Title
Point Process Models for the spread of Coccidioidomycosis, an infectious disease, in California

Author
Wang, Jiajia

Publication Date
2019

Peer reviewed|Thesis/dissertation

University of California
Los Angeles

Point Process Models for the spread of Coccidioidomycosis, an infectious disease, in California

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Applied Statistics

by

Jiajia Wang

2019

ABSTRACT OF THE THESIS

Point Process Models for the spread of Coccidioidomycosis, an infectious disease, in California

by

Jiajia Wang
Master of Applied Statistics
University of California, Los Angeles, 2019
Professor Frederic Paik Schoenberg, Chair

Increasing incidence of coccidioidomycosis in California recent years catches our attention. To avoid future widespread contagion and decrease the risk for coccidioidomycosis, statistical models are used to forecast the spread of this epidemic disease. Developed methods for the estimation of Hawkes point process models and recursive point process models facilitate their application for exploring and foretelling the distribution of this type of infection. It is necessary that we can make a preparation in advance when coccidioidomycosis emerges and spreads. We use coccidioidomycosis data from Project Tycho to fit recursive model and Hawkes model with different triggering functions. Serval methods of evaluation are used to value each model's performance. We also use the first 80% of the data for estimation and the subsequent 20% of the data for evaluation to further check the goodness of models. Three point process models show big similarity, but recursive point process model still does better than others because of its situation-based productivity. We use the recursive model with training and testing data to see how well it can forecast the spread of coccidioidomycosis in California. We demonstrate that the recursive model is shown to fit well to our data and can provide us a reliable prediction.

The thesis of Jiajia Wang is approved.

Hongquan Xu

Vivian Lew

Frederic Paik Schoenberg, Committee Chair

University of California, Los Angeles

2019

To my parents . . .

for their unconditional love and support

Table of Contents

List of Figures

Acknowledgments

I would first like to express my sincere gratitude to my thesis advisor professor Frederic Paik Schoenberg for the continuous support of my master study. The door to Prof. Schoenberg office was always open whenever I ran into a trouble spot or had a question about my code or writing. He consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it. His patience, motivation, enthusiasm, and immense knowledge helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my master study.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Vivian Lew, and Prof. Hongquan Xu, for their help and encouragement. I am gratefully indebted to them for their very valuable comments on this thesis.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

CHAPTER 1

Introduction

Coccidioidomycosis, mostly named valley fever, is a disease of the skin or viscera, caused by infection with the pathogen soil-dwelling fungus Coccidioides immitis or Coccidioides posadasii. These two etiologic agents cause similar clinical diseases but are present in different endemic areas [1]. Coccidioides Immitis is largely found in California, in many areas of the San Joaquin Valley, whereas Coccidioides Posadasii can also be found in southwestern United States and northwestern Mexico [2].

Most people get this infection from inhalation of dust containing a form of the Coccidioides, named spores, after soil disruption, but a few can also start with skin infections. Based on the Epidemiologic Profile of Coccidioidomycosis from San Luis Obispo County [1], nearly 60% of people infected with coccidioidomycosis have minimal to no symptoms, while 40% of them, those with a relatively weak immune system, will have a range of possible clinical symptoms such as flu or pneumonia, including fatigue, cough, chest pain, fever, etc., and need to be hospitalized. In a small percentage of cases, the disease spreads, or disseminates, to other parts of the body, which may affect meninges, soft tissues, bones and even cause death [1].

In United States, there have been around 150,000 infections in the endemic areas each year and nearly 25,000 new infections occurring annually [3]. Although Coccidioidomycosis has higher annually cumulative total number of illness and death than most of other diseases, Coccidioidomycosis has attracted compara-

tively little attention over the past decade [4]. To alleviate future outbreaks of this type of infectious diseases, research and government's attention are something Coccidioidomycosis needs. To achieve this goal, statistical models can be used to discover the spread of infectious diseases and even forecast them both during and after an outbreak.

The Hawkes self-exciting point process model is a mathematical model created by Alan G. Hawkes [5]. Hawkes models are currently widely applied in diverse areas, from earthquake modeling to financial analysis. Although rarely have they been applied to the spread of infectious diseases, Hawkes models have some features making them amenable to modeling incidence of infectious diseases. A new type of point process model named recursive point process model [6] was introduced in 2017 to describe the incidence of infection diseases. The model incorporates different premises which improved the Hawkes point process model when expected number of transmissions is not static [6]. Although they both belong to point process family, comparisons between Hawkes self-exciting point process models and recursive point process models may help us to have a deeper understanding and even a better prediction of the spread of this disease.

The structure of this paper is as follows. Following a description of the Coccidioidomycosis data in Chapter 2, we briefly review point processes models, especially Hawkes self-exciting point process models and recursive point process models in Chapter 3. Methods of Evaluation are discussed in Chapter 4, and in Chapter 5, we fit the models to our Coccidioidomycosis data. We train models with first part of data and evaluate models using the remaining data in Chapter 6. Forecasting is discussed in Chapter 7, and Chapter 8 contains some concluding remarks and future discussion.

CHAPTER 2

Dataset

We download the Coccidioidomycosis dataset [7] from Project Tycho, a database provides open access to U.S. notifiable disease data that have been reported by cities and states. The dataset contains information from external sources of disease surveillance data, such as the United States Centers for Disease Control or the World Health Organization. Data from some sources were already in the public domain, whereas data from other sources were not public before inclusion in Project Tycho datasets [7].

Coccidioidomycosis dataset contains case counts for coccidioidomycosis in United States reported for time intervals. In addition to case counts, this dataset includes information about these attributes, such as the location, age group, diagnostic certainty, place of acquisition, and the source where Project Tycho team obtained case counts. Coccidioidomycosis dataset contains 10678 attributes in total from December 2002 to December 2017 among the world. It does not include time intervals for which no case count was reported but include time intervals for which a case count value of zero was reported [7]. Because of our interests, we use cases happened in California in this paper.

Project Tycho uses two ways to case count time series which caused two type of time interval series in this dataset. One is cumulative time interval series, and the other one is non-cumulative, in other word, fixed time interval series. Fixed-interval case count time series consist of mutually exclusive time intervals that all start and end on different date, and all have identical length. For example,

it contains a time interval from Jan 1 to Jan 7 with 10 cases and a second time interval from Jan 8 to Jan 14 with 7 cases. Cumulative time interval series, for example, has the same first time interval from Jan 1 to Jan 7 with 10 cases, but a different second time interval from Jan 1 to Jan 14 with 17 cases. We need to separate of case count intervals before we fit models to our dataset. After separating the dataset with different methods of case count, we found that cumulative time interval series started from December 2002 to December 2017, and fixed time interval series started from January 2016 to December 2017. The information they include are repeated. Based on our needs, we decide to use the fixed time interval series.

Finally, because the data consists of weekly counts of confirmed cases of Coccidioidomycosis in United States, to fit the point process model, the onset time for each individual case was drawn uniformly within each seven-day(weekly) time interval [6], and Coccidioidomycosis data is transferred as a list of sorted numbers starting from 0. There are total 12202 cases from Jan. 2006 to Dec. 2017.

CHAPTER 3

Point Process Models

3.1 Introduction to Point Process

A point process is a probabilistic model, sometimes known as counting processes or random scatters [8]. It is a collection of points randomly falling in some space S. Here we assume S has interval [0, T] in time. Let B be a subset of some complete separable metric space equipped with Lebesgue measure, ℓ , and assuming the spatial region is scaled so that $\wp$(B) =1. K=B $\times$ [0,T] is a bounded region [9]. We can conclude that a point process $\wp$ is defined as a mapping such that

$$\forall\ \omega \in \Omega,\ \wp(\omega) = \{k_1, k_2, ..., k_n\}, where\ k_i \in K.$$

In most applications, each point, ω_i ,represents the time or location of an event, such as an earthquake. When modeling purely temporal data, the space in which the points fall is simply a portion of the real line. Point process data arise in a wide variety of scientific applications, including epidemiology, ecology, and so on.

A point process is typically modeled via its conditional intensity, $\lambda(t)$, which represents the infinitesimal expected rate at which points are accumulating at time t, given information on all points occurring prior to time t [8].The simplest and most ubiquitous example of a point process is the Poisson point process.

3.2 Hawkes Models

3.2.1 Self-exciting

A point process $\wp$ may be called self-exciting [6] if

$$cov\{\ \wp(s,t),\ \wp(t,u)\ \} > 0\ for\ s < t\ <\ u.$$

$\wp$ is self-correcting if instead this covariance is negative. Thus, the occurrence of points in a self-exciting point process causes other points to be more likely to occur, whereas in a self-correcting process, the points have an inhibitory effect [6].

3.2.2 Hawkes Process

The Hawkes point process model is a mathematical model created by Alan G. Hawkes [5]. A purely temporal Hawkes process is a common self-exciting point process model, and it is a counting process that models a sequence of occurrence of some type over time, for example, infectious diseases or earthquakes.

Using $\{t_1, t_2, ..., t_k\}$ to denote the observed sequence of past arrival times of the point process up to time t, then the conditional intensity is given by

$$\lambda(t) = \mu(t) + K * \sum_{i:t_i<t} g(t - t_i),$$

where μ(t) represents the deterministic background rate and the function g(v) $\geq$ 0 is the triggering function. Every occurrence t_i contributes a secondary series of occurrences (aftershocks) occurring at a time varying rate K $*$ g(t - t_i), which in turn produces its own aftershock sequence, and so on [10]. K represents the expected number of new infections directly attributable to each case, $0 \leq K \leq 1$ [10].

$\lambda(t)$ is flexible and only needs specification of the background rate $\mu(t) > 0$ and the excitation function g(.) [10]. For many processes, the triggering density

g(u) decays gradually as the time delay u increases. A common choice for the triggering functions is for example, an exponential density function.

3.3 Recursive Models

Recursive point process model [6] , unlike Hawkes Model, is a model where the productivity for a subject infected at time t is inversely related to the conditional intensity at time t. Using $\{t_1, t_2, ..., t_k\}$ to denote the observed sequence of past arrival times of the point process up to time t, the conditional intensity is given by

$$\lambda(t) = \mu(t) + \sum_{i:t_i<t} H(\lambda(t_i)) \ * \ g(t - t_i),$$

where similarly μ(t) represents the deterministic background rate, the function g(v) ≥ 0 is the triggering function, but in the case of the recursive model, the productivity of any point t_i is given by H($\lambda(t_i)$). Thus the total productivity, for n points $t_1, t_2, ..., t_n$ is $\sum_{i=1}^{n} H(\lambda(t_i)))$ [6].

This shows important difference between Hawkes models and recursive models. For a Hawkes process, the points $t_1, t_2, ..., t_n$ all have constant productivity K, whereas for a standard recursive process, the productivity changes depends its situation. Recursive models incorporates different premises which improved the Hawkes models when expected number of transmissions is not static. It enables a greater precision of forecasts and can also provide a more detailed and precise account of heterogeneity and clustering when dynamic situation occurred [6].

CHAPTER 4

Methods of Model Evaluation

To evaluate Hawkes and recursive models, there are several ways we can use to test the goodness of fit.

4.1 Stoyan-Grabarnik diagnostic

Stoyan and Grabarnik proposed a diagnostic for point process models based on the reciprocal of the conditional intensity in 1991 [11]. We called Stoyan-Grabarnik diagnostic, given by $\frac{\sum \frac{1}{\lambda_i}}{T}$. Let t denote a point pattern dataset, consisting of the time $t_1, t_2, ..., t_n$ of events observed in a past arrival times.

Suppose that a point process model $\wp$ has been fitted to the data t. To each data point t_i, attach the weight $w_i = \frac{1}{\lambda(t_i)}$, where λ denotes the conditional intensity of the point process model. Stoyan and Grabarnik showed that

$$E \sum_{i=1}^{n} \frac{1}{\lambda(t_i)} = E \int \frac{1}{\lambda(t)} dN.$$

By martingale formula, we can get

$$= E \int \frac{1}{\lambda(t)} * \lambda(t) dN$$

$$= E \int_0^T 1 \; dt = \; T.$$

Therefore, $\frac{\sum \frac{1}{\lambda_i}}{T}$ should be ~ 1. Stoyan and Grabarni suggested that the weights w_i can be used for exploratory data analysis and goodness of fit testing [11]. Specically, they recommended that (1) extreme values of individual weight may

indicate 'outliers', (2) groups of extreme values may indicate regions of irregularity, and (3) global departures between the left and right sides of (1) may be used to test goodness of fit [11].

4.2 Harte's Ratio

The second method is the ratio suggested by Harte in 2015 [12]. It is a common way to check that the estimates obtained by MLE are reasonable and not merely local rather than global optima.The ratio is given by $\frac{\int_0^T \lambda(t)dt}{n}$.

By martingale formula, We can get that

$$E\int_0^T \lambda(t)dt$$

$$= E\int_0^T dn$$

$$= E(n).$$

Therefore, $\frac{\int_0^T \lambda(t)dt}{n}$ should be approximate to 1.

4.3 Super-thinning

The third way to evaluate models is using super-thinning. Super-thinning residuals [13] is a type of transformation based residuals for space-time point processes based on both thinned residuals [14] and superposed residuals [15].

Thinning is defined as suppose inf $\lambda(t_i, x_i, y_i) = b$. Keep each point (t_i, x_i, y_i) with probability $\frac{b}{\lambda(t_i,x_i,y_i)}$ [14]. One problem is that when $b = inf\ \lambda(t_i, x_i, y_i)$ is small, it will end up with very few points. Superposition also has a weakness. We suppose $sup\ \lambda(t, x, y) = c$. Superpose N with a simulated Poisson process of rate $c - \lambda(t, x, y)$[15]. This transformation is not as powerful as it should be when c = sup $\lambda(t_i, x_i, y_i)$ is large. Most of the residual points will be simulated in this case.

Therefore, to evaluate the goodness of fit, we used super-thinning [13] in this paper. In super-thinning, the existing data points are first thinned where each point is randomly kept independently of the others with probability $min\{\frac{b}{\hat{\lambda}(t)}, 1\}$, and then new points are superposed according to a Poisson process over the observed time window $[0, T]$,with rate $(b + \hat{\lambda}(t))^+$ [16]. Super-thinning requires an initial choice of the tuning parameter, b, and as suggested in Clements et al. [13], we used the simple default value of the total number of cases divided by the length, in days, of the observation period [16].

Choose some constant number b $\sim$ mean($\hat{\lambda}$) as suggested in Gordon et al. [17]. Superpose the observations where $\hat{\lambda}(t\ , x\ , y) < b$, add in points of a simulated Possion process of rate $b - \hat{\lambda}(t, x, y)$. Thin the observations where $\hat{\lambda}(t_i, x_i, y_i) > b$, keep each point (t_i, x_i, y_i) with probability $\frac{b}{\hat{\lambda}(t_i, x_i, y_i)}$. The resulting super-thinned residuals form a homogeneous Poisson process with rate b if and only if $\hat{\lambda}$ is the true conditional rate of the observed point process [13]. If t_i are the times of the super-thinned points, one may consider the interevent times, $r_i = t_i - t_{i-1}$ (with the convention $t_0 = 0$), which should be exponential with mean 1/b if the fitted model $\hat{\lambda}$ is correct, and it is natural therefore to inspect the uniformity of the standardized interevent times $u_i = F^{-1}(r_i)$, where F is the cumulative distribution function of the exponential with mean 1/b [6]. Any patterns or inter-point interaction in the residuals indicates a lack of fit of the model. Super-thinning is a better way to check the goodness of the fit visually.

4.4 Akaike information criterion

Hirotugu Akaike introduced a Akaike information criterion in the 1974 [18], which is now widely used for model selection, commonly the most difficult aspect of statistical inference. It is an estimator of the relative quality of statistical models for a given set of data. Given a collection of models for the data, AIC estimates

the quality of each model, relative to each of the other models. AIC not only rewards goodness of fit, but also includes a penalty that is an increasing function of the number of estimated parameters [19].

Suppose that we have a statistical model of some data. Let

$$AIC \ = \ 2 \times p - 2 \times ln(\hat{L}),$$

where p is the number of estimated parameters in the model, and L be the maximum value of the likelihood function for the model [19]. The preferred model is the one with the minimum AIC value. AIC plays an important role in model selection.

CHAPTER 5

Application to Coccidioidomycosis in California

Recorded cases of Coccidioidomycosis in California as we discussed in Chapter 2 is from Jan 2006 to Dec 2017. Figure 5.1 displays a histogram of the cases. There is an obvious one-year gap from 2010 to 2011 due to no case count was reported in this time period. The peak in the data is in 2009, and we can see the number of events in recent years seems to be fewer than previous years.

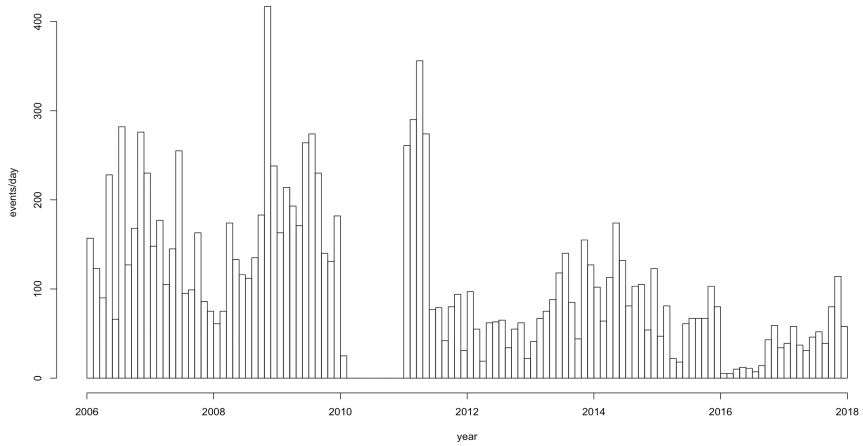
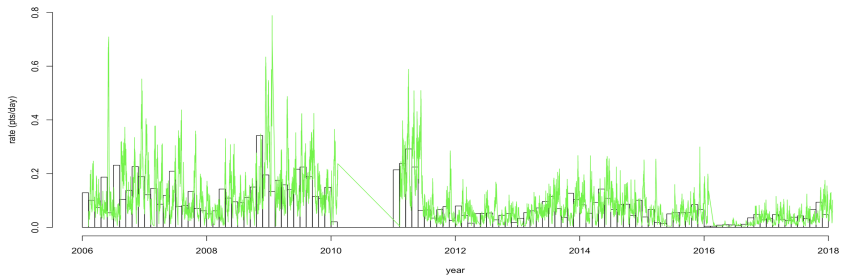


Figure 5.1: Histogram of Coccidioidomycosis in California from Jan 2006 to Dec 2017

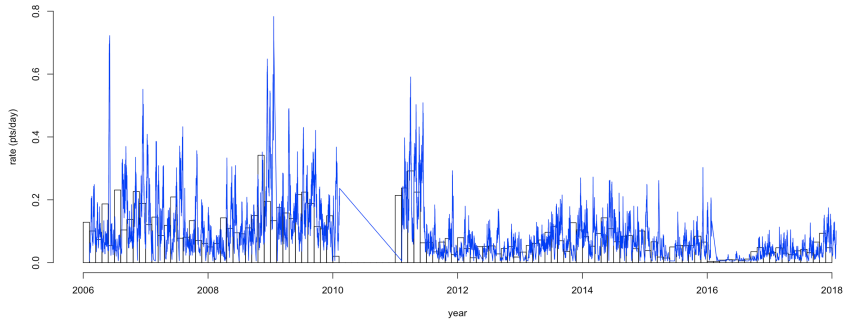
5.1 Model fitting

In this section, we fit Hawkes models with two different types of triggering functions: power-law and exponential. It is interesting to see whether same Hawkes

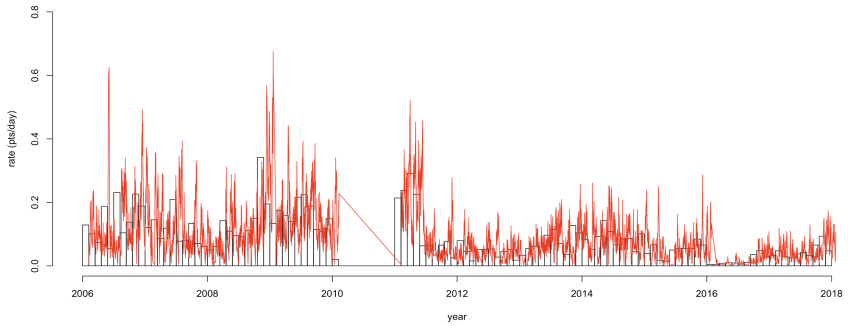
model with different triggering functions will have any affect on the results. We also fit recursive model with exponential function to see whether recursive model will perform better than Hawkes model or not.



(a) Estimated rate of Hawkes model with power-law function



(b) Estimated rate of Hawkes model with exponential function



(c) Estimated rate of recursive model with exponential function

Figure 5.2: Histogram of Coccidioidomycosis in California along with different estimated rates of three models

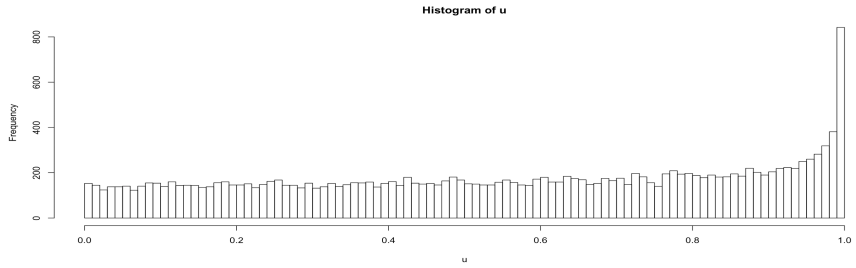
Firstly, we use Hawkes model with power-law function $g(u) = (p-1) \times$

$c^{\,p-1} \times (u+c)^{-p}$. The estimated parameters are (μ, κ, c, p) = (0.1390437 points/-day, 0.9503494 triggered points/observed point, 4.6699837 points/day, 4.4809440), with corresponding standard error estimates (0.02513146, 0.01243992, 2.11838361, 1.47667453). The estimated conditional intensity of the fitted Hawkes model is shown in Figure 5.2(a). We fit the Hawkes model with exponential function $g(u) = \beta \times e^{\,\beta u}$ to our data as well, and the estimated parameters in this model are (μ, c, β) = (0.1818148 points/day, 0.934889 triggered points/observed point, 0.695121 points/day). Its corresponding standard error estimates are (0.01724456, 0.01045866, 0.02335972). The estimated conditional intensity of the fitted Hawkes model with exponential function is also shown in Figure 5.2 (b). To compare with the Hawkes models, we fit the recursive model with exponential function to this Coccidioidomycosis data. The recursive model with exponential function is identical to Hawkes models with exponential function as above, but with one more parameter α. The estimated parameters are (μ, c, β, α) = (0.1138 points/day, 1.115 triggered points/observed point, 0.646 points/day, 0.1038), with corresponding standard error estimates (0.01627903, 0.02959700, 0.02307343, 0.01544262). Figure 5.2(c) shows the histogram of Coccidioidomycosis in California along with the estimated rates of the recursive model. From Figure 5.2, we can see the estimates rate of three models looks pretty similar. By calculating the mean of estimated rate for Hawkes power-law model, Hawkes exponential model, and Recursive model, we get similar mean: 5.5487, 5.726951, and 5.749835 correspondingly.

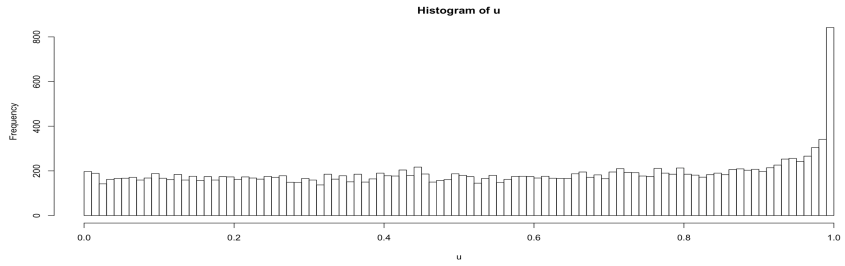
5.2 Model evaluation

As we mentioned in Section 4.1 and 4.2, it is important to use Harte's ratio and Stoyan-Grabarnik diagnostic to check whether the estimates are reasonable. The Harte's ratio for Hawkes models with power-law function is 1.00002541 and

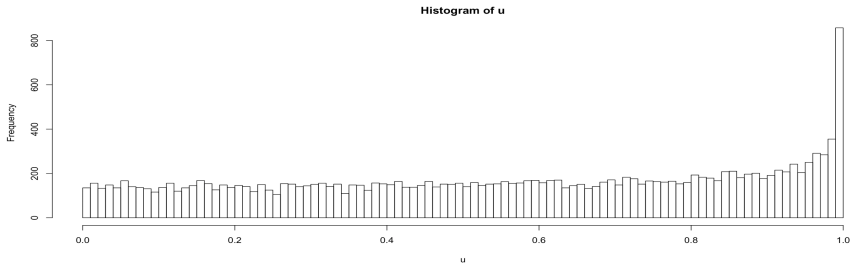
Stoyan-Grabarnik diagnostic is 0.9996276. Hawkes models with exponential function has a Harte's ratio equal to 0.999974 and Stoyan-Grabarnik diagnostic is equal to 0.9999556. For the recursive model, we got a Harte's ration equal to 1.002272, and Stoyan-Grabarnik diagnostic is equal to 1.000336. Both of these two diagnostics for all three models are approximate to 1, but recursive model's numbers are much closer to 1 than the other two's. This result indicates the estimates provided by all of these three models are all good and reasonable, whereas recursive model performs the best among three models.



(a) Histogram of standardized interevent times u for Hawkes with power law function



(b) Histogram of standardized interevent times u for Hawkes with exponential function

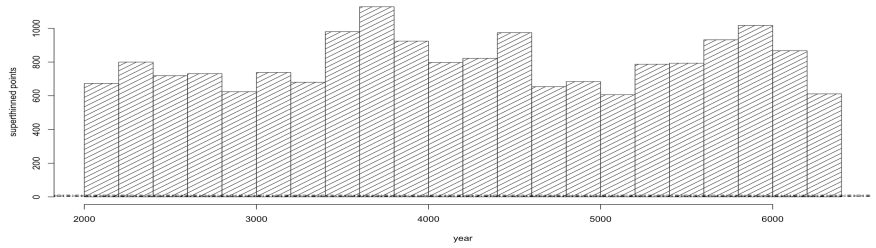


(c) Histogram of standardized interevent times u for recursive with exponential function

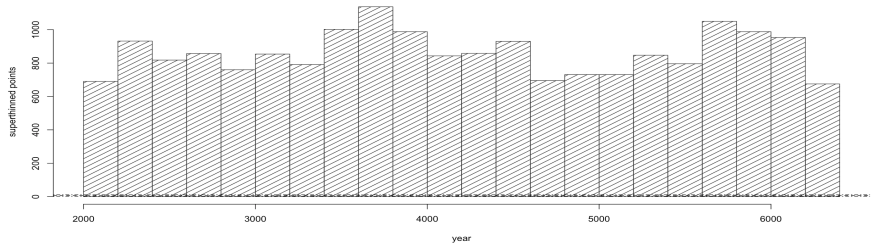
Figure 5.3: Histogram of standardized interevent times u for three models

Figure 5.3 shows the plot of the standardized interevent times u_i as we men-

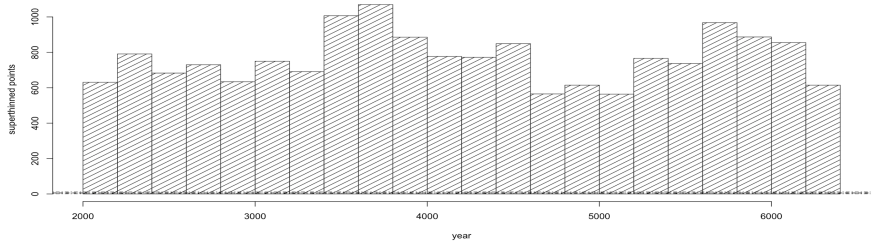
tioned in section 4.3. The plots for all three models looks similar, except tiny differences. All of these three plots show u_i spreads uniform, except there is a peak at u equal to 1, indicating many large time gaps between cases in our dataset.



(a) Super-thinned points for Hawkes model with power law function



(b) Super-thinned points for Hawkes model with exponential function



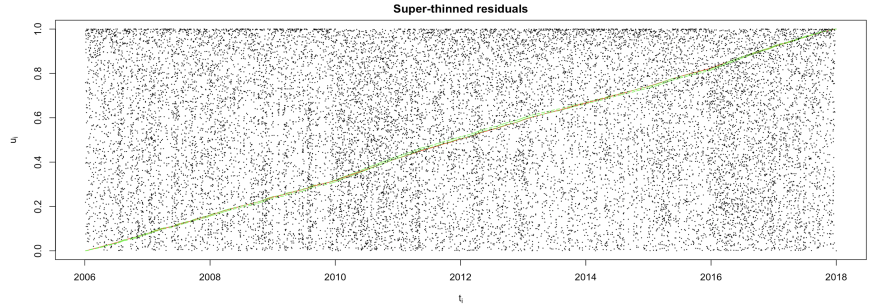
(c) Super-thinned points for recursive model with exponential function

Figure 5.4: Super-thinned points for three different models

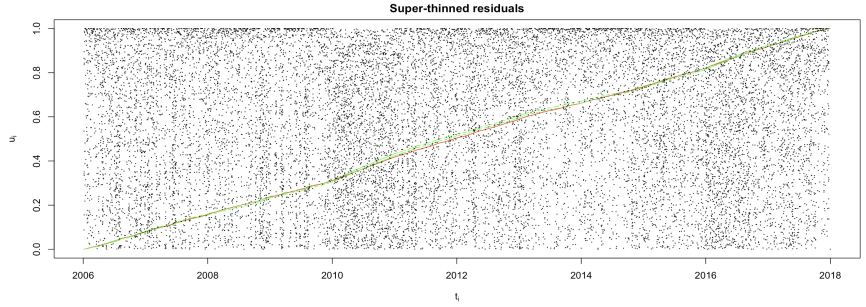
Figure 5.4 displays the super-thinned points of Coccidioidomycosis data for three models. Since super-thinned points are uniform in the plots, we can conclude everything goes well. Figure 5.5 shows the super-thinned residuals t_i along with their corresponding standardized interevent times u_i, as well as the cumulative sum of the standardized interevent times, and the individual 95% confidence

bounds based on 1000 simulations of an equivalent number of uniform random variables. The solid red line shows, for each value of t_k , the normalized cumulative sum $\frac{\sum_{i=1}^{k} u_i}{\sum_{i=1}^{m} u_i}$,where m is the number of super-thinned residuals. Dotted blue lines show lower and upper simultaneous 95% confidence bounds.

(a) t_i *vs* u_i *plot* for Hawkes model with power law function



(b) t_i *vs* u_i *plot* for Hawkes model with exponential function



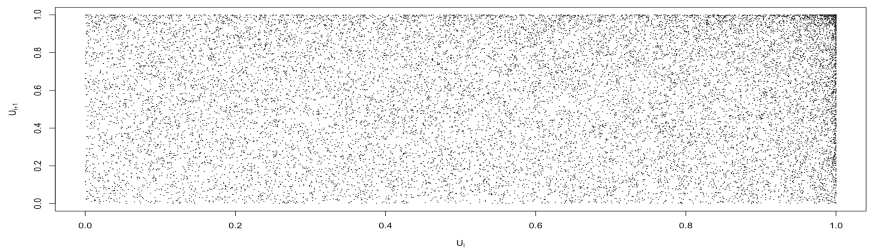
(c) t_i *vs* u_i *plot* for recursive model with exponential function

Figure 5.5: Super-thinned residuals t_k using b = 100 points/year and their corresponding standardized interevent times u_k with their 95% confidence bounds for three different models

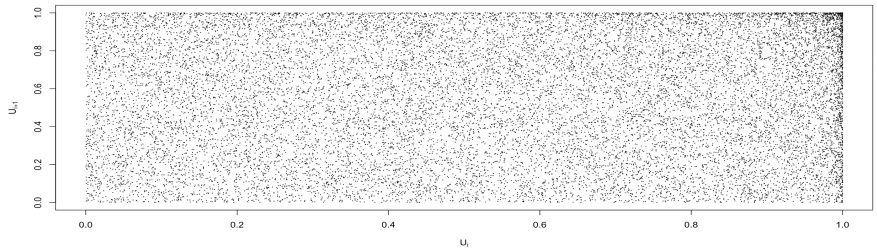
The super-thinned residuals for three models appear to be well scattered, and in general the interevent times also appear to be largely well. However, in figure 5.5(a) , the normalized cumulative sum is higher than its upper confidence bound from 2008 to 2010, and it is lower than the lower bound from the middle of 2010 to 2012. In figure 5.5(b), the normalized cumulative sum is lower than the lower bound from 2011 to 2013. And we also can see the normalized cumulative sum is obviously lower than the lower confidence bound from 2010 to 2014 in figure 5.5(c). Three models show us similar information, and these problem might be because of large gap from 2010 to 2011.

Figure 5.6 shows the lags plot of the standardized interevent times u_i of the super-thinned residuals using b = 100 points/year. The lag plot for three models looks good and similar, although there are more points in the upper right corner than we would expect. We use Ripley's K-function [20] for each of the plots to test for clustering for our three models. We compute the K function with confidence bounds. One can obtain bounds via simulation or theoretically. Theoretical K(h) function is $\frac{1}{\mu}\times$ E(number of extra events within distance t of a randomly chosen event). K is estimated in the obvious way using data, where μ is the number per unit area of events, estimated as n divided by the size of the spatial area, and E(number of extra events within distance h of a randomly chosen event) is estimated just by taking the average number of points within distance t of any point, averaged over the points in the dataset. But various edge correction ideas are available [20]. For many point processes the expectation in the numerator of K(h) function can be analytically evaluated [21], for a homogeneous stationary Poisson process, K(h) can be written as: K(h)= $\pi \times h^2$. The K function K(h) can indicate whether the points in the dataset have more or less clustering within distance h compared to a stationary Poisson process.

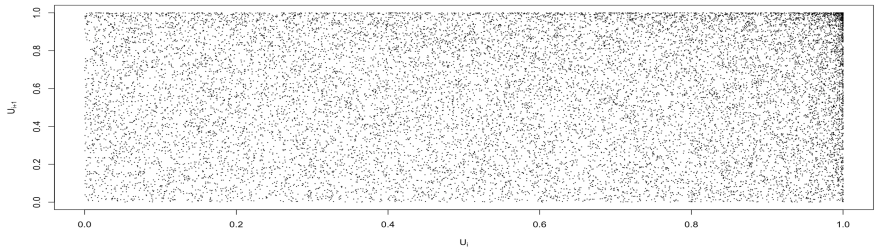
When $K(h) > \pi \times h^2$, this implies a mean point count higher than would be expected, and hence indicates some degree of clustering at scale h. Similarly,



(a) Lag plot for Hawkes model with power law function



(b) Lag plot for Hawkes model with exponential function

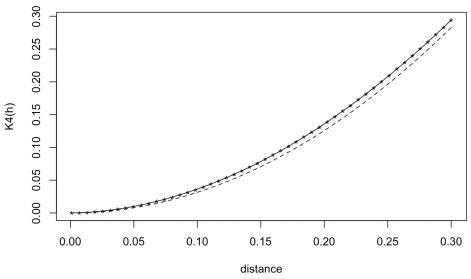


(c) Lag plot for recursive model with exponential function

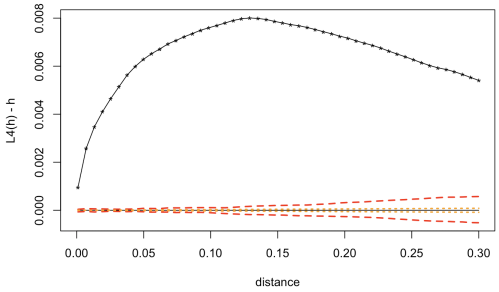
Figure 5.6: Lag plot of the standardized interevent times u_i of the super-thinned residuals using b = 100 points/yea for three different models

$K(h) < \pi \times h^2$ indicates some degree of dispersion at scale h. To further standardize K-functions in a manner that eliminates the need for considering these values [21], we get L function $L(h) = \sqrt{\frac{K(h)}{\pi}} - h$. And when $L(h) > 0$, it means some clustering exist at scale r, and when $L(h) < 0$, it indicates some statistically significant degree of dispersion at scale h [21].

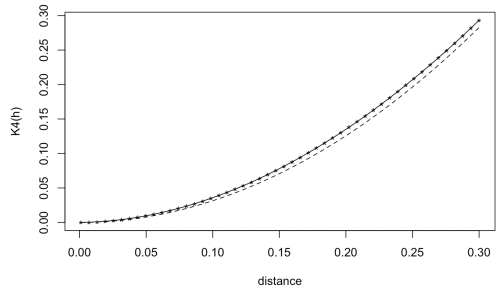
Figure 5.7 shows the results using K-function for the lag plot for three different models. K4 in the plots is our standard edge correction. In figure 5.7(a), the straight line K lies above the expected value of $\pi * h^2$ for all distances between



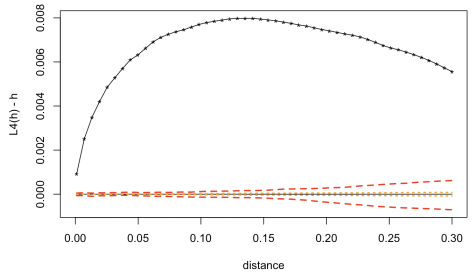
(a) Hawkes model with Power-law function: Plot of K(h) vs. h up to 0.3



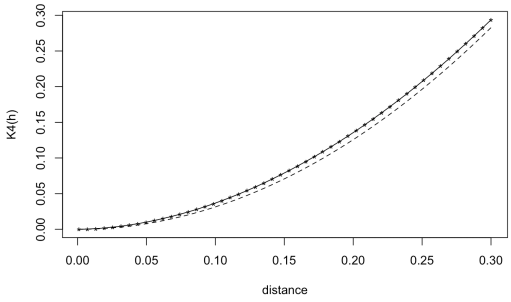
(b) Hawkes model with Power-law function: Plot of L(h) - h for all cases



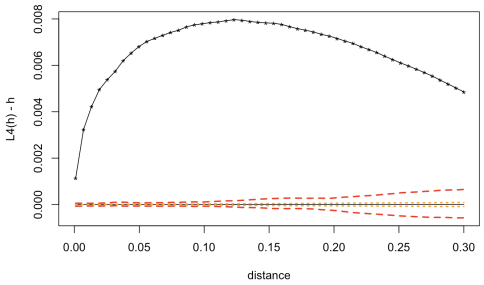
(c) Hawkes model with exponential function: Plot of K(h) vs. h up to 0.3



(d) Hawkes model with exponential function: Plot of L(h) - h for all cases



(e) Recursive model: Plot of K(t) vs. distance up to 0.3 for all cases.



(f) Recursive model: Plot of L(h) - h for all cases.

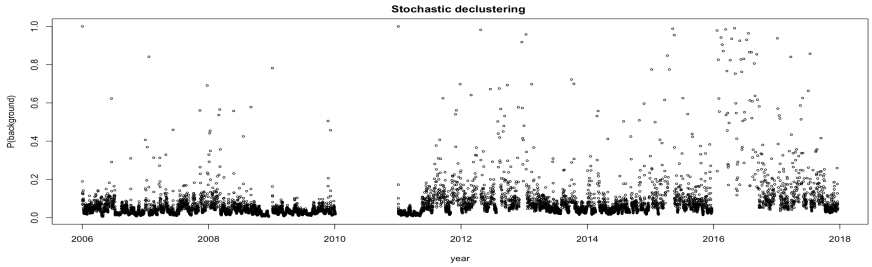
Figure 5.7: K(t) and L(t) plots for standardized interevent times u_i for three models

0 and 0.3. $K > \pi * h^2$ indicates some degree of clustering in the data. And in figure 5.7(b), solid horizontal line provides a reference for L(h) under com-

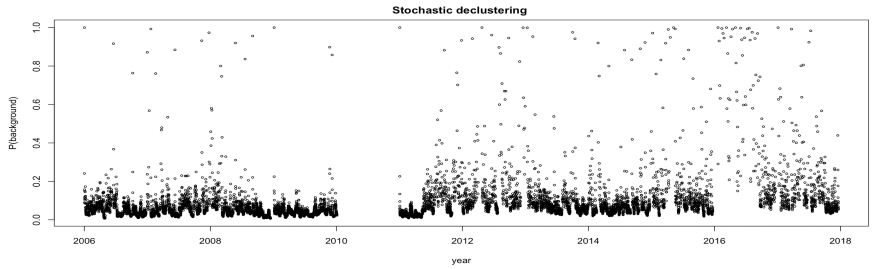
plete spatial randomness. Red dashed lines are 95% confidence bounds of L(h)-h estimated from 100 simulations and orange dashed lines are 95% theoretical confidence bounds of L(h)-h. We can see $L(h) - h > 0$ in the plot. Therefore, there is statistically significant clustering of cases at distances in the lag plot for Hawkes model with power-law function. Similarly, plots in figure 5.7(c) and (e) indicate $K > \pi * h^2$. Figure 5.7(d) and (f) also display $L(h) - h > 0$ for both models. Thus, although clustering does not look too significant by eye in three lag plots, based on the K-function applied to three models, we conclude that some significant degree of clustering exist in our dataset.

Figure 5.8 shows the stochastic declustering of the Coccidioidomycosis. For each observed point t_i , the y-coordinate, $\frac{\mu}{\lambda(t_i)}$, is the probability, based on fitted model, that the point is attributed to the background rate (μ) as opposed to contagion from previous points. In figure 5.8, we can see the vast majority of points are attributed to contagion rather than novel outbreaks. For Hawkes model with power-law function in figure 5.8(a), 17 points are assigned near certainty of being attributed to new outbreaks, especially 1 point with a probability equal to 1. 50 points are assigned substantially higher probability(>50%) of being attributable to new outbreaks rather than contagion from one of the other points in the dataset. In figure 5.8(b), based on Hawkes model with exponential function, we find that 39 points are assigned near certainty of being attributed to new occurrence. Two of them have a probability equal to 1, and 126 points are assigned substantially higher probability(>50%) of being attributable to new outbreaks. In figure 5.8(c), 28 points are assigned near certainty of being attributed to new outbreaks, especially 1 point has a probability equal to 1. And two of them have a probability equal to 1. 82 points are assigned substantially higher probability(>50%) of being attributable to new outbreaks rather than contagion according to the recursive fitted model. High probability of being attributable to new outbreaks meaning these people might be the ones who spread the disease to the public. This may

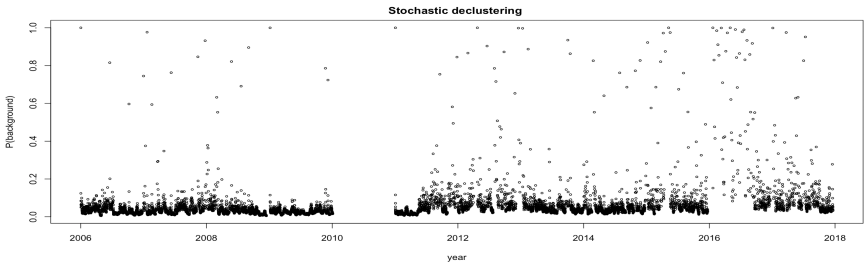
suggest that government or researchers should pay more attention to these people, which can effectively prevent or avoid spreading infection. From plots, we can also find that more points are attributable to new occurrence in 2012 to 2018 rather than in 2006 to 2010.



(a) Stochastic declustering for Hawkes model with power law function



(b) Stochastic declustering for Hawkes model with exponential function



(c) Stochastic declustering for recursive model with exponential function

Figure 5.8: Stochastic declustering for three different models

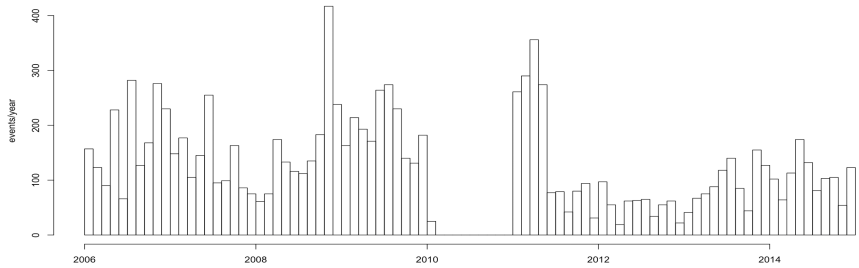
Finally, to evaluate our models in statistical ways, we calculate the log-likelihood for each model. For the Hawkes model with power law function, the log-likelihood is 5499.553; the log-likelihood for Hawkes model with exponential function is 5498.71; and the log-likelihood for Recursive model is 5525.479. AIC values are calculated as we mentioned in section 5.4. For the Hawkes model with power law

function, we get AIC value is equal to -10991.106. The AIC value for Hawkes model with exponential function is -10991.42, which is a little bit lower than the AIC value for Hawkes model with power-law function. For the recursive model with exponential function, we get an AIC value equal to -11042.994. For this Coccidioidomycosis dataset, compared with other two models, recursive model has significantly lower AIC value, informing that it is doing much better than both Hawkes models.

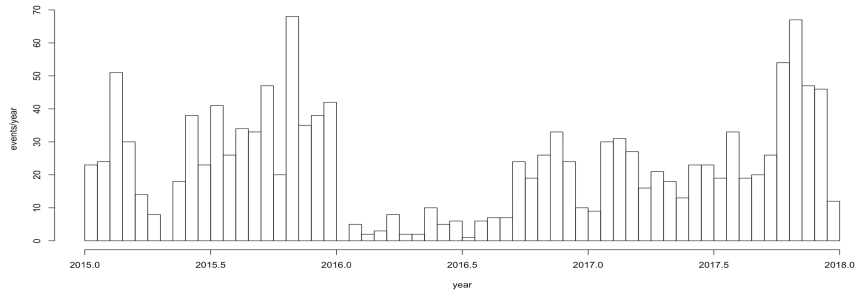
CHAPTER 6

80/20 Evaluation

In this chapter, we separate our Coccidioidomycosis into two different parts, the first 80% of dates, from Jan. 1, 2006 to Jan. 3, 2015, as our training data and the last 20% of dates, from Jan. 4, 2015 to Dec. 16, 2017, as our testing data. We will use the first part of the data to fit our models, and then use the last part of the data to evaluate our models. Two separated datasets are shown in Figure 6.1.



(a) Histogram of Coccidioidomycosis in California from Jan 2006 to Dec 2014.

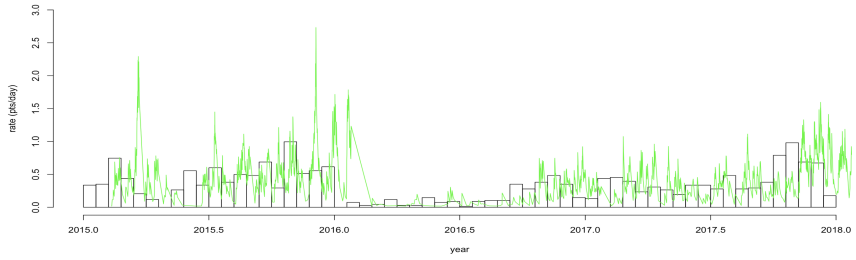


(b) Histogram of Coccidioidomycosis in California from Jan 2015 to Dec 2017.

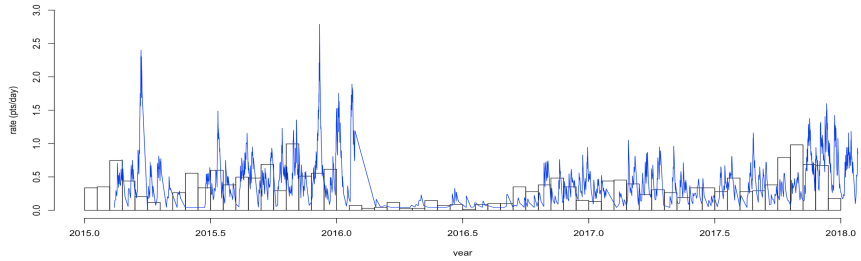
Figure 6.1: Histogram of Coccidioidomycosis after separating in two parts

6.1 Model fitting using training dataset

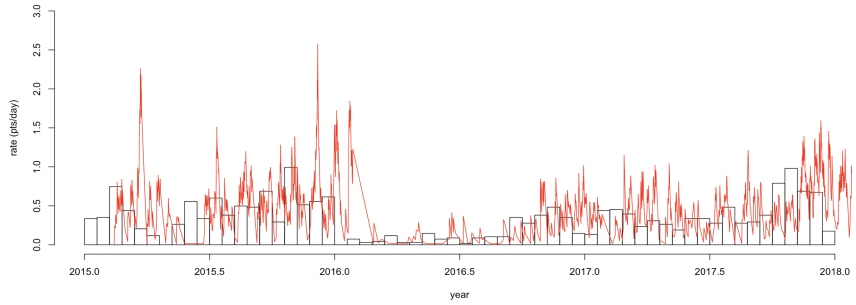
We use the first 80% of our data to fit our Hawkes and Recursive models.



(a) Testing data: Estimated rate of Hawkes model with power-law function



(b) Testing data: Estimated rate of Hawkes model with exponential function



(c) Testing data: Estimated rate of recursive model with exponential function

Figure 6.2: Histogram of Coccidioidomycosis in California along with different estimated rates of three models using testing dataset

The optimal Hawkes model with power law function using training dataset has estimated parameters $(\mu, \kappa, c, p) = (0.06106282$ points/day, 0.98135918 triggered points/observed point, 2.76599042 points/day, $3.21616419)$, with corresponding standard error estimates (0.02655895, 0.01239711, 0.62622026, 0.44570360). The

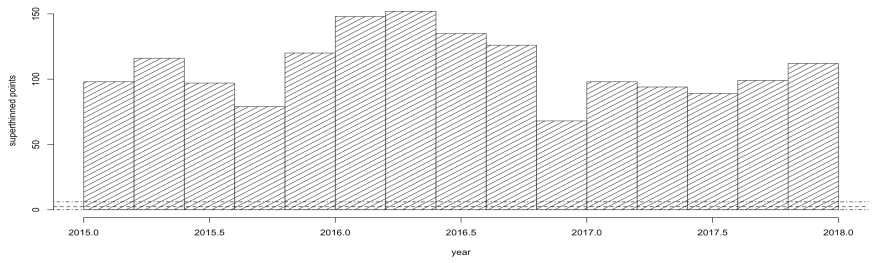
estimated parameters for Hawkes model with exponential function is (μ, c, β) = (0.1571098 points/day, 0.9522980 triggered points/observed point, 0.7246614 points/day). Its corresponding standard error estimates is (0.02159658, 0.01124194, 0.02617143). The optimal recursive model using training data is with estimated parameters (μ, c, β, α) = (0.05074475 points/day, 1.28445653 triggered points/observed point, 0.65005831 points/day, 0.17365545). It has corresponding standard error estimates (0.01458985, 0.04004347, 0.02561124, 0.01825384). Then we use our estimated parameters from training data to fit models to our testing data. Figure 6.2 shows the estimated rate λ for three models. We can see they perform pretty similarly using testing data.

6.2 Super-thinning evaluation using testing dataset

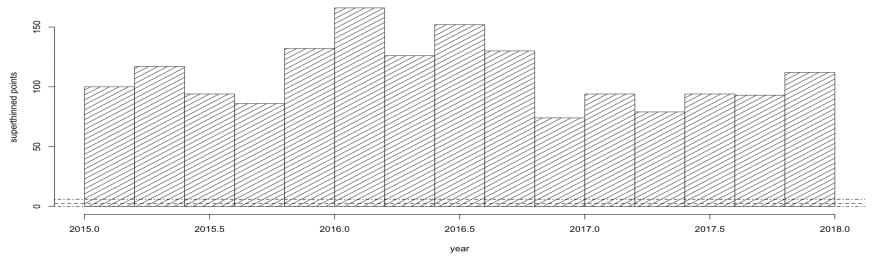
To visualize the differences among our Hawkes and recursive models, as we did in previous chapter, we use super-thinning to evaluate our models. In figure 6.3, we can see super-thinned points for all three models are nearly uniform.

Lag plots in figure 6.4 seems to be fine, except there is a cluster in the up right corner for all three models. This result is the same as we observed from our full models in Chapter 5. After using K function to test the clustering further, based on figure 6.5(a), (c), and (e), we can get $K > \pi * h^2$ and from figure 6.5(b), (d) and (f), we can know $L(h) - h > 0$ for all three models. Here, we may say some degree of clustering exist among the data.

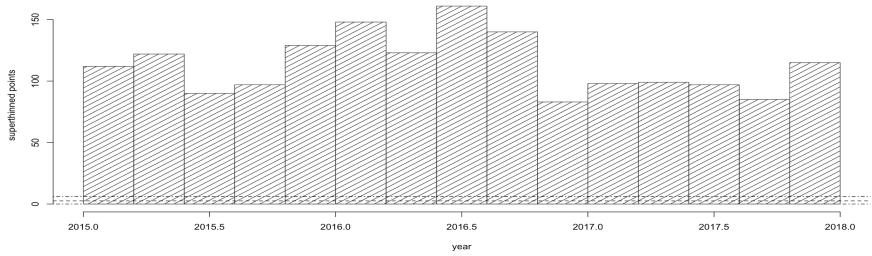
In figure 6.6, the super-thinned residuals for three models seem to be well scattered. We can see the normalized cumulative sums are little lower than lower confidence bounds from 2016 to 2017 in figure 6.6(a) and (b). Generally, there is almost no pattern among these plots and we can conclude that three models fit our Coccidioidomycosis data well.



(a) Testing data: Super-thinned points for Hawkes model with power law function

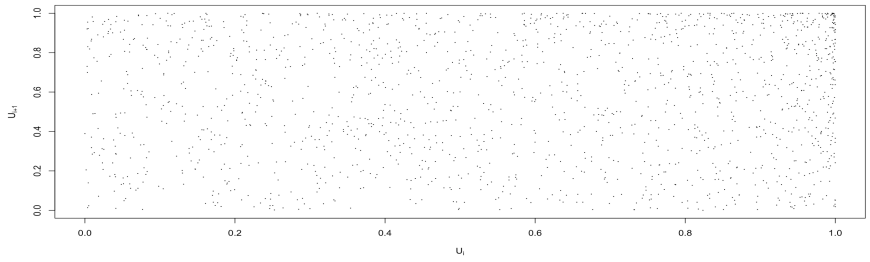


(b) Testing data: Super-thinned points for Hawkes model with exponential function

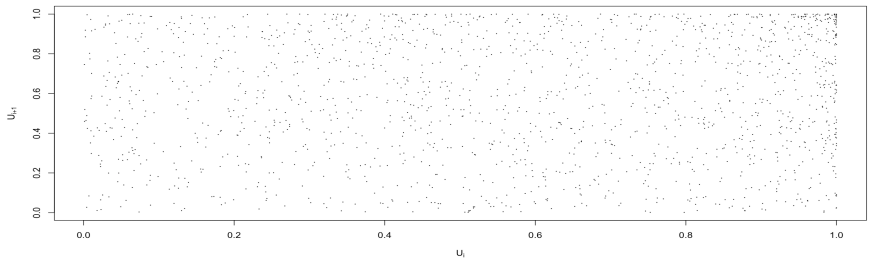


(c) Testing data: Super-thinned points for recursive model with exponential function

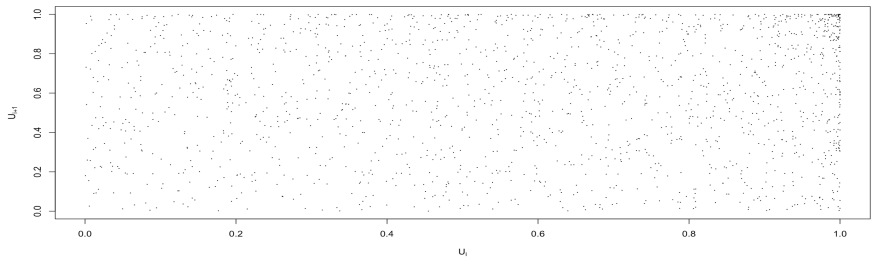
Figure 6.3: Super-thinned points for three different models using testing dataset



(a) Testing data: Lag plot for Hawkes model with power law function

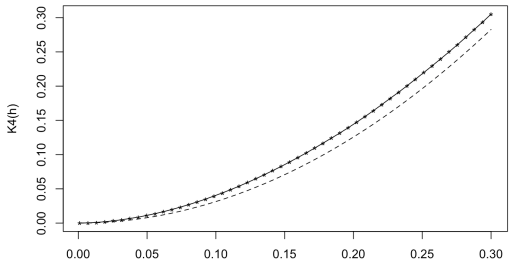


(b) Testing data: Lag plot for Hawkes model with exponential function

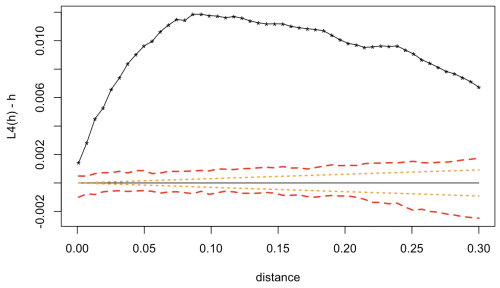


(c) Testing data: Lag plot for recursive model with exponential function

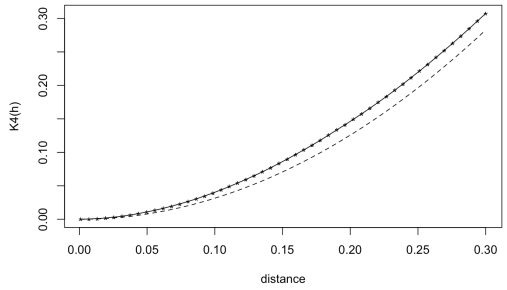
Figure 6.4: Lag plot of the standardized interevent times ui of the super-thinned residuals using b = 100 points/yea for three different models using testing data



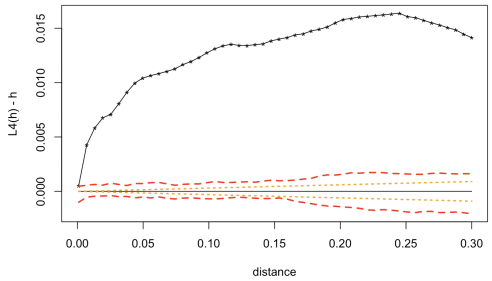
(a) Testing data: Hawkes model with Power-law function - Plot of K(h) vs. h up to 0.3



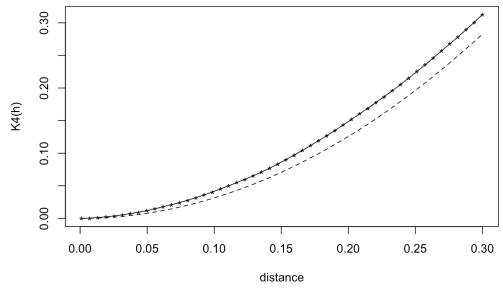
(b) Testing data: Hawkes model with Power-law function - Plot of L(t) - t for all cases



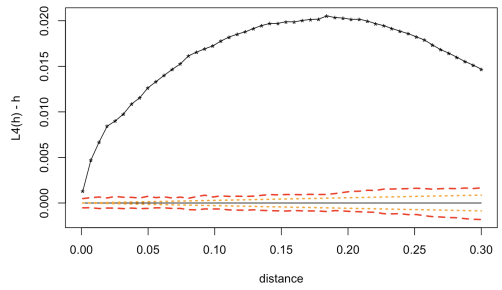
(c) Testing data: Hawkes model with exponential function- Plot of K(h) vs. h up to 0.3



(d) Testing data: Hawkes model with exponential function - Plot of L(t) - t for all cases

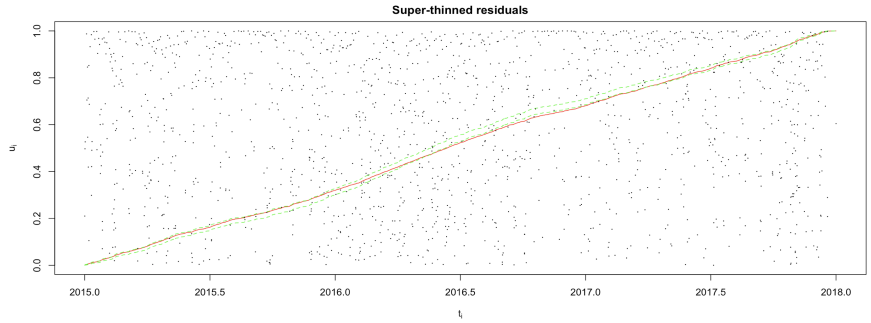


(e) Testing data: Recursive model - Plot of K(t) vs. distance up to 0.3 for all cases.

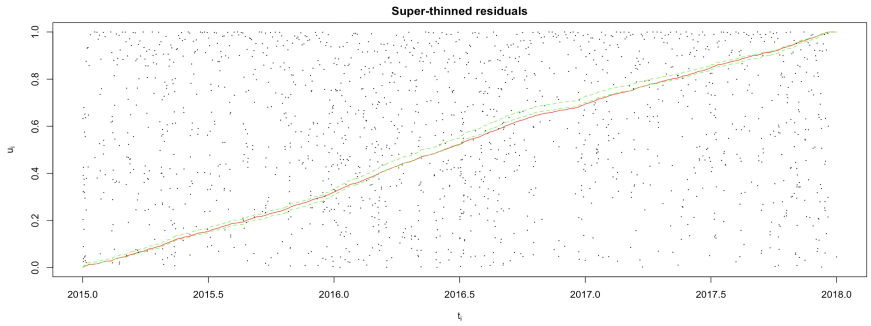


(f) Testing data: Recursive model - Plot of L(t) - t for all cases.

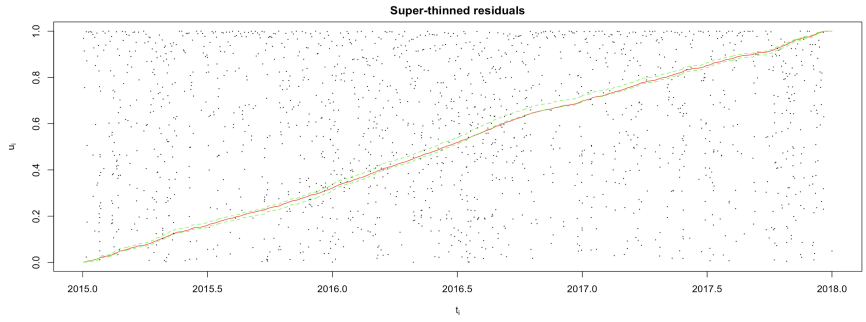
Figure 6.5: Testing data: K(t) and L(t) plots for standardized interevent times u_i for three models



(a) Testing data: t_i *vs* u_i *plot* for Hawkes model with power law function



(b) Testing data: t_i *vs* u_i *plot* for Hawkes model with exponential function



(c) Testing data: t_i *vs* u_i *plot* for recursive model with exponential function

Figure 6.6: Super-thinned residuals t_k using b = 100 points/year and their corresponding standardized interevent times u_k with their 95% confidence bounds for three different models using testing data

CHAPTER 7

Forecasting using recursive model

In this section, we are going to use recursive model, the best model which we selected in Section 5, to do the future prediction. Figure 7.1 shows the cumulative total number of events for Coccidioidomycosis dataset from 2006 to 2017. Blue line separated the training and testing data at Dec. 28, 2014.

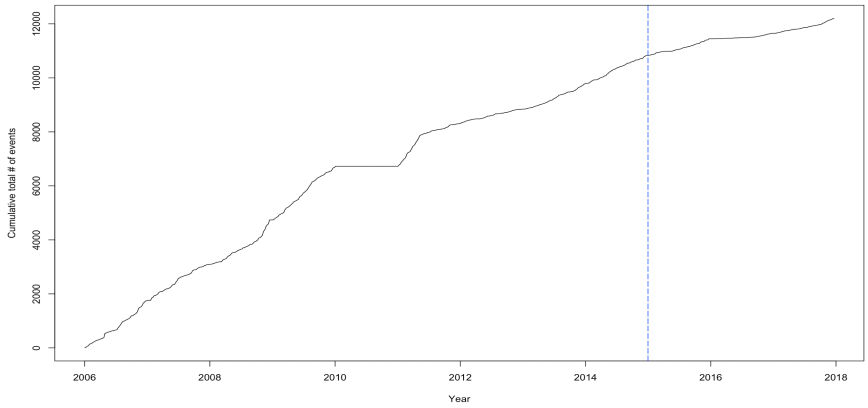
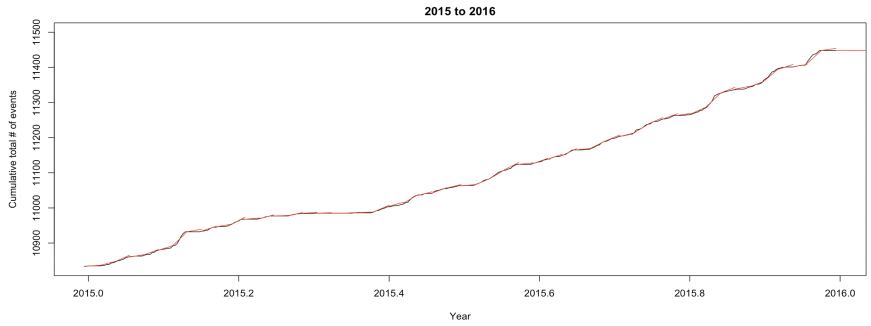


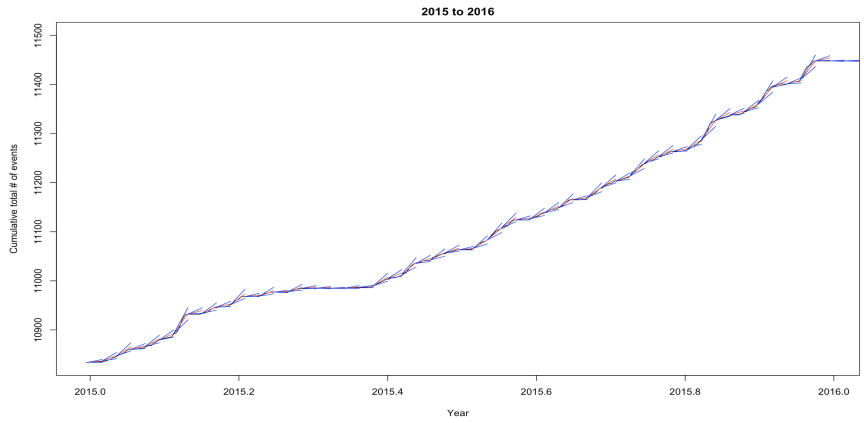
Figure 7.1: Cumulative total number of events

We first fitted our recursive model to the training data, from Jan.1, 2006 to Dec.24, 2017, giving us a set of estimated parameters (0.05074475, 1.28445653, 0.65005831, 0.17365545). For any given week, we used the parameters from training and all the data up to the beginning of the week in question, and added up the estimated lambda, which is the average of lambdas times seven days for the week, to get our expected number of events. And this would also be the variance of the number of predicted events in the week. From 2015 to 2018, we have 155 weeks in the testing period. And because we are interested in predicting for three

years with a large number of weeks, we separated the plots for each year so we can see all details more clearly. Figure 7.2, 7.3, and 7.4 are the forecasting results from our recursive model.



(a) Cumulative forecasts with the observed number of events

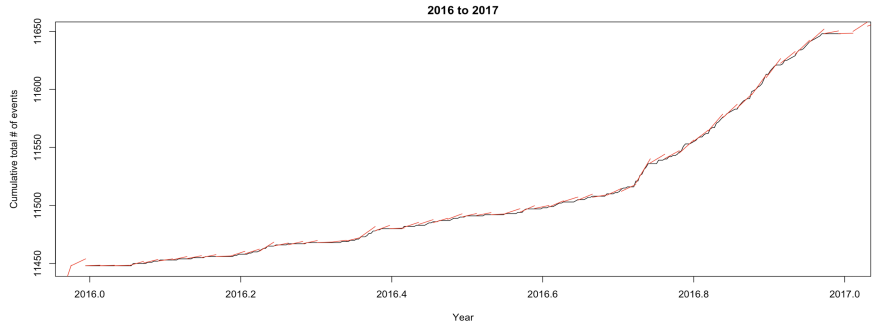


(b) Cumulative forecasts and its confidence bounds with the observed number of events

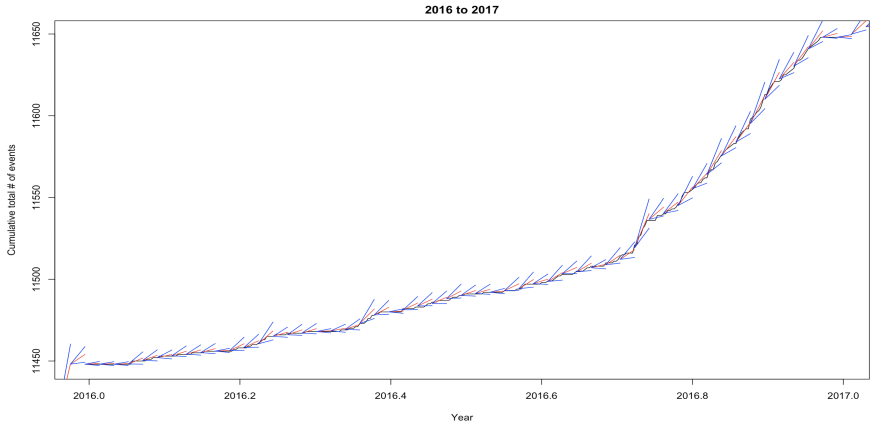
Figure 7.2: Prediction by fitting recursive model (2015-2016)

The cumulative forecasts follow the observed number of events relatively well. The red line in the plot is our estimated number of events from recursive model, and blue lines are the confidence bounds for this estimation. The total cumulative number of events is matched well. In figure 7.2(a), we can see some forecasts are higher than actual number events, especially at the end of 2015-2016. We realize that forecasts may not do well when many fluctuations appear in a short period.

In figure 7.3(a), we can see most of our estimated values are higher than actual number of events. In 2016, the curve for the cumulative number of events is more



(a) Cumulative forecasts with the observed number of events

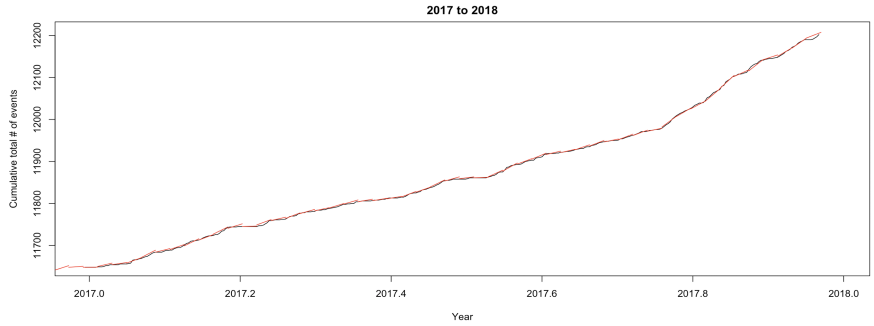


(b) Cumulative forecasts and its confidence bounds with the observed number of events

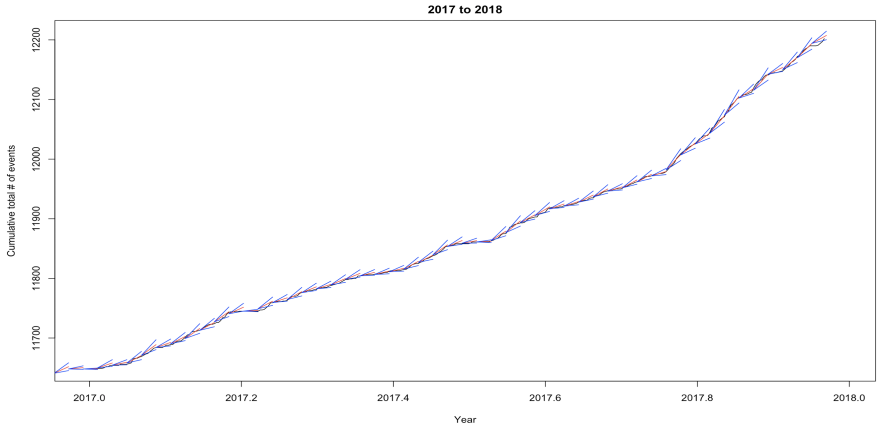
Figure 7.3: Prediction by fitting recursive model (2016-2017)

gradual than in previous years. The rate of increasing of events tends to be a little bit slower. This may cause our over-prediction in this period of time. However, from figure 7.3(b), although the prediction is higher, the recursive model seems to perform well in the testing period, with most realizations falling within two-standard errors from our predictions.

Similarly with figure 7.2(a), in figure7.4(a), some of our estimated values are a little higher than our actual observed number of events, especially in the beginning of 2017. The forecasts turn to be better when the slope become much steeper. And the forecasts at the end of year 2017 do not perform well since there are some small fluctuations exist. However, from figure 7.4(b), it seems that almost all realizations falling within our confidence bounds, indicating a good prediction.



(a) Cumulative forecasts with the observed number of events



(b) Cumulative forecasts and its confidence bounds with the observed number of events

Figure 7.4: Prediction by fitting recursive model (2017- 2018)

To evaluate our forecasting statistically, we will use RMS Error, a measure of the error around the regression line [22]. It defined as $RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$ [22]. RMS error is a measure of accuracy, to show the forecasting errors of models for a particular dataset [23]. In order to calculated our RMS error, we compared the estimated number of events for the last day in each week with the observed number of event on that day. Our RMS error is equal to 3.615393, indicating that the difference between our forecasts and the actual observed cases is only around 3.62 events/day. Based on our large number of data in the testing dataset, this RMS error can prove we have a pretty good forecasting.

CHAPTER 8

Conclusion

8.1 Concluding Remarks

The application of Hawkes point process models and recursive point process model to predict the spread of Coccidioidomycosis in California present that these methods have the potential to be a useful addition to the pallet of available methods for disease forecasting. In all aspects of fitting and evaluation of the spread of the infection we performed, recursive point process models with exponential function performed as well as, or better than, Hawkes point process models with power-law or exponential functions. From model fitting and evaluation, we can realize that all three point process models perform well, and it is also hard to visually see any differences among some methods of evaluation. It is reasonable since although they are different models, they are all belongs to point process family, which may cause large similarity. Statistically, based on AIC value, recursive model perform significantly better than the other two models. This shows that recursive point process model truly has an improvement over Hawkes point process models because of its flexibility, enabling it to account for changes in the spread rate over the course of the disease. From forecasting, with a small number of RMS error, we can conclude our recursive model can predict future events well. This could help us to make preparations for the spread of Coccidioidomycosis in California in advance.

8.2 Future Discussion

In the Coccidioidomycosis data we used, there is a large gap from 2010 to 2011. Some small gaps also exist among the data. For example, our data does not have the information for week from Dec. 23, 2006 to Dec. 30, 2006. No case count was reported during these periods in California according to Project Tycho. These time periods with no case count reported might be incorrect, but we do not know what actually happened. Large gap in the data may cause some inaccurate model fitting and training, giving us unreliable predictions when we do the forecasting. It is important for us to verify the accuracy and integrality of our data in future further research and analysis. Moreover, since our data is collected in a weekly format, our data are not ideally precise as the times of the points had to be randomized within weeks. It would be better if we can get a more accurate data, which contains information about the exact time and date each case happened.

Finally, in this paper, we only use point process models to analyze the Coccidioidomycosis. We receive plenty similar results from using Hawkes modes and recursive model. To discover something new and different, for infection disease, SEIR compartmental model [24] is a good choice. This model is especially popular and it is one of the simplest compartmental models. The name of the SEIR model derives from the fact that they involve coupled variables relating the number of susceptible people (S), a latent/exposed population (E), number of people infected (I), and number of people who have recovered (R) [24]. For many important infections we know, there is an important incubation period during which the individual has been infected, but is not yet infectious themselves [24] . Unlike other models, SEIR models take into the consideration this exposed period. Since our point process models give us pretty similar result, in the future, it will be interesting if we can try SEIR model and compare the results with our point process models. We hope that we could get some distinguished and surprising

results from fitting SEIR models to our dataset.

Appendix: R code

```
#read data
t <- read.csv("~/desktop/converted_data.csv")
t <- sort(t[, 2])
n = length(t)
T = 4368
hist(
t / 365 + 2006,
nclass = 100,
main = "",
xlim = c(2006, 2018),
prob = F,
xlab = "year",
ylab = "events/day"
)

#Hawkes model fitting - power-law function
##### Fit the Hawkes model.
##### lambda(t,x,y) = mu + K SUM g(t-t_i).
##### Here, g(u) = (p-1)c^(p-1) (u+c)^(-p),
##### g(x,y) = (q-1)d^(q-1)/? (r^2+d)^(-q), with x^2+y^2=r^2,
##### For any theta, the integral of lambda over
##### the whole time region is approx.
##### mu T + nK.
##### Here the parameter vector theta = (mu, K, c, p)

m3 = function(x)
signif(x, 3)
```

```
## First we will write the loglikelihood function in R.
logl = function(theta2) {
## returns the negative loglikelihood.
mu = theta2[1]
K = theta2[2]
c = theta2[3]
p = theta2[4]
cat("\n mu = ", m3(mu), ", K = ", m3(K), ",
c = ", m3(c), ", p = ", m3(p), "\n")
if (min(c(mu, K, c, p - 1)) < 0.000000001)
return(99999)
if (K > .99999)
return(99999)
sumlog = log(mu)
intlam = mu * T + n * K
const = K * (p - 1) * c ^ (p - 1)  # k*beta
for (j in 2:(n)) {
gij = 0
for (i in 1:(j - 1)) {
gij = gij + (t[j] - t[i] + c) ^ (-p) #gij=gij+e^(-beta*(tj-ti))
}
lamj = mu + const * gij
if (lamj < 0) {
cat("lambda ", j, " is less than 0.")
return(99999)
}
sumlog = sumlog + log(lamj)
}
loglik = sumlog - intlam
```

```
cat("loglike␣is␣",
loglik,
"sumlog␣=␣",
sumlog,
".␣integral/n␣=␣",
intlam / n,
".\n")
#if(draw) lines(u,Kg,col="white",lty=2)
return(-1.0 * loglik)
}

theta1 = c(.08, .00075, .2, 3.3) / 2
b1 = optim(theta1, logl)
#b2 = optim(b1$par,logl,hessian=T) b1 b2 are similar
theta2 = b1$par ## final parameter estimates.
sqrt(diag(solve(b2$hess))) ## SEs

#plot
## compare the fit.
u2 = seq(0, 3, length = 100)
Kg = theta1[2] * (theta1[4] - 1) * theta1[3] ^ (theta1[4] - 1)
* (u2 + theta1[3]) ^ (-theta1[4])
plot(
u2,
Kg,
col = "green",
xlab = "u",
ylab = "Kg(u)",
type = "l",
ylim = c(0, 0.8)
```

```
) #starting guess

Kg2 = theta2[2] * (theta2[4] - 1) * theta2[3] ^ (theta2[4] - 1)
* (u2 + theta2[3]) ^ (-theta2[4]) ## mu K c p.  #final estimated
lines(u2, Kg2, col = "blue")
legend(
"topright",
lty = c(1, 1),
c("starting guess", "final estimated"),
col = c("green", "blue")
)

#lambda
mu =  0.1390437
K =  0.9503494
c =  4.6699837
p =  4.4809440

lam_h = rep(mu, n)
const = K * (p - 1) * c ^ (p - 1)  # k*beta
for (j in 2:(n)) {
gij = 0
for (i in 1:(j - 1)) {
gij = gij
+ (t[j] - t[i] + c) ^ (-p)
}
lam_h[j] = mu + const * gij
}

#Stoyan-Grabarnik diagnostic
```

```
sum(1 / lam_h) / T

hist(
t / 365 + 2006,
nclass = 100,
main = "",
prob = T,
ylab = "rate␣(pts/day)",
xlim = c(2006, 2018),
ylim = c(0, 0.8),
xlab = "year",
axes = T
)
lines(t / 365 + 2006.1, lam_h * 365 / n, col = "green")

# Plot: Stochastic declustering
#theta =  0.000139 0.002045 0.001508 1.093
theta = c(0.1390437, 0.9503494, 4.6699837, 4.48094400)
mu = theta[1]
K = theta[2]
c = theta[3]
p = theta[4]

lam_h = rep(mu, n)
const = K * (p - 1) * c ^ (p - 1)  # k*beta
for (j in 2:(n)) {
gij = 0
for (i in 1:(j - 1)) {
gij = gij + (t[j] - t[i] + c) ^ (-p)
}
```

```
lam_h[j] = mu + const * gij
cat("␣", t[i] / 365 + 2006, "␣", i, "\n")
}

plot(
t / 365 + 2006,
mu / lam_h,
cex = .5,
xlab = "year",
ylab = "P(background)",
main = "Stochastic␣declustering␣"
)

# Plot: Lag plot
## superthin with b = mean(lam).
b = mean(lam_h)
thinkeep = (runif(n) < b / lam_h) & (lam_h > b)
nsup = rpois(1, b * T)
cand = sort(runif(nsup) * T) + 4.1
supkeep = rep(0, nsup)
for (i in 1:nsup) {
cat(i, "␣")
if (cand[i] < t[1])
supkeep[i] = (runif(1) < (b - mu) / b)
else{
j = max((1:n)[t < cand[i]])
a = mu + K * (p - 1) * c ^ (p - 1)
* sum((cand[i] - t[1:j] + c) ^ (-p))
supkeep[i] = (runif(1) < (b - a) / b)
cat(supkeep[i], cand[i], j, a, "\n")
```

```
}
}
supers = cand[supkeep > 0.5]
keepers = sort(c(t[thinkeep > .5], supers))
tkeep = keepers
hist(
tkeep + 2006,
main = "",
xlab = "year",
ylab = "superthinned␣points",
density = 20,
col = grey(0.2)
)
abline(h = b, lty = 2)
abline(h = qpois(.975, lambda = b), lty = 4)
abline(h = qpois(.025, lambda = b), lty = 4)
m = length(tkeep)
u = pexp(diff(c(4.1, tkeep, 48)), rate = b)
hist(u, nclass = 100)

#Lag plot
plot(u[1:(m)],
u[2:(m + 1)],
xlab = expression(U[i]),
ylab = expression(U[i + 1]),
cex = .1)

library(spatstat)
library(splancs)
library(spatial)
```

```
x1 = u[1:(m)]
y1 = u[2:(m + 1)]
n = length(x1)
plot(c(0, 1),
c(0, 1),
type = "n",
xlab = "x-coordinate",
ylab = "y-coordinate")
points(x1, y1, pch = 3)
b1 = as.points(x1, y1)
bdry = matrix(c(0, 0, 1, 0, 1, 1, 0, 1, 0, 0), ncol = 2, byrow = T)

s = seq(.001, 0.3, length = 50)
k4 = khat(b1, bdry, s)
plot(s,
k4,
xlab = "distance",
ylab = "K4(h)",
pch = "*")
lines(s, k4)
lines(s, pi * s ^ 2, lty = 2)
L4 = sqrt(k4 / pi) - s
plot(c(0, 0.3),
c(0, max(L4)),
type = "n",
xlab = "lag, h",
ylab = "L4(h) - h")
points(s, L4, pch = "*")
lines(s, L4)
lines(s, rep(0, 50), lty = 2)
```

```
### CONFIDENCE BOUNDS FOR K-FUNCTION via simulation
k4conf = Kenv.csr(npts(b1), bdry, 100, s)
plot(c(0, max(s)),
c(0, max(k4conf$upper, k4)),
type = "n",
xlab = "distance␣h",
ylab = "K4(h)")
points(s, k4, pch = "*")
lines(s, k4)
lines(s, pi * s ^ 2, col = "black")
lines(s,
k4conf$upper,
lty = 3,
col = "red",
lwd = 2)
lines(s,
k4conf$lower,
lty = 3,
col = "red",
lwd = 2)
L4upper = sqrt(k4conf$upper / pi) - s
L4lower = sqrt(k4conf$lower / pi) - s

plot(
c(0, max(s)),
c(min(L4lower, L4), max(L4upper, L4)),
type = "n",
xlab = "distance",
ylab = "L4(h)␣-␣h"
```

```
)
points(s, L4, pch = "*")
lines(s, L4)
lines(s,
L4upper,
lty = 2,
col = "red",
lwd = 2)
lines(s,
L4lower,
lty = 2,
col = "red",
lwd = 2)
lines(s, rep(0, length(s)))

### THEORETICAL BOUNDS for L-function
## bounds = 1.96 * sqrt(2*pi*A) * h / E(N), where
## A = area of space, and
## E(N) = expected # of pts in the space (approximated here using
## the observed # of pts

L4upper = 1.96 * sqrt(2 * pi * 1 * 1) * s / n
L4lower = -1.0 * L4upper
lines(s,
L4upper,
lty = 3,
col = "orange",
lwd = 2)
lines(s,
L4lower,
```

```
lty = 3,
col = "orange",
lwd = 2)

#Plot: Super-thinned residuals
#and their corresponding standardized interevent times
plot(
  c(tkeep / 365 + 2006, 2018),
  cumsum(u) / sum(u),
  type = "l",
  xlab = expression(t[i]),
  ylab = expression(u[i]),
  col = "red",
  main = "Super-thinned residuals"
  )

  ## do sims to get confidence bounds.
  r = function(x)
  quantile(x, .975)
  d = function(x)
  quantile(x, .025)
  w = matrix(0, ncol = m + 1, nrow = 1000)
  for (j in 1:1000) {
  k = runif(m + 1)
  w[j, ] = cumsum(k) / sum(k)
  }
  up = apply(w, 2, r)
  down = apply(w, 2, d)
  lines(c(tkeep / 365 + 2006, 2018), up, lty = 2, col = "green")
  lines(c(tkeep / 365 + 2006, 2018), down, lty = 2, col = "green")
```

```
  ## Plot t versus ut.
  points(c(tkeep / 365 + 2006, 2018), u, cex = .1, col = "black")


 # Hawkes model fitting- exponential function
 ##### Fit the Hawkes model.
##### lambda(t,x,y) = mu + K SUM g(t-t_i).
##### Here, g(u) = beta*e^(-beta*u),
##### For any theta, the integral of lambda
##### over the whole time region is approx.
##### mu T + nK.
##### Here the parameter vector theta = (mu, c, beta)

m3 = function(x)
signif(x, 3)

## First we will write the loglikelihood function in R.
logle = function(p) {
## returns the negative loglikelihood.
mu = p[1]
c = p[2]
beta = p[3]
cat("parms = ", mu, ", ", c, ", ", beta, ". \n")
lam = rep(mu, n)
for (i in 2:n) {
lam[i] = mu + c * beta * sum(exp(-beta * (t[i] - t[1:(i - 1)])))
}
if (min(lam) < 0.00000001)
return(99999)
if (is.na(min(lam)))
```

```
return(99999)
intgrl = mu * T + c * n
negloglam = intgrl - sum(log(lam))
cat("int/n is ", intgrl / n, "and neglog = ", negloglam, ".\n")
negloglam
}

theta0 = c(0.4, 2, 1) / 2
b_e = optim(theta0, logle, hessian = T)
theta = c(0.1818148, 0.9348890, 0.6951210, 0)
mu3 = theta[1]
c3 = theta[2]
beta3 = theta[3]
# 0.1818148 0.934889 0.695121
# integral is  12201.68 and neglog =  -5498.71 .
sqrt(diag(solve(b_e$hess))) ## SEs
#0.01724456 0.01045866 0.02335972


## Now show the fitted conditional intensity and the actual one.
lametas = rep(mu3, n)
for (i in 2:n) {
lametas[i] = mu3 + c3 * beta3 * sum(exp(-beta3 * (t[i] - t[1:(i - 1)])))
}


#Stoyan-Grabarnik diagnostic
sum(1 / lametas) / T
#0.9999556
#int/n is  0.999974 and neglog =  -5498.715
```

```
hist(
t / 365 + 2006,
nclass = 100,
main = "",
prob = T,
ylab = "rate␣(pts/day)",
xlim = c(2006, 2018),
ylim = c(0, 0.8),
xlab = "year",
axes = T
)
lines(t / 365 + 2006.1, lametas * 365 / n, col = "blue")

mean(a3$lam)# [1] 5.5487
mean(lam_h) #[1] 5.726951
mean(lametas)

# Plot: Stochastic declustering
#b_e$par
#theta =  0.000139 0.002045 0.001508 1.093
theta = c(0.1818148, 0.9348890, 0.6951210, 0)
mu = theta[1]
c = theta[2]
beta = theta[3]

plot(
t / 365 + 2006,
mu / lametas,
cex = .5,
```

```
xlab = "year",
ylab = "P(background)",
main = "Stochastic declustering "
)

# Plot: Lag plot
## superthin with b = mean(lam).
b = mean(lametas)
thinkeep = (runif(n) < b / lametas) & (lametas > b)
nsup = rpois(1, b * T)
cand = sort(runif(nsup) * T) + 4.1
supkeep = rep(0, nsup)
for (i in 1:nsup) {
cat(i, " ")
if (cand[i] < t[1])
supkeep[i] = (runif(1) < (b - mu) / b)
else{
j = max((1:n)[t < cand[i]])
a = mu + c * beta * sum(exp(-beta * (cand[i] - t[1:j])))
supkeep[i] = (runif(1) < (b - a) / b)
cat(supkeep[i], cand[i], j, a, "\n")
}
}
supers = cand[supkeep > 0.5]
keepers = sort(c(t[thinkeep > .5], supers))
tkeep = keepers
hist(
tkeep + 2006,
main = "",
xlab = "year",
```

```
ylab = "superthinned␣points",
density = 20,
col = grey(0.2)
)
abline(h = b, lty = 2)
abline(h = qpois(.975, lambda = b), lty = 4)
abline(h = qpois(.025, lambda = b), lty = 4)
m = length(tkeep)
u = pexp(diff(c(4.1, tkeep, 48)), rate = b)
hist(u, nclass = 100)

#Lag plot
plot(u[1:(m)],
u[2:(m + 1)],
xlab = expression(U[i]),
ylab = expression(U[i + 1]),
cex = .1)

library(spatstat)
library(splancs)
library(spatial)
x1 = u[1:(m)]
y1 = u[2:(m + 1)]
n = length(x1)
plot(c(0, 1),
c(0, 1),
type = "n",
xlab = "x-coordinate",
ylab = "y-coordinate")
points(x1, y1, pch = 3)
```

```
b1 = as.points(x1, y1)
bdry = matrix(c(0, 0, 1, 0, 1, 1, 0, 1, 0, 0), ncol = 2, byrow = T)

s = seq(.001, 0.3, length = 50)
k4 = khat(b1, bdry, s)
plot(s,
k4,
xlab = "distance",
ylab = "K4(h)",
pch = "*")
lines(s, k4)
lines(s, pi * s ^ 2, lty = 2)
L4 = sqrt(k4 / pi) - s
plot(c(0, 0.3),
c(0, max(L4)),
type = "n",
xlab = "lag, h",
ylab = "L4(h) - h")
points(s, L4, pch = "*")
lines(s, L4)
lines(s, rep(0, 50), lty = 2)

### CONFIDENCE BOUNDS FOR K-FUNCTION via simulation
k4conf = Kenv.csr(npts(b1), bdry, 100, s)
plot(c(0, max(s)),
c(0, max(k4conf$upper, k4)),
type = "n",
xlab = "distance h",
ylab = "K4(h)")
points(s, k4, pch = "*")
```

```
lines(s, k4)
lines(s, pi * s ^ 2, col = "black")
lines(s,
k4conf$upper,
lty = 3,
col = "red",
lwd = 2)
lines(s,
k4conf$lower,
lty = 3,
col = "red",
lwd = 2)
L4upper = sqrt(k4conf$upper / pi) - s
L4lower = sqrt(k4conf$lower / pi) - s

plot(
c(0, max(s)),
c(min(L4lower, L4), max(L4upper, L4)),
type = "n",
xlab = "distance",
ylab = "L4(h) - h"
)
points(s, L4, pch = "*")
lines(s, L4)
lines(s,
L4upper,
lty = 2,
col = "red",
lwd = 2)
lines(s,
```

```
L4lower,
lty = 2,
col = "red",
lwd = 2)
lines(s, rep(0, length(s)))

### THEORETICAL BOUNDS for L-function
## bounds = 1.96 * sqrt(2*pi*A) * h / E(N), where
## A = area of space, and
## E(N) = expected # of pts in the space (approximated here using
## the observed # of pts

L4upper = 1.96 * sqrt(2 * pi * 1 * 1) * s / n
L4lower = -1.0 * L4upper
lines(s,
L4upper,
lty = 3,
col = "orange",
lwd = 2)
lines(s,
L4lower,
lty = 3,
col = "orange",
lwd = 2)

#Plot: Super-thinned residuals
#and their corresponding standardized interevent times
plot(
  c(tkeep / 365 + 2006, 2018),
  cumsum(u) / sum(u),
```

```
  type = "l",
  xlab = expression(t[i]),
  ylab = expression(u[i]),
  col = "red",
  main = "Super-thinned␣residuals"
  )


## do sims to get confidence bounds.
r = function(x)
quantile(x, .975)
d = function(x)
quantile(x, .025)
w = matrix(0, ncol = m + 1, nrow = 1000)
for (j in 1:1000) {
k = runif(m + 1)
w[j, ] = cumsum(k) / sum(k)
}
up = apply(w, 2, r)
down = apply(w, 2, d)
lines(c(tkeep / 365 + 2006, 2018), up, lty = 2, col = "green")
lines(c(tkeep / 365 + 2006, 2018), down, lty = 2, col = "green")
## Plot t versus ut.
points(c(tkeep / 365 + 2006, 2018), u, cex = .1, col = "black")

#fit recursive- exponential function
## ESTIMATION
## We need n, T and the data t.
## We assume the size S of the spatial region =1 here.
```

```
## This needs to be done in terminal using R.
system("R CMD SHLIB measles.c")
dyn.load("measles.so")

## theta = (mu, c, beta, p).

neglogc = function(theta1) {
t1 <<- t1 + 1
if (min(c(theta1[1], theta1[2], theta1[3])) < 0.00000001)
return(9e20)
else
a3 = .C(
"neglogmea",
as.double(t),
as.integer(n),
as.double(T),
as.double(theta1),
a2 = double(1),
onelam = double(1),
inte = double(1),
as.double(eps),
lam = as.double(lam)
)
if (p1 > 1) {
cat("\n iter =", t1, " theta =", signif(theta1, 4))
cat("\n loglik is ", -1.0 * a3$a2, ". Int/n =", a3$inte / n, ".")
cat("\n sum of 1/lam/T is ", a3$onelam / T, ".\n\n")
## Both Int/n and sum of 1/lam/ST should be about 1.
}
a3$a2  ## or equivalently a[[5]]
```

```
}


t1 = 0
lam = rep(n / T, n)
p1 = 2
eps = .00000001
theta = c(.0001, .001, .002, .5)
#theta = c(.0001484, .9576, .00555, 0)
neglogc(theta)
b2 = optim(theta, neglogc, hessian = T)

theta = b2$par
# iter =  426  theta =  0.1138 1.115 0.646 0.1038
#loglik is  5525.479 . Int/n =  1.002272 .
#sum of 1/lam/T is  1.000336 .

sqrt(diag(solve(b2$hess))) ## for SEs
#0.01627903 0.02959700 0.02307343 0.01544262
```
##plot
```{
r
}
theta = c(0.1137992, 1.1145431, 0.6459955, 0.1058030)
a3 = .C(
"neglogmea",
as.double(t),
as.integer(n),
as.double(T),
as.double(theta),
```

```
a2 = double(1),
onelam = double(1),
inte = double(1),
as.double(eps),
lam = as.double(lam)
)
hist(
t / 365 + 2006,
nclass = 100,
main = "",
prob = T,
ylab = "rate␣(pts/day)",
xlim = c(2006, 2018),
ylim = c(0, 0.8),
xlab = "year",
axes = T
)
lines(t / 365 + 2006.1, a3$lam * 365 / n, col = "red")

# Plot: Stochastic declustering
#theta =  0.000139 0.002045 0.001508 1.093
theta = c(0.1137992, 1.1145431, 0.6459955, 0.1058030)
mu = theta[1]
K = theta[2]
beta = theta[3]
alpha = theta[4]
lam_r = rep(mu, n)
ks = rep(K / mu ^ alpha, n)
for (i in 2:n) {
lam_r[i] = mu + sum(ks[1:(i - 1)] * beta * exp(-beta * (t[i] - t[1:(i -
```

```
1)])))
ks[i] = K / (lam_r[i] ^ alpha)
cat("␣", t[i] / 365 + 2006, "␣", i, "\n")
}

plot(
t / 365 + 2006,
mu / lam_r,
cex = .5,
xlab = "year",
ylab = "P(background)",
main = "Stochastic␣declustering␣"
)

p <- (mu / lam_r) > 0.5
summary(p)
# 82 greater than 0.5

# Plot: Lag plot
## superthin with b = mean(lam).
b = mean(lam_r)
thinkeep = (runif(n) < b / lam_r) & (lam_r > b)
nsup = rpois(1, b * T)
cand = sort(runif(nsup) * T) + 4.1
supkeep = rep(0, nsup)
for (i in 1:nsup) {
cat(i, "␣")
if (cand[i] < t[1])
supkeep[i] = (runif(1) < (b - mu) / b)
else{
```

```
j = max((1:n)[t < cand[i]])
a = mu + sum(ks[1:j] * beta * exp(-beta * (cand[i] - t[1:j])))
supkeep[i] = (runif(1) < (b - a) / b)
cat(supkeep[i], cand[i], j, a, "\n")
}
}
supers = cand[supkeep > 0.5]
keepers = sort(c(t[thinkeep > .5], supers))
tkeep = keepers
hist(
tkeep / 365 + 2006,
main = "",
xlab = "year",
ylab = "superthinned␣points",
density = 20,
col = grey(0.2)
)
abline(h = b, lty = 2)
abline(h = qpois(.975, lambda = b), lty = 4)
abline(h = qpois(.025, lambda = b), lty = 4)
m = length(tkeep)
u = pexp(diff(c(4.1, tkeep, 48)), rate = b)
hist(u, nclass = 100)

#Lag plot
plot(u[1:(m)],
u[2:(m + 1)],
xlab = expression(U[i]),
ylab = expression(U[i + 1]),
cex = .1)
```

```
#library(sm)
library(spatstat)
library(splancs)
library(spatial)
x1 = u[1:(m)]
y1 = u[2:(m + 1)]
n = length(x1)
plot(c(0, 1),
c(0, 1),
type = "n",
xlab = "x-coordinate",
ylab = "y-coordinate")
points(x1, y1, pch = 3)
b1 = as.points(x1, y1)
bdry = matrix(c(0, 0, 1, 0, 1, 1, 0, 1, 0, 0), ncol = 2, byrow = T)

s = seq(.001, 0.3, length = 50)
k4 = khat(b1, bdry, s)
plot(s,
k4,
xlab = "distance",
ylab = "K4(h)",
pch = "*")
lines(s, k4)
lines(s, pi * s ^ 2, lty = 2)
L4 = sqrt(k4 / pi) - s
plot(c(0, 0.3),
c(0, max(L4)),
type = "n",
```

```
xlab = "lag, h",
ylab = "L4(h) - h")
points(s, L4, pch = "*")
lines(s, L4)
lines(s, rep(0, 50), lty = 2)

### CONFIDENCE BOUNDS FOR K-FUNCTION via simulation
k4conf = Kenv.csr(npts(b1), bdry, 100, s)
plot(c(0, max(s)),
c(0, max(k4conf$upper, k4)),
type = "n",
xlab = "distance h",
ylab = "K4(h)")
points(s, k4, pch = "*")
lines(s, k4)
lines(s, pi * s ^ 2, col = "black")
lines(s,
k4conf$upper,
lty = 3,
col = "red",
lwd = 2)
lines(s,
k4conf$lower,
lty = 3,
col = "red",
lwd = 2)
L4upper = sqrt(k4conf$upper / pi) - s
L4lower = sqrt(k4conf$lower / pi) - s

plot(
```

```
c(0, max(s)),
c(min(L4lower, L4), max(L4upper, L4)),
type = "n",
xlab = "distance",
ylab = "L4(h) - h"
)
points(s, L4, pch = "*")
lines(s, L4)
lines(s,
L4upper,
lty = 2,
col = "red",
lwd = 2)
lines(s,
L4lower,
lty = 2,
col = "red",
lwd = 2)
lines(s, rep(0, length(s)))

### THEORETICAL BOUNDS for L-function
## bounds = 1.96 * sqrt(2*pi*A) * h / E(N), where
## A = area of space, and
## E(N) = expected # of pts in the space (approximated here using
## the observed # of pts
L4upper = 1.96 * sqrt(2 * pi * 1 * 1) * s / n
L4lower = -1.0 * L4upper
lines(s,
L4upper,
lty = 3,
```

```
col = "orange",
lwd = 2)
lines(s,
L4lower,
lty = 3,
col = "orange",
lwd = 2)

# Plot: Super-thinned residuals
#and their corresponding standardized interevent times
plot(
  c(tkeep / 365 + 2006, 2018),
  cumsum(u) / sum(u),
  type = "l",
  xlab = expression(t[i]),
  ylab = expression(u[i]),
  col = "red",
  main = "Super-thinned residuals"
  )

## do sims to get confidence bounds.
r = function(x)
  quantile(x, .975)
d = function(x)
  quantile(x, .025)
w = matrix(0, ncol = m + 1, nrow = 1000)
for (j in 1:1000) {
  k = runif(m + 1)
  w[j, ] = cumsum(k) / sum(k)
}
```

```
up = apply(w, 2, r)
down = apply(w, 2, d)
lines(c(tkeep / 365 + 2006, 2018), up, lty = 2, col = "green")
lines(c(tkeep / 365 + 2006, 2018), down, lty = 2, col = "green")
## Plot t versus ut.
points(c(tkeep / 365 + 2006, 2018), u, cex = .1, col = "black")

# Forecasting
#-cumulative total # of events
#a month at a time: sum actual lambda
t <- read.csv("~/desktop/converted_data.csv")
t <- sort(t[, 2])
n = length(t)
T = 4368
t11 <- sort(t[1:10835])
t22 <- sort(t[10836:12202])
n1 = length(t11)
n2 = length(t22)
T1 <- 3285
T2 <- T - T1
event <- c()
for (i in 1:T) {
# i=1
event[i] = sum(t <= i)
}

#lambda
#paremeters from training dataset
theta = c(0.05074475, 1.28445653, 0.65005831, 0.17365545)
mu = theta[1]
```

```
K = theta[2]
beta = theta[3]
alpha = theta[4]

lam_r = rep(mu, n)
ks = rep(K / mu ^ alpha, n)
for (i in 2:n) {
lam_r[i] = mu + sum(ks[1:(i - 1)] * beta * exp(-beta * (t[i] - t[1:(i -
1)])))
ks[i] = K / (lam_r[i] ^ alpha)
}

#using all the data up to the begining of the week
tp = T1 + 1
if (t[1] < tp)
IndexForDayt = max((j = 1:n)[t[j] < tp])
if (t[1] < tp)
LamforDayt = mu + sum(ks[1:IndexForDayt] * beta
 * exp(-beta * (tp - t[1:IndexForDayt])))

#lambda for a year
lamn <- c()
for (i in 1:365) {
tp = T1 + i
if (t[1] < tp)
IndexForDayt = max((j = 1:n)[t[j] < tp])
if (t[1] < tp)
LamforDayt = mu + sum(ks[1:IndexForDayt] * beta
 * exp(-beta * (tp - t[1:IndexForDayt])))#lambda at time t
lamn[i] <- LamforDayt
```

```
}

# average lambda for a month
tp_n <- T1 + 1 + 365
avelam <- mean(lamn)

# 95% confidence
upp <- avelam + 1.96 * sqrt(avelam)
low <- avelam - 1.96 * sqrt(avelam)

tp = T1

plot((1:T) / 365 + 2006,
event,
type = "l",
xlab = "Year",
ylab = "Cumulative␣total␣#␣of␣events")
abline(v = (tp) / 365 + 2006,
col = "blue",
lty = 2)
points((tp) / 365 + 2006,
event[tp] + LamforDayt,
pch = 19,
cex = 0.5)
#2015
segments((tp) / 365 + 2006,
event[tp] + LamforDayt,
(tp_n) / 365 + 2006,
event[tp] + avelam,
col = "red")
```

```
#confidence interval
segments((tp) / 365 + 2006,
event[tp] + LamforDayt,
(tp_n) / 365 + 2006,
event[tp] + upp,
col = "blue")
#par(new=T)

##80/20 evaluation

# recursive
t <- read.csv("~/desktop/converted_data.csv")
t <- sort(t[, 2])
t11 <- sort(t[1:10835])
t22 <- sort(t[10836:12202])
n1 = length(t11)
n2 = length(t22)
T1 <- 3285
T2 <- 4368 - 3285
hist(
t / 365 + 2006,
nclass = 100,
main = "",
xlim = c(2006, 2018),
prob = F,
xlab = "year",
ylab = "events/year"
)
hist(
t11 / 365 + 2006,
```

```
nclass = 100,
main = "",
xlim = c(2006, 2015),
prob = F,
xlab = "year",
ylab = "events/year"
)
hist(
t22 / 365 + 2006,
nclass = 100,
main = "",
xlim = c(2015, 2018),
prob = F,
xlab = "year",
ylab = "events/year"
)

t1 = 0
lam = rep(n1 / T1, n1)
p1 = 2
eps = .00000001
theta = c(.0001, .001, .002, .5)
#theta = c(.0001484, .9576, .00555, 0)
neglogc(theta)
b2 = optim(theta, neglogc, hessian = T)

theta = b2$par
# iter =  316  theta =  0.05074475 1.28445653 0.65005831 0.17365545
# loglik is  6245.884 . Int/n =  1.002498 .
# sum of 1/lam/T is  1.017088 .
```

```
sqrt(diag(solve(b2$hess)))
theta = c(0.05074475, 1.28445653, 0.65005831, 0.17365545)
t1 = 0
lam = rep(n2 / T2, n2)
p1 = 2
eps = .00000001
a3 = .C(
"neglogmea",
as.double(t22),
as.integer(n2),
as.double(T2),
as.double(theta),
a2 = double(1),
onelam = double(1),
inte = double(1),
as.double(eps),
lam = as.double(lam)
)
hist(
t22 / 365 + 2006,
nclass = 100,
main = "",
prob = T,
ylab = "rate␣(pts/day)",
xlim = c(2015, 2018),
ylim = c(0, 3),
xlab = "year",
axes = T
)
```

```
lines(t22 / 365 + 2006.1, a3$lam * 365 / n2, col = "red")

theta = c(0.05074475, 1.28445653, 0.65005831, 0.17365545)
mu = theta[1]
K = theta[2]
beta = theta[3]
alpha = theta[4]
lam_r = rep(mu, n2)
ks = rep(K / mu ^ alpha, n2)
for (i in 2:n2) {
lam_r[i] = mu + sum(ks[1:(i - 1)] * beta
 * exp(-beta * (t22[i] - t22[1:(i -
1)])))
ks[i] = K / (lam_r[i] ^ alpha)
cat("␣", t22[i] / 365 + 2006, "␣", i, "\n")
}

plot(
t22 / 365 + 2006,
mu / lam_r,
cex = .5,
xlab = "year",
ylab = "P(background)",
main = "Stochastic␣declustering␣for␣80/20␣"
)

# Plot: Lag plot
#t22,T2,n2
## superthin with b = mean(lam).
b = mean(lam_r)
```

```
thinkeep = (runif(n2) < b / lam_r) & (lam_r > b)
nsup = rpois(1, b * T2)
cand = sort(runif(nsup) * T2) + T1
supkeep = rep(0, nsup)
for (i in 1:nsup) {
cat(i, "␣")
if (cand[i] < t22[1])
supkeep[i] = (runif(1) < (b - mu) / b)
else{
j = max((1:n2)[t22 < cand[i]])
a = mu + sum(ks[1:j] * beta * exp(-beta * (cand[i] - t22[1:j])))
supkeep[i] = (runif(1) < (b - a) / b)
cat(supkeep[i], cand[i], j, a, "\n")
}
}
supers = cand[supkeep > 0.5]
keepers = sort(c(t22[thinkeep > .5], supers))
tkeep = keepers
hist(
tkeep / 365 + 2006,
main = "",
xlab = "year",
ylab = "superthinned␣points",
density = 20,
col = grey(0.2)
)

abline(h = b, lty = 2)
abline(h = qpois(.975, lambda = b), lty = 4)
abline(h = qpois(.025, lambda = b), lty = 4)
```

```
m = length(tkeep)
u = pexp(diff(c(T1, tkeep, 4368)), rate = b)
hist(u, nclass = 100)

#Lag plot
plot(u[1:(m)],
u[2:(m + 1)],
xlab = expression(U[i]),
ylab = expression(U[i + 1]),
cex = .1)

library(spatstat)
library(splancs)
library(spatial)
x1 = u[1:(m)]
y1 = u[2:(m + 1)]
n = length(x1)
plot(c(0, 1),
c(0, 1),
type = "n",
xlab = "x-coordinate",
ylab = "y-coordinate")
points(x1, y1, pch = 3)
b1 = as.points(x1, y1)
bdry = matrix(c(0, 0, 1, 0, 1, 1, 0, 1, 0, 0), ncol = 2, byrow = T)

s = seq(.001, 0.3, length = 50)
k4 = khat(b1, bdry, s)
plot(s,
k4,
```

```
xlab = "distance",
ylab = "K4(h)",
pch = "*")
lines(s, k4)
lines(s, pi * s ^ 2, lty = 2)
L4 = sqrt(k4 / pi) - s
plot(c(0, 0.3),
c(0, max(L4)),
type = "n",
xlab = "lag, h",
ylab = "L4(h) - h")
points(s, L4, pch = "*")
lines(s, L4)
lines(s, rep(0, 50), lty = 2)

### CONFIDENCE BOUNDS FOR K-FUNCTION via simulation
k4conf = Kenv.csr(npts(b1), bdry, 100, s)
plot(c(0, max(s)),
c(0, max(k4conf$upper, k4)),
type = "n",
xlab = "distance h",
ylab = "K4(h)")
points(s, k4, pch = "*")
lines(s, k4)
lines(s, pi * s ^ 2, col = "black")
lines(s,
k4conf$upper,
lty = 3,
col = "red",
lwd = 2)
```

```
lines(s,
k4conf$lower,
lty = 3,
col = "red",
lwd = 2)
L4upper = sqrt(k4conf$upper / pi) - s
L4lower = sqrt(k4conf$lower / pi) - s

plot(
c(0, max(s)),
c(min(L4lower, L4), max(L4upper, L4)),
type = "n",
xlab = "distance",
ylab = "L4(h) - h"
)
points(s, L4, pch = "*")
lines(s, L4)
lines(s,
L4upper,
lty = 2,
col = "red",
lwd = 2)
lines(s,
L4lower,
lty = 2,
col = "red",
lwd = 2)
lines(s, rep(0, length(s)))

### THEORETICAL BOUNDS for L-function
```

```
## bounds = 1.96 * sqrt(2*pi*A) * h / E(N), where
## A = area of space, and
## E(N) = expected # of pts in the space (approximated here using
## the observed # of pts
L4upper = 1.96 * sqrt(2 * pi * 1 * 1) * s / n
L4lower = -1.0 * L4upper
lines(s,
L4upper,
lty = 3,
col = "orange",
lwd = 2)
lines(s,
L4lower,
lty = 3,
col = "orange",
lwd = 2)

#Plot: Super-thinned residuals
#and their corresponding standardized interevent times
plot(
c(tkeep / 365 + 2006, 2018),
cumsum(u) / sum(u),
type = "l",
xlab = expression(t[i]),
ylab = expression(u[i]),
col = "red",
main = "Super-thinned␣residuals"
)

## do sims to get confidence bounds.
```

```
r = function(x)
quantile(x, .975)
d = function(x)
quantile(x, .025)
w = matrix(0, ncol = m + 1, nrow = 1000)
for (j in 1:1000) {
k = runif(m + 1)
w[j, ] = cumsum(k) / sum(k)
}
up = apply(w, 2, r)
down = apply(w, 2, d)
lines(c(tkeep / 365 + 2006, 2018), up, lty = 2, col = "green")
lines(c(tkeep / 365 + 2006, 2018), down, lty = 2, col = "green")
## Plot t versus ut.
points(c(tkeep / 365 + 2006, 2018), u, cex = .1, col = "black")

# Hawkes power law
theta1 = c(.08, .00075, .2, 3.3) / 2
b1 = optim(theta1, logl, hessian = T)
#b2 = optim(b1$par,logl,hessian=T) b1 b2 are similar
theta2 = b1$par ## final parameter estimates.
sqrt(diag(solve(b1$hess))) ## SEs

theta = c(0.06106282, 0.98135918, 2.76599042, 3.21616419)
mu = theta[1]
K = theta[2]
c = theta[3]
p = theta[4]

#using t22
```

```
lam_h = rep(mu, n2)
const = K * (p - 1) * c ^ (p - 1)  # k*beta
for (j in 2:(n2)) {
gij = 0
for (i in 1:(j - 1)) {
gij = gij + (t22[j] - t22[i] + c) ^ (-p)
}
lam_h[j] = mu + const * gij
}

hist(
t22 / 365 + 2006,
nclass = 100,
main = "",
prob = T,
ylab = "rate (pts/day)",
xlim = c(2015, 2018),
ylim = c(0, 3),
xlab = "year",
axes = T
)
lines(t22 / 365 + 2006.1, lam_h * 365 / n2, col = "green")

plot(
t22 / 365 + 2006,
mu / lam_h,
cex = .5,
xlab = "year",
ylab = "P(background)",
main = "Stochastic declustering for 80/20 "
```

```
)


#t22
## superthin with b = mean(lam).
b = mean(lam_h)
thinkeep = (runif(n2) < b / lam_h) & (lam_h > b)
nsup = rpois(1, b * T2)
cand = sort(runif(nsup) * T2) + T1
supkeep = rep(0, nsup)
for (i in 1:nsup) {
cat(i, "␣")
if (cand[i] < t22[1])
supkeep[i] = (runif(1) < (b - mu) / b)
else{
j = max((1:n2)[t22 < cand[i]])
a = mu + K * (p - 1) * c ^ (p - 1) * sum((cand[i] - t22[1:j] + c) ^ (-p))
supkeep[i] = (runif(1) < (b - a) / b)
cat(supkeep[i], cand[i], j, a, "\n")
}
}
supers = cand[supkeep > 0.5]
keepers = sort(c(t22[thinkeep > .5], supers))
tkeep = keepers
hist(
tkeep / 365 + 2006,
main = "",
xlab = "year",
ylab = "superthinned␣points",
density = 20,
col = grey(0.2)
```

```
)
abline(h = b, lty = 2)
abline(h = qpois(.975, lambda = b), lty = 4)
abline(h = qpois(.025, lambda = b), lty = 4)
m = length(tkeep)
u = pexp(diff(c(T1, tkeep, 4368)), rate = b)
hist(u, nclass = 100)

#Lag plot
plot(u[1:(m)],
u[2:(m + 1)],
xlab = expression(U[i]),
ylab = expression(U[i + 1]),
cex = .1)

library(spatstat)
library(splancs)
library(spatial)
x1 = u[1:(m)]
y1 = u[2:(m + 1)]
n = length(x1)
plot(c(0, 1),
c(0, 1),
type = "n",
xlab = "x-coordinate",
ylab = "y-coordinate")
points(x1, y1, pch = 3)
b1 = as.points(x1, y1)
bdry = matrix(c(0, 0, 1, 0, 1, 1, 0, 1, 0, 0), ncol = 2, byrow = T)
```

```
s = seq(.001, 0.3, length = 50)
k4 = khat(b1, bdry, s)
plot(s,
k4,
xlab = "distance",
ylab = "K4(h)",
pch = "*")
lines(s, k4)
lines(s, pi * s ^ 2, lty = 2)
L4 = sqrt(k4 / pi) - s
plot(c(0, 0.3),
c(0, max(L4)),
type = "n",
xlab = "lag, h",
ylab = "L4(h) - h")
points(s, L4, pch = "*")
lines(s, L4)
lines(s, rep(0, 50), lty = 2)

### CONFIDENCE BOUNDS FOR K-FUNCTION via simulation
k4conf = Kenv.csr(npts(b1), bdry, 100, s)
plot(c(0, max(s)),
c(0, max(k4conf$upper, k4)),
type = "n",
xlab = "distance h",
ylab = "K4(h)")
points(s, k4, pch = "*")
lines(s, k4)
lines(s, pi * s ^ 2, col = "black")
lines(s,
```

```
k4conf$upper,
lty = 3,
col = "red",
lwd = 2)
lines(s,
k4conf$lower,
lty = 3,
col = "red",
lwd = 2)
L4upper = sqrt(k4conf$upper / pi) - s
L4lower = sqrt(k4conf$lower / pi) - s

plot(
c(0, max(s)),
c(min(L4lower, L4), max(L4upper, L4)),
type = "n",
xlab = "distance",
ylab = "L4(h) - h"
)
points(s, L4, pch = "*")
lines(s, L4)
lines(s,
L4upper,
lty = 2,
col = "red",
lwd = 2)
lines(s,
L4lower,
lty = 2,
col = "red",
```

```
lwd = 2)
lines(s, rep(0, length(s)))

### THEORETICAL BOUNDS for L-function
## bounds = 1.96 * sqrt(2*pi*A) * h / E(N), where
## A = area of space, and
## E(N) = expected # of pts in the space (approximated here using
## the observed # of pts
L4upper = 1.96 * sqrt(2 * pi * 1 * 1) * s / n
L4lower = -1.0 * L4upper
lines(s,
L4upper,
lty = 3,
col = "orange",
lwd = 2)
lines(s,
L4lower,
lty = 3,
col = "orange",
lwd = 2)

plot(
  c(tkeep / 365 + 2006, 2018),
  cumsum(u) / sum(u),
  type = "l",
  xlab = expression(t[i]),
  ylab = expression(u[i]),
  col = "red",
  main = "Super-thinned␣residuals"
  )
```

```
## do sims to get confidence bounds.
r = function(x)
  quantile(x, .975)
d = function(x)
  quantile(x, .025)
w = matrix(0, ncol = m + 1, nrow = 1000)
for (j in 1:1000) {
  k = runif(m + 1)
  w[j, ] = cumsum(k) / sum(k)
}
up = apply(w, 2, r)
down = apply(w, 2, d)
lines(c(tkeep / 365 + 2006, 2018), up, lty = 2, col = "green")
lines(c(tkeep / 365 + 2006, 2018), down, lty = 2, col = "green")
## Plot t versus ut.
points(c(tkeep / 365 + 2006, 2018), u, cex = .1, col = "black")

# Hawkes exponential
theta0 = c(0.4, 2, 1) / 2
b_e = optim(theta0, logle, hessian = T)
theta00 = b_e$par
sqrt(diag(solve(b_e$hess)))

theta = c(0.1571098, 0.9522980, 0.7246614)
mu = theta[1]
c = theta[2]
beta = theta[3]
lametas = rep(mu, n2)
for (i in 2:n2) {
```

```
lametas[i] = mu + c * beta * sum(exp(-beta * (t22[i] - t22[1:(i - 1)])))
}
hist(
t22 / 365 + 2006,
nclass = 100,
main = "",
prob = T,
ylab = "rate␣(pts/day)",
xlim = c(2015, 2018),
ylim = c(0, 3),
xlab = "year",
axes = T
)
lines(t22 / 365 + 2006.1, lametas * 365 / n2, col = "blue")

plot(
t22 / 365 + 2006,
mu / lametas,
cex = .5,
xlab = "year",
ylab = "P(background)",
main = "Stochastic␣declustering␣"
)

#t22
## superthin with b = mean(lam).
b = mean(lametas)
thinkeep = (runif(n2) < b / lametas) & (lametas > b)
nsup = rpois(1, b * T2)
cand = sort(runif(nsup) * T2) + T1
```

```
supkeep = rep(0, nsup)
for (i in 1:nsup) {
cat(i, "␣")
if (cand[i] < t22[1])
supkeep[i] = (runif(1) < (b - mu) / b)
else{
j = max((1:n2)[t22 < cand[i]])
a = mu + c * beta * sum(exp(-beta * (cand[i] - t22[1:j])))
supkeep[i] = (runif(1) < (b - a) / b)
cat(supkeep[i], cand[i], j, a, "\n")
}
}
supers = cand[supkeep > 0.5]
keepers = sort(c(t22[thinkeep > .5], supers))
tkeep = keepers
hist(
tkeep / 365 + 2006,
main = "",
xlab = "year",
ylab = "superthinned␣points",
density = 20,
col = grey(0.2)
)
abline(h = b, lty = 2)
abline(h = qpois(.975, lambda = b), lty = 4)
abline(h = qpois(.025, lambda = b), lty = 4)
m = length(tkeep)
u = pexp(diff(c(T1, tkeep, 4368)), rate = b)
hist(u, nclass = 100)
```

```
#Lag plot
plot(u[1:(m)],
u[2:(m + 1)],
xlab = expression(U[i]),
ylab = expression(U[i + 1]),
cex = .1)

library(spatstat)
library(splancs)
library(spatial)
x1 = u[1:(m)]
y1 = u[2:(m + 1)]
n = length(x1)
plot(c(0, 1),
c(0, 1),
type = "n",
xlab = "x-coordinate",
ylab = "y-coordinate")
points(x1, y1, pch = 3)
b1 = as.points(x1, y1)
bdry = matrix(c(0, 0, 1, 0, 1, 1, 0, 1, 0, 0), ncol = 2, byrow = T)

s = seq(.001, 0.3, length = 50)
k4 = khat(b1, bdry, s)
plot(s,
k4,
xlab = "distance",
ylab = "K4(h)",
pch = "*")
lines(s, k4)
```

```
lines(s, pi * s ^ 2, lty = 2)
L4 = sqrt(k4 / pi) - s
plot(c(0, 0.3),
c(0, max(L4)),
type = "n",
xlab = "lag, h",
ylab = "L4(h) - h")
points(s, L4, pch = "*")
lines(s, L4)
lines(s, rep(0, 50), lty = 2)

### CONFIDENCE BOUNDS FOR K-FUNCTION via simulation
k4conf = Kenv.csr(npts(b1), bdry, 100, s)
plot(c(0, max(s)),
c(0, max(k4conf$upper, k4)),
type = "n",
xlab = "distance h",
ylab = "K4(h)")
points(s, k4, pch = "*")
lines(s, k4)
lines(s, pi * s ^ 2, col = "black")
lines(s,
k4conf$upper,
lty = 3,
col = "red",
lwd = 2)
lines(s,
k4conf$lower,
lty = 3,
col = "red",
```

```
lwd = 2)
L4upper = sqrt(k4conf$upper / pi) - s
L4lower = sqrt(k4conf$lower / pi) - s

plot(
c(0, max(s)),
c(min(L4lower, L4), max(L4upper, L4)),
type = "n",
xlab = "distance",
ylab = "L4(h) - h"
)
points(s, L4, pch = "*")
lines(s, L4)
lines(s,
L4upper,
lty = 2,
col = "red",
lwd = 2)
lines(s,
L4lower,
lty = 2,
col = "red",
lwd = 2)
lines(s, rep(0, length(s)))

### THEORETICAL BOUNDS for L-function
## bounds = 1.96 * sqrt(2*pi*A) * h / E(N), where
## A = area of space, and
## E(N) = expected # of pts in the space (approximated here using
## the observed # of pts
```

```
L4upper = 1.96 * sqrt(2 * pi * 1 * 1) * s / n
L4lower = -1.0 * L4upper
lines(s,
L4upper,
lty = 3,
col = "orange",
lwd = 2)
lines(s,
L4lower,
lty = 3,
col = "orange",
lwd = 2)

plot(
  c(tkeep / 365 + 2006, 2018),
  cumsum(u) / sum(u),
  type = "l",
  xlab = expression(t[i]),
  ylab = expression(u[i]),
  col = "red",
  main = "Super-thinned␣residuals"
  )

## do sims to get confidence bounds.
r = function(x)
  quantile(x, .975)
d = function(x)
  quantile(x, .025)
w = matrix(0, ncol = m + 1, nrow = 1000)
for (j in 1:1000) {
```

```
  k = runif(m + 1)
  w[j, ] = cumsum(k) / sum(k)
}
up = apply(w, 2, r)
down = apply(w, 2, d)
lines(c(tkeep / 365 + 2006, 2018), up, lty = 2, col = "green")
lines(c(tkeep / 365 + 2006, 2018), down, lty = 2, col = "green")
## Plot t versus ut.
points(c(tkeep / 365 + 2006, 2018), u, cex = .1, col = "black")


# Forecasting
#whole dataset
t <- read.csv("~/desktop/converted_data.csv")
t <- sort(t[, 2])
n = length(t)
T = 4368
#training dataset
t11 <- sort(t[1:10834])
#testing dataset
t22 <- sort(t[10835:12202])
n1 = length(t11)
n2 = length(t22)
T1 <- 3283
T2 <- T - T1

#cumulative total # of events
event <- c()
for (i in 1:T) {
# i=1
```

```
event[i] = sum(t <= i)
}

#lambda
#paremeters from training dataset
theta = c(0.05074475, 1.28445653, 0.65005831, 0.17365545)
mu = theta[1]
K = theta[2]
beta = theta[3]
alpha = theta[4]

lam_r = rep(mu, n)
ks = rep(K / mu ^ alpha, n)
for (i in 2:n) {
lam_r[i] = mu + sum(ks[1:(i - 1)] * beta * exp(-beta * (t[i] - t[1:(i -
1)])))
ks[i] = K / (lam_r[i] ^ alpha)
}

mult_seg <-  data.frame(
  x0 = double(),
  y0 = double(),
  x1 = double(),
  y1 = double()
  )

  #drawing segments
  #starting point :  x-coordinate
  #x0
  for (i in 1:155) {
```

```
mult_seg[i, 1] = (T1 + 1 + 7 * (i - 1)) / 365 + 2006
}

#end point: x-coordinate
#x1
for (i in 1:155) {
mult_seg[i, 3] = (T1 + 1 + 7 * i) / 365 + 2006
}

# starting point: y-coordinate
# end point : y-coordinate
for (i in 1:155) {
lamn <- c()
for (k in 1:7) {
#lambda for a year
tpp = T1 + 7 * (i - 1) + k # T1+i = day t
if (t[1] < tpp)
IndexForDayt = max((j = 1:n)[t[j] < tpp])
if (t[1] < tpp)
LamforDayt = mu + sum(ks[1:IndexForDayt] * beta
 * exp(-beta * (tpp - t[1:IndexForDayt])))
lamn[k] <- LamforDayt
}
avelam <- mean(lamn)
mult_seg[i, 2] <-
event[T1 + (i - 1) * 7] + LamforDayt
mult_seg[i, 4] <- mult_seg[i, 2] + avelam * 7
}

#upper and lower bound
```

```
mult_low <-  mult_seg
mult_upp <-  mult_seg
for (i in 1:155) {
lamn <- c()
for (k in 1:7) {
#lambda for a year
tpp = T1 + 7 * (i - 1) + k # T1+i = day t
if (t[1] < tpp)
IndexForDayt = max((j = 1:n)[t[j] < tpp])
if (t[1] < tpp)
LamforDayt = mu + sum(ks[1:IndexForDayt] * beta
 * exp(-beta * (tpp - t[1:IndexForDayt])))
lamn[k] <- LamforDayt
}
avelam <- mean(lamn)
upp <- avelam * 7 + 1.96 * sqrt(avelam * 7)
low <- avelam * 7 - 1.96 * sqrt(avelam * 7)
mult_low[i, 4] <- mult_seg[i, 2] + low
mult_upp[i, 4] <- mult_seg[i, 2] + upp
}

#plot
plot((1:T) / 365 + 2006,
event,
type = "l",
xlab = "Year",
ylab = "Cumulative␣total␣#␣of␣events")
abline(v = (T1 + 1) / 365 + 2006,
col = "blue",
lty = 5)
```

```
#plot
plot((1:T) / 365 + 2006,
event,
type = "l",
xlab = "Year",
ylab = "Cumulative total # of events",
main = "Predicting Period"
)
segments(
x0 = mult_seg$x0,
y0 = mult_seg$y0,
x1 = mult_seg$x1,
y1 = mult_seg$y1,
col = "red"
)
segments(
x0 = mult_low$x0,
y0 = mult_low$y0,
x1 = mult_low$x1,
y1 = mult_low$y1,
col = "blue"
)
segments(
x0 = mult_upp$x0,
y0 = mult_upp$y0,
x1 = mult_upp$x1,
y1 = mult_upp$y1,
col = "blue"
)
```

```
#plot
plot((T1:(T1 + 365 * 3)) / 365 + 2006,
event[T1:(T1 + 365 * 3)],
type = "l",
xlab = "Year",
ylab = "Cumulative total # of events",
main = "Predicting Period",
ylim = c(10834, 12202)
)
segments(
x0 = mult_seg$x0,
y0 = mult_seg$y0,
x1 = mult_seg$x1,
y1 = mult_seg$y1,
col = "red"
)
segments(
x0 = mult_low$x0,
y0 = mult_low$y0,
x1 = mult_low$x1,
y1 = mult_low$y1,
col = "blue"
)
segments(
x0 = mult_upp$x0,
y0 = mult_upp$y0,
x1 = mult_upp$x1,
y1 = mult_upp$y1,
col = "blue"
)
```

```
#plot
plot((T1:(T1 + 365)) / 365 + 2006,
event[T1:(T1 + 365)],
type = "l",
xlab = "Year",
ylab = "Cumulative total # of events",
main = "2015 to 2016",
ylim = c(10834, 11500)
)
segments(
x0 = mult_seg$x0,
y0 = mult_seg$y0,
x1 = mult_seg$x1,
y1 = mult_seg$y1,
col = "red"
)
segments(
x0 = mult_low$x0,
y0 = mult_low$y0,
x1 = mult_low$x1,
y1 = mult_low$y1,
col = "blue"
)
segments(
x0 = mult_upp$x0,
y0 = mult_upp$y0,
x1 = mult_upp$x1,
y1 = mult_upp$y1,
```

```
col = "blue"
)

#plot
plot(((T1 + 365):(T1 + 365 * 2)) / 365 + 2006,
event[(T1 + 365):(T1 + 365 * 2)],
type = "l",
xlab = "Year",
ylab = "Cumulative total # of events",
main = "2016 to 2017",
ylim = c(11447, 11650)
)
segments(
x0 = mult_seg$x0,
y0 = mult_seg$y0,
x1 = mult_seg$x1,
y1 = mult_seg$y1,
col = "red"
)
segments(
x0 = mult_low$x0,
y0 = mult_low$y0,
x1 = mult_low$x1,
y1 = mult_low$y1,
col = "blue"
)
segments(
x0 = mult_upp$x0,
y0 = mult_upp$y0,
x1 = mult_upp$x1,
```

```
y1 = mult_upp$y1,
col = "blue"
)

#plot
plot((T1:(T1 + 365)) / 365 + 2006,
event[T1:(T1 + 365)],
type = "l",
xlab = "Year",
ylab = "Cumulative total # of events",
main = "2015 to 2016",
ylim = c(10834, 11500)
)
segments(
x0 = mult_seg$x0,
y0 = mult_seg$y0,
x1 = mult_seg$x1,
y1 = mult_seg$y1,
col = "red"
)
segments(
x0 = mult_low$x0,
y0 = mult_low$y0,
x1 = mult_low$x1,
y1 = mult_low$y1,
col = "blue"
)
segments(
x0 = mult_upp$x0,
y0 = mult_upp$y0,
```

```
x1 = mult_upp$x1,
y1 = mult_upp$y1,
col = "blue"
)

observed <- c()
for (i in 1:155) {
observed[i] = event[T1 + (i) * 7]
}

(sum((mult_seg$y1 - observed) ^ 2) / 155) ^ (1 / 2)



plot((T1:(T1 + 365)) / 365 + 2006,
   event[T1:(T1 + 365)],
   type = "l",
   xlab = "Year",
   ylab = "Cumulative total # of events",
   main = "2015 to 2016",
   ylim = c(10834, 11500)
   )
   segments(
   x0 = mult_seg$x0,
   y0 = mult_seg$y0,
   x1 = mult_seg$x1,
   y1 = mult_seg$y1,
   col = "red"
   )
```

```
plot(((T1 + 365):(T1 + 365 * 2)) / 365 + 2006,
event[(T1 + 365):(T1 + 365 * 2)],
type = "l",
xlab = "Year",
ylab = "Cumulative total # of events",
main = "2016 to 2017",
ylim = c(11447, 11650)
)
segments(
x0 = mult_seg$x0,
y0 = mult_seg$y0,
x1 = mult_seg$x1,
y1 = mult_seg$y1,
col = "red"
)
plot(((T1 + 365 * 2):(T1 + 365 * 3)) / 365 + 2006,
event[(T1 + 365 * 2):(T1 + 365 * 3)],
type = "l",
xlab = "Year",
ylab = "Cumulative total # of events",
main = "2017 to 2018",
ylim = c(11650, 12210)
)
segments(
x0 = mult_seg$x0,
y0 = mult_seg$y0,
x1 = mult_seg$x1,
y1 = mult_seg$y1,
col = "red"
)
```

References

[1] Epidemiologic profile of coccidiodomycosis in san luis obispo county, ca. Technical report, San Luis Obispo County Public Health Department, May 2014.

[2] Jennifer Brown, Kaitlin Benedict, Benjamin J Park, and George R Thompson 3rd. Coccidioidomycosis: epidemiology. *Clin Epidemiol*, June 2013.

[3] Wikipedia contributors. Coccidioidomycosis. `https://en.wikipedia.org/wiki/Coccidioidomycosis`, May 2019.

[4] HAROLD PIERCE. Forecasting an epidemic: How weather contributes to valley fever outbreaks, November 2016.

[5] ALAN G. HAWKES. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 51(1):83, 1971.

[6] Frederic Schoenberg, Marc Hoffmann, and Ryan Harrigan. A recursive point process model for infectious diseases. *arXiv e-prints*, page arXiv:1703.08202, March 2017.

[7] Wilbert Van Panhuis, Anne Cross, and Donald S. Burke. Counts of coccidioidomycosis reported in united states of america. page DOI: 10.25337/T7, April 2018.

[8] David R. Brillinger, Peter M. Guttorp, and Frederic Paik Schoenberg. Point processes, temporal. *Encyclopedia of Environmetrics*, January 2013.

[9] Frederic Paik Schoenberg. Introduction to point processes. 2000.

[10] Junhyung Park, Adam W. Chaffee, Ryan J. Harrigan, and Frederic Paik Schoenberg. A non-parametric hawkes model of the spread of ebola in west africa. December 2018.

[11] Dietrich Stoyan and Pavel Grabarnik. Secondorder characteristics for stochastic structures connected with gibbs point processes. *Mathematische Nachrichten*, 151:95–100, 1991.

[12] D.S. Harte. Log-likelihood of earthquake models: Evaluation ofmodels and forecasts. *Geophysical Journal International*, 201:711–723, March 2015.

[13] Robert Alan Clements, Frederic Paik Schoenberg, and Alejandro Veen. Evaluation of space-time point process models using super-thinning. *Environmetrics*, 23:606–616, September 2012.

[14] Frederic Paik Schoenberg. Multi-dimensional residuals analysis of point process models for earthquake occurrences. *Journal of the American Statistical Association*, 98(464):789–795, December 2003.

[15] Pierre Brémaud. *Point Processes and Queues: Martingale Dynamics*. Springer Series in Statistics. Springer, 1981 edition, September 1981.

[16] Junhyung Park, Adam W. Chaffee, Ryan J. Harrigan, and Frederic Paik Schoenberg. A non-parametric hawkes model of the spread of ebola in west africa. December 2018.

[17] Joshua Seth Gordon, Robert Alan Clements, Frederic Paik Schoenberg, and Danijel Schorlemmer. Voronoi residuals and other residual analyses applied to csep earthquake forecasts. *Spatial Statistics*, 14:133–150, November 2015.

[18] Hirotugu Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19, December 1974.

[19] Wikipedia contributors. Akaike information criterion. `https://en.wikipedia.org/wiki/Akaike_information_criterion`.

[20] Brian D. Ripley. The second-order analysis of stationary point processes. *Journal of Applied Probability*, 13(2):255–266, June 1976.

[21] Philip M. Dixon. Ripley's k function. *Encyclopedia of Environmetrics*, 3:1796–1803, 2002.

[22] Susan Holmes. Rms error. `http://statweb.stanford.edu/~susan/courses/s60/split/node60.html`, 2000.

[23] Rob J Hyndman and Anne B Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting.*, November 2005.

[24] Hans Nesse. Global health - seir model. `http://www.public.asu.edu/~hnesse/classes/seir.html`.