \theta) + \sum_j \log a(t, z_j, j; \theta)\Big)$$

where $p_{i,j}$ is computed using θ_n. As we saw before, it is generally not profitable to expand this directly, but rather exploit linearity of the expectation.

References

[1] M. Adepeju, G. Rosser, T. Cheng, "Novel evaluation metrics for sparse spatio-temporal point process hotspot predictions – a crime case study", International Journal of Geographical Information Science, 30:11, 2133-2154, DOI:10.1080/13658816.2016.1159684

[2] W. Bernasco, P. Nieuwbeerta, "How Do Residential Burglars Select Target Areas?: A New Approach to the Analysis of Criminal Location Choice", The British Journal of Criminology (2005) 45 296–315.

[3] K. J. Bowers, S. D. Johnson, K. Pease, "Prospective Hot-Spotting. The future of crime mapping?", The British Journal of Criminology (2004) 44 641–658.

[4] Chicago Data Portal, "Crimes - 2001 to present", available at `https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2`

[5] N. A. C. Cressie, "Statistics for spatial data", (Wiley, 1993).

[6] D. J. Daley, D. Vere-Jones, "An introduction to the theory of point processes: Volume 1. 2nd edition", (Springer, 2003).

[7] M. Daws, "Open crime data-sets: Pitfalls of geocoding", preprint.

[8] M. Daws, "Comparison of short range spatial predictive policing forecasts", preprint.

[9] M. Daws, “Self-excited point process models of crime”, open source Python package, available at https://github.com/MatthewDaws/SEPP/

[10] A. E. Gelfand et al., eds, “Handbook of Spatial Statistics”, (CRC Press, 2010).

[11] G. Grimmett, D. Stirzaker, “Probability And Random Processes”, (Oxford University Press, 2001).

[12] E. Groff, N. La Vigne, “Forecasting the future of predictive crime mapping”, Crime Prevention Studies (2002) 13 29–57.

[13] S. D. Johnson, “Repeat burglary victimisation: a tale of two theories”, Journal of Experimental Criminology (2008) 4 215–240

[14] S. D. Johnson, K. J. Bowers, “The Burglary as Clue to the Future. The Beginnings of Prospective Hot-Spotting”, European Journal of Criminology 2004 (1) 237–255.

[15] S. D. Johnson et al, “Space–Time Patterns of Risk: A Cross National Assessment of Residential Burglary Victimization”, Journal of Quantitative Criminology (2007) 23 201–219.

[16] S. D. Johnson et al, “Prospective crime mapping in operational context: final report”, Home Office Online Report 19/07. Available at library.college.police.uk/docs/hordsolr/rdsolr1907.pdf

[17] P. J. Laub, T. Taimre, P. K. Pollett, “Hawkes Processes”, arXiv:1507.02822 [math.PR]

[18] E. Lewis, G. Mohler, “A Nonparametric EM algorithm for Multiscale Hawkes Processes”, preprint available at http://math.scu.edu/~gmohler/EM_paper.pdf

[19] D. Marsan, O. Lengliné, “A new estimation of the decay of aftershock density with distance to the mainshock”, Journal of Geophysical Research (2010) 115 B09302.

[20] D. Marsan, O. Lengliné, “Extending Earthquakes’ Reach Through Cascading”, Science (2008) 319, 1076–1079.

[21] G. J. McLachlan, T. Krishnan, “The EM algorithm and extensions” (John Wiley, 1996).

[22] G. Mohler, “Marked point process hotspot maps for homicide and gun crime prediction in Chicago”, International Journal of Forecasting (2014) 30 491–497.

[23] G. O. Mohler, M. B. Short, P. J. Brantingham, F. P. Schoenberg, G. E. Tita, “Self-Exciting Point Process Modeling of Crime”, Journal of the American Statistical Association (2011) 106 100–108.

[24] G. O. Mohler, M. B. Short, S. Malinowski, M. Johnson, G. E. Tita, A. L. Bertozzi, P. J. Brantingham, “Randomized Controlled Field Trials of Predictive Policing”, Journal of the American Statistical Association, 110:512, 1399-1411, DOI: 10.1080/01621459.2015.1077710

[25] J. Møller, J. G. Rasmussen, “Perfect simulation of Hawkes processes”, Adv. Appl. Prob. (2005) 37 629–646.

[26] J. Møller, J. G. Rasmussen, “Approximate simulation of Hawkes processes”, Methodol. Comput. Appl. Probab. (2006) 8 53–64.

[27] W. L. Perry, B. McInnis, C. C. Price, S. C. Smith, J. S. Hollywood, “Predictive Policing. The Role of Crime Forecasting in Law Enforcement Operations”, (RAND Corporation 2013).

[28] Predpol website, “UCLA Study on Predictive Policing”, see http://www.predpol.com/ucla-predictive-policing-study/

[29] Quantitative Criminology at Leeds, "`open_cp`", open source Python library, available at `https://github.com/QuantCrimAtLeeds/PredictCode`

[30] J. G. Rasmussen, "Temporal point processes: the conditional intensity function", lecture notes available at `http://people.math.aau.dk/ jgr/teaching/punktproc11/tpp.pdf`

[31] J. H. Ratcliffe, "Aoristic analysis: the spatial interpretation of unspecific temporal events", International Journal of Geographical Information Science, 14:7, 669-679, DOI: 10.1080/136588100424963

[32] B. D. Ripley, "Spatial Statistics", (John Wiley, 1981).

[33] G. Rosser, T. Cheng, "Improving the Robustness and Accuracy of Crime Prediction with the Self-Exciting Point Process Through Isotropic Triggering" Appl. Spatial Analysis (2016) `https://doi.org/10.1007/s12061-016-9198-y`

[34] G. Rosser, T. Davies, K. J. Bowers, S. D. Johnson, T. Cheng, "Predictive Crime Mapping: Arbitrary Grids or Street Networks?", Journal of Quantitative Criminology 2017 (33) 569–594.

[35] B. W. Silverman, "Density estimation for statistics and data analysis", (Chapman and Hall, 1986).

[36] A. Veen, F. P. Schoenberg, "Estimation of Space-Time Branching Process Models in Seismology Using an EM-Type Algorithm", Journal of the American Statistical Association (2008) 103 614–624.

[37] D. Vere-Jones, "Probabilities and Information Gain for Earthquake Forecasting", in Selected Papers From Volume 30 of Vychislitel'naya Seysmologiya, (American Geophysical Union, 2003). doi: 10.1029/CS005p0104

[38] J. Zhuang, Y. Ogata, D. Vere-Jones, "Stochastic Declustering of Space-Time Earthquake Occurrences", Journal of the American Statistical Association (2002) 97 369–380.

Author's Address: Jeremiah Horrocks Institute
University of Central Lancashire
Preston
PR1 2HE

Email: `matt.daws@cantab.net`