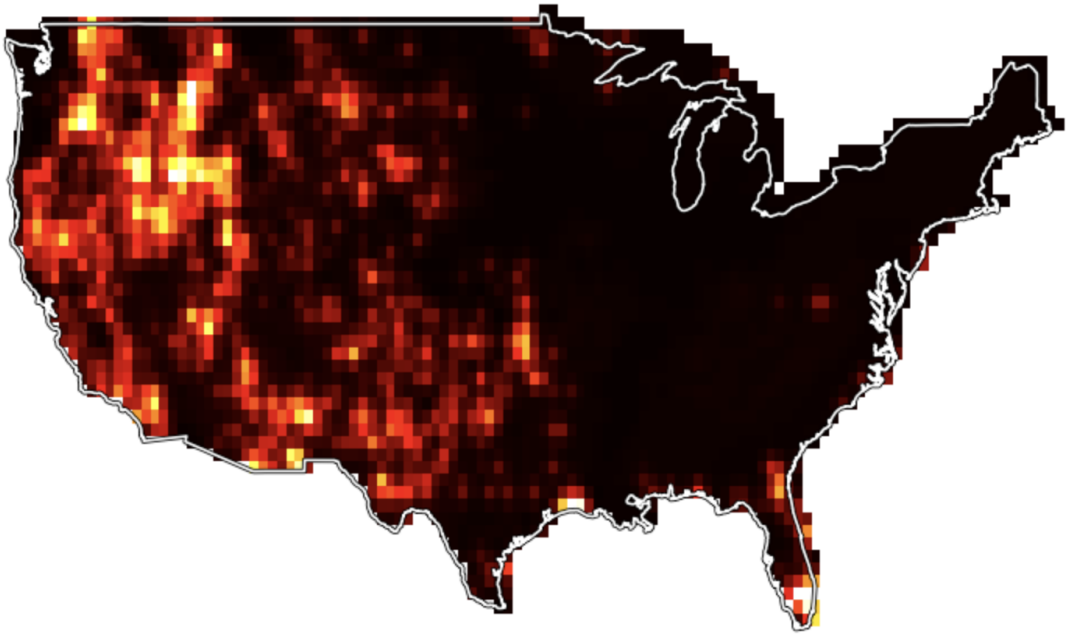
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Spatial Extreme Value Analysis using Point Processes

A Spatio-temporal Approach to Modeling Extreme Wildfire Activity

Master's thesis in Data Science and AI

ANNE ENGSTRÖM

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

Master's thesis 2021

Spatial Extreme Value Analysis using Point Processes

A Spatio-temporal Approach to Modeling Extreme Wildfire Activity

Anne Engström

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2021

Spatial Extreme Value Analysis using Point Processes
A Spatio-temporal Approach to Modeling Extreme Wildfire Activity
ANNE ENGSTRÖM

© ANNE ENGSTRÖM, 2021.

Supervisor: Mike Pereira, Department of Mathematical Sciences
Examiner: Annika Lang, Department of Mathematical Sciences

Master's Thesis 2021
Department of Mathematical Sciences
Division of Applied Mathematics and Statistics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The estimated risk of extensive wildfire activity in different locations of the continental United States in July 2015.

Typeset in LaTeX
Printed by Department of Mathematical Sciences, Sweden 2021

Spatial Extreme Value Analysis using Point Processes
A Spatio-temporal Approach to Modeling Extreme Wildfire Activity
ANNE ENGSTRÖM
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Extreme value theory is a field in statistics dealing with the occurrence of rare events, also known as extreme events. It is commonly found in financial applications, but also when modeling rare environmental events. This Master's thesis proposes a new methodology for modeling spatio-temporal extreme values, by generalising a method for extremes of a time series to both space and time. The method is applied to data of extreme wildfire activity in the United States, with the goal to predict the risk of wildfires exceeding some large threshold. The model created for predicting extreme wildfires in the spatio-temporal dimension is created from a self-exciting model, with previous events triggering new events. In the spatio-temporal dimension, the self-excitation is dependant on both the time and the distance to previous events, making point processes suitable for modeling these.

The resulting model manages to capture both the influence of previous wildfires, as well as the impact of a regions vegetation when assessing the risk of large wildfires in the area. While the model cannot predict the exact locations of all wildfires at a selected time point, it accurately shows the regions with an elevated risk of extensive wildfire activity. It also captures the trend of increasing wildfire activity in the studied area.

Keywords: spatio-temporal extreme value analysis, point processes, peaks over threshold, risk

Acknowledgements

I want to say a big thank you to my supervisor Mike Pereira for guiding me through this project, always making sure that it was headed in the right direction. Most importantly he always kept the spirits high, making the process so much more enjoyable.

Anne Engström, Gothenburg, May 2021

Contents

List of Figures **xiii**

List of Tables **xv**

1 Introduction **1**
1.1 Background . . . 1
1.2 Purpose . . . 2
1.3 Limitations . . . 2
1.4 Structure of the report . . . 2

2 Theory **3**
2.1 Point Processes . . . 3
2.1.1 Intensity of the one-dimensional Poisson point process . . . 4
2.1.2 Multidimensional point processes . . . 6
2.1.3 The spatio-temporal point process . . . 7
2.1.4 The marked point process . . . 8
2.2 Extreme value analysis . . . 9
2.2.1 Block maxima . . . 9
2.2.2 Peaks over threshold . . . 10
2.2.3 Choosing a method . . . 11
2.2.4 Defining extreme events with threshold selection . . . 11

3 Modeling spatio-temporal extremes using point processes **15**
3.1 The model . . . 15
3.1.1 The temporal model . . . 15
3.1.2 The spatio-temporal model . . . 19
3.1.3 Parameter estimation . . . 24
3.2 Measuring the risk of wildfires . . . 26

4 Application of the wildfire dataset **29**
4.1 Description of dataset . . . 29
4.1.1 Training, validation and test datasets . . . 30
4.1.2 Dealing with missing data . . . 30
4.2 Choosing the threshold . . . 30
4.3 Results . . . 32
4.3.1 Validation of model . . . 32
4.3.2 Intensity of the model . . . 35

4.3.3 Risk assessment 38

5 Discussion and conclusion **41**
5.1 Interpretation of results 41
5.2 Limiting the information used in the model 42
5.3 Future work 43
5.4 Conclusion 44

Bibliography **45**

A Appendix 1 **47**
A.1 Derivation of the GPD from the GEV distribution 47

B Appendix 2 **49**
B.1 Proof of Proposition 3.1.1 49
B.2 Proof of Proposition 3.1.2 49
B.3 Proof Proposition 3.1.3 50

C Appendix 3 **51**
C.1 Dataset 51

D Appendix 4 **55**
D.1 Code 55

List of Figures

2.1 The MRL plot of the daily rainfall data. A suitable threshold for the data is $u_0 = 30$, with a positive linear trend for $u \in [30, 60]$. 13

2.2 The parameter stability plots for the scale parameter $\hat{\sigma}^*$ with modified scale, and the shape parameter $\hat{\xi}$. The 95% confidence interval of the parameters are shown for each value of u. A suitable threshold is chosen at $u_0 = 30$, since the parameters are approximately constant up until this value. 14

3.1 The negative log returns of Z exceeding the threshold u. 17

4.1 The proportion of burnt area in each location recorded between March 1993 and September 2015. 31

4.2 The MRL plots of the wildfire dataset. The left figure is evaluated for $u \in [0, 0.2]$ and the right for $u \in [0, 0.05]$. 31

4.3 The cumulative number of predicted events plotted against the actual number of events taking place between 1993 and 2015 in blue. The orange line indicates perfect model fit. 33

4.4 The exponential probability plot of the residuals of the validation set, for the GPD$(\hat{\xi}, \hat{\sigma})$ model with $\xi < 0$. 34

4.5 The exponential probability plot of the residuals of the validation set, for the GPD$(\hat{\xi}, \hat{\sigma})$ model with $\xi \geq 0$. 35

4.6 The ACF of the residuals (left) and of the squared residuals (right), with the 95% confidence interval in gray and $\xi \geq 0$. 35

4.7 The background intensity. 36

4.8 The auxiliary variables contribution to the background intensity in different areas of the United States, with cropland (top left), forest (top right) and low vegetation (bottom). 37

4.9 The estimated trigger density at time $t = 50$. 37

4.10 The estimated total intensity at time $t = 50$. 38

4.11 The probability of the occurrence of wildfires exceeding the threshold $u = 0.01$ at the times $t = 10$ (top left), $t = 50$ (top right), $t = 99$ (lower left) and at $t = 151$ (lower right). These times are from different years, but all in July. 39

4.12 The wildfires that occurred at the times $t = 10$ (top left), $t = 100$ (top right), $t = 150$ (lower left) and at $t = 150$ (lower right), indicated in orange. These times are from different years, but all in July. 40

4.13 The VaR y_q^t in each location, for $t = 150$ and $q = 0.9999$. This indicates that the probability of the proportion of burnt area not exceeding y_q^t in each location is 99.99%. 40

List of Tables

4.1 The estimated parameters $\hat{\theta}_A = [\hat{\alpha}_0, \hat{\alpha}_1, \hat{\alpha}_2, \hat{\alpha}_3, \hat{\gamma}, \hat{\psi}, \hat{\beta}, \hat{\lambda}_1, \hat{\lambda}_2, \hat{\eta}]$. 32
4.2 The parameters $\hat{\theta}_B = [\hat{\xi}, \hat{\sigma}]$, estimated using finite support. 32
4.3 The parameters $\hat{\theta}_B = [\hat{\xi}, \hat{\sigma}]$, estimated with the sigmoid transformation for $\xi \geq 0$. 34

1
Introduction

With the climate changing more rapidly with each passing year the occurrence of rare environmental events are increasing. Examples of such events are extreme temperatures, heavy rainfall, droughts and massive wildfires. Not only are these events increasing in frequency, but the size of these events are also changing. In addition, extreme events such as wildfires further contribute to the increased amount of greenhouse gas emissions each year, allowing this problem to spiral further. This calls for ways to model these events to be able to recognize how they can impact the future and what risks they might bring, but also in order to prevent them.

Extreme value theory, the statistical theory of rare events, is widely used in financial applications but can also be used to describe the spatial events mentioned above. Such events generally follow distributions with heavier tails, since these capture the nature of extreme events better than for example the normal distribution. Extreme events have a tendency to occur in clusters, meaning that if one event has taken place it is more likely for another one to occur. This means that the events are dependent and self-exciting. Point processes capture this dependency between events nicely, while also being randomly located in the domain just like extreme events are. This makes point processes suitable for modeling extreme events.

1.1 Background

The EVA Conference is an annual conference for researchers in extreme value theory, this year hosted by The School of Mathematics at The University of Edinburgh. Each year the conference organises a data challenge in the area of extreme value analysis and the 2021 challenge is set in the field of spatio-temporal regression modeling of extremes, treating the prediction of extreme wildfire activity in the spatio-temporal dimension. This project deals with modeling the extreme wildfire activity in the United States, based on this challenge set by the EVA Conference.

As mentioned above, extreme values often occur in clusters and extreme wildfire activity is no exception. Therefore a self-exciting model is implemented, where previous events trigger the occurrence of new events. By introducing a spatial component to the triggering effect, a model is created for predicting extreme wildfire activity in the United States. From this model it is possible to assess the risk of large wildfires occurring in different locations and predict the size these fires are likely to reach. With this information sufficient preventive measures can be set in place to avoid future extreme wildfire activity.

1.2 Purpose

The goal of this project was to propose a new methodology for modeling spatial extremes, by generalising a method for extremes of a time series to space and time. This was done by connecting extreme value theory, in particular peaks over threshold analysis, with spatial statistics, to define a new model for spatio-temporal data. This was done with the purpose to predict extreme events taking place in the spatio-temporal dimension, where in particular wildfire prediction was of interest.

1.3 Limitations

The aim of this project was not to perform an exhaustive search of possible methods to model spatio-temporal extremes, but to explore and apply a point process approach to spatio-temporal data. Therefore, other methods were not explored. The approach followed in this project is mainly based on the work by Chavez-Demoulin et al. [1], which propose a method for modeling extreme returns in financial time series with a marked point process model for exceedances over a high threshold. Inspiration for generalising the model to the spatio-temporal dimension was gathered from multiple sources, but the work by Chavez-Demoulin et al. set the foundation and main direction of the project.

1.4 Structure of the report

This report starts off with a chapter treating the underlying theory needed for modeling extremes of a spatio-temporal time series using marked point processes. The theory of point processes is introduced, where extra care is given to spatio-temporal point processes. The theory of extreme value analysis follows, where the peaks over threshold approach is introduced. In this chapter, the notation used throughout the report is also established.

The spatio-temporal model for wildfire prediction is found in chapter 3. To get a full understanding of the spatio-temporal model, its components and how they are derived, a description of a similar temporal model is first given. This is followed by a thorough description of the spatio-temporal model. At the end of the chapter a risk measure is introduced in order to assess the risk of wildfires occurring in different locations.

In chapter 4 the EVA 2021 wildfire dataset is applied to the spatio-temporal model created in chapter 3. The chapter starts of with a description of the wildfire dataset, as well as defining extreme wildfire activity. The results of this application are then stated by presenting the predicted wildfire intensities and risk measures. These results are then discussed in chapter 5, where some indication of possible directions of future work is given. The report ends with a presentation of the conclusions drawn from carrying out this study.

2
Theory

2.1 Point Processes

When modeling events occurring at random locations in a given domain, *point processes* are useful. They are found in a wide variety of applications, for example at insurance companies when modeling insurance claims [2] or when modeling the number of hurricanes in some region [3]. Other common applications are in the modeling of earthquakes or epidemics [4]. The domain of the events can hence be of different dimensions, but simplest is the one-dimensional time domain $\mathbb{R}^+$. Here, the process can be thought of as marking down points on the continuous time line whenever some event occurs. These events have in common that the times at which they occur are random. The events could be almost anything, for instance the random times at which wildfires occur in a forest.

Formally, point processes on $\mathbb{R}^+$ are defined through the counts of events falling in any (borelian) subset of $\mathbb{R}^+$, as follows.

Definition 2.1.1. Let $(\Omega, \mathcal{A}, \mathbb{P})$ be a probability space. We denote by $\mathfrak{B}(\mathbb{R}^+)$ the set of Borel sets of $\mathbb{R}^+$.

A point process X on $\mathbb{R}^+$ is a locally finite random counting measure on $\mathfrak{B}(\mathbb{R}^+)$, i.e. X is a mapping $X : \Omega \times \mathfrak{B}(\mathbb{R}^+) \to \mathbb{Z}^+ \cup +\infty$ such that

- for each $\omega \in \Omega$, $X(\omega, \cdot)$ is a measure on $\mathfrak{B}(\mathbb{R}^+)$,
- for each $A \in \mathfrak{B}(\mathbb{R}^+)$, $X(\cdot, A)$ is a random variable taking values in $\mathbb{Z}^+ \cup +\infty$: in particular, if A is bounded then $X(\cdot, A)$ is finite, and called number of events in A.

Another characterization of one-dimensional point processes is given by considering directly the (random) times at which events occur: the so-called *arrival times*. The arrival time of the i^{th} point is denoted by the random variable T_i and defined such that the i^{th} point always arrives before the $(i+1)^{th}$ point: $T_i < T_{i+1}$. One-dimensional point processes hence have a natural ordering in time, which means that the arrival times of a point process are dependent on each other. To avoid the complications arising from this dependence, it can be easier to look at the *inter-arrival times* between points. These are defined as $S_i = T_{i+1} - T_i$ and for some processes, like the Poisson process, the random variables $S_1, S_2, \ldots$ are independent [5].

A one-dimensional point process can also be described by its *counting process* N, defined as the process that associates to each $t \geq 0$ the random number N_t given by

$$N_t = \sum_{i=1}^{\infty} \mathbf{1}\{T_i \leq t\},$$

where $\mathbf{1}\{T_i \leq t\}$ is the random variable equal to 1 if $T_i \leq t$, 0 otherwise. Hence, N_t describes the number of points arriving up to time t. Then, for $0 \leq a \leq b$, the random variable $N(a, b]$ defined by

$$N(a, b] = N_b - N_a,$$

describes the number of points arriving in the time interval between a and b. To circle back to Definition 2.1.1, note that for a point process X its associated counting process N is no other than the random count $X(\cdot, (a, b])$ of the interval $(a, b]$.

2.1.1 Intensity of the one-dimensional Poisson point process

The *intensity* of a one-dimensional point process measures the frequency at which points fall into a given interval on the time line. Formally, for a point process X on $\mathbb{R}^+$, it is defined as the density λ, with respect to the Lebesgue measure, of the measure Λ given by

$$\Lambda(A) = \mathbb{E}[X(\cdot, A)] = \int_A \lambda(t)\mathrm{d}t, \quad A \in \mathfrak{B}(\mathbb{R}^+),$$

assuming of course that this density exists. In particular, X is called homogeneous if its intensity is equal to a constant $\lambda > 0$, meaning that for any $0 \leq a < b$,

$$\mathbb{E}[N(a, b]] = \lambda \cdot (b - a).$$

Otherwise, the process is called inhomogeneous. From the definition above, it is clear that the intensity over some time interval $(a, b]$ is the expected number of points falling on the time line during this interval. The one-dimensional Poisson process is then defined as follows.

Definition 2.1.2. The *one-dimensional Poisson process*, with intensity $\lambda(t) > 0$, is a point process in $\mathbb{R}$ such that

1. for every bounded interval $(a, b]$, the count $N(a, b]$ has a Poisson distribution with mean $\int_b^a \lambda(t)\mathrm{d}t$.

2. if $(a_1, b_1], \ldots, (a_m, b_m]$ are disjoint bounded intervals, then the counts $N(a_1, b_1], \ldots, N(a_m, b_m]$ are independent random variables.

The conditional intensity $\lambda(t|H_t)$ of a point process in one dimension is defined as (cf. [6]):

$$\lambda(t|H_t) = \lim_{h \to 0} \frac{\mathbb{E}[N(t, t+h]|H_t]}{h},$$

where H_t is the *history* of the process defined as follows.

Definition 2.1.3. The *history* of a process up until some time $t > 0$, is in one dimension defined as (the sigma-algebra generated by) the set of arrival times of the events that occurred before (but not including) time t and is denoted by H_t.

From this definition it can be seen that the conditional intensity function is the expected number of points dropped on the time line, given the history H_t up until that point.

The conditional intensity function can be homogeneous or inhomogeneous. If a process is homogeneous and hence neither dependent on the time t nor on its history H_t, the conditional intensity function is constant $\lambda(t|H_t) = \lambda$. If the process instead is inhomogeneous there are two possible scenarios for the intensity function. It can either be dependent only on the time t or be dependent on both t and H_t. This results in intensity functions $\lambda(t|H_t) = \lambda(t)$ and $\lambda(t|H_t)$, respectively. There are a variety of ways to model the conditional intensity function. The constant intensity function λ needs no further discussion, but the ones dependent on time and history can be of different types.

Linear and log-linear models

When the intensity is dependent on the time t but not on the history H_t, the simplest way to express this is with a linear model,

$$\lambda(t) = \alpha + \beta t, \quad t \geq 0, \tag{2.1}$$

where α and β take values such that $\lambda(t)$ is non-negative. Here, β describes a trend in intensity over time [6]. By definition of the intensity function, the expected number of points falling in a domain cannot be negative. To avoid this, the model in (2.1) is turned into the log-linear model

$$\log \lambda(t) = \alpha + \beta t, \quad t \geq 0.$$

The log-linear model is useful for describing the risk of some events occurring [6].

Self-correcting models

Another type of model is the *self-correcting* model. With this type of model the risk of some event builds up over time before it finally occurs. At this point the system experiences a kind of stress-release, where the risk of it occurring again is instantaneously decreased. The name of these models comes from the logarithm of the conditional intensity being adjusted each time an event occurs. These models are of the form

$$\log \lambda(t|H_t) = \alpha + \beta t - \nu N(0, t], \quad t \geq 0, \tag{2.2}$$

where $\nu \geq 0$ [6]. As can be seen in (2.2), the probability of an event increases linearly with time t, but each time an event occurs it decreases by a factor $\exp(-\nu)$.

Self-exciting models

The models used later in this thesis have the property that the occurrence of one event elevates the chance of another event occurring. This causes clustering of events. Stemming from this behaviour, this type of model is called *self-exciting*. As the time from one event grows larger, the probability of another event occurring decreases with these models, but surges again as soon as another event occurs. In the

time domain, this can be described as the counting processes of two non-overlapping adjacent intervals $(a, b]$ and $(b, c]$ having positive covariance, $\mathrm{Cov}\Big(N(a, b], N(b, c]\Big) > 0$ [7]. For each event occurring, the intensity of the process increases.

Since previous events play a role in determining when the next event is to happen, the self-exciting models clearly depend on the history H_t, defined as in Definition 2.1.3. The self-exciting models are called Hawkes processes when the conditional intensity function is defined as

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

where $\mu \geq 0$ is a constant, $t_1, t_2, \ldots$ are the actual arrival times of the process and g is a non-increasing non-negative function. The parameter μ can be considered to be the background rate of events and, for any $t > 0$, each event occurring at time $t_i < t$ adds a positive contribution $g(t - t_i)$ to the overall intensity at time t. It is worth noting that in these models, recent events typically increase the probability of another event occurring more than events that happened in the distant past. It is hence common to use a function such as

$$g(s) = \sum_{k=1}^{K} \phi_k \cdot s^{k-1} \cdot e^{-\alpha s}, \quad s > 0$$

for a defined number of steps $K \geq 1$, and some weights $\phi_1, \ldots, \phi_K$. Since g is a non-increasing function, the increased probability of an event occurring caused by a previous event decays as the previous event become part of the distant history. One could say that g triggers the events, which is why it is sometimes referred to as the *trigger density* [6].

Another option for the trigger density g is the model

$$g(s) = \frac{\kappa}{(s + \phi)^\theta}, \quad s > 0, \tag{2.3}$$

where κ, θ and ϕ are parameters. With the trigger density on this form, the Hawkes process is called an *Epidemic-Type Aftershock Sequence* model, in short ETAS model [6]. This type of model is often used to describe earthquakes and their aftershock frequency. Indeed, one extreme event, like an earthquake, often causes after-events which in turn cause other after-events and so on. The events branch out from the original extreme event and is called a *branching process* [1]. The probability of an event being triggered by an extreme event is, however, largest close in time to the extreme event and then decays with time.

2.1.2 Multidimensional point processes

When increasing the dimension of the domain to $d \geq 2$, multidimensional point processes are used to model the occurrence of these random events. Reusing the example of the wildfires in a forest from above, it is then possible to create a two-dimensional map of the locations of the fires during some time frame. This map will consist of a random number of points, scattered in space at random.

A multidimensional point process is a point process on $S \subseteq \mathbb{R}^d$. When discussing these, it is hence the number of points falling in a specific region $S \subseteq \mathbb{R}^d$ that is of interest, where d no longer is equal to one. Similarly as in Definition 2.1.1, a point process on $S \subseteq \mathbb{R}^d$ is formally defined as a random counting measure on the Borel sets of S. The associated counting process N is then defined as the process that associates to each bounded closed set $A \subset S$ the random variable defined as

$$N(A) = \text{number of points falling in } A.$$

It is generally assumed that the point process is simple, meaning that for all $x \in S$

$$N\{x\} \leq 1 \tag{2.4}$$

with probability 1. This means that no two points coincide.

To specify the intensity of the multidimensional point process, an *intensity measure* is used. For a point process X on S, it is defined as

$$\Lambda(A) = \mathbb{E}[N(A)] = \int_A \lambda(a)\mathrm{d}a,$$

where $A \subset S$ is a compact set and λ is a non-negative and locally integrable function called the *intensity function*. For further reference, see work by Baddeley [5]. From the definition above, it is clear that the intensity measure over some region A is the expected number of points falling in the region. With the intensity measure Λ, the multidimensional Poisson point process is defined according to the definition below [5].

Definition 2.1.4. Let S be a space and Λ a measure on S. (S is required to be a locally compact metric space and Λ a measure which is finite on every compact set and which has no atoms.) The multidimensional Poisson process on S with intensity measure Λ is a point process on S such that

1. for every compact set $A \subset S$, the count $N(A)$ has a Poisson distribution with mean $\Lambda(A)$.

2. if $A_1, \ldots, A_m$ are disjoint compact sets, then $N(A_1), \ldots, N(A_m)$ are independent.

2.1.3 The spatio-temporal point process

In this thesis, the *spatio-temporal point process* is central. As mentioned in the introduction, spatio-temporal point processes are useful when modeling for example earthquakes or wildfires. They capture how events occur at some location in a multidimensional space during some time interval, which can be used for preventive measures. In this thesis the region of events is described by $S \subseteq \mathbb{R}^+ \times \mathbb{R}^2$.

The conditional intensity of the spatio-temporal point process is defined by

$$\lambda(t, \boldsymbol{x}|H_t) = \lim_{|\mathrm{d}t, \mathrm{d}\boldsymbol{x}| \to 0} \frac{\mathbb{E}[N(\mathrm{d}t \times \mathrm{d}\boldsymbol{x})|H_t]}{\mathrm{d}t\mathrm{d}\boldsymbol{x}},$$

where $\boldsymbol{x} \in \mathbb{R}^2$ are the location coordinates, $\mathrm{d}t$ is an infinitesimal time interval containing t and $\mathrm{d}\boldsymbol{x}$ is an infinitesimal disk surrounding the location $\boldsymbol{x}$ [7]. The history H_t is now defined as in the following definition, including both the time and location of previous events up until time t.

Definition 2.1.5. The *history* of a process up until some time $t > 0$, is in the spatio-temporal dimension defined as (the sigma-algebra generated by) the set of arrival times and locations of the events that occurred before (but not including) time t and is denoted by H_t.

The conditional intensity for the spatio-temporal process is then the expected number of points dropped in some time-space region $S \subseteq \mathbb{R}^+ \times \mathbb{R}^2$, knowing the history H_t.

As for the one-dimensional Poisson process, there is a variety of ways to model the conditional intensity function for the spatio-temporal point process [7]. In particular, it can be modeled with the approaches mentioned in section 2.1.1, generalised to the region $A \subset \mathbb{R}^+ \times \mathbb{R}^2$. The model of interest in this thesis is the self-exciting model called ETAS from section 2.1.1, generalised to the spatio-temporal dimension. It is defined as

$$\lambda(t, \boldsymbol{x}|H_t) = \mu(\boldsymbol{x}) + \sum_{t_i<t} \omega(t - t_i, \boldsymbol{x} - \boldsymbol{x_i}), \tag{2.5}$$

where μ is the non-negative and integrable background rate, ω is the non-negative trigger density dependant on both time and location and $\boldsymbol{x}_i$ is the spatial coordinate of the event occurring at time t_i [7]. As can be seen in (2.5) the background rate is now dependent on the spatial coordinates. Following the approach by Ilhan and Kozat [8], the trigger density is factorised into a temporal function ω_T and a spatial function ω_X,

$$\lambda(t, \boldsymbol{x}|H_t) = \mu(\boldsymbol{x}) + \sum_{t_i<t} \omega_T(t - t_i)\omega_X(\boldsymbol{x} - \boldsymbol{x_i}). \tag{2.6}$$

This is often done to the trigger density ω for simplicity [9]. Here, ω_T is equivalent to the trigger density for the temporal process in (2.3). The spatial trigger density can be defined in different ways depending on the application and the properties the model should capture.

2.1.4 The marked point process

So far, point processes have only been introduced as points falling on the time line or in some region A. The points can, however, also hold additional information called *marks*. These marks can be of different nature and can for example indicate the size of the wildfires in some region.

If X is some point process in the region $S \subseteq \mathbb{R}^d$, each point $x_i \in X$ can have a mark $w_i \in M$ attached to it. The mark space M is either a finite set or $M \subset \mathbb{R}^m$. The *marked point process* is then

$$Y = \{(x_k, w_k)|x_k \in X, w_k \in M\}$$

and defined as follows [5].

Definition 2.1.6. A *marked point process* on a space S with marks in a space M is a point process Y on $S \times M$ s.t. $N(S \times M) < \infty$ a.s. for all compact $S \subseteq \mathbb{R}^d$. That is, the corresponding projected process (of points without marks) is locally finite.

The intensity measure of the marked point process is given as

$$\Lambda(A \times M) = \mathbb{E}[N(A \times M)],$$

where $A \subset S$ and M is either a finite set or $M \subset \mathbb{R}^m$ [4].

Just like Chavez-Demoulin et al. [1] use marks in the trigger density for a temporal point process, the marks can be added similarly to a spatio-temporal process. Building on the model in (2.6), the marks w_i are added to the temporal function ω_T. The conditional intensity function for the spatio-temporal model with marks is then

$$\lambda(t, \boldsymbol{x}|H_t) = \mu(\boldsymbol{x}) + \sum_{t_i<t} \omega_T(t - t_i, w_i)\omega_X(\boldsymbol{x} - \boldsymbol{x_i}),$$

where H_t now also includes the marks of previous events. The temporal and spatial functions ω_T and ω_X will be specified in more detail in sections 3.1.1 and 3.1.2.

2.2 Extreme value analysis

The main objective of *extreme value theory (EVT)* is to predict the occurrence of rare events outside the sample data. Unlike classical statistics, EVT deals with the prediction of rare events, also called extreme events. If for example wildfires are recorded, the extreme events would be recorded on days when the proportion of burnt area in a region is much larger than in the majority of data points recorded.

To model these types of events, distributions well suited to describe the tails of other distributions are used. Before finding such distributions, however, the notion of extreme events has to be defined in a suitable way. Depending on the method used, this is done in different ways.

2.2.1 Block maxima

The first method used to model extreme events is called *block maxima (BM)*. With this method the data is divided into sequential blocks of equal size, where the maxima in each block are chosen to be part of the extreme value data. These extreme values approximately follow the generalised extreme value distribution. Depending on the block size the quality of this approximation can vary, since choosing a block size is a trade-off between bias and variance. Small blocks leads to poor approximation, with the maxima in each block not necessarily being a very large (or small) value. Large blocks, on the other hand, mean that there cannot be as many blocks, which leads to large estimation variance.

If the sample process $X_1, \ldots, X_n$ consists of a sequence of n i.i.d. random variables with common unknown cumulative distribution function F, BM focuses on the behaviour of

$$M_n := \max\{X_1, \ldots, X_n\},$$

where M_n represents the maximum of a process observed over n time steps. Since all random variables are i.i.d. the probability of M_n being smaller than or equal to the value $x \in \mathbb{R}$ is then

$$P(M_n \leq x) = P(X_1 \leq x, \ldots, X_n \leq x) = F^n(x).$$

Since F is unknown, small discrepancies in the approximation of it may cause important deviations in F^n. According to Coles [10], a better approach is to instead approximate models of F^n directly and not go by F. As Pickands [11] states, if there exists sequences $\{a_n\}_{n\in\mathbb{N}} \subset \mathbb{R}^+$ and $\{b_n\}_{n\in\mathbb{N}} \subset \mathbb{R}$ and a distribution function EV such that

$$\lim_{n\to\infty} P\Big(\frac{M_n - b_n}{a_n} \leq x\Big) = \lim_{n\to\infty} F^n(a_n x + b_n) = EV(x) \quad \forall x \in \mathbb{R},$$

then the distribution function EV is the so-called *generalized extreme value distribution (GEV)*, defined by

$$EV(x) = \begin{cases} \exp\Big(-\Big(1+\xi\big(\frac{x-\mu}{\sigma}\big)\Big)^{-1/\xi}\Big), & \xi \neq 0, \;\; x \in \{x \in \mathbb{R} : 1+\xi\big(\frac{x-\mu}{\sigma}\big) > 0\} \\ \exp\Big(-\exp\Big(-\big(\frac{x-\mu}{\sigma}\big)\Big)\Big), & \xi = 0, \quad x \in \mathbb{R}, \end{cases} \tag{2.7}$$

for some parameters $\mu \in \mathbb{R}$, $\xi \in \mathbb{R}$ and $\sigma \in \mathbb{R}^+$. In particular, the choices $\xi = 0$ defines the Gumbel distribution, $\xi > 0$ the Fréchet distribution and $\xi < 0$ the Weibull distribution.

2.2.2 Peaks over threshold

Another way to define extreme events is by setting some threshold u and letting all data points exceeding u be extreme events. If for example wildfires in a region have been recorded over some period, all wildfires exceeding some size u are then considered to be extreme events. By defining the extreme events in this way the data is used more effectively, with not as many data points missed or overlooked. Using the exceedances over a threshold to model extreme events is called the *peaks over threshold (POT)* method and the extreme events approximately follow the generalised Pareto distribution [10].

If the sample $X_1, \ldots, X_n$ is a sequence of n i.i.d. random variables with cumulative distribution function $F_{(X_1,\ldots,X_n)}$, the variables exceeding some threshold u can be denoted by

$$W_j = (X_i - u | X_i > u) \quad \text{for } i = 1, \ldots, n$$

for $j \in \{1, \ldots, n_u\}$, where n_u is the number of extreme events. If u is large enough, it can be proven that the distribution of exceedances $W_1, \ldots, W_{n_u}$ can be approximated by the *generalized Pareto distribution (GPD)*,

$$G(w) = \begin{cases} 1 - \Big(1 + \frac{\xi w}{\tilde{\sigma}}\Big)^{-1/\xi}, & \xi \neq 0, \\ 1 - \exp\Big(-\frac{w}{\tilde{\sigma}}\Big), & \xi = 0, \end{cases} \tag{2.8}$$

where the support of W is defined on $w \geq 0$ for $\xi \geq 0$, and on $0 \leq w \leq -\tilde{\sigma}/\xi$ for $\xi < 0$ [10]. Here, ξ is the shape parameter and $\tilde{\sigma}$ the scale parameter defined as $\tilde{\sigma} = \sigma + \xi(u - \mu)$, with σ and μ being the scale and location parameters of the corresponding GEV distribution. This means that there is a connection between the GPD and GEV distributions. For the interested reader the derivation of the GPD from the GEV can be found in Appendix A.1.

2.2.3 Choosing a method

There are advantages and disadvantages to both BM and the POT method. For example, the BM approach can be more useful when the observations are not i.i.d. but not suitable if the data is not easily divided into natural blocks [12]. There are also situations where the only data available are block maxima, like the annual maxima of the temperature recorded over a long period of time. Then the BM approach is to be preferred. Generally, however, the BM method wastes a lot of the data available by only picking one maxima from each block. It can also include lower observations than what normally would be considered an extreme value, when some blocks do not contain high values. The POT method uses all the information of the extreme values and utilizes the information more effectively. This projects expands on the POT approach by Chavez-Demoulin et al. [1] and therefore POT will be used to define the extreme events.

2.2.4 Defining extreme events with threshold selection

To model exceedances over a high threshold, as is done with the POT method, a suitable threshold has to be selected. The balance between choosing a threshold that is not too high and not too low can be seen as a trade-off between bias and variance. If a low threshold is chosen, the exceedances over the threshold is not really a limiting distribution and this leads to bias. If instead the threshold is too high there will not be many data points exceeding the threshold, which in turn leads to high variance. As a rule of thumb, the threshold should be set such that approximately 10% of the data are deemed extreme events.

Finding a suitable threshold can be done in various ways, but two common methods are the mean residual life plot and the parameter stability plots.

The mean residual life plot

The *mean residual life (MRL) plot* is a graphical method for choosing a suitable threshold, suggested by Coles [10] for analysis of extremes of data. It is a method based on the mean of the GPD. If X follows a GPD with scale and shape parameters σ and ξ respectively, the mean is described by

$$\mathbb{E}[X] = \frac{\sigma}{1 - \xi}$$

for $\xi < 1$ [10]. For a stochastic process where the random variables exceeding the threshold u_0 are denoted by $Y_1, Y_2, \ldots$, the mean of the excesses over the threshold

is then

$$\mathbb{E}[Y - u_0 | Y > u_0] = \frac{\sigma_{u_0}}{1 - \xi}, \tag{2.9}$$

where σ_{u_0} denotes the scale parameter for the excesses over the threshold u_0. However any threshold $u > u_0$ is valid if the GPD is valid for u_0, which means that (2.9) can be written as

$$\mathbb{E}[Y - u | Y > u] = \frac{\sigma_u}{1 - \xi}.$$

It is then called the *mean residual life function*. The mean of the excess over the threshold u changes linearly with u. By calculating the sample mean of the excesses, an empirical estimate of the mean excess is given. From this, a suitable value of u can be determined. This is done by locating the points

$$\left\{ \left(u, \frac{1}{n_u} \sum_{i=1}^{n_u} (y_i - u) \right) : u < y_{\max} \right) \right\},$$

where y_i are the observations exceeding the threshold u and n_u denotes the number of such observations for the current u. The value of u ranges from a suitable lower bound, usually zero, to $y_{\max}$, which is the largest value of any random variable in the process. For the chosen range of values of u, these points make out the MRL plot. From this plot, a suitable threshold u_0 is chosen such that the threshold values following u_0 are growing approximately linearly with u.

As an example of how to perform threshold selection, the MRL plot of daily rainfall data in south-west England over the period 1914-1962 [10] can be seen in Figure 2.1. The MRL function is computed for values $u \in [0, 90]$. The smallest value for which a linear trend can be seen is $u_0 = 30$, which is a suitable threshold for this data. A negative linear trend can be seen for values over $u = 60$. This, however, does not make a good threshold, since the number of exceedances for these high thresholds are very few. With only a couple of data points exceeding the threshold, it is hard to fit a good model to the data.

Depending on the context, the mean residual life function can go under other names. In insurance contexts it is often referred to as the *mean excess loss function*, while used in financial risk management it is referred to as *expected shortfall* [13]. In reliability and medical settings, however, the name mean residual life function is well adopted.

The parameter stability plot

Another graphical method for selecting the threshold, also suggested by Coles [10], is the *parameter stability plot*. This is a good complement to the MRL plot, which can be hard to interpret at times.

If the exceedances over a threshold u_0 follow a GPD, then the exceedances over $u > u_0$ also follow a GPD. The shape parameter ξ does not depend on the threshold and should hence stay approximately constant when plotted against a range of values of u. As mentioned in section 2.2.4, the scale parameter σ_u does change with u. This can be seen from the definition of σ_u given by Coles [10],

$$\sigma_u = \sigma_{u_0} + \xi(u - u_0).$$

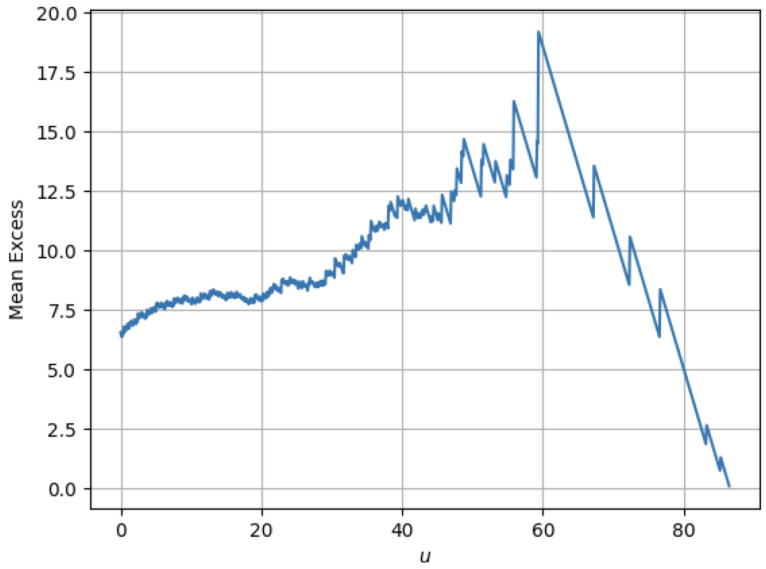


Figure 2.1: The MRL plot of the daily rainfall data. A suitable threshold for the data is $u_0 = 30$, with a positive linear trend for $u \in [30, 60]$.

By reparametrising the scale parameter to

$$\sigma^* = \sigma_u - \xi u,$$

it also becomes constant with respect to u. When plotting the estimates $\hat{\xi}$ and $\hat{\sigma}^*$ against u, however, it is often found that these are not constant with respect to u. This is due to sampling variability. Nevertheless, when taking their sampling errors into account, these estimates should be stable. This is why the 95% confidence interval is plotted together with these parameters.

Revisiting the daily rainfall data from section 2.2.4, the parameter stability plots for this data, seen in Figure 2.2, further indicate that $u_0 = 30$ is a reasonable threshold. From looking at the plots, it is possible to interpret threshold values up until approximately $u = 30$ as having somewhat constant parameters. For larger values of u, the parameter values start to deviate more from the constant value, with the variance growing larger.

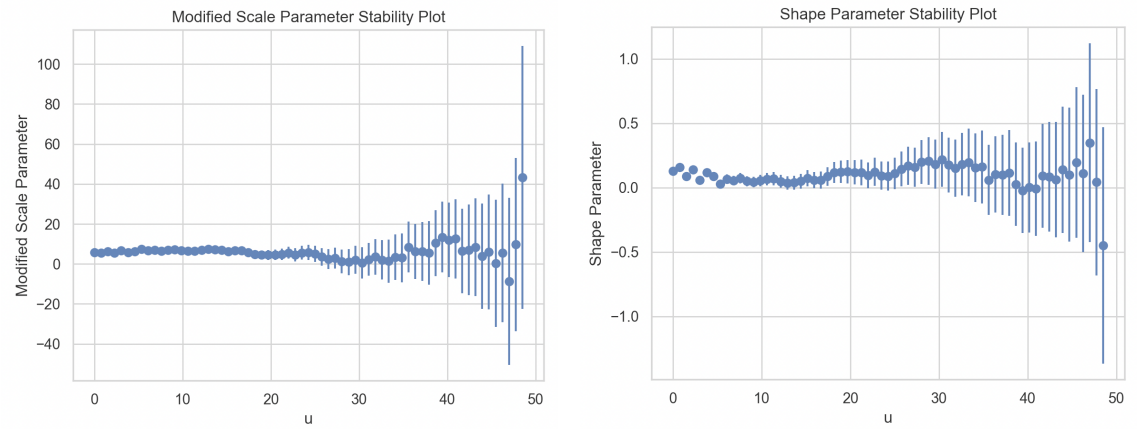


Figure 2.2: The parameter stability plots for the scale parameter $\hat{\sigma}^*$ with modified scale, and the shape parameter $\hat{\xi}$. The 95% confidence interval of the parameters are shown for each value of u. A suitable threshold is chosen at $u_0 = 30$, since the parameters are approximately constant up until this value.

3

Modeling spatio-temporal extremes using point processes

As mentioned in the introduction, wildfires can become very destructive when allowed to grow out of control. To predict the risk of large wildfires, a model for spatio-temporal extreme value data is created. The aim with the model is to predict the risk of an area having wildfires larger than some set threshold. With such predictions, preventive measures can be set in place to ward of future fires. Furthermore the model can give insight into what is waiting unless the cause of the increase in wildfire activity – climate change – is not dealt with properly.

3.1 The model

The spatio-temporal model used for predicting wildfires in the United States is inspired by the approach by Chavez-Demoulin et al. [1] for the estimation of the Value-at-Risk using point processes. To generalise their model to the spatio-temporal dimension, a thorough understanding of the temporal model is needed. Therefore, an explanation of the temporal model is first given, before moving on to the description of the spatio-temporal model created to model wildfires in the United States.

3.1.1 The temporal model

By predicting the evolution of financial time series, it is possible to assess the financial risk in for example a portfolio. This is done with a measure called *Value-at-Risk (VaR)*. This measure allows to determine an upper bound for the loss y_q, which holds with probability q. Common values for q are 0.95 and 0.99, and by recognizing that big losses are rare, they can be thought of as extreme events. Rather than following the normal distribution, these extreme events follow distributions with heavier tails.

In their work, Chavez-Demoulin et al. [1] consider the modeling of extremes of financial time series with the purpose to estimate the VaR. The model presented is a one-step model, predicting the time at which an extreme event occurs separately from the prediction of the size of the extreme event. Using the closing price of a stock Z, the negative daily log returns are

$$Y_i = \log(Z_{i-1}) - \log(Z_i) \quad i \in \mathbb{N},$$

where Z_i is the closing price at day i. The extreme events to be modeled are the occurrence and size of extreme negative daily log returns, that is, whenever large

losses occur from one day to the next. These extremes are modeled using the POT approach, which describes events over some high threshold u as extreme events, see section 2.2.2. The exceedances over the threshold are hence

$$W_j = (Y_i - u | Y_i > u), \quad i \in \mathbb{N},$$

where $j \in \{1, \ldots, N\}$ and N is the number of losses larger than the threshold u. To model these threshold exceedances a marked point process is used, where the events are the log-returns exceeding the threshold u and marks are the amount W_j of these exceedances. Each extreme event is therefore described by both its mark size W_j and time of occurrence T_j, and is denoted by $(T_j, W_j), j \in \{1, \ldots, N\}$.

When observing the log-returns over the time interval $(0, \tau]$, $n \in \mathbb{N}$ exceedances $(T_1, W_1), \ldots, (T_n, W_n)$ occur for some $\tau > 0$. The history of the process up until some time t is denoted by H_t and includes the event times T_i and marks W_i of all events occurring up until but not including time t, $T_i < t$. This gives the joint density of the observations

$$f_{T_1,W_1}(t_1, w_1) \prod_{i=2}^{n} f_{T_i,W_i|H_{t_{i-1}}}(t_i, w_i | H_{t_{i-1}}).$$

Noting that the process observed over the period $(0, \tau]$ gives n observations, it is also known that the $(n+1)^{\text{th}}$ exceedance only can be observed at some time $T_{n+1} > \tau$. With this additional information the joint density over $(0, \tau]$ is

$$f_{T_1,W_1}(t_1, w_1) \prod_{i=2}^{n} f_{T_i,W_i|H_{t_{i-1}}}(t_i, w_i | H_{t_{i-1}}) P(T_{n+1} > \tau | H_{t_n}).$$

Here, an assumption of conditional independence between times and marks is made, given the history of all previous events. The likelihood of the model can therefore be stated as

$$L(\theta) = f_{T_1}(t_1) \prod_{i=2}^{n} f_{T_i|H_{t_{i-1}}}(t_i | H_{t_{i-1}}) P(T_{n+1} > \tau | H_{t_n}) \cdot f_{W_1}(w_1) \prod_{i=2}^{n} f_{W_i|H_{t_{i-1}}}(w_i | H_{t_{i-1}}),$$

where the parameters to be estimated in θ each are in $\mathbb{R}^+$. The assumption of conditional independence is made to simplify the model, as it allows the log-likelihood ℓ to be split into two parts corresponding to the time of occurrences and the marks:

$$\begin{aligned} \ell(\theta) = &\overbrace{\log\left(f_{T_1}(t_1) \prod_{i=2}^{n} f_{T_i|H_{t_{i-1}}}(t_i | H_{t_{i-1}}) P(T_{n+1} > \tau | H_{t_n}) \right)}^{\text{A}} \\ &+ \underbrace{\log\left(f_{W_1}(w_1) \prod_{i=2}^{n} f_{W_i|H_{t_{i-1}}}(w_i | H_{t_{i-1}}) \right)}_{\text{B}}. \end{aligned} \tag{3.1}$$

For classical POT models, the homogeneous Poisson process is used to model the times of the extreme events and the GPD the size of the events [1]. This, however, is not an accurate representation of the extreme returns. As can be seen in Figure 3.1,

the times at which exceedances over the threshold u occurs are clustered. A model which captures this short-range dependence between extreme events is needed. This is done with a marked point process model, where the event times are modelled with a self-exciting structure dependent on the history of previous event times and marks. The GPD is still used for modelling the size of the marks.

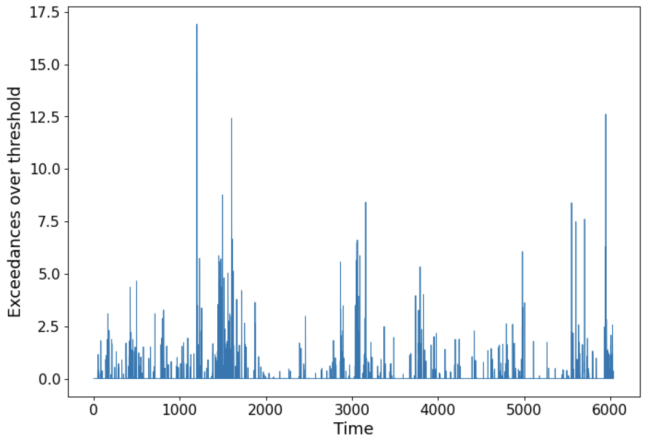


Figure 3.1: The negative log returns of Z exceeding the threshold u.

Part A: Time of events

As explained in section 2.1.1, the conditional intensity is the expected number of points dropped on the time line, given the history H_t up until that point,

$$\lambda(t|H_t) = \lim_{\mathrm{d}t \to 0} \frac{\mathbb{E}[N(t, t+\mathrm{d}t]|H_t]}{\mathrm{d}t}. \tag{3.2}$$

Here, $N(t, t+\mathrm{d}t]$ is the number of events dropped on the time line in $(t, t+\mathrm{d}t]$. From (3.2) the following proposition can be derived.

Proposition 3.1.1. *The conditional intensity function $\lambda(t|H_t)$ is defined by*

$$\lambda(t|H_t) = \frac{f_T(t|H_{t_n})}{1 - F_T(t|H_{t_n})}, \tag{3.3}$$

where t_n is the time of the last event before t.

See B.1 for the proof. The density function f_T is needed in part A of the log-likelihood and can be extracted from (3.3), giving the following proposition.

Proposition 3.1.2. *The probability density function $f_T(t|H_{t_n})$ is*

$$f_T(t|H_{t_n}) = \lambda(t|H_{t_n}) \cdot \exp\Big(- \int_{t_n}^{t} \lambda(t'|H_{t_n})\mathrm{d}t' \Big). \tag{3.4}$$

See B.2 for the proof. Using the probability density function in (3.4) and that $P(T_{n+1} > \tau|H_{t_n}) = 1 - F_T(\tau|H_{t_n}) = f_T(\tau|H_{t_n})/\lambda(\tau|H_{t_n})$, results in the following proposition.

Proposition 3.1.3. *Part A of the log-likelihood in* (3.1) *is*

$$\ell_A = -\int_0^\tau \lambda(t'|H_{t'})\mathrm{d}t' + \sum_{i=1}^{n} \log\Big(\lambda(t_i|H_{t_{i-1}})\Big). \tag{3.5}$$

See B.3 for the proof.

Since the process should be self-exciting for the times at which the events occur, the intensity function $\lambda(t_i|H_{t_{i-1}})$ is one of the self-exciting models discussed in section 2.1.1. It takes the form

$$\lambda(t|H_{t_n}) = \mu + \sum_{j:t_j<t} \omega(t - t_j; w_j),$$

where $\mu > 0$ is the constant background intensity and ω the trigger density. Here, $\omega(u) \geq 0$ for $u > 0$ and otherwise zero. The trigger density does not only depend on the times of previous events, but also on their marks w_j, which is common in the financial context [1]. It takes the form

$$\omega(t - t_j; w_j) = \frac{\psi e^{\beta w_j}}{t - t_j + \gamma}, \quad t > t_j,$$

resulting in the log-likelihood

$$\ell_A = -\int_0^\tau \Big(\mu + \sum_{j:t_j<t'} \frac{\psi e^{\beta w_j}}{t' - t_j + \gamma}\Big)\mathrm{d}t' + \sum_{i=1}^{n} \log\Big(\mu + \sum_{j:t_j<t_i} \frac{\psi e^{\beta w_j}}{t_i - t_j + \gamma}\Big).$$

The parameters to be estimated for part A of the log-likelihood is therefore $\theta_A = [\mu, \gamma, \psi, \beta]$.

Part B: Mark sizes

In the model presented by Chavez-Demoulin et al. [1], the marks are modelled with the GPD(ξ_i, σ_i). The probability density function of a random variable $W_i \sim$ GPD(ξ_i, σ_i) is

$$f_{W_i}(w_i) = \frac{1}{\sigma_i}\Big(1 + \frac{\xi_i \cdot w_i}{\sigma_i}\Big)^{\left(-\frac{1}{\xi_i}-1\right)},$$

where the shape and scale parameters ξ_i and σ_i are set to constants $\xi_i = \xi \in \mathbb{R}^+$ and $\sigma_i = \sigma \in \mathbb{R}^+$ in the model. This results in the probability density function

$$f_{W_i}(w_i) = \frac{1}{\sigma}\left(1 + \frac{\xi \cdot w_i}{\sigma}\right)^{\left(-\frac{1}{\xi}-1\right)} \tag{3.6}$$

and the log-likelihood

$$\begin{aligned}\ell_B &= \log\Big(f_{W_1}(w_1)\prod_{i=2}^{n} f_{W_i|H_{t_{i-1}}}(w_i|H_{t_{i-1}})\Big) \\ &= \sum_{i=1}^{n} \log\left(\frac{1}{\sigma}\Big(1 + \frac{\xi \cdot w_i}{\sigma}\Big)^{\left(-\frac{1}{\xi}-1\right)}\right),\end{aligned}$$

with the parameters to be estimated $\theta_B = [\xi, \sigma]$.

3.1.2 The spatio-temporal model

When generalising the model in section 3.1.1 to the spatio-temporal dimension, the attributes of the temporal model are kept. Where Y was the log-returns of a stock in the temporal implementation, it is now the proportion of burnt area in a location on the continental United States. Hence, W now marks the size of wildfires exceeding some threshold, where the size is measured in the proportion of burnt area in each region.

One major difference to the temporal implementation in section 3.1.1 is that the probability of an event being triggered at some location by a previous event is not only dependent on the time between these two events. It is now also dependent on the spatial distance to this previous event. To capture this, the trigger density ω is extended to the spatio-temporal dimension and then split into a temporal and a spatial part,

$$\omega(t - t_j, \boldsymbol{x} - \boldsymbol{x_j}; w_j) = \omega_T(t - t_j; w_j)\omega_X(\boldsymbol{x} - \boldsymbol{x_j}),$$

where ω_T and ω_X are the temporal and spatial densities respectively, with $t \in \mathbb{R}^+$ and $\boldsymbol{x} \in \mathbb{R}^2$. This approach is inspired by Diggle [14], Ilhan and Kozat [8] and allows the temporal trigger density to remain the same as that in section 3.1.1. The spatial density $\omega_X(\boldsymbol{x} - \boldsymbol{x_j})$ is a new addition, allowing wildfires at different locations to influence the occurrence of wildfires in other locations. To model this spatial dimension of the trigger density, the joint normal distribution is used and takes the form

$$\omega_X(\boldsymbol{x} - \boldsymbol{x_j}) = \frac{1}{2\pi\sqrt{|\Sigma|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x_j})^\top \Sigma^{-1}(\boldsymbol{x}-\boldsymbol{x_j})},$$

where Σ is a positive definite matrix to be estimated, and $|\Sigma|$ its determinant.

With the extended trigger density, the conditional intensity function used to model the wildfires takes the form

$$\lambda(t, \boldsymbol{x}|H_t) = \mu(\boldsymbol{x}) + \sum_{j:t_j<t} \omega_T(t - t_j; w_j)\omega_X(\boldsymbol{x} - \boldsymbol{x_j}). \tag{3.7}$$

As can be seen in (3.7), the background intensity $\mu(\boldsymbol{x})$ is now dependent on the spatial locations. It seems reasonable that the vegetation, or lack thereof, should influence a locations susceptibility of wildfires. By using a regression sum of the three auxiliary variables describing the land cover, the model takes geographical factors into account when determining the intensity. The three auxiliary variables are the proportion of land at each location covered by cropland, trees and low vegetation, denoted by q_1, q_2 and q_3. These are used in the background intensity as

$$\mu(\boldsymbol{x}) = \alpha_0 + \sum_{i=1}^{3} \alpha_i \cdot q_i(\boldsymbol{x})$$

for each location, where α_i are the weights for the auxiliary variables to be estimated and α_0 is an offset.

It should be noted that more than one event can occur at the same time point when extending the conditional intensity function to the spatio-temporal dimension, since it is possible for one event to occur at each location. The sum over the trigger densities in (3.7) should take this into account and sum over all these previous events.

The spatio-temporal likelihood function

As in the temporal case, the process observed over the period $(0, \tau]$ gives n observations. Now, however, the locations of these events are recorded as well. The events are denoted by $(T_1, X_1, W_1), \ldots, (T_n, X_n, W_n)$, where $T_i \leq T_{i+1}$ since multiple events can happen at the same time at different locations. The history of the process, still denoted by H_t, includes both the time and location of previous events, as well as the mark size.

An assumption of conditional independence between the locations is made and this, together with the knowledge that the time of the $(n+1)^{\text{th}}$ observation is $T_{n+1} > \tau$, gives that the joint density over the period $(0, \tau]$ is

$$f_{T_1,X_1,W_1}(t_1, \boldsymbol{x_1}, w_1) \prod_{i=2}^{n} f_{T_i,X_i,W_i|H_{t_{i-1}}}(t_i, \boldsymbol{x_i}, w_i|H_{t_{i-1}}) P(T_{n+1} > \tau|H_{t_n}).$$

To simplify the model, an assumption of conditional independence between the marks and the space-time locations is made. This results in the likelihood function

$$\begin{aligned} L(\theta) = f_{T_1,X_1}(t_1, \boldsymbol{x}_1) \prod_{i=2}^{n} f_{T_i,X_i|H_{t_{i-1}}}(t_i, \boldsymbol{x}_i|H_{t_{i-1}}) P(T_{n+1} > \tau|H_{t_n}) \\ \cdot f_{W_1}(w_1) \prod_{i=2}^{n} f_{W_i|H_{t_{i-1}}}(w_i|H_{t_{i-1}}). \end{aligned}$$

Taking the logarithm of this yields the log-likelihood

$$\begin{aligned} \ell = \overbrace{\log\left(f_{T_1,X_1}(t_1, \boldsymbol{x}_1) \prod_{i=2}^{n} f_{T_i,X_i|H_{t_{i-1}}}(t_i, \boldsymbol{x}_i|H_{t_{i-1}}) P(T_{n+1} > \tau|H_{t_n})\right)}^{\text{A}} \\ \underbrace{+ \log\left(f_{W_1}(w_1) \prod_{i=2}^{n} f_{W_i|H_{t_{i-1}}}(w_i|H_{t_{i-1}})\right)}_{\text{B}}. \end{aligned} \tag{3.8}$$

So far, this looks very similar to the temporal log-likelihood in section 3.1.1. The log-likelihood can still be split into two parts, where part A models the time and locations of events and part B the mark sizes.

Part A: Time and location of events

The conditional intensity function at time t is defined as the expected number of points dropped in some region in a small time frame around t, given the history H_t up until that point,

$$\lambda(t, \boldsymbol{x}|H_t) = \lim_{\mathrm{d}t, \mathrm{d}\boldsymbol{x} \to 0} \frac{\mathbb{E}[N(t + \mathrm{d}t, \boldsymbol{x} + \mathrm{d}\boldsymbol{x})|H_t]}{\mathrm{d}t \mathrm{d}\boldsymbol{x}}. \tag{3.9}$$

This can be interpreted as the result in the following proposition.

Proposition 3.1.4. *For $t > 0$ and $t_1, \ldots, t_n$ denoting the events prior to t, the conditional intensity function $\lambda(t, \boldsymbol{x}|H_t)$ is defined by*

$$\lambda(t, \boldsymbol{x}|H_t) = \frac{f_{T,X}(t, \boldsymbol{x}|H_{t_n})}{1 - \mathbb{P}\Big[t_{n+1} \in [t_n, t)|H_{t_n}\Big]}. \tag{3.10}$$

Proof. From the definition of the conditional intensity function in (3.9), the intensity function can be stated as

$$\begin{aligned}\lambda(t,\boldsymbol{x}|H_t) &= \lim_{\mathrm{d}t,\mathrm{d}\boldsymbol{x}\to 0} \frac{\mathbb{E}[N(t+\mathrm{d}t,\boldsymbol{x}+\mathrm{d}\boldsymbol{x})|H_t]}{\mathrm{d}t\mathrm{d}\boldsymbol{x}} \\ &= \lim_{\mathrm{d}t,\mathrm{d}\boldsymbol{x}\to 0} \frac{\mathbb{P}\Big[t_{n+1}\in[t,t+\mathrm{d}t),\boldsymbol{x_{n+1}}\in[\boldsymbol{x}+\mathrm{d}\boldsymbol{x}]|H_t\Big]}{\mathrm{d}t\mathrm{d}\boldsymbol{x}} \\ &= \lim_{\mathrm{d}t,\mathrm{d}\boldsymbol{x}\to 0} \frac{\mathbb{P}\Big[t_{n+1}\in[t,t+\mathrm{d}t),\boldsymbol{x_{n+1}}\in[\boldsymbol{x}+\mathrm{d}\boldsymbol{x}]|H_{t_n},t_{n+1}\notin[t_n,t)\Big]}{\mathrm{d}t\mathrm{d}\boldsymbol{x}},\end{aligned}$$

since $H_t = H_{t_n}\cap\{t_{n+1}\notin[t_n,t)\}$. In the above equation $[\boldsymbol{x}+\mathrm{d}\boldsymbol{x}]$ denotes a small ball centered around $\boldsymbol{x}$. Using the formula for conditional probability, this can be expressed as

$$\lambda(t,\boldsymbol{x}|H_t) = \lim_{\mathrm{d}t,\mathrm{d}\boldsymbol{x}\to 0} \frac{\mathbb{P}\Big[t_{n+1}\in[t,t+\mathrm{d}t),\boldsymbol{x_{n+1}}\in[\boldsymbol{x}+\mathrm{d}\boldsymbol{x}],t_{n+1}\notin[t_n,t)|H_{t_n}\Big]}{\mathbb{P}\Big[t_{n+1}\notin[t_n,t)|H_{t_n}\Big]\cdot\mathrm{d}t\mathrm{d}\boldsymbol{x}},$$

where it can be noted that $t_{n+1}\notin[t_n,t)|H_{t_n}$ is redundant when $t_{n+1}\in[t,t+\mathrm{d}t)$ is already known. The conditional intensity function can therefore be written as

$$\begin{aligned}\lambda(t,\boldsymbol{x}|H_t) &= \lim_{\mathrm{d}t,\mathrm{d}\boldsymbol{x}\to 0} \frac{\mathbb{P}\Big[t_{n+1}\in[t,t+\mathrm{d}t),\boldsymbol{x_{n+1}}\in[\boldsymbol{x}+\mathrm{d}\boldsymbol{x}]|H_{t_n}\Big]}{1-\mathbb{P}\Big[t_{n+1}\in[t_n,t)|H_{t_n}\Big]\cdot\mathrm{d}t\mathrm{d}\boldsymbol{x}} \\ &= \frac{f_{T,X}(t,\boldsymbol{x}|H_{t_n})}{1-\mathbb{P}\Big[t_{n+1}\in[t_n,t)|H_{t_n}\Big]},\end{aligned}$$

which completes the proof. □

The probability density function $f_{T,X}$, which is used in part A of the log-likelihood in (3.8), can be derived from Proposition 3.1.4. This derivation is given in the following proposition.

Proposition 3.1.5. *For $t>0$ and $t_1,\ldots,t_n$ denoting the events prior to t, the probability density function $f_{T,X}$ given the history H_{t_n} is given by*

$$f_{T,X}(t,\boldsymbol{x}|H_{t_n}) = \lambda(t,\boldsymbol{x}|H_t)\cdot\exp\Big(-\int_{\mathbb{R}^2}\int_{t_n}^{t}\lambda(t,\boldsymbol{x'}|H_{t_n})\mathrm{d}t'\mathrm{d}\boldsymbol{x'}\Big), \tag{3.11}$$

where λ is the conditional intensity function.

Proof. The result in (3.10) can be written as

$$f_{T,X}(t,\boldsymbol{x}|H_{t_n}) = \lambda(t,\boldsymbol{x}|H_t)\cdot\Big(1-\mathbb{P}\Big[t_{n+1}\in[t_n,t)|H_{t_n}\Big]\Big).$$

The first term in the above equation is dependant on both time and space, while the second term only depends on time. To find the value of the probability density function $f_{T,X}$, the second term has to be stated as a function of both time and space as well.

Taking the marginal of the second term gives

$$f_{T,X}(t,\boldsymbol{x}|H_{t_n}) = \lambda(t,\boldsymbol{x}|H_t)\cdot\Big(1-\int_{\mathbb{R}^2}\mathbb{P}\big[t_{n+1}\in[t_n,t),\boldsymbol{x_{n+1}}=\boldsymbol{x}'|H_{t_n}\big]d\boldsymbol{x}'\Big),$$
$$= \lambda(t,\boldsymbol{x}|H_t)\cdot\Big(1-\int_{\mathbb{R}^2}\int_{t_n}^{t} f_{T,X}(t',\boldsymbol{x}'|H_{t_n})\mathrm{d}t'\mathrm{d}\boldsymbol{x}'\Big). \tag{3.12}$$

The value of $\Big(1-\int_{\mathbb{R}^2}\int_{t_n}^{t} f_{T,X}(t',\boldsymbol{x}'|H_{t_n})\mathrm{d}t'\mathrm{d}\boldsymbol{x}'\Big)$ in terms of $\lambda(t,\boldsymbol{x}|H_t)$ is then obtained in a few steps. By writing (3.12) as

$$\lambda(t,\boldsymbol{x}|H_t) = \frac{f_{T,X}(t,\boldsymbol{x}|H_{t_n})}{1-\int_{\mathbb{R}^2}\int_{t_n}^{t} f_{T,X}(t',\boldsymbol{x}'|H_{t_n})\mathrm{d}t'\mathrm{d}\boldsymbol{x}'}$$

and integrating over all values $\boldsymbol{x}\in\mathbb{R}^2$ on both sides gives

$$\int_{\mathbb{R}^2}\lambda(t,\boldsymbol{x}'|H_t)d\boldsymbol{x}' = \frac{\int_{\mathbb{R}^2} f_{T,X}(t,\boldsymbol{x}'|H_{t_n})d\boldsymbol{x}'}{1-\int_{\mathbb{R}^2}\int_{t_n}^{t} f_{T,X}(t',\boldsymbol{x}'|H_{t_n})\mathrm{d}t'\mathrm{d}\boldsymbol{x}'}.$$

Letting $\varphi(t)=\int_{\mathbb{R}^2} f_{T,X}(t,x'|H_{t_n})d\boldsymbol{x}'$, the above equation can be written as

$$\int_{\mathbb{R}^2}\lambda(t,\boldsymbol{x}'|H_t)d\boldsymbol{x}' = \frac{\varphi(t)}{1-\int_{t_n}^{t}\varphi(t')\mathrm{d}'} \tag{3.13}$$

and setting $\Phi(t)=1-\int_{t_n}^{t}\varphi(t')\mathrm{d}'$ makes it possible to write (3.13) as

$$\begin{aligned}\int_{\mathbb{R}^2}\lambda(t,\boldsymbol{x}'|H_t)d\boldsymbol{x}' &= -\frac{\frac{\mathrm{d}}{\mathrm{d}t}\Phi(t)}{\Phi(t)}\\ &= -\frac{\mathrm{d}}{\mathrm{d}t}\log\Phi(t)\\ &= -\frac{\mathrm{d}}{\mathrm{d}t}\log\Big(1-\int_{\mathbb{R}^2}\int_{t_n}^{t} f_{T,X}(t',\boldsymbol{x}'|H_{t_n})\mathrm{d}t'\mathrm{d}\boldsymbol{x}'\Big).\end{aligned}$$

By then multiplying this with -1 and integrating over time gives

$$-\int_{\mathbb{R}^2}\int_{t_n}^{t}\lambda(t,\boldsymbol{x}'|H_{t_n})\mathrm{d}t'\mathrm{d}\boldsymbol{x}' = \log\Big(1-\int_{\mathbb{R}^2}\int_{t_n}^{t} f_{T,X}(t',\boldsymbol{x}'|H_{t_n})\mathrm{d}t'\mathrm{d}\boldsymbol{x}'\Big),$$

which exponentiated results in

$$\exp\Big(-\int_{\mathbb{R}^2}\int_{t_n}^{t}\lambda(t,\boldsymbol{x}'|H_{t_n})\mathrm{d}t'\mathrm{d}\boldsymbol{x}'\Big) = 1-\int_{\mathbb{R}^2}\int_{t_n}^{t} f_{T,X}(t',\boldsymbol{x}'|H_{t_n})\mathrm{d}t'\mathrm{d}\boldsymbol{x}'. \tag{3.14}$$

Using the expression obtained in (3.14) into (3.12) gives the probability density function

$$f_{T,X}(t,\boldsymbol{x}|H_{t_n}) = \lambda(t,\boldsymbol{x}|H_t)\cdot\exp\Big(-\int_{\mathbb{R}^2}\int_{t_n}^{t}\lambda(t,\boldsymbol{x}'|H_{t_n})\mathrm{d}t'\mathrm{d}\boldsymbol{x}'\Big). \qquad \square$$

Using the probability density function $f_{T,X}$ in (3.11), part A of the log-likelihood ℓ is given by Proposition 3.1.6.

Proposition 3.1.6. *Part A of the log-likelihood is given by*

$$\ell_A = -\int_{\mathbb{R}^2}\int_0^\tau \lambda(t', \boldsymbol{x}'|H_{t'})\mathrm{d}t'd\boldsymbol{x}' + \sum_{i=1}^{n} \log\Big(\lambda(t_i, \boldsymbol{x_i}|H_{t_{i-1}})\Big). \tag{3.15}$$

Proof. The log-likelihood of part A in (3.8) is

$$\ell_A = \log\Big(f_{T_1,X_1}(t_1, \boldsymbol{x}_1)\prod_{i=2}^{n} f_{T_i,X_i|H_{t_{i-1}}}(t_i, \boldsymbol{x}_i|H_{t_{i-1}})\mathbb{P}(T_{n+1} > \tau|H_{t_n})\Big). \tag{3.16}$$

Just as in the temporal implementation the information that the $(n+1)^{\text{th}}$ event occurs at time $T_{n+1} > \tau$ is stated as

$$\mathbb{P}(T_{n+1} > \tau|H_{t_n}) = \exp\Big(-\int_{t_n}^{\tau}\lambda(t'|H_{t_n}\Big)\mathrm{d}t'. \tag{3.17}$$

By taking the marginal of (3.17), the probability is stated in terms of both t and $\boldsymbol{x}$,

$$\mathbb{P}(T_{n+1} > \tau|H_{t_n}) = \exp\Big(-\int_{\mathbb{R}^2}\int_{t_n}^{\tau}\lambda(t', \boldsymbol{x}'|H_{t_n}\Big)\mathrm{d}t'\mathrm{d}\boldsymbol{x}'. \tag{3.18}$$

Using the result in (3.18) and Proposition 3.1.5 in (3.16) results in the log-likelihood

$$\begin{aligned}\ell_A = \log\Bigg(&\prod_{i=1}^{n}\lambda(t_i, \boldsymbol{x_i}|H_{t_{i-1}})\cdot\exp\Big(-\int_{\mathbb{R}^2}\int_{t_{i-1}}^{t_i}\lambda(t', \boldsymbol{x}'|H_{t_{i-1}})\mathrm{d}t'\mathrm{d}\boldsymbol{x}'\Big)\\ &\cdot\exp\Big(-\int_{\mathbb{R}^2}\int_{t_n}^{\tau}\lambda(t', \boldsymbol{x}'|H_{t_n})\mathrm{d}t'\mathrm{d}\boldsymbol{x'}\Big)\Bigg)\\ =&\sum_{i=1}^{n}\log\Big(\lambda(t_i, \boldsymbol{x_i}|H_{t_{i-1}})\Big) - \int_{\mathbb{R}^2}\int_0^\tau\lambda(t', \boldsymbol{x}'|H_{t'})\mathrm{d}t'\mathrm{d}\boldsymbol{x'}.\end{aligned}$$

□

With ℓ_A stated only in terms of $\lambda(t, \boldsymbol{x}|H_t)$, it is possible to estimate the parameters of part A of the log-likelihood. As mentioned in section 3.1.2, the conditional intensity function is

$$\lambda(t, \boldsymbol{x}|H_t) = \mu(\boldsymbol{x}) + \sum_{j:t_j<t}\omega_T(t - t_j; w_j)\omega_X(\boldsymbol{x_k} - \boldsymbol{x_j}) \tag{3.19}$$

and putting this into (3.15) results in

$$\begin{aligned}\ell_A =& \sum_{i=1}^{n}\log\Big(\mu(\boldsymbol{x_i}) + \sum_{j:t_j<t_i}\frac{\psi e^{\beta w_j}}{t_i - t_j + \gamma}\cdot\frac{1}{2\pi\sqrt{|\Sigma|}}e^{-\frac{1}{2}(\boldsymbol{x_i}-\boldsymbol{x_j})^\top\Sigma^{-1}(\boldsymbol{x_i}-\boldsymbol{x_j})}\Big)\\ &-\int_{\mathbb{R}^2}\int_0^\tau\mu(\boldsymbol{x'}) + \sum_{j:t_j<t'}\frac{\psi e^{\beta w_j}}{t' - t_j + \gamma}\cdot\frac{1}{2\pi\sqrt{|\Sigma|}}e^{-\frac{1}{2}(\boldsymbol{x'}-\boldsymbol{x_j})^\top\Sigma^{-1}(\boldsymbol{x'}-\boldsymbol{x_j})}\mathrm{d}t'\mathrm{d}\boldsymbol{x}'\end{aligned}$$

.

Noting that

$$\int_{\mathbb{R}^2} e^{-\frac{1}{2}(\boldsymbol{x'}-\boldsymbol{x_j})^\top\Sigma^{-1}(\boldsymbol{x'}-\boldsymbol{x_j})}d\boldsymbol{x'} = 2\pi\sqrt{|\Sigma|}$$

as stated by Petersen and Pedersen [15], the spatial trigger density ω_X integrated over $\boldsymbol{x} \in \mathbb{R}^2$ can be simplified to 1. This results in the log-likelihood

$$\begin{aligned}\ell_A = &\sum_{i=1}^{n} \log\left(\mu(\boldsymbol{x_i}) + \sum_{j:t_j<t_i} \frac{\psi e^{\beta w_j}}{t_i - t_j + \gamma} \cdot \frac{1}{2\pi\sqrt{|\Sigma|}} e^{-\frac{1}{2}(\boldsymbol{x_i}-\boldsymbol{x_j})^\top \Sigma^{-1}(\boldsymbol{x_i}-\boldsymbol{x_j})}\right) \\ &- \int_{\mathbb{R}^2}\int_0^\tau \mu(\boldsymbol{x'})\mathrm{d}t'\mathrm{d}\boldsymbol{x'} + \int_0^\tau \sum_{j:t_j<t'} \frac{\psi e^{\beta w_j}}{t' - t_j + \gamma}\mathrm{d}t'\end{aligned}$$

and the parameters to be estimated for ℓ_A are therefore $\theta_A = [\alpha_0, \alpha_1, \alpha_2, \alpha_3, \gamma, \psi, \beta, \Sigma]$.

Part B: Mark sizes

The marks are modeled similar to the temporal implementation in section 3.1.1, with the GPD(ξ, σ). This results in the probability density function of a random variable $W_i \sim \text{GPD}(\xi, \sigma)$ being

$$f_{W_i}(w_i) = \frac{1}{\sigma}\left(1 + \frac{\xi \cdot w_i}{\sigma}\right)^{\left(-\frac{1}{\xi}-1\right)}, \tag{3.20}$$

with shape and scale parameters $\xi \in \mathbb{R}$ and $\sigma \in \mathbb{R}^+$. Using (3.20) in part B of the log-likelihood in (3.8) gives

$$\begin{aligned}\ell_B &= \log\left(f_{W_1}(w_1)\prod_{i=2}^{n} f_{W_i|H_{t_{i-1}}}(w_i|H_{t_{i-1}})\right) \\ &= \log f_{W_1}(w_1) + \sum_{i=2}^{n} \log f_{W_i|H_{t_{i-1}}}(w_i|H_{t_{i-1}}) \\ &= \sum_{i=1}^{n} \log\left(\frac{1}{\sigma}\left(1 + \frac{\xi \cdot w_i}{\sigma}\right)^{\left(-\frac{1}{\xi}-1\right)}\right),\end{aligned}$$

where it can be noted that the marks are not dependent on the history.

However, it should be noted that the extreme values w_i are modeled as a proportion (proportion of burnt area of a grid cell) and are hence in $(0, 1)$. The distribution should therefore be of finite support and only achieve values in this range. This is done by only allowing $\xi < 0$, leading to the distribution achieving values in $0 \leq w_i \leq -\sigma/\xi$, see (2.8). Then, if $\xi < 0$ and $\sigma \leq |\xi|$, the random variables are bounded, taking values in $0 \leq w_i \leq 1$.

The parameters to be estimated for part B of the log-likelihood are $\theta_B = [\xi, \sigma]$.

3.1.3 Parameter estimation

With the log-likelihood separated into parts A and B, the parameters in θ_A and θ_B can be estimated unrelated to one another. The parameters of part A are estimated using the *Expectation-maximization (EM) algorithm*, while the parameters for part B are estimated using *Maximum likelihood estimation (MLE)*. The parameters are estimated using different methods since part A is computationally more complex than part B. While MLE works nicely to get $\hat{\theta}_B$, the EM algorithm is more efficient when finding the optimal parameter estimates of $\hat{\theta}_A$.

Estimating θ_A

When studying events of self-exciting processes, an event i can be considered to either come from the background intensity $\mu(\boldsymbol{x_i})$ or that it is triggered by a previous event j with $\omega_T(t_i - t_j; w_j) \cdot \omega_X(\boldsymbol{x_i} - \boldsymbol{x_j})$ [9]. By introducing a hidden quantity u_i for each event i, indicating whether the event came from the background intensity ($u_i = 0$) or from a previous event j ($u_i = j$), the log-likelihood ℓ_A can be stated as

$$\begin{aligned}\ell_A = &\sum_{i=1}^{n} \mathbf{1}\{u_i = 0\} \log \mu(\boldsymbol{x_i}) \\ &+ \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbf{1}\{u_i = j\} \log\Big(\omega_T(t_i - t_j; w_j) \cdot \omega_X(\boldsymbol{x_i} - \boldsymbol{x_j})\Big) \\ &- \int_{\mathbb{R}^2}\int_0^{\tau} \lambda(t', \boldsymbol{x'}) \mathrm{d}t' \mathrm{d}\boldsymbol{x'},\end{aligned}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function and equals one when its argument is true and zero otherwise. With the log-likelihood of this form, $\hat{\theta}_A$ can be estimated using the EM algorithm [9].

The first step in the EM algorithm is the *expectation step*, also known as the *E-step*. In this step the expectation of the likelihood ℓ_A is computed,

$$\begin{aligned}\mathbb{E}[\ell_A] = &\sum_{i=1}^{n} \mathbb{E}[\mathbf{1}\{u_i = 0\}] \log \mu(\boldsymbol{x_i}) \\ &+ \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{E}[\mathbf{1}\{u_i = j\}] \log\Big(\omega_T(t_i - t_j; w_j) \cdot \omega_X(\boldsymbol{x_i} - \boldsymbol{x_j})\Big) \\ &- \int_{\mathbb{R}^2}\int_0^{\tau} \lambda(t', \boldsymbol{x'}) \mathrm{d}t' \mathrm{d}\boldsymbol{x'},\end{aligned}$$

where the triggering probabilities $\mathbb{P}(u_i = j) = \mathbb{E}[\mathbf{1}\{u_i = j\}]$ are estimated as

$$\mathbb{P}(u_i = j) = \begin{cases} \frac{\omega_T(t_i - t_j; w_j) \cdot \omega_X(\boldsymbol{x_i} - \boldsymbol{x_j})}{\lambda(t_i, \boldsymbol{x_i})} & t_j < t_i \\ 0 & t_j \geq t_i \end{cases}$$

and $\mathbb{P}(u_i = 0) = \mathbb{E}[\mathbf{1}\{u_i = 0\}]$ as

$$\mathbb{P}(u_i = 0) = 1 - \sum_{j=1}^{i-1} \mathbb{P}(u_i = j) = \frac{\mu(\boldsymbol{x_i})}{\lambda(t_i, \boldsymbol{x_i})},$$

following the approach by Reinhart [9]. This results in the expected log-likelihood

$$\begin{aligned}\mathbb{E}[\ell_A] = &\sum_{i=1}^{n} \mathbb{P}(u_i = 0) \log \mu(\boldsymbol{x_i}) \\ &+ \sum_{i=1}^{n}\sum_{j=1}^{n} \mathbb{P}(u_i = j) \log\Big(\omega_T(t_i - t_j; w_j) \cdot \omega_X(\boldsymbol{x_i} - \boldsymbol{x_j})\Big) \\ &- \int_{\mathbb{R}^2}\int_0^{\tau} \lambda(t', \boldsymbol{x'}) \mathrm{d}t' \mathrm{d}\boldsymbol{x'}.\end{aligned}$$

Since only the trigger densities with $\mathbb{P}(u_i = j) \neq 0$ are computed, this approach is more efficient than MLE.

When the expected log-likelihood has been computed, the algorithm moves on to the *maximization step*, or *M-step*. As the name implies this is where the parameters are maximized. These parameters are then sent back into the E-step, where new trigger probabilities are computed together with a new expected log-likelihood. By iterating between the E-step and the M-step until the log-likelihood has converged, the optimal parameters are found and returned as $\hat{\theta}_A$.

Estimating θ_B

The parameter estimate $\hat{\theta}_B$ is obtained with MLE due to ℓ_B in (3.8) not being as computationally heavy as ℓ_A. The estimate $\hat{\theta}_B = [\hat{\xi}, \hat{\sigma}]$ is computed as

$$\hat{\theta}_B = \arg\max_{\theta_B} \ell_B.$$

3.2 Measuring the risk of wildfires

When studying wildfires it is useful to have some risk measure in place to assess possible wildfires. Primarily, the probability of wildfires exceeding the set threshold u should be evaluated, however it is also of interest to get an understanding of how large a wildfire is likely to get. By having a measure of this maximum size, it is possible to prepare for these worst case scenarios. This can be compared to the VaR used in the financial context, where for instance the 95^{th} percentile for the predictive distribution is estimated for the return of some asset over the next day. With this measure, it is possible to get an understanding of the potential loss of this asset, and how the risk of the loss exceeding the 95^{th} percentile only is 5%. Similarly, when modeling wildfires it is of interest to evaluate how large the fires are likely to get. By doing so, precautions can be made such that for example 95% of the time, the systems set in place are enough to stop the wildfires.

To evaluate these risk measures, the first step is to compute the probability of the proportion of burnt area exceeding u at an arbitrary time $t+1$ in any location, given the history up until that point,

$$\mathbb{P}(Y_{t+1} > u | H_t).$$

This can be approximated by the probability of an event occurring in $(t, t+1]$,

$$\mathbb{P}(Y_{t+1} > u | H_t) \approx 1 - \mathbb{P}\Big(N(t, t+1) = 0 | H_t\Big) \tag{3.21}$$

$$= 1 - \exp\Big(-\int_t^{t+1} \lambda(t', \boldsymbol{x}) \mathrm{d}t'\Big) \tag{3.22}$$

for each location $\boldsymbol{x} \in \mathbb{R}^2$.

When the probability of a wildfire exceeding u is known for all locations, the VaR is computed. The estimated conditional quantile

$$y_q^t = \inf\{y \in \mathbb{R} : F_{Y_{t+1}|H_t}(y) \geq q\}$$

is estimated for each location $\boldsymbol{x} \in \mathbb{R}^2$ for some set value q. As mentioned above, it is common to use $q = 0.95$ in the financial context. One has to note though that q has to be set such that $1 - q \leq \mathbb{P}(Y_{t+1} > u|H_t)$ holds. This stems from the fact that the probability of $Y_{t+1} > y_q^t$ cannot be larger than the probability of $Y_{t+1} > u$, since $y_q^t > u$. To compute y_q^t for each location, it is first noted that

$$F_{Y_{t+1}|H_t}(y) = \mathbb{P}(Y_{t+1} \leq y|H_t) = 1 - \mathbb{P}(Y_{t+1} > y|H_t)$$

and that

$$\mathbb{P}(Y_{t+1} > y|H_t) = \mathbb{P}(Y_{t+1} - u > y - u|Y_{t+1} > u, H_t) \cdot \mathbb{P}(Y_{t+1} > u|H_t).$$

The first term on the r.h.s. in the above equation is estimated using the GPD model

$$(Y_{t+1} - u|Y_{t+1} > u, H_t) \sim \text{GPD}(\xi, \sigma)$$

and the second term is known from (3.22). From this it follows that

$$\begin{aligned} F_{Y_{t+1}|H_t}(y) &= 1 - \mathbb{P}(Y_{t+1} > y|H_t) \\ &= 1 - \mathbb{P}(Y_{t+1} - u > y - u|Y_{t+1} > u, H_t) \cdot \mathbb{P}(Y_{t+1} > u|H_t) \\ &= 1 - \left[\left(1 + \frac{\xi(y-u)}{\sigma}\right)^{-1/\xi} \cdot \left(1 - \exp\Big(-\int_t^{t+1} \lambda(t', \boldsymbol{x})\mathrm{d}t'\Big)\right)\right]. \end{aligned}$$

The values of y for which $q \leq F_{Y_{t+1}}(y)$ are sought, which now can be expressed as

$$q \leq 1 - \left[\left(1 + \frac{\xi(y-u)}{\sigma}\right)^{-1/\xi} \cdot \left(1 - \exp\Big(-\int_t^{t+1} \lambda(t', \boldsymbol{x})\mathrm{d}t'\Big)\right)\right].$$

This inequality solved for y results in

$$y \geq \frac{\sigma}{\xi}\left[\left(\frac{1-q}{1 - \exp\Big(-\int_t^{t+1} \lambda(t', \boldsymbol{x})\mathrm{d}t'\Big)}\right)^{-\xi} - 1\right] + u.$$

The smallest value of y for which $F_{Y_{t+1}|H_t}(y) \geq q$ holds is then the quantile y_q^t which results in $F_{Y_{t+1}|H_t}(y_q^t) = q$. The spatio-temporal VaR in each location is the value y_q^t computed at all $\boldsymbol{x} \in \mathbb{R}^2$.

4
Application of the wildfire dataset

The wildfire dataset is taken from the Data Challenge of the EVA 2021 Conference at The School of Mathematics at The University of Edinburgh [16]. The Data Challenge is a competition in the field of spatio-temporal regression modelling of extremes, this year treating the problem of predicting extreme wildfire activity in space and time. The data used in the challenge is gathered from multiple wildfire inventories and has already been preprocessed for the purpose of the challenge, see the website of the challenge for more information [16]. It is here applied to the model presented in chapter 3 to predict wildfire activity in the United States.

4.1 Description of dataset

The wildfire dataset covers the period between March to September each year from 1993 to 2015 for the continental United States, containing more than 500,000 data points. The spatial locations of the data are given in longitude and latitude coordinates following a $0.5° \times 0.5°$ grid, fully covering the studied area. The components of interest for the wildfire activity are wildfire occurrence and wildfire size. For each geographic position in the grid, the time of occurrence and burnt area of individual wildfires have hence been recorded. The number of wildfires and the aggregated burnt area have then been summed up monthly for each geographic location.

The dataset as a whole contains 37 variables, where 19 of these are used for the model. The prediction variable of interest is the proportion of burnt area in each grid cell, called *PBA*. This variable is not originally part of the dataset, but is created from a variable of the aggregated burnt area divided by a variable for the total area of the grid cell, at each location.

Out of the 19 variables used in the model, 13 are related to land cover. The European Union's COPERNICUS service provides land cover classification maps, where 18 of its categories are present in the studied area and 13 of these will be used in the model. These 13 categories are grouped into the 3 larger groups `cropland`, `forest` and `low vegetation`. This is done to simplify the model. For a description of all 18 categories and how they are grouped, see Appendix C.1. For each grid cell the proportions of each of the 3 types of land cover is averaged over all time points, since these generally do not change that much over time. The proportions give an idea of what underlying properties each grid cell might have, which will be used when estimating the background intensity of wildfires. The 3 auxiliary variables $q_1(\boldsymbol{x}), q_2(\boldsymbol{x}), q_3(\boldsymbol{x})$ correspond to the proportional presence of each of these 3 types of vegetation in location $\boldsymbol{x} \in \mathbb{R}^2$, given in the same order as mentioned above.

Out of the original 37 variables, 10 are meteorological variables from the COPERNICUS Climate Data Service. These variables typically show seasonal behaviour and for simplicity are not used in the model. For more information about these 10 variables, see Appendix C.1. Variables related to the altitude at each location are also not used in the model for the same reason.

This results in the dataset used in the model:

- *PBA* – proportion of burnt area in each grid cell (values to be predicted are given as `NA`)
- *lon* – longitude coordinate of grid cell center
- *lat* – latitude coordinate of grid cell center
- t – time of observation (in months), where $t = 0$ corresponds to March 1993
- $q_1, \ldots, q_3$ – proportion of the 3 land cover classes in the grid cell, where $q_i \in (0, 1)$ for all $i \in \{1, 2, 3\}$ and $\sum_{i=1}^{3} q_i \approx 1$.

4.1.1 Training, validation and test datasets

The wildfire dataset was split into a training dataset and a test dataset by the Data Challenge providers. From the original dataset of over 500,000 data points, 80,000 observations of the variable for the aggregated burnt area are removed for testing. These are all taken from even years, $(1994, \ldots, 2014)$ and results in 80,000 data points of the *PBA* variable being masked in the training dataset. The training dataset consists of the rest of the data, with the $80,000$ masked variables set to `NA`. The test dataset is not yet released and is unavailable.

To be able to validate the model a validation data set is created. 100 data points are removed randomly in time and space from the training set and put into a validation set.

4.1.2 Dealing with missing data

The only missing data in the wildfire dataset are the $80,000$ variables removed to be used in the test data. Whenever the variable $PBA =$ `NA` in a data point, the entire data point is removed from the dataset.

4.2 Choosing the threshold

As discussed in section 2.2.4 the extreme values are defined by the choice of threshold. For the wildfire dataset described in section 4.1, the prediction variable is the *proportion of burnt area* in a grid cell and it is hence the threshold of this proportion that will indicate the extreme values. In Figure 4.1 the proportion of burnt area at all locations and all time points can be seen. Note that the locations are two-dimensional and therefore time cannot be specified on the x-axis, as the burnt area in different locations are shown after one another in the plot, while in reality they occur at the same time point.

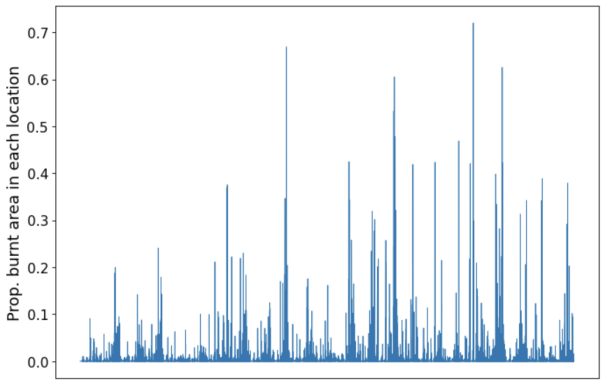


Figure 4.1: The proportion of burnt area in each location recorded between March 1993 and September 2015.

By creating the MRL plot and parameter stability plot as described in section 2.2.4, some indication is given of what a suitable threshold is. Sometimes, however, these indications are not very accurate and some extra thought is required when selecting the threshold. This is the case for the wildfire dataset. The MRL plots of this dataset can be seen in Figure 4.2.

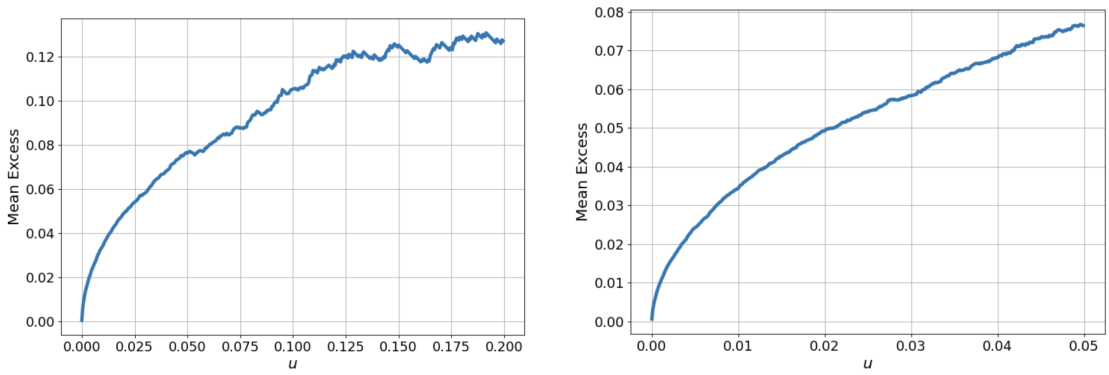


Figure 4.2: The MRL plots of the wildfire dataset. The left figure is evaluated for $u \in [0, 0.2]$ and the right for $u \in [0, 0.05]$.

In the left figure an approximate linear trend can be seen from $u = 0.025$, which indicates that this is a suitable threshold. However, setting the threshold to $u = 0.025$ results in only 0.16% of the data being set to extreme values. The rule of thumb is to have approximately 10% of data being extreme values. This allows for enough data points to fit the model, while still only using data points that are significantly larger than the majority of data points. Since the training set of the wildfire dataset consists of almost 500.000 data points, which is a large dataset, it is reasonable to allow less than 10% to be extreme values. However, 0.16% of the dataset corresponds to only 790 extreme data points. Lowering the threshold allows for more extreme data points and therefore a lower threshold is sought. By potting the MRL plot for the values of $u \in [0, 0.05]$, an approximate linear trend is seen from $u = 0.01$ in the right of Figure 4.2. For this value of u, 0.36% of the data

is considered extreme values, which results in 1740 extreme data points. This is enough data to fit the model, and checking for wildfires that exceed 1% of the grid area in each grid cell is a nice and easy definition of an extreme value. Thus, the threshold is chosen to be $u = 0.01$.

The parameter stability plots for the scale and shape parameters σ, ξ are not considered here since they are too unstable for the wildfire dataset to give any further information about what threshold to choose.

4.3 Results

Parameters are fitted to the model as described in section 3.1.3. The results of the estimated parameters $\hat{\theta}_A$ can be found in Table 4.1 and the estimated parameters $\hat{\theta}_B$ in Table 4.2. For the results in Table 4.1, note that $\Sigma = P^\top DP$ for

$$D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

and

$$P = \begin{bmatrix} \cos(\eta) & -\sin(\eta) \\ \sin(\eta) & \cos(\eta) \end{bmatrix}.$$

Table 4.1: The estimated parameters $\hat{\theta}_A = [\hat{\alpha}_0, \hat{\alpha}_1, \hat{\alpha}_2, \hat{\alpha}_3, \hat{\gamma}, \hat{\psi}, \hat{\beta}, \hat{\lambda}_1, \hat{\lambda}_2, \hat{\eta}]$.

$\hat{\alpha}_0$	$\hat{\alpha}_1$	$\hat{\alpha}_2$	$\hat{\alpha}_3$	$\hat{\gamma}$	$\hat{\psi}$	$\hat{\beta}$	$\hat{\lambda}_1$	$\hat{\lambda}_2$	$\hat{\eta}$
0.100	-1.820	0.209	1.588	50.014	0.630	0.072	0.085	0.083	0.937

Table 4.2: The parameters $\hat{\theta}_B = [\hat{\xi}, \hat{\sigma}]$, estimated using finite support.

$\hat{\xi}$	$\hat{\sigma}$
$-1.75 \cdot 10^{-5}$	0.0321

4.3.1 Validation of model

First of all the model with estimated parameters $\hat{\theta}_A$ and $\hat{\theta}_B$ has to be validated. This can be done separately for part A and part B. Starting with part A, the model is validated by checking the predicted times of events. The cumulative number of predicted events at time t is given by

$$\Lambda(t) = \int_0^t \int_{\boldsymbol{x} \in \mathbb{R}^2} \lambda(t', \boldsymbol{x'}) d\boldsymbol{x}' dt' \tag{4.1}$$

and can be seen for all times in Figure 4.3. The predicted events are plotted against the actual number of events that took place. As can be seen in the figure, the predictions deviate a bit from the true values, which can be seen from the blue dots generally being positioned over the orange line indicating perfect fit. However, this

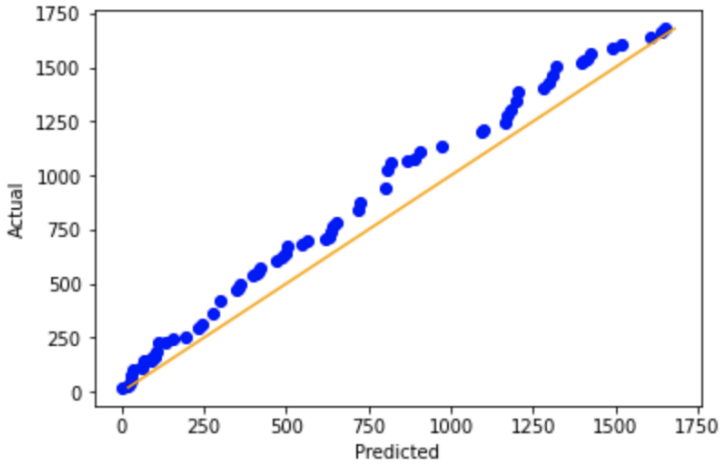


Figure 4.3: The cumulative number of predicted events plotted against the actual number of events taking place between 1993 and 2015 in blue. The orange line indicates perfect model fit.

deviation is not too large and the model can be said to be a good fit for the event times.

Moving on to part B, the model is validated by checking if the exceedances over the threshold u in the validation set follow the GPD$(\hat{\xi}, \hat{\sigma})$ distribution. This is done by checking if the residuals,

$$R_i = \frac{1}{\hat{\xi}} \log\left(\frac{w_i \cdot \hat{\xi}}{\hat{\sigma}} + 1\right),$$

are unit exponential variables for all events i exceeding the threshold. The reasoning behind this is that a random variable X following the GPD distribution with parameters ξ and σ can be generated from a random variable U uniformly distributed on $(0, 1]$ as

$$X = \frac{\sigma(U^{-\xi} - 1)}{\xi}.$$

This can be written in terms of X,

$$U = \left(\frac{X \cdot \xi}{\sigma} + 1\right)^{-1/\xi},$$

and taking the negative logarithm of this gives

$$-\log U = \frac{1}{\xi} \log\left(\frac{X \cdot \xi}{\sigma} + 1\right).$$

The negative logarithm of a uniformly distributed random variable is an exponential random variable, distributed as $exp(1)$. It then follows that if R_i approximately follow $exp(1)$, the exceedances w_i approximately follow the GPD$(\hat{\xi}, \hat{\sigma})$.

The result of computing R_i for the exceedances w_i in the validation set and plotting these in an exponential probability plot can be seen in Figure 4.4. The

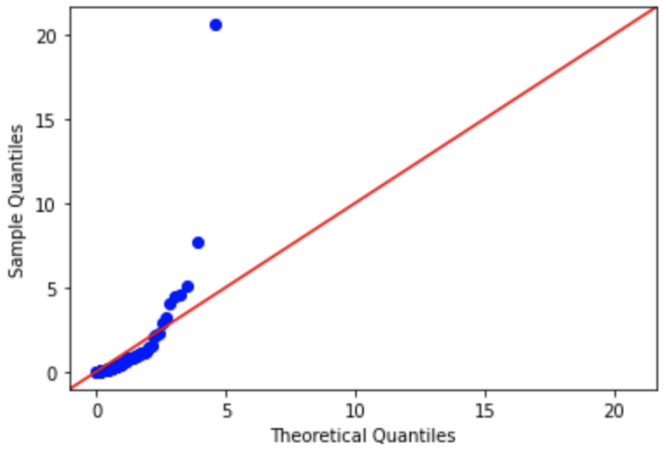


Figure 4.4: The exponential probability plot of the residuals of the validation set, for the GPD($\hat{\xi}, \hat{\sigma}$) model with $\xi < 0$.

model is clearly not a very good fit, indicating that the estimated parameters $\hat{\xi}$ and $\hat{\sigma}$ are wrong.

The parameters ξ and σ were estimated with finite support of the GPD, only allowing $\xi < 0$ since the proportion of burnt area is in $(0, 1)$ for each grid square. The choice of modeling the proportion of burnt area, rather than the cumulative burnt area at each time point, was made so that the model cannot predict more burnt area than is available in a grid square. This, however, forces $\xi < 0$, making the GPD a bad fit for the model as was seen in Figure 4.4. To get around this issue the data is transformed using the reverse sigmoid function. In this way $\xi \geq 0$ can be allowed.

The sigmoid function, or more specifically the logistic function,

$$\frac{1}{1 + \exp(-\tilde{w})} = w, \tag{4.2}$$

takes any value $\tilde{w} \in \mathbb{R}$ and returns a value $w \in (0, 1)$. By using the inverse logistic function,

$$\log\left(\frac{w}{1-w}\right) = \tilde{w}, \tag{4.3}$$

the reverse can instead be performed. Any value $w \in (0, 1)$ can be transformed to the unbounded set $\mathbb{R}$, which results in the exceedances no longer being bounded. By transforming the threshold as well and allowing $\xi \geq 0$, the parameters in θ_B are estimated again, now found in Table 4.3. The new exponential probability plot of the residuals can be seen in Figure 4.5, where it becomes evident that this estimate of the parameters gives a good fit of the model.

Table 4.3: The parameters $\hat{\theta}_B = [\hat{\xi}, \hat{\sigma}]$, estimated with the sigmoid transformation for $\xi \geq 0$.

$\hat{\xi}$	$\hat{\sigma}$
$4.31 \cdot 10^{-5}$	0.962

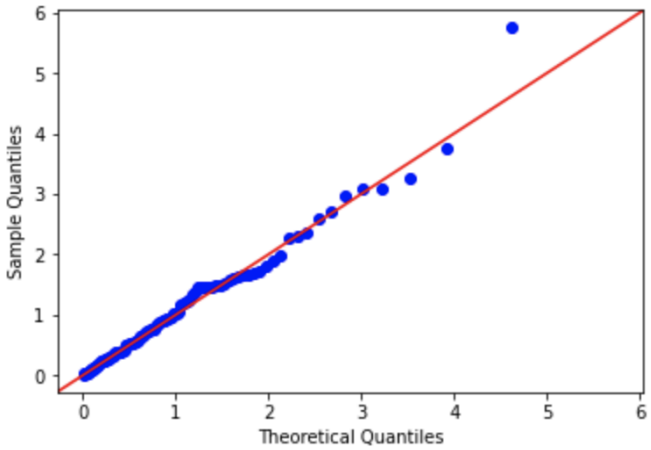


Figure 4.5: The exponential probability plot of the residuals of the validation set, for the GPD($\hat{\xi}, \hat{\sigma}$) model with $\xi \geq 0$.

By looking at the ACF of the residuals and squared residuals for the new estimates in Figure 4.6, it can be seen that the residuals look like white noise. There is no time series structure left, which further indicates that the model is a good fit.

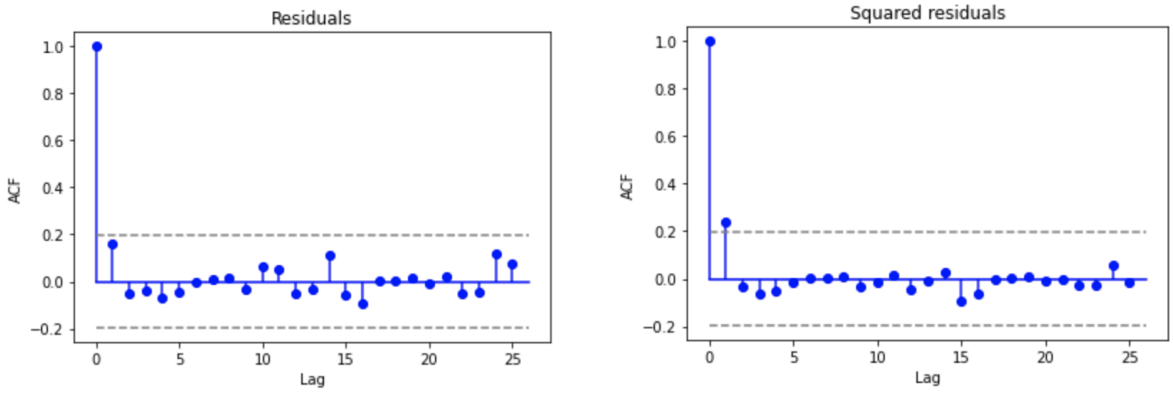


Figure 4.6: The ACF of the residuals (left) and of the squared residuals (right), with the 95% confidence interval in gray and $\xi \geq 0$.

The result in Tables 4.1 and 4.3 give good estimates of the parameters θ_A and θ_B, and are hence used in the model.

4.3.2 Intensity of the model

The estimated intensity of the model is shown in three steps as the background intensity $\mu(\boldsymbol{x})$, the trigger density $\sum_{j:tj<t} \omega_T(t-t_j; w_j)\omega(\boldsymbol{x}-\boldsymbol{x_j})$ and finally the full intensity $\lambda(t, \boldsymbol{x}|H_t)$.

Estimated background intensity

The estimated background intensity, which can be seen in Figure 4.7, is only dependent of space and is hence the same at all time points. The intensity is generally larger in the Southwest and West of the United States, and smaller in the Midwest and at borders neighbouring water. The contribution from the three auxiliary variables representing cropland, forests and low vegetation in different areas of the United States can be seen in Figure 4.8. By comparing Figures 4.7 and 4.8, it is possible to tell that areas with low vegetation have the highest impact in the background intensity. These are regions in the Southwest and West, with dry climate and deserts. Areas with cropland show the lowest intensities, while regions with forest end up somewhere in between the other two. This seems reasonable for the background intensity.

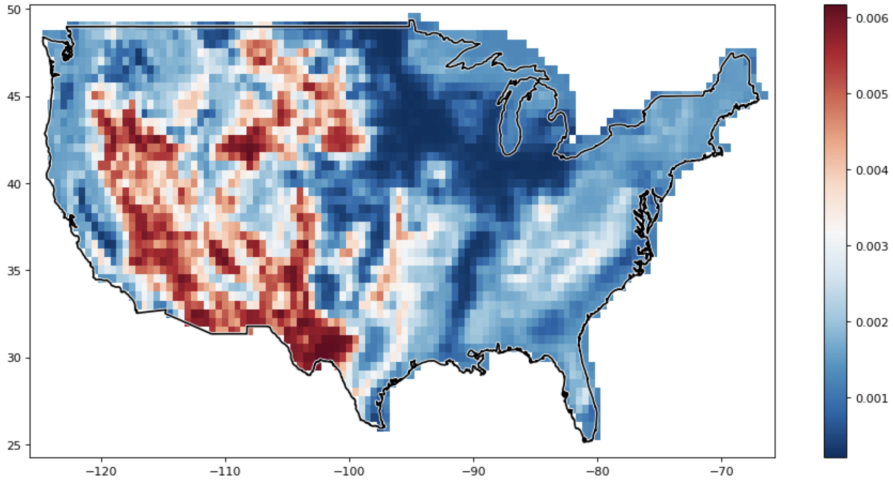


Figure 4.7: The background intensity.

Estimated trigger density

The estimated trigger density depends on both time and space. This makes it harder to evaluate than the background intensity, since it changes from one time step to the next. Looking at the estimated trigger density at an arbitrary time point $t = 50$, it can be noted that the highest trigger density is found in the dry areas of western United States and in Florida, see Figure 4.9. Extensive wildfire activity is common in these areas, and due to the spatial part of the trigger density areas in close proximity to locations of previous wildfires are estimated to have higher intensity than regions far from these locations.

It can also be noted that the intensity is point wise elevated in almost circular shapes. This is due to the parameters $\hat{\lambda}_1$ and $\hat{\lambda}_2$ of the model being almost equal. These parameters are the eigenvalues of Σ and determines the shape of the normal distribution. When these are equal, the intensity of all locations at a certain distance from a previous event is increased by the same trigger density.

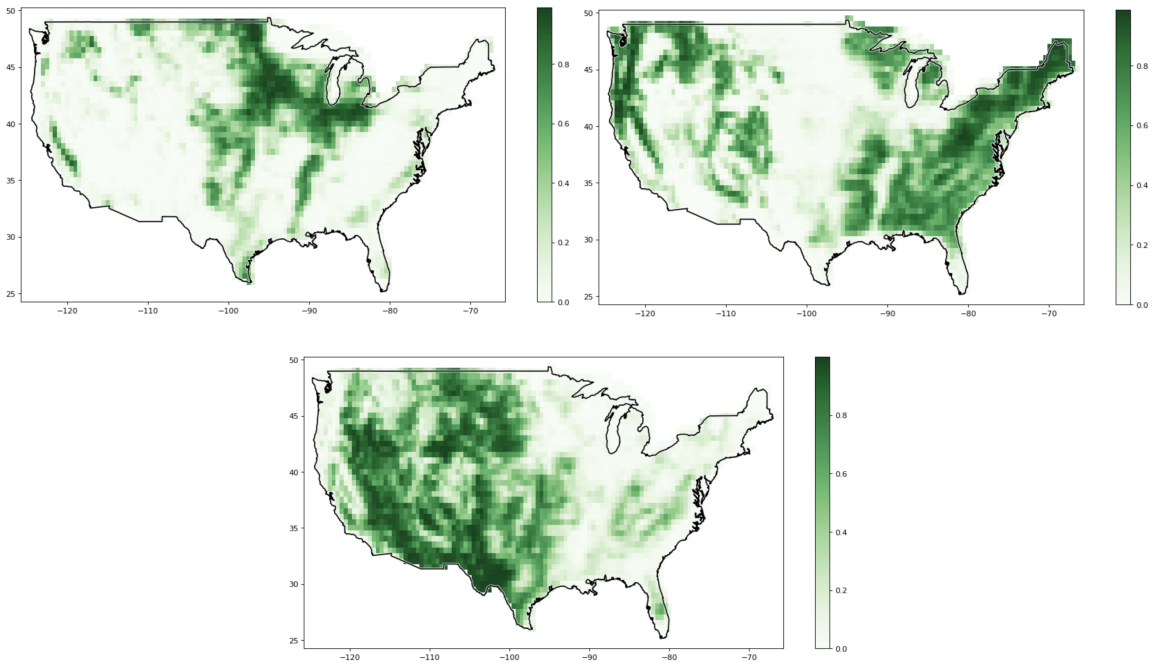


Figure 4.8: The auxiliary variables contribution to the background intensity in different areas of the United States, with cropland (top left), forest (top right) and low vegetation (bottom).

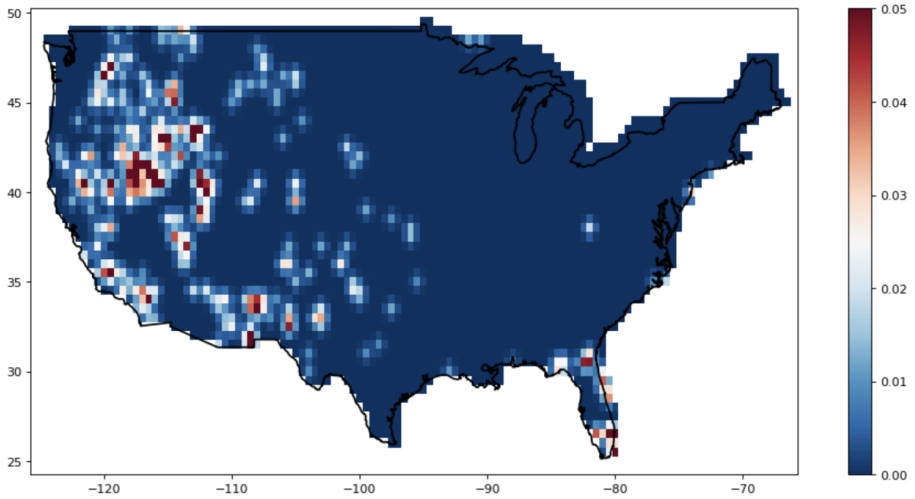


Figure 4.9: The estimated trigger density at time $t = 50$.

The total estimated intensity

The result of adding the trigger density to the background intensity can be seen in Figure 4.10. From this figure it is possible to draw the conclusion that, according to the model, the proximity to previous wildfires is the most important factor when estimating how prone an area is of wildfires. Noting that the scale of the background intensity in Figure 4.7 are much lower than the scale of the trigger densities in Figure 4.9 leads to the same conclusion. Even though the underlying vegetation contributes to the estimated intensity, it is the trigger density that has the largest impact on the estimated intensity.

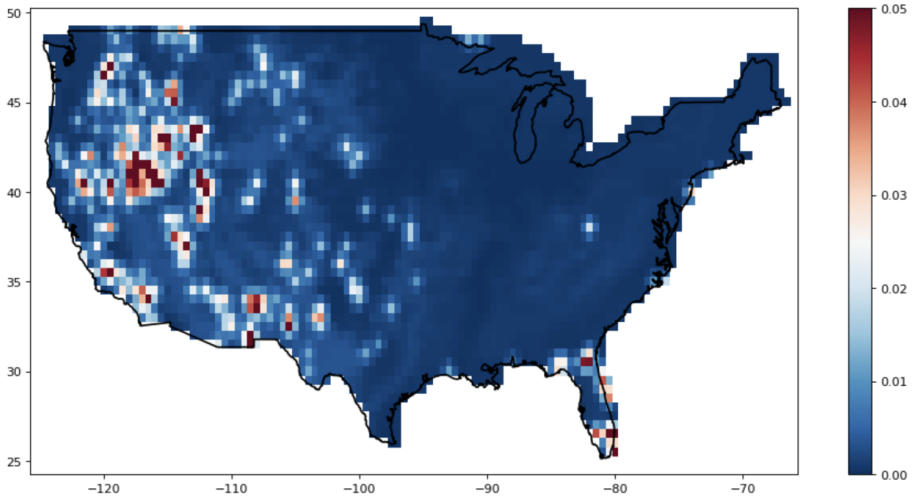


Figure 4.10: The estimated total intensity at time $t = 50$.

4.3.3 Risk assessment

The probability of a fire occurring at time $t + 1$ given the history up until H_t is evaluated according to section 3.2 in each location. The risk of fires exceeding $u = 0.01$ can be seen at four spaced out time points in Figure 4.11. All these time points represent July but for different years. The general trend is that the risk of large wildfires grows larger with time, especially in areas prone to wildfires. The high risk areas are also growing in size and spreading from Florida and the West inwards. This accurately captures how extensive wildfire activity is becoming more common in many places.

When comparing the risk of wildfires in Figure 4.11 to the actual wildfires that took place at the time in Figure 4.12, the same trend of increasing wildfire activity is indicated in both figures. From these figures it can be seen that the model does not manage to predict the exact location of future wildfires, but rather manages to indicate larger regions prone to wildfires.

Finally, the spatio-temporal VaR y_q^t is computed for $q = 0.9999$ at each location, with the result seen in Figure 4.13. The reason for choosing such a large value of q is because $1 - q \leq \mathbb{P}(Y_{t+1} > u|H_t)$ has to hold, which is a requirement when computing the VaR. This is interpreted as the probability of $Y_{t+1} \leq y_q^t$ being 0.9999. The VaR has different values in different locations, depending on the underlying

risk of a wildfire exceeding u occurring. Here, the VaR is a kind of measure of the maximum proportion of a grid square that is likely to burn up. Since q has such a high value, it is possible to say with near certainty that wildfires exceeding the sizes given in Figure 4.13 will not occur. With this knowledge it is possible to put enough resources in place to fight the fires of the indicated size in each location.

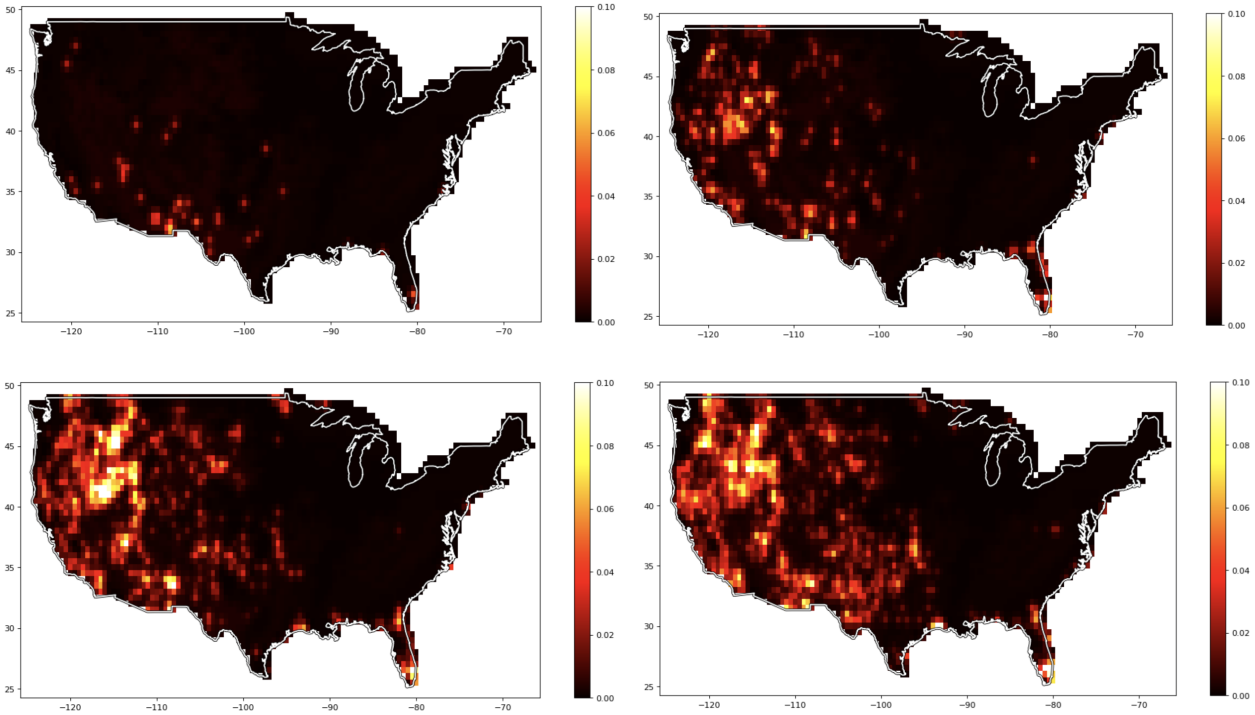


Figure 4.11: The probability of the occurrence of wildfires exceeding the threshold $u = 0.01$ at the times $t = 10$ (top left), $t = 50$ (top right), $t = 99$ (lower left) and at $t = 151$ (lower right). These times are from different years, but all in July.

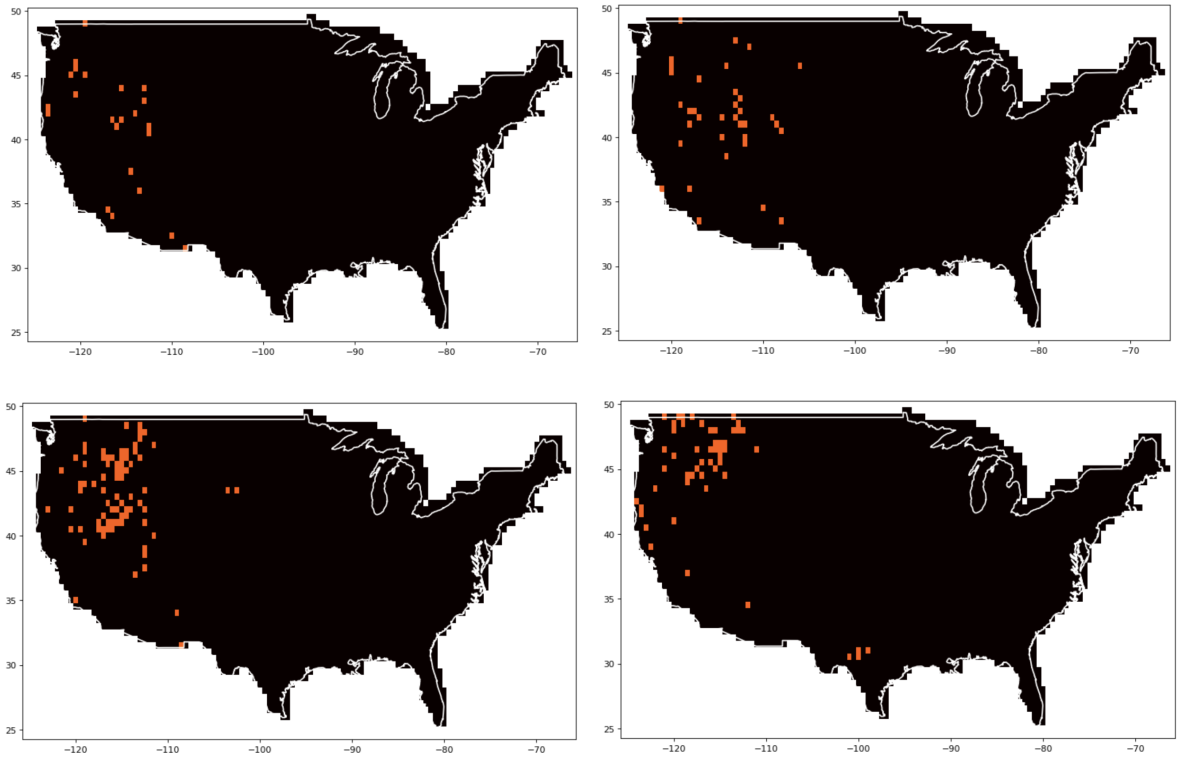


Figure 4.12: The wildfires that occurred at the times $t = 10$ (top left), $t = 100$ (top right), $t = 150$ (lower left) and at $t = 150$ (lower right), indicated in orange. These times are from different years, but all in July.

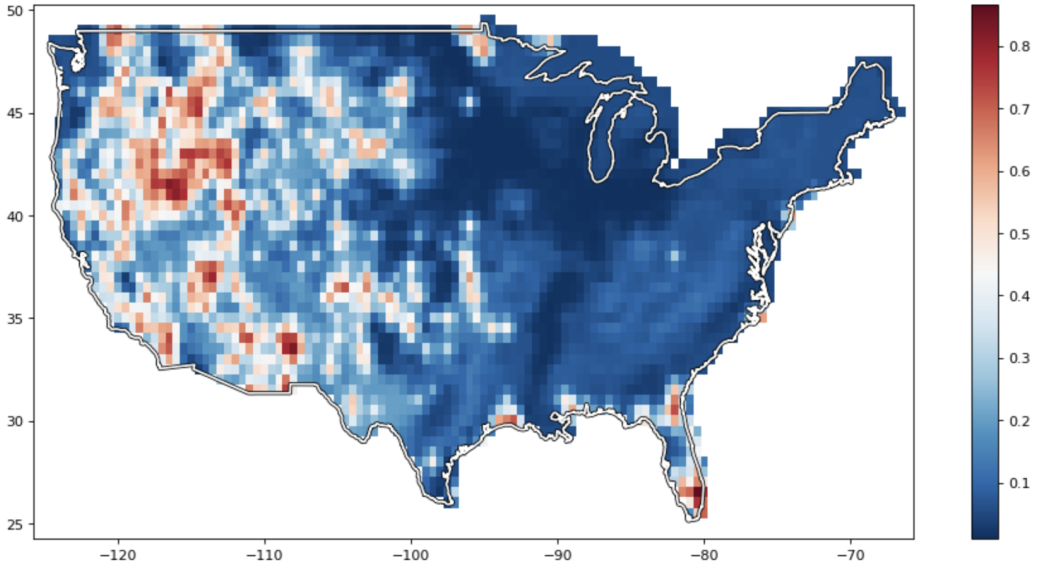


Figure 4.13: The VaR y_q^t in each location, for $t = 150$ and $q = 0.9999$. This indicates that the probability of the proportion of burnt area not exceeding y_q^t in each location is 99.99%.

5
Discussion and conclusion

In this thesis a new approach for modeling spatio-temporal extreme values was created, with a focus on modeling wildfires. The model is based on a self-exciting approach, with intensities dependent on the time and location of previous events. From this model and its predicted intensities, the risk of future extensive wildfire activity can be assessed in each location.

5.1 Interpretation of results

Looking at the intensities in section 4.3.2, it can be noted that the background intensity is considerably smaller than the trigger density. With the underlying assumption that wildfires follow a self-exciting behaviour, this seems reasonable. Some events should be caused by ecological and environmental factors, while most events are caused by previous events. However, it should be noted that the ETAS model described in section 2.1.3 and used in the model for wildfire prediction, might not reflect the wildfire activity as accurately as it does its intended purpose – modeling earthquake activity. One major difference between modeling earthquakes and wildfires, is that it does not seem to be as many contributing background factors to the earthquake activity as there is to wildfire behaviour. Earthquakes occur when tectonic plates shift, with the cause being the pressure under the plates building up before being released with the moving plates. Wildfires, on the other hand, have multiple causes. For one, there are natural causes such as lightning, but human activity is the most common source of ignition [16]. On top of that, the vegetation, temperature and dryness of an area at the time of a wildfire being ignited plays a large role in determining whether this grows into a large wildfire. Furthermore, the self-exciting structure of the model increases the probability of a wildfire occurring in the same place more than once. Considering that wildfires destroy the materials they feed off, it might not be very likely that a wildfire occurs in the same place without allowing enough time for the vegetation to grow back. It is reasonable, however, to consider the risk of wildfires increased in neighbouring areas, since the risk of wildfires spreading is large in the following time steps.

Moving on to the trigger densities, these often show circular behaviour, see Figure 4.9. This can be seen from the parameters $\hat{\lambda}_1$ and $\hat{\lambda}_2$ too, which are almost identical. An event hence increases the intensity in other locations equally in all direction, with only the distance from the event being of importance. This was a bit surprising, since it is generally seen that places located on the same latitude have the same climate and hence can be expected to be predisposed to having wildfires

at similar times. This behaviour was not seen in the model. This can mean that the assumption of regions on the same latitude being equally vulnerable to wildfires is wrong. Another possible explanation is that even though locations on the same latitude might influence each other more, the changing landscapes, coastlines and boundary data makes this negligible. The most probable reason, however, is that the spatial trigger density is a short range influence. This would mean that neither the longitude nor latitude coordinates have time to change much, before the influence of the trigger density dies down.

Looking at the risk of wildfires in section 4.3.3, it can be seen that the probability of extensive wildfire activity is increasing with time. This is an accurate representation of the increase in the number of wildfires recorded each year. It should also be noted that the increase is largest in western United States as well as in Florida, which accurately describes where the actual fires take place. When comparing the results in Figures 4.11 and 4.12, showing the expected probability of fires exceeding $u = 0.01$ and the actual fires exceeding this threshold at separate time points, both figures show the same increase in wildfire activity. While the model cannot predict the exact locations of the wildfires, it is good at approximating the regions where the wildfires will occur. It can also be noted that the model gets more accurate when more data is part of the history, indicating that the model would get better with time.

When looking at the VaR y_q^t in Figure 4.13, it can be interpreted that the largest proportions of burnt area in a grid square is found in Florida and in western United States. This is not very surprising, since these are the areas with the highest risk of wildfires. The highest values of y_q^t reach almost 0.9, which might be considered to be very large proportions of possible burnt area. One has to have in mind, however, that this is almost surely the largest proportion that will burn, since q has such a high value. If q was smaller this value would probably drop considerably. Furthermore in most regions it is possible to say that the probability of wildfires burning more than $1 - 40\%$ of the area is highly unlikely.

5.2 Limiting the information used in the model

Another point of discussion regarding the background intensity is that only the proportion of cropland, forests and low vegetation present in each grid square is used in its construction. Other land cover variables such as flooded, bare and urban areas are not used. The reason for this is that these show large variance and trying to fit a model to these are like fitting a model to white noise.

The reason for not including the meteorological variables at all in the background intensity is another. These variables generally show seasonal behavior, which is not present in the land cover variables. Due to this lack of seasonality, the land cover variables are determined by taking the mean over all time points at each location. This can be done since these typically do not change much with time. Taking the mean of the meteorological variables is not a good solution, since these tend to vary a lot with time. For these variables to be part of the model, the background intensity would have to be dependent on both time and space. For simplicity, this was not done in this project. Furthermore, the main purpose of this thesis was to evaluate

how a self-exciting spatio-temporal model can predict wildfire activity. Therefore the background intensity of the model was chosen to be of a more simple nature so that the self-exciting part could be evaluated.

5.3 Future work

As has been seen in previous sections, the model is built such that wildfires part of the history increase the probability of new wildfires occurring at later time points. It should be noted, however, that the recorded data starts in March 1993. No wildfires prior to this point are regarded when determining the risk of wildfires, leading to the predictions in 1993 being based on very few historic wildfires. Looking at the resulting model, the general trend as time goes on is that the intensities grow. It is hard to say whether the model captures the growing number of events accurately, or if the intensities just build up with the growing numbers of wildfires added to the history. One way to investigate this would be to only allow the history to cover a certain number of time steps, making the length of the history equal at all times. This would be an interesting question to investigate further.

Another part of the model that would be interesting to look into further is the background intensity. In the current model, the background intensity $\mu(\boldsymbol{x})$ is only dependent on space. More specifically, the background intensity depends on the vegetation at each location. It would be of interest to investigate if some of the meteorological variables made available in the dataset improve the model if added to the background intensity, thus making the background intensity spatio-temporal. Two of these variables that would be especially interesting to evaluate are the ones holding the temperature and precipitation at each location and all time points. Since wildfires have a tendency to occur in dry areas and the temperature and precipitation are key factors of this, it seems reasonable that the temperature and precipitation would affect the occurrence of large wildfires. Introducing these two variables would make the model more complicated, especially since these variables in many areas show seasonal behavior. This could lead to the model becoming computationally more complex. However, if enough computational power is available, this can be a very interesting development of the model.

Finally, the models performance in comparison to other models created for the same purpose would be interesting to analyse. Two models that would be interesting to compare the model to, is a non-self-exciting spatio-temporal model and a temporal self-exciting model. In the first of these the background intensity would be extended to include the variables holding the temperature and precipitation, with the trigger density removed entirely. With this model the self-exciting part of the model created in this project could be analysed. In the second, the model would be a temporal self-exciting model implemented once at each location, allowing the spatial aspect of the original model to be analysed. These comparisons could help determine if the spatio-temporal self-exciting model is useful when modeling extensive wildfire activity, or if other types of models might be more suitable.

5.4 Conclusion

The spatio-temporal model for predicting extreme wildfire activity shows promise when it comes to determining the approximate locations of high risk areas. It does not however indicate the exact locations of future wildfires. It can be used to some extent, but only to get an idea of the regions wildfires can be expected in, rather than nailing down the exact locations of these.

Some extensions can be done to the proposed model that have the potential of increasing the accuracy of it, but at the cost of making the model more complex. The model presented in this thesis manages to accurately capture some features seen in the real world, while remaining relatively simplistic.

Assuming that the model is a good representation of the real world, there is no indication of the number of large wildfires reducing in the future. The opposite seems more likely, with the number of large wildfires increasing. This means that future wildfire prevention is likely to be of even higher importance than it is today. With the spatio-temporal Value-at-Risk, some indication can be given as to how large the wildfires are likely to get and can be of great use when planning for future wildfire management.

Bibliography

[1] V. Chavez-Demoulin, A. Davison, and A. McNeil. "Estimating value-at-risk: A point process approach". In: *Quantitative Finance* 5 (Apr. 2005), pp. 227–234. DOI: 10.1080/14697680500039613.

[2] N. Sunusi, R. Aidawayati, and Irmayani. "Study of Insurance Claim using Point Process Models". In: *Indian Journal of Science and Technology* 9 (July 2016). DOI: 10.17485/ijst/2016/v9i28/97780.

[3] R. Katz. "Stochastic Modeling of Hurricane Damage". In: *Journal of Applied Meteorology* 41 (2002), pp. 754–762. DOI: 10.1175/1520-0450(2002)041<0754:SMOHD>2.0.CO;2.

[4] O. Ibe. *Markov Processes for Stochastic Modeling*. Elsevier, 2013, pp. 453–480. DOI: 10.1016/B978-0-12-407795-9.00015-3.

[5] A. Baddeley et al. *Stochastic Geometry*. Springer, 2004. DOI: 10.1007/3-540-38174-0.

[6] R Peng. *A Very Short Course on Time Series Analysis*. URL: https://bookdown.org/rdpeng/timeseriesbook/.

[7] J. González et al. "Spatio-temporal point process statistics: A review". In: *Spatial Statistics* 8 (2016), pp. 505–544. DOI: 10.1016/j.spasta.2016.10.002.

[8] F. Ilhan and S. Kozat. "Modeling of Spatio-Temporal Hawkes Processes With Randomized Kernels". In: *IEEE Transactions on Signal Processing* 68 (2020), pp. 4946–4958. DOI: 10.1109/tsp.2020.3019329.

[9] A. Reinhart. "A Review of Self-Exciting Spatio-Temporal Point Processes and Their Applications". In: *Statistical Science* 33.3 (Aug. 2018). DOI: 10.1214/17-sts629.

[10] S. Coles. *An Introduction to statistical Modeling of Extreme Values*. Springer, 2004. DOI: 10.1007/978-1-4471-3675-0.

[11] J. Pickands. "Statistical Inference using Extreme Order Statistics". In: *The Annals of Statistics* 3 (1975), pp. 119–131. DOI: 10.1214/aos/1176343003.

[12] A. Ferreira and L. De Hann. "On the Block Maxima Method in Extreme Value Theory: PWM Estimators". In: *The Annals of Statistics* 43 (2015), pp. 276–298. DOI: 10.1214/14-AOS1280.

[13] T. Mikosch. *Non-Life Insurance Mathematics*. Springer, 2009. DOI: 10.1007/978-3-540-88233-6.

[14] P. Diggle. “Spatio-temporal Point Processes: Methods and Applications”. In: *Johns Hopkins University, Dept. of Biostatistics Working Papers* 78 (June 2005). URL: https://biostats.bepress.com/jhubiostat/paper78.

[15] K. B. Petersen and M. S. Pedersen. *The Matrix Cookbook*. Version 20081110. Oct. 2008. URL: https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf.

[16] *Data Challenge, EVA 2021 Conference, School of Mathematics, The University of Edinburgh*. URL: https://www.maths.ed.ac.uk/school-of-mathematics/eva-2021/competitions/data-challenge.

A

Appendix 1

A.1 Derivation of the GPD from the GEV distribution

For an arbitrary random variable $X_i > u$, with marginal distribution function F, it is noted that the extreme event $W_i = (X_i - u | X_i > u)$ can be described by the conditional probability

$$\mathbb{P}(X_i > u + w_i | X_i > u) = \frac{1 - F(u + w_i)}{1 - F(u)}. \tag{A.1}$$

Also, from the GEV distribution it is known that, for large n,

$$\mathbb{P}(M_n \leq x) = F^n(x) \approx EV(x) = \exp\Big(- \Big[1 + \xi\Big(\frac{x - \mu}{\sigma}\Big)\Big]^{-1/\xi}\Big)$$

for $\sigma \in \mathbb{R}^+$ and $\mu, \xi \in \mathbb{R}$. Taking the logarithm of this gives

$$n \log F(x) \approx -\Big[1 + \xi\Big(\frac{x - \mu}{\sigma}\Big)\Big]^{-1/\xi}.$$

Using Taylor expansion, $\log F(x)$ can be approximated by $-(1 - F(x))$ and the above equation can be rewritten to

$$n(1 - F(x)) \approx \Big[1 + \xi\Big(\frac{x - \mu}{\sigma}\Big)\Big]^{-1/\xi}.$$

Dividing both sides by n and replacing x with $u + w$ gives

$$1 - F(u + w) \approx \frac{1}{n}\Big[1 + \xi\Big(\frac{u + w - \mu}{\sigma}\Big)\Big]^{-1/\xi}.$$

Using this in (A.1) results in

$$\begin{aligned}
\mathbb{P}(X_i > u + w_i | X_i > u) &\approx \frac{n\Big(1 + \xi(u + w_i - \mu)/\sigma\Big)^{-1/\xi}}{n\Big(1 + \xi(u - \mu)/\sigma\Big)^{-1/\xi}} \\
&= \Big(1 + \frac{\xi w_i/\sigma}{1 + \xi(u - \mu)/\sigma}\Big)^{-1/\xi} \\
&= \Big(1 + \frac{\xi w_i}{\tilde{\sigma}}\Big)^{-1/\xi},
\end{aligned}$$

where $\tilde{\sigma} = \sigma + \xi(u - \mu)$. The distribution function of the exceedance is then

$$\mathbb{P}(X_i \leq u + w_i | X_i > u) \approx 1 - \left(1 + \frac{\xi w_i}{\tilde{\sigma}}\right)^{-1/\xi}$$

when $\xi \neq 0$. The GPD distribution for $\xi = 0$ is obtained in the same way from equation (2.7) for $\xi = 0$.

B

Appendix 2

B.1 Proof of Proposition 3.1.1

Proposition 3.1.1. *The conditional intensity function $\lambda(t|H_t)$ is defined by*

$$\lambda(t|H_t) = \frac{f_T(t|H_{t_n})}{1 - F_T(t|H_{t_n})}. \tag{B.1}$$

Proof. From the definition of the conditional intensity function in (3.2), the intensity function can be stated as

$$\begin{aligned}
\lambda(t|H_t) &= \lim_{\mathrm{d}t\to 0} \frac{\mathbb{E}\Big[N([t,t+\mathrm{d}t]|H_t\Big]}{\mathrm{d}t} \\
&= \lim_{\mathrm{d}t\to 0} \frac{\mathbb{P}(t_{n+1} \in [t,t+\mathrm{d}t]|H_t)}{\mathrm{d}t} \\
&= \lim_{\mathrm{d}t\to 0} \frac{\mathbb{P}(t_{n+1} \in [t,t+\mathrm{d}t]|t_{n+1} \notin (t_n,t), H_{t_n})}{\mathrm{d}t},
\end{aligned}$$

since $H_t = H_{t_n} \cap \{t_{n+1} \notin [t_n, t)\}$. By then using the formula for conditional probability, this can be expressed as

$$\lambda(t|H_t) = \lim_{\mathrm{d}t\to 0} \frac{\mathbb{P}(t_{n+1} \in [t,t+\mathrm{d}t], t_{n+1} \notin (t_n,t)|H_{t_n})}{\mathbb{P}(t_{n+1} \notin (t_n,t)|H_{t_n}) \cdot \mathrm{d}t}.$$

However, since it is known that $t_{n+1} \in [t, t+\mathrm{d}t)$, it is redundant to state $t_{n+1} \notin [t_n, t)$. The above equation can hence be written as

$$\begin{aligned}
\lambda(t|H_t) &= \lim_{\mathrm{d}t\to 0} \frac{\mathbb{P}(t_{n+1} \in [t,t+\mathrm{d}t]|H_{t_n})}{\Big(1 - \mathbb{P}(t_{n+1} \in (t_n,t)|H_{t_n})\Big) \cdot \mathrm{d}t} \\
&= \frac{f_T(t|H_{t_n})}{1 - F_T(t|H_{t_n})},
\end{aligned}$$

completing the proof. □

B.2 Proof of Proposition 3.1.2

Proposition 3.1.2. *The probability density function $f_T(t|H_{t_n})$ is*

$$f_T(t|H_{t_n}) = \lambda(t|H_{t_n}) \cdot \exp\Big(- \int_{t_n}^{t} \lambda(t'|H_{t_n})\mathrm{d}t'\Big). \tag{B.2}$$

Proof. To get f_T the intensity function is rewritten as

$$\lambda(t|H_{t_n}) = \frac{f(t|H_{t_n})}{1 - F(t|H_{t_n})} = \frac{\frac{\mathrm{d}}{\mathrm{d}t}F(t|H_{t_n})}{1 - F(t|H_{t_n})} = -\frac{\mathrm{d}}{\mathrm{d}t}\log(1 - F(t|H_{t_n})).$$

Integrating both sides over time gives

$$\int_{t_n}^{t} \lambda(t'|H_{t_n})\mathrm{d}t' = -\log(1 - F(t|H_{t_n})) + \log\Big(1 - F(t_n|H_{t_n})\Big),$$

where $\log(1 - F(t_n|H_{t_n})) = 0$ since $F(t_n|H_{t_n}) = 0$. The equation can therefore be written as

$$-\int_{t_n}^{t} \lambda(t'|H_{t_n})\mathrm{d}t' = \log(1 - F(t|H_{t_n})),$$

where exponentiating both sides results in

$$F(t|H_{t_n}) = 1 - \exp\Big(-\int_{t_n}^{t} \lambda(t'|H_{t_n})\mathrm{d}t'\Big).$$

From this it is easy to derive that

$$f(t|H_{t_n}) = \frac{\mathrm{d}}{\mathrm{d}t}F(t|H_{t_n}) = \lambda(t|H_{t_n}) \cdot \exp\Big(-\int_{t_n}^{t} \lambda(t'|H_{t_n})\mathrm{d}t'\Big). \qquad \square$$

B.3 Proof Proposition 3.1.3

Proposition 3.1.3. *Part A of the log-likelihood is*

$$\ell_A = -\int_0^{\tau} \lambda(t'|H_{t'})\mathrm{d}t' + \sum_{i=1}^{n} \log\Big(\lambda(t_i|H_{t_{i-1}})\Big). \tag{B.3}$$

Proof. Part A of the log-likelihood is given by

$$\ell_A = \log\left(f_{T_1}(t_1)\prod_{i=2}^{n} f_{T_i|H_{t_{i-1}}}(t_i|H_{t_{i-1}})\frac{f_T(\tau|H_{t_n})}{\lambda(\tau|H_{t_n})}\right).$$

From Proposition 3.1.2 the density f_T is known, giving

$$\begin{aligned}
\ell_A &= \log\left(\prod_{i=1}^{n}\left(\lambda(t_i|H_{t_{i-1}}) \cdot \exp\Big(-\int_{t_{i-1}}^{t_i} \lambda(t'|H_{t_{i-1}})\mathrm{d}t'\Big)\right) \cdot \exp\Big(-\int_{t_n}^{\tau} \lambda(t'|H_{t_n})\mathrm{d}t'\Big)\right) \\
&= \log\left(\prod_{i=1}^{n} \lambda(t_i|H_{t_{i-1}}) \cdot \exp\Big(-\int_0^{\tau} \lambda(t'|H_{t_n})\mathrm{d}t'\Big)\right) \\
&= \sum_{i=1}^{n} \log\Big(\lambda(t_i|H_{t_{i-1}})\Big) - \int_0^{\tau} \lambda(t'|H_{t'})\mathrm{d}t'. \qquad \square
\end{aligned}$$

C

Appendix 3

C.1 Dataset

The original dataset variables

- CNT – number of wildfires
- BA – aggregated burnt area of wildfires in acres
- lon – longitude coordinate of grid cell center
- lat – latitude coordinate of grid cell center
- $area$ – the proportion of a grid cell that overlaps the continental US (a value in (0, 1], which can be smaller than 1 for grid cells on the boundary of the US territory)
- $month$ – month of observation (integer value between 3 and 9)
- $year$ – year of observation (integer value between 1 and 23, with 1 corresponding to year 1993 and 23 to year 2015)
- $lc_1, \ldots, lc_{18}$ – area proportion of 18 land cover classes in the grid cell (see details below)
- $altiMean$, $altiSD$ – altitude-related variables given as mean and standard devation in the grid cell
- $clim_1, \ldots, clim_{10}$ – monthly means of 10 meteorological variables in the grid cell (see details below)

Land cover variables

The 18 auxiliary variables related to land cover from the COPERNICUS Climate Data Service:

1. cropland rainfed
2. cropland rainfed herbaceous cover
3. mosaic cropland

4. mosaic natural vegetation
5. tree broadleaved deciduous closed to open
6. tree broadleaved deciduous closed
7. tree needleleave evergreen closed to open
8. tree needleleave evergreen closed
9. tree mixed
10. mosaic tree and shrub
11. shrubland
12. grassland
13. sparse vegetation
14. tree cover flooded fresh or brakish water
15. shrub or herbaceous cover flooded
16. urban
17. bare areas
18. water

These are grouped as follows:

1. cropland: 1, 2, 3
2. forest: 4,..., 10
3. low vegetation: 11, 12, 13
4. flodded areas: 14, 15
5. urban areas: 16
6. bare areas: 17
7. water: 18

Meteorological variables

10 of the auxiliary variables are meteorological variables from the COPERNICUS Climate Data Service. They are the monthly means of the following:

1. 10m U-component of wind (the wind speed in Eastern direction) (m/s)
2. 10m V-component of wind (the wind speed in Northern direction) (m/s)
3. Dewpoint temperature (temperature at 2m from ground to which air must be cooled to become saturated with water vapor, such that condensation ensues) (Kelvin)
4. Temperature (at 2m from ground) (Kelvin)
5. Potential evaporation (the amount of evaporation of water that would take place if a sufficient source of water were available) (m)
6. Surface net solar radiation (net flux of shortwave radiation; mostly radiation coming from the sun) (J/m2)
7. Surface net thermal radiation (net flux of longwave radiation; mostly radiation emitted by the surface) (J/m2)
8. Surface pressure (Pa)
9. Evaporation (of water) (m)
10. Precipitation (m)

D

Appendix 4

D.1 Code

Spatio-temporal model

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import statsmodels.graphics.gofplots as statsm
import math
import scipy.optimize as opt
import scipy.integrate as integrate
import scipy
import random


from oneDcase import mean_res_plot
from oneDcase import exceedances
from oneDcase import days_betw_events
from oneDcase import qq_plot
from oneDcase import time_of_events
from oneDcase import define_data


from scipy.optimize import minimize
from scipy.stats import norm

from statsmodels.graphics.tsaplots import plot_acf
from itertools import compress

from matplotlib.pyplot import figure
```

Index & rearrange data

[ ]:
```
# Put all training data into dataframe
df = pd.read_csv('data_train_DF.csv')
```

[ ]:
```
def order_events_in_time(df):
    '''
    The events happening at the same time at different locations should have the␣
 ↪same time stamp
    '''

    year = df['year']
    month = df['month']

    time = np.zeros(len(year))
    time[0] = 0
```

```
    for i in range(len(year)-1):

        y = year[i+1]
        m = month[i+1]

        time[i+1] = time[i] + 12*(y - year[i]) + (m - month[i]) # compute time␣
↪in months from March 1993

    df.drop(columns=['year', 'month'], inplace=True)
    df.insert(1, "time", time, True)
    return df
```

[ ]:
```
def prop_burnt_area(df):
    '''
    The proportion of burnt area in each grid cell:
    (How much is burning)/(How much is there in each grid)
    '''

    ba = df['BA']             # burnt area in acres
    ba_km2 = ba*0.0040469     # burnt area in square km, 0.0040469 km^2 in one␣
↪acre

    area = df['area']         # proportion af square grid that is land
    grid_size = 55*55         # each grid square is 55 by 55 km
    tot_area = area*grid_size

    data = ba_km2/tot_area
    data = np.nan_to_num(data)

    return np.log(data/(1-data)) # reverse sigmoid transformed data
```

[ ]:
```
def aux_var(df):

    ### LAND COVER VARIABLES ###

    lc1 = df['lc1']
    lc2 = df['lc2']
    lc3 = df['lc3']
    cropland = lc1 + lc2 + lc3 # group land cover vars. as cropland

    lc4 = df['lc4']
    lc5 = df['lc5']
    lc6 = df['lc6']
    lc7 = df['lc7']
    lc8 = df['lc8']
```

```
    lc9 = df['lc9']
    lc10 = df['lc10']
    trees = lc4 + lc5 + lc6 + lc7 + lc8 + lc9 + lc10 # group land cover vars. as␣
↪trees

    lc11 = df['lc11']
    lc12 = df['lc12']
    lc13 = df['lc13']
    lowveg = lc11 + lc12 + lc13 # group land cover vars. as low vegetation


    df = df.drop(['lc1', 'lc2', 'lc3', 'lc4', 'lc5', 'lc6', 'lc7', 'lc8', 'lc9',␣
↪'lc10', 'lc11', 'lc12', 'lc13', 'lc14', 'lc15', 'lc16', 'lc17', 'lc18'], axis␣
↪= 1)

    df.insert(5, 'cropland', cropland)
    df.insert(6, 'trees', trees)
    df.insert(7, 'lowveg', lowveg)

    ### METEOROLOGICAL VARIABLES ### not included
    df = df.drop(['clim1', 'clim2', 'clim3', 'clim4', 'clim5', 'clim6', 'clim7',␣
↪'clim8', 'clim9', 'clim10'], axis = 1)

    ### ALTITUDE ### not included
    df = df.drop(['altiMean', 'altiSD'], axis = 1)


    return df
```

[ ]:
```
# create data frame with, index, time, prop. burnt area, aux_var
year = df['year']
test_idx = max(list(compress(range(len( (year < 2016) )),  (year < 2016) )))
df_test = df.take(range(0, test_idx+1))

# create column in df with "real times"
df_test = order_events_in_time(df_test)

# drop CNT and take away all rows with BA=NAN
df_test.pop('CNT')
df_test = df_test.dropna(axis=0, how='any')


# location-x, location-y
second_col = df_test.pop('lon')
third_col = df_test.pop('lat')
df_test.insert(2, 'lon', second_col)
```

```
df_test.insert(3, 'lat', third_col)

# create marks BA/area
prop_ba = prop_burnt_area(df_test)
df_test.pop('BA')
#df_test.pop('area')
df_test.insert(4, 'prop_ba', prop_ba)

# choose vegetation variables
df_test = aux_var(df_test)

# Use df_test (2 years of data) when writing model
df_test = df_test.dropna(axis=0, how='any')


# index data
idx = range(0, len(df_test["prop_ba"]))
df_test.insert(0, "idx", idx, True)
df_test
```

Make mean of aux vars.

[ ]:
```
# unique positions in lon lat coordinates
lon = df_test['lon']
lat = df_test['lat']

lonlat = np.unique(np.transpose(np.array([lon, lat])), axis=0)
```

[ ]:
```
# Arange aux. vars. to be put in dataframe
aux_vars = np.zeros([len(lonlat), 5])

for i,l in enumerate(lonlat):
    coord = lonlat[i]
    new_df = df_test.loc[(df_test['lon'] == coord[0]) & (df_test['lat'] ==␣
 ↪coord[1])]
    m_cropland = np.mean(new_df['cropland'])
    m_trees = np.mean(new_df['trees'])
    m_lowveg = np.mean(new_df['lowveg'])

    aux_vars[i, 0] = coord[0]
    aux_vars[i, 1] = coord[1]
    aux_vars[i, 2] = m_cropland
    aux_vars[i, 3] = m_trees
    aux_vars[i, 4] = m_lowveg
```

[ ]:
```
# create dataframe for aux. vars.
df_av = pd.DataFrame(aux_vars)
df_av.columns = ['lon', 'lat', 'crop', 'trees', 'lowveg']
df_av
```

Threshold select

[ ]:
```
# Plot prop. burnt area in each location
data = df_test['prop_ba'].to_numpy()

figure(figsize=(10, 7), dpi=80)

#%matplotlib inline
plt.plot( df_test['idx'].to_numpy(), data, linewidth=0.7)
plt.xticks([])
plt.ylabel('Prop. burnt area in each location', size=16)
#plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```

[ ]:
```
# Mean residual life plot

# We use proportions
low = 0              # lower limit when creating MRL plot
high = 0.05          # upper limit when creating MRL plot
step = 0.0001        # step length
l = np.arange(low, high, step)

%matplotlib inline
figure(figsize=(10, 7), dpi=80)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.xlabel('$u$', size=18)
plt.ylabel('Mean Excess',size=20)

#MRL plot
mean_exceed = mean_res_plot(data, l)
```

From looking at the MRL plot the threshold is $u = 0.01$.

[ ]:
```
# Create reverse sigmoid transformed threshold
u = np.log((0.01/(1-0.01)))
#u = 0.01

# Get Exceedance over u (the marks)
all_W, W = exceedances(data, u)
```

```
e_idx = (all_W > 0) # index of the exceedances

# ALL DATA
T_all = df_test['time'][e_idx].to_numpy()
X_all = df_test[['idx','lon','lat']][e_idx].values
W_all = W


# select random data points to put in validation dataset
index = df_test['idx'][e_idx]
random.seed(45)
valid_idx = np.sort(random.choices(index.to_numpy(), k=100))

# VALIDATION DATA FRAME
df_valid = df_test[df_test['idx'].isin(valid_idx)]
valid_T = df_valid['time'].to_numpy()
valid_X = df_valid[['idx','lon','lat']].values
valid_W = all_W[valid_idx]



# TRAIN DATA FRAME
d = df_test[e_idx]
train_idx = [i for i in index if i not in valid_idx]
df_train = d[d['idx'].isin(train_idx)]
# Time and index of exceedances
T = df_train['time'].to_numpy()

# Location and index of exceedances
X = df_train[['idx','lon','lat']].values

W = all_W[train_idx]
```

[ ]:
```
# Number of datapoints in dataframe
sum(data>u)
```

Training Data

[ ]:
```
plt.hist(W, bins=50)
plt.xlabel('Size of exceedance over $u$')
plt.ylabel('Frequency')
plt.title('Histogram over exceedances over threshold $u$')
```

Spatio-temporal model Part A

[ ]:
```
def omega_T_est(theta, u, w):
    '''
    Temporal trigger density
    '''

    # parameters
    gamma = theta[1]
    psi = theta[2]
    beta = theta[3]

    return psi * np.exp(beta * w) / (u + gamma)
```

[ ]:
```
def omega_X_est(theta, x, xj):
    '''
    Spatial trigger density, joint normal distribution
    '''

    # parameters
    lam1= theta[4]
    lam2 = theta[5]

    # Positive definite matrix Sigma in joint normal dist
    # Sigma = sigma(theta)

    # Inverse of Sigma
    inv_Sigma = inv_sigma(theta)

    return 1/(2*np.pi*np.sqrt( lam1*lam2 ))* np.exp(-1/2 * np.dot( np.dot(␣
 ↪(x-xj), inv_Sigma ) ,(x-xj)))
```

[ ]:
```
def sigma(theta):
    '''
    The positive definite matrix Sigma in the spatial trigger densuty (joint␣
 ↪normal distribution)
    '''

    # parameters
    lam1 = theta[4]
    lam2 = theta[5]
    eta = theta[6]

    # Sigma = transpose(P)LP
    P = np.array([[np.cos(eta), -np.sin(eta)], [np.sin(eta), np.cos(eta)]])
    L = np.array([[lam1, 0], [0, lam2]])
```

```
    Sigma = np.dot(np.dot(np.transpose(P), L) , P )

    return Sigma
```

[ ]:
```
def inv_sigma(theta):
    '''
    The inverse of the positive definite matrix Sigma
    '''

    # parameters
    lam1 = theta[4]
    lam2 = theta[5]
    eta = -theta[6]

    # inv_Sigma = transpose(P)L^(-1)P
    P = np.array([[np.cos(eta), -np.sin(eta)], [np.sin(eta), np.cos(eta)]])
    # invert L
    L = np.array([[1/lam1, 0], [0, 1/lam2]])

    inv_Sigma = np.dot(np.dot(np.transpose(P), L) , P )

    return inv_Sigma
```

[ ]:
```
def mu_func(theta, x, df):
    '''
    The background intensity, consisting of the aux. vars. cropland, trees and␣
 ↪low vegetation
    '''

    offset = theta[7]

    Dx = 0.5
    Dy = 0.5

    if len(x) == 3:
        a_vars = df_av.loc[(df_av['lon'] == x[1]) & (df_av['lat'] == x[2])] #␣
 ↪take out vars. for location x
    else:
        a_vars = df_av.loc[(df_av['lon'] == x[0]) & (df_av['lat'] == x[1])]


    a_vars = sum(np.array(a_vars[['crop','trees','lowveg']]))   # remove␣
 ↪coordinates

    alpha = np.zeros(3)   # length = number of auxiliary variables

    for i,a in enumerate(alpha): # set parameters
```

```
        alpha[i] = theta[i+8]


    mu = sum(alpha*a_vars) + offset

                            # exp(mu), dont want negative background rate but dont␣
↪want to bound the parameters
                            # to individually be positive

    return  np.exp(mu)/(len(lonlat)*Dx*Dy)
```

[ ]:
```
def int_mu_over_x(theta, df):
    '''
    The background intensity mu(x) integrated over all values x
    '''

    sum_mu = 0

    # grid cell sides
    Dx = 0.5
    Dy = 0.5

    for i,l in enumerate(lonlat):

        m = mu_func(theta, l, df) # background intensity at location x, mu(x)

        sum_mu += Dx*Dy * m       # background intensity integrated over the␣
↪grid cell area

    return sum_mu
```

[ ]:
```
# NOT USED WITH TRIGGER PROBS
def lam_est(theta, t, W, T, X, df):
    """
    The conditional intensity function
    t: time we are at right know, unique event times
    T: events times in history
    X: locations of events in history
    W: mark sizes of events in history
    df: dataframe of auxiliary variables, used to compute background intensity
    """

    l = 0
    idx = (T == t)     # index of events occurring at time t
    x = X[idx][:]      # all locations of events occurring at time t
```

```
    for k in range(len(x)):    # sum over all events happening at time t

        l +=  mu_func(theta, x[k], df)    # background intensity

        for j in range(len(T)):
            if t <= T[j]:    # only sum over previous events
                break
            elif t > T[j]:
                # (temporal trigger density)*(spatial trigger density) from␣
↪previous events
                l += omega_T_est(theta, t-T[j], W[j]) * omega_X_est(theta, x[k],␣
↪X[j])




    return l
```

EM algorithm

[ ]:
```
def trigger_prob(theta, T, X, W, df):
    '''
    Compute trigger and background probabilities to use in the EM-algorithm
    '''

    p = np.array([0, 0, 0])
    minus_P = np.zeros(len(T))    # background probabilities

    for i,t in enumerate(T):

        P = np.zeros([len(T), 3])
        count = 0
        mu = mu_func(theta, X[i][1:3], df)

        G = 0
        for j,_ in enumerate(T):

            if T[j] >= t:
                break
            else:

                g = omega_T_est(theta, T[i]-T[j], W[j]) * omega_X_est(theta,␣
↪X[i][1:3], X[j][1:3])
                G += g
```

```
                # store efficiently, intensity in column 2
                P[count][0] = i
                P[count][1] = j
                P[count][2] = g #    numerator
                count += 1


        # denominator
        lam = mu + G
        P[:,2] = P[:,2]/lam                # trigger probability

        idx = (P[:,2] > 10**(-6))          # filter out small probabilities

        minus_P[i] = 1 - sum(P[idx, 2])    # compute background probabilities
        p = np.vstack((p, P[idx, :]))      # stack trigger probabilities


    p = np.delete(p, (0), axis=0)


    return p, minus_P
```

[ ]:
```
def neg_loglike_A(theta, W, T, X, df, prob, minus_prob):
    '''
    Part A of the log-likelihood, to be maximized (hence minimze the negative␣
↪log-likelihood)

    theta: parameters
    W: size of previous events
    T: time of previous events
    X: location of previous events
    df: dataframe with auxiliary variabkes, used to compute background intensity
    prob: trigger probabilities
    minus_prob: background probabilities

    '''

    # parameters
    gamma = theta[1]
    psi = theta[2]
    beta = theta[3]

    lam1 = theta[4]
    lam2 = theta[5]
    eta = theta[6]

    Wb = np.array(W)
```

```
    # integral I ################################################
    mu_for_int = int_mu_over_x(theta, df)     # background intensity integrated␣
↪over space
    I = mu_for_int * T[0]

    uniq_T = np.unique(T)

    for i in range(len(uniq_T) - 1):

        I += mu_for_int * (uniq_T[i + 1] - uniq_T[i]) # background integrated␣
↪over time

        tj = min(np.where( (T==uniq_T[i]))[0])

        # trigger denisty integarted over time and space
        I  += sum( psi * np.exp(beta * Wb[:tj]) * (np.log(uniq_T[i + 1] - T[:tj]␣
↪+ gamma)
                                                   - np.log(uniq_T[i] - T[:tj] +␣
↪gamma)) )



    # sum over intensities ######################################
    # use trigger probabilities & background probabilities to make more efficient

    ll = 0


    for i,x in enumerate(X):
        ll += minus_prob[i] * np.log(mu_func(theta, x, df)) # background
        idx = np.where(prob[:,0] == i)[0]

        if len(idx) > 0:
            p = prob[idx]    # trigger probabilities

            t = T[i]         # times
            x = X[i][1:3]    # locations

            pu = p[:, 1].astype(int)

            u = x - X[pu, 1:3]
            v1 = 1/np.sqrt(lam1) * (u[:, 0] * np.cos(eta) - u[:, 1]* np.sin(eta))
            v2 = 1/np.sqrt(lam2) * (u[:, 0] * np.sin(eta) + u[:, 1]* np.cos(eta))
            v = v1**2 + v2**2

            # spatial trigger intensity
```

```
            omega_X = -np.log(2) - np.log(np.pi) -1/2 * (np.log(lam1) + np.
↪log(lam2) ) -1/2 * v


            tu = t - T[pu]
            wu = Wb[pu]

            # temporal trigger intensity
            omega_T = np.log(psi)  + beta * wu - np.log(tu + gamma)

            # trigger prob. * trigger density
            ll += sum( p[:,-1] * ( omega_T + omega_X ))


    ll = ll - I

    return -ll
```

Create a for-loop, in this loop we compute trigger_prob, send that in as extra argument to neg_loglike in the minimize function. Stop the for loop when theta doesn't change that much

[ ]:
```
import warnings
warnings.filterwarnings("ignore")

## Create file to store params and function value in
file = create_file_name()

# Initialise params. to reasonable values
theta = [ 0, 20, 0.5, 0.05, 0.1, 0.1, 1, 0.1, 0, 0, 0]

## EM-alg ##

for i in range(100):
    print('i: ', i)

    prob, minus_prob = trigger_prob(theta, T, X, W, df_av) # move as input

    # set boundaries of parameters
            # mu        gamma       psi      beta.      lam1       lam2         eta␣
↪        offset          a1            a2           a3
    bnds=((0, None), (0, None), (0, None),(0, None),(0, None),(0, None), (0, np.
↪pi), (None, None),(None, None),(None, None),(None, None))

    res_veg = minimize(neg_loglike_A, theta, args=( W, T, X, df_av, prob,␣
↪minus_prob), method='L-BFGS-B', bounds=bnds, options={'disp': True, 'gtol': 0.
↪1, 'ftol': 0.1})
```

```
    ## Save params and function value in file
    list_of_params_funval = np.concatenate(( i, res_veg.x, res_veg.fun),␣
↪axis=None)
    append_list_as_row(file, [list_of_params_funval])

    theta = res_veg.x
    print(res_veg.fun)
    print(theta)
```

i=50, res_50

[ ]:
```
res_veg
```

Likelihood Part B

[ ]:
```
def neg_loglike_B(theta, T, W):
    '''
    Part B of the log-likelihood
    '''

    #### xi < 0
    #a = np.exp(theta[0])
    #xi = -np.exp(theta[1])
    #sigma = (-xi*6 + a)

    #### xi >= 0
    sigma = np.exp(theta[0])
    xi = np.exp(theta[1])


    mu = 0
    ll = 0

    for i in range(len(W)):

        # log-likelihood of GPD
        ll += -np.log(sigma) + (-1 / xi - 1) * np.log( (1 + xi * (W[i] - mu) /␣
↪sigma) )


    return -ll
```

[ ]:
```
# initialize parameter values
theta_start_B = [5 , 5] # sigma, xi

# maximize log-likelihood
```

```
res = minimize(neg_loglike_B, theta_start_B, args=(T, W), method='L-BFGS-B',
               options={'disp': True, 'ftol':2.220446049250313e-09})

print(res.x)
print(-np.exp(res.x))
```

[ ]:
```
# Results
theta = res.x

# Xi >= 0
sigma = np.exp(theta[0])
xi = np.exp(theta[1])

# xi < 0
#a = np.exp(theta[0])
#xi = -np.exp(theta[1])
#sigma = (-xi*6 + a)


xi, sigma
```

[ ]:
```
res
```

Verify part A

Verification of model is performed seperately for part A and part B

[ ]:
```
## Load in optimal pramaters in theta_hat from file
p_df = pd.read_csv('130521-171711.csv')
v = p_df.iloc[[-1]].to_numpy()
theta_hat = v[0][1:-1]
funval = v[0][-1]

# theta_hat = [mu  gamma  psi  beta  lam1  lam2  eta  offset  a1  a2  a3]
theta_hat
```

[ ]:
```
# Create History of process with event times, locations and mark sizes

# Event times
T = df_train['time'].to_numpy()

# Location and index of exceedances
X = df_train[['idx','lon','lat']].values

# Exceedances
W = all_W[train_idx]
```

[ ]:
```
## Predict occurrences of wildfires

I = 0
Pred_at_T = np.zeros(len(np.unique(valid_T)))    # store predicted wildfire times
mu = int_mu_over_x(theta_hat, df_av)    # background intensity integrated over␣
 ↪all locations

# parameters
gamma = theta_hat[1]
psi = theta_hat[2]
beta = theta_hat[3]

# Integrate intensity over each timestep to find all wildfires
# that are predicted to occur in this timestep
for i,t in enumerate(np.unique(valid_T)):

    if i < len(np.unique(valid_T))-1:
        I += mu * (np.unique(valid_T)[i + 1] - np.unique(valid_T)[i])

        sum_over_history = 0
        for j in range(len(T_all)):
                if t <= T_all[j]:
                    break
                elif t > T_all[j]:

                    sum_over_history += (psi * np.exp(beta * W_all[j])
                                         *(np.log(np.unique(valid_T)[i + 1] -␣
 ↪T_all[j] + gamma)
                                           - np.log(np.unique(valid_T)[i] -␣
 ↪T_all[j] + gamma)) )

        I += sum_over_history

        Pred_at_T[i+1] = I    # Predicted wildfires at time i+1

Pred_at_T
```

[ ]:
```
## The actual number of wildfires at all event times

Actual_at_T = np.zeros(len(np.unique(valid_T)))
cum = 0

for i,t in enumerate(np.unique(valid_T)):
    cum = 0
    for j,tj in enumerate(np.unique(T_all)):
        if tj < t:    # events occurring before time t
             cum += sum((T_all == tj))
```

```
    Actual_at_T[i] = cum

Actual_at_T
```

[ ]:
```
# Plot the predicted number of wildfires against the actual number of wildfires␣
 ↪at different times

plt.scatter(Pred_at_T, Actual_at_T, color='blue')
plt.plot(Actual_at_T, Actual_at_T, color='orange')
plt.ylabel('Actual')
plt.xlabel('Predicted')

plt.show()
```

Verify part B

[ ]:
```
### Compute residuals ###

R = np.zeros(0)

#SET OPTIMAL PARAMETERS PREDICTED IN neg_loglike_B
xi, sigma = -1.7522904376784435e-05, 0.03214314273674598

# History of validation dataset
T = valid_T
X = valid_X
W = valid_W


# Compute residuals
for i,w in enumerate(W):

    r = (1/xi) * np.log(1 + (xi * w) / sigma)

    R = np.append(R, r)


### Compute ACF of residuals and squared residuals ###

H = np.linspace(0,25,26)

mu = np.mean(R)         # mean resdiuals
mu2 = np.mean(R**2)     # mean residuals^2

ACF = []
```

```
ACF2 = []


for i,h in enumerate(H):

    pairs = []
    pairs2 = []

    for j,t in enumerate(T):

        for k in range(j, len(T)):
            if np.abs(t-T[k]) > h:
                break


            if T[k]-t == h:     # compute acf
                pairs.append( (R[j]-mu) * (R[k]-mu) )

                pairs2.append( (R[j]**2-mu2) * (R[k]**2-mu2) )



    ACF.append(sum(pairs)/len(R))

    ACF2.append(sum(pairs2)/len(R))

# normalise
ACFnorm = ACF/ACF[0]
ACF2norm = ACF2/ACF2[0]
```

[ ]:
```
##### PLOT ACF #####

# compute conf. int. of ACF
conf_bounds = np.ones(27)*1.96/np.sqrt(len(R))

## ACF
plt.vlines(H, ymin= 0, ymax=ACFnorm, alpha=0.7, linewidth=2, color='blue')␣
 ↪#color='C0')#, width=0.15)
plt.scatter(H, ACFnorm, color='blue')

plt.plot(range(27), conf_bounds, '--', color='gray')
plt.plot(range(27), -conf_bounds, '--', color='gray')
plt.plot(range(27), np.zeros(27), '-', color='blue')

plt.title('Residuals')
plt.xlabel('Lag')
plt.ylabel('ACF')
```

```
plt.show()


## Squared res ACF
plt.vlines(H, ymin= 0, ymax=ACF2norm, alpha=0.7, linewidth=2, color='blue')␣
 ↪#color='C0')#, width=0.15)
plt.scatter(H, ACF2norm, color='blue')


plt.plot(range(27), conf_bounds, '--', color='gray')
plt.plot(range(27), -conf_bounds, '--', color='gray')
plt.plot(range(27), np.zeros(27), '-', color='blue')

plt.title('Squared residuals')
plt.xlabel('Lag')
plt.ylabel('ACF')
plt.show()
```

[ ]:
```
## Probability plot of residuals

statsm.qqplot(R, dist=stats.distributions.expon, line='45')
#plt.title('Exponential probability plot of residuals')
plt.show()
```

[ ]:
```
from datetime import datetime
import csv


def create_file_name():
    # Hämtar datum o klock-slag
    now = datetime.now()
    now_string = now.strftime("%d%m%y-%H%M%S")
    # Lägg till mapp-namn om du vill samla körnignarna någonstans
    name_of_folder = ""

    file_name = name_of_folder + now_string + ".csv"

    return file_name


def append_list_as_row(file_name, list_of_elem):
    # Open file in append mode
    with open(file_name, 'a+', newline='') as write_obj:
        # Create a writer object from csv module
        csv_writer = csv.writer(write_obj)
        # Add contents of list as last row in the csv file
        csv_writer.writerow(*list_of_elem)
```

```
def load_from_file(path_to_file):
    # Laddar in data från en fil
    return np.loadtxt(path_to_file, delimiter=",")


def write_loop_to_file():
    # Test funktion för att spara en random vektor till en fil

    path_to_file = create_file_name()
    for i in range(10):
        # Generera random data
        vec = np.random.rand(1, 10)
        append_list_as_row(path_to_file, vec)

    return path_to_file

def main():
    # Bara testar att skapa en vektor, och sen ladda in den
    path_to_file = write_loop_to_file()

    data = load_from_file(path_to_file)
    print(data)
```

[ ]:

Visualisation of results

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stats
import statsmodels.graphics.gofplots as statsm
import math
import scipy.optimize as opt
import scipy.integrate as integrate
import scipy
import random


from oneDcase import mean_res_plot
from oneDcase import exceedances
from oneDcase import days_betw_events
from oneDcase import qq_plot
from oneDcase import time_of_events
from oneDcase import define_data


from scipy.optimize import minimize
from scipy.stats import norm

from statsmodels.graphics.tsaplots import plot_acf
from itertools import compress

from matplotlib.pyplot import figure
```

[ ]:
```
df = pd.read_csv('data_train_DF.csv')
df
```

[ ]:
```
def order_events_in_time(df):
    '''
    The events happening at the same time at different locations should have the␣
 ↪same time stamp
    '''

    year = df['year']
    month = df['month']

    time = np.zeros(len(year))
    time[0] = 0

    for i in range(len(year)-1):
```

```
        y = year[i+1]
        m = month[i+1]

        time[i+1] = time[i] + 12*(y - year[i]) + (m - month[i]) # compute time␣
 ↪in months from March 1993

    df.drop(columns=['year', 'month'], inplace=True)
    df.insert(1, "time", time, True)
    return df



def prop_burnt_area(df):
    '''
    The proportion of burnt area in each grid cell:
    (How much is burning)/(How much is there in each grid)
    '''

    ba = df['BA']             # burnt area in acres
    ba_km2 = ba*0.0040469     # burnt area in square km, 0.0040469 km^2 in one␣
 ↪acre

    area = df['area']         # proportion af square grid that is land
    grid_size = 55*55         # each grid square is 55 by 55 km
    tot_area = area*grid_size

    data = ba_km2/tot_area
    data = np.nan_to_num(data)

    return np.log(data/(1-data)) # reverse sigmoid transformed data



def aux_var(df):

    ### LAND COVER VARIABLES ###

    lc1 = df['lc1']
    lc2 = df['lc2']
    lc3 = df['lc3']
    cropland = lc1 + lc2 + lc3 # group land cover vars. as cropland

    lc4 = df['lc4']
    lc5 = df['lc5']
    lc6 = df['lc6']
    lc7 = df['lc7']
    lc8 = df['lc8']
```

```
    lc9 = df['lc9']
    lc10 = df['lc10']
    trees = lc4 + lc5 + lc6 + lc7 + lc8 + lc9 + lc10 # group land cover vars. as␣
 ↪trees

    lc11 = df['lc11']
    lc12 = df['lc12']
    lc13 = df['lc13']
    lowveg = lc11 + lc12 + lc13 # group land cover vars. as low vegetation


    df = df.drop(['lc1', 'lc2', 'lc3', 'lc4', 'lc5', 'lc6', 'lc7', 'lc8', 'lc9',␣
 ↪'lc10', 'lc11', 'lc12', 'lc13', 'lc14', 'lc15', 'lc16', 'lc17', 'lc18'], axis␣
 ↪= 1)

    df.insert(5, 'cropland', cropland)
    df.insert(6, 'trees', trees)
    df.insert(7, 'lowveg', lowveg)

    ### METEOROLOGICAL VARIABLES ### not included
    df = df.drop(['clim1', 'clim2', 'clim3', 'clim4', 'clim5', 'clim6', 'clim7',␣
 ↪'clim8', 'clim9', 'clim10'], axis = 1)

    ### ALTITUDE ### not included
    df = df.drop(['altiMean', 'altiSD'], axis = 1)


    return df



# create data frame with, index, time, prop. burnt area, aux_var
year = df['year']
test_idx = max(list(compress(range(len( (year < 2016) )),  (year < 2016) )))
df_test = df.take(range(0, test_idx+1))

# create column in df with "real times"
df_test = order_events_in_time(df_test)

# drop CNT and take away all rows with BA=NAN
df_test.pop('CNT')
df_test = df_test.dropna(axis=0, how='any')


# location-x, location-y
second_col = df_test.pop('lon')
third_col = df_test.pop('lat')
```

```
df_test.insert(2, 'lon', second_col)
df_test.insert(3, 'lat', third_col)

# create marks BA/area
prop_ba = prop_burnt_area(df_test)
df_test.pop('BA')
#df_test.pop('area')
df_test.insert(4, 'prop_ba', prop_ba)

# choose vegetation variables
df_test = aux_var(df_test)

# Use df_test (2 years of data) when writing model
df_test = df_test.dropna(axis=0, how='any')


# index data
idx = range(0, len(df_test["prop_ba"]))
df_test.insert(0, "idx", idx, True)
df_test
```

Make mean of aux vars.

[ ]:
```
# unique position
lon = df_test['lon']
lat = df_test['lat']

lonlat = np.unique(np.transpose(np.array([lon, lat])), axis=0)
```

[ ]:
```
# Arange aux. vars. to be put in dataframe
aux_vars = np.zeros([len(lonlat), 5])

for i,l in enumerate(lonlat):
    coord = lonlat[i]
    new_df = df_test.loc[(df_test['lon'] == coord[0]) & (df_test['lat'] ==␣
 ↪coord[1])]
    m_cropland = np.mean(new_df['cropland'])
    m_trees = np.mean(new_df['trees'])
    m_lowveg = np.mean(new_df['lowveg'])

    aux_vars[i, 0] = coord[0]
    aux_vars[i, 1] = coord[1]
    aux_vars[i, 2] = m_cropland
    aux_vars[i, 3] = m_trees
    aux_vars[i, 4] = m_lowveg
```

[ ]:
```
# create dataframe for aux. vars.
df_av = pd.DataFrame(aux_vars)
df_av.columns = ['lon', 'lat', 'crop', 'trees', 'lowveg']
df_av
```

Split training and validation data

[ ]:
```
# Create reverse sigmoid transformed threshold
u = np.log((0.01/(1-0.01)))
#u = 0.01

# Get Exceedance over u (the marks)
all_W, W = exceedances(data, u)

e_idx = (all_W > 0) # index of the exceedances

# ALL DATA
T_all = df_test['time'][e_idx].to_numpy()
X_all = df_test[['idx','lon','lat']][e_idx].values
W_all = W


# select random data points to put in validation dataset
index = df_test['idx'][e_idx]
random.seed(45)
valid_idx = np.sort(random.choices(index.to_numpy(), k=100))

# VALIDATION DATA FRAME
df_valid = df_test[df_test['idx'].isin(valid_idx)]
valid_T = df_valid['time'].to_numpy()
valid_X = df_valid[['idx','lon','lat']].values
valid_W = all_W[valid_idx]



# TRAIN DATA FRAME
d = df_test[e_idx]
train_idx = [i for i in index if i not in valid_idx]
df_train = d[d['idx'].isin(train_idx)]
# Time and index of exceedances
T = df_train['time'].to_numpy()

# Location and index of exceedances
X = df_train[['idx','lon','lat']].values

W = all_W[train_idx]
```

[ ]:
```
a = df_test['area'][0:len(lonlat)]

for i,l in enumerate(lonlat):
    coord = lonlat[i]
    new_df = df_test.loc[(df_test['lon'] == coord[0]) & (df_test['lat'] ==␣
 ↪coord[1])]
```

Training Data

Spatio-temporal model Part A

[ ]:
```
def omega_T_est(theta, u, w):
    '''
    Temporal trigger density
    '''

    # parameters
    gamma = theta[1]
    psi = theta[2]
    beta = theta[3]

    return psi * np.exp(beta * w) / (u + gamma)
```

[ ]:
```
def omega_X_est(theta, x, xj):
    '''
    Spatial trigger density, joint normal distribution
    '''

    # parameters
    lam1= theta[4]
    lam2 = theta[5]

    # Positive definite matrix Sigma in joint normal dist
    # Sigma = sigma(theta)

    # Inverse of Sigma
    inv_Sigma = inv_sigma(theta)

    return 1/(2*np.pi*np.sqrt( lam1*lam2 ))* np.exp(-1/2 * np.dot( np.dot(␣
 ↪(x-xj), inv_Sigma ) ,(x-xj)))
```

[ ]:
```
def mu_func(theta, x, df):
    '''
    The background intensity, consisting of the aux. vars. cropland, trees and␣
 ↪low vegetation
```

```
    '''

    offset = theta[7]

    Dx = 0.5
    Dy = 0.5

    if len(x) == 3:
        a_vars = df_av.loc[(df_av['lon'] == x[1]) & (df_av['lat'] == x[2])] #␣
↪take out vars. for location x
    else:
        a_vars = df_av.loc[(df_av['lon'] == x[0]) & (df_av['lat'] == x[1])]


    a_vars = sum(np.array(a_vars[['crop','trees','lowveg']]))   # remove␣
↪coordinates

    alpha = np.zeros(3)   # length = number of auxiliary variables

    for i,a in enumerate(alpha): # set parameters
        alpha[i] = theta[i+8]


    mu = sum(alpha*a_vars) + offset

                           # exp(mu), dont want negative background rate but dont␣
↪want to bound the parameters
                           # to individually be positive

    return  np.exp(mu)/(len(lonlat)*Dx*Dy)




def int_mu_over_x(theta, df):
    '''
    The background intensity mu(x) integrated over all values x
    '''

    sum_mu = 0

    # grid cell sides
    Dx = 0.5
    Dy = 0.5

    for i,l in enumerate(lonlat):
```

```
        m = mu_func(theta, l, df) # background intensity at location x, mu(x)

        sum_mu += Dx*Dy * m       # background intensity integrated over the␣
↪grid cell area

    return sum_mu
```

[ ]:
```
def inv_sigma(theta):
    '''
    The inverse of the positive definite matrix Sigma
    '''

    # parameters
    lam1 = theta[4]
    lam2 = theta[5]
    eta = -theta[6]

    # inv_Sigma = transpose(P)L^(-1)P
    P = np.array([[np.cos(eta), -np.sin(eta)], [np.sin(eta), np.cos(eta)]])
    # invert L
    L = np.array([[1/lam1, 0], [0, 1/lam2]])

    inv_Sigma = np.dot(np.dot(np.transpose(P), L) , P )

    return inv_Sigma
```

RESULTS:

Bakground intensities

[ ]:
```
df_av
```

[ ]:
```
# side of grid cells
Dx = 0.5
Dy = 0.5

def f_mu(x):
    '''
    Background intensity
    '''

    background = mu_func(theta_hat, x, df_av)

    return background
```

[ ]:
```
#%% USA Border

## Load file containing points delimiting US border
usa=pd.read_csv("usa_border.csv")

## Select points giving corresonponding to the main land
usa=usa[['long','lat','region']]
usa=usa[usa['region']=='main']

## Convert to numpy array
usa=usa[['long','lat']].to_numpy()

## Plot
plt.plot(usa[:,0], usa[:,1],color="black",linewidth=1)
plt.show()
```

[ ]:
```
## Load all events
dat=pd.read_csv("dat_BA.csv")

## Unique coordinates of events
points=dat[['lon','lat']]
points=points.drop_duplicates().to_numpy()


## Convert to numpy
dat=dat.to_numpy()


#%%

## Grid step size
dg=0.5

## Grid extent
extent = [points[:,0].min()-2*dg, points[:,0].max()+2*dg,
          points[:,1].min()-2*dg, points[:,1].max()+2*dg]

## Lon/Lat coordinates defining the grid
lon = np.arange(extent[0], extent[1], dg)
lat = np.arange(extent[2],  extent[3], dg)

## Create grid
lon2d, lat2d = np.meshgrid(lon, lat)
gridShape=lon2d.shape

## Array of coordinates containing the coordinates of each grid point
lon2d, lat2d = lon2d.flatten(),lat2d.flatten()
```

```
Ngrid=lon2d.shape[0]

## Create Tree to speed up search
lonlat2 = np.column_stack((lon2d,lat2d))
tree = scipy.spatial.cKDTree(lonlat2)


#%%

## For each point in "points", search the nearest point in the grid
## Return distannce to grid point and index
dist,idx = tree.query(points,k=1)



I = np.zeros(len(lonlat2))
I_with_mu = np.zeros(len(lonlat2))

## Choose time to estimate intensity at
j = 39
t1 = np.unique(T)[j]
t2 = np.unique(T)[j+1]




## Initialize grid of nan
gridDat_mu=np.zeros(Ngrid)+np.nan
## Fill the US points with values
gridDat_mu[idx]=np.array([f_mu(x) for x in points])
```

[ ]:
```
# Choose figure size
figure(figsize=(15, 7), dpi=80)

## Plot Grid
plt.plot(usa[:,0], usa[:,1],color="white",linewidth=3)
plt.plot(usa[:,0], usa[:,1],color="black",linewidth=1.5)

plt.imshow(gridDat_mu.reshape(gridShape),
           origin="lower",
           extent=extent,
           aspect=1.4,
           cmap="RdBu_r")

#plt.clim(0.0,0.0015)
plt.colorbar()
plt.show()
```

Trigger intensities (spatial + temporal but the pattern is spatial)

[ ]:
```
def f_omega(x):
    '''
    Trigger density
    '''

    # parameters
    gamma = theta_hat[1]
    psi = theta_hat[2]
    beta = theta_hat[3]

    background = 0

    sum_over_history = 0
    for j in range(len(T)):
        if t2 <= T[j]:    # only sum over previous events
            break
        elif t2 > T[j]:
            # add triggers from previous events
            sum_over_history += omega_X_est(theta_hat, x, X[j][1:3]) *␣
↪omega_T_est(theta_hat,t2-T[j], W[j])

    I = background + sum_over_history
    return I
```

[ ]:
```
## Load all events
dat=pd.read_csv("dat_BA.csv")

## Unique coordinates of events
points=dat[['lon','lat']]
points=points.drop_duplicates().to_numpy()


## Convert to numpy
dat=dat.to_numpy()


#%%

## Grid step size
dg=0.5

## Grid extent
extent = [points[:,0].min()-2*dg, points[:,0].max()+2*dg,
          points[:,1].min()-2*dg, points[:,1].max()+2*dg]
```

```
## Lon/Lat coordinates defining the grid
lon = np.arange(extent[0], extent[1], dg)
lat = np.arange(extent[2],  extent[3], dg)

## Create grid
lon2d, lat2d = np.meshgrid(lon, lat)
gridShape=lon2d.shape

## Array of coordinates containing the coordinates of each grid point
lon2d, lat2d = lon2d.flatten(),lat2d.flatten()
Ngrid=lon2d.shape[0]

## Create Tree to speed up search
lonlat = np.column_stack((lon2d,lat2d))
tree = scipy.spatial.cKDTree(lonlat)


#%%

## For each point in "points", search the nearest point in the grid
## Return distannce to grid point and index
dist,idx = tree.query(points,k=1)



#I = np.zeros(len(lonlat))
#I_with_mu = np.zeros(len(lonlat))

# Choose time to estimate intensity at
j = 49
t1 = np.unique(T)[j]
t2 = np.unique(T)[j+1]


## Initialize grid of nan
gridDat_omega=np.zeros(Ngrid)+np.nan
## Fill the US points with values
gridDat_omega[idx]=np.array([f_omega(x) for x in points])
```

[ ]:
```
# Chose figure size
figure(figsize=(15, 7), dpi=80)

## Plot Grid
plt.plot(usa[:,0], usa[:,1],color="black",linewidth=1.5)
plt.imshow(gridDat_omega.reshape(gridShape),
           origin="lower",
           extent=extent,
```

```
            aspect=1.4,
            cmap="RdBu_r")

plt.clim(0,0.06)
plt.colorbar()
plt.show()
```

Total intensity

[ ]:
```
def f_all(x):
    '''
    The full intensity
    '''

    # parameters
    gamma = theta_hat[1]
    psi = theta_hat[2]
    beta = theta_hat[3]
    sum_over_history = 0

    background = mu_func(theta_hat, x, df_av)    # background intensity

    for j in range(len(T)):
        if t2 <= T[j]:     # only sum over previous events
            break
        elif t2 > T[j]:
            # add triggers for all previous events
            sum_over_history += omega_X_est(theta_hat, x, X[j][1:3]) *␣
↪omega_T_est(theta_hat,t2-T[j], W[j])

    I = background + sum_over_history
    return I
```

[ ]:
```
## Load all events
dat=pd.read_csv("dat_BA.csv")

## Unique coordinates of events
points=dat[['lon','lat']]
points=points.drop_duplicates().to_numpy()


## Convert to numpy
dat=dat.to_numpy()


#%%
```

```
## Grid step size
dg=0.5

## Grid extent
extent = [points[:,0].min()-2*dg, points[:,0].max()+2*dg,
          points[:,1].min()-2*dg, points[:,1].max()+2*dg]

## Lon/Lat coordinates defining the grid
lon = np.arange(extent[0], extent[1], dg)
lat = np.arange(extent[2],  extent[3], dg)

## Create grid
lon2d, lat2d = np.meshgrid(lon, lat)
gridShape=lon2d.shape

## Array of coordinates containing the coordinates of each grid point
lon2d, lat2d = lon2d.flatten(),lat2d.flatten()
Ngrid=lon2d.shape[0]

## Create Tree to speed up search
lonlat = np.column_stack((lon2d,lat2d))
tree = scipy.spatial.cKDTree(lonlat)


#%%

## For each point in "points", search the nearest point in the grid
## Return distannce to grid point and index
dist,idx = tree.query(points,k=1)

## Choose time to estimate intensity at
i= 49
t1 = np.unique(T)[i]
t2 = np.unique(T)[i+1]



## Initialize grid of nan
gridDat_all=np.zeros(Ngrid)+np.nan
## Fill the US points with values
gridDat_all[idx]=np.array([f_all(x) for x in points])
```

[ ]:
```
# Choose figure size
figure(figsize=(15, 7), dpi=80)

## Plot Grid
```

```
plt.plot(usa[:,0], usa[:,1],color="black",linewidth=1.5)
plt.imshow(gridDat_all.reshape(gridShape),
           origin="lower",
           extent=extent,
           aspect=1.4,
           cmap="RdBu_r")

plt.clim(0,0.06)
plt.colorbar()
plt.show()
```

Risk measure

Prob $Z_{t+1} > u$

[ ]:
```
I = 0
I_with_mu = 0


def prob(x):
    '''
    Probability of wildfire occurring larger than threshold u
    '''

    # parameters
    gamma = theta_hat[1]
    psi = theta_hat[2]
    beta = theta_hat[3]

    background = mu_func(theta_hat, x, df_av) * (t2-t1)  # backgroudnd intensity

    sum_over_history = 0
    for j in range(len(T)-1):
        if t2 <= T[j]:
            break
        elif t2 > T[j]:

            # triggers from all previous events
            sum_over_history +=  psi * np.exp(beta * W[j]) * (np.log(t2 - T[j] +␣
↪gamma)
                                    - np.log(t1 - T[j] + gamma)) *␣
↪omega_X_est(theta_hat, x, X[j][1:3])

    I = sum_over_history

    # add background intensity and trigger densities
```

```
    I_with_mu = background + sum_over_history

    return I_with_mu
```

[ ]:
```
## Load all events
dat=pd.read_csv("dat_BA.csv")

## Unique coordinates of events
points=dat[['lon','lat']]
points=points.drop_duplicates().to_numpy()


## Convert to numpy
dat=dat.to_numpy()


#%%

## Grid step size
dg=0.5

## Grid extent
extent = [points[:,0].min()-2*dg, points[:,0].max()+2*dg,
          points[:,1].min()-2*dg, points[:,1].max()+2*dg]

## Lon/Lat coordinates defining the grid
lon = np.arange(extent[0], extent[1], dg)
lat = np.arange(extent[2],  extent[3], dg)

## Create grid
lon2d, lat2d = np.meshgrid(lon, lat)
gridShape=lon2d.shape

## Array of coordinates containing the coordinates of each grid point
lon2d, lat2d = lon2d.flatten(),lat2d.flatten()
Ngrid=lon2d.shape[0]

## Create Tree to speed up search
lonlat2 = np.column_stack((lon2d,lat2d))
tree = scipy.spatial.cKDTree(lonlat2)


#%%

## For each point in "points", search the nearest point in the grid
## Return distannce to grid point and index
dist,idx = tree.query(points,k=1)
```

```
# Choose time to estimate risk at
j = 151
t1 = np.unique(T)[j]
t2 = np.unique(T)[j+1]




## Initialize grid of nan
gridDat=np.zeros(Ngrid)+np.nan
## Fill the US points with values
gridDat[idx]=np.array([prob(x) for x in points])
```

[ ]:
```
# Compute probability in each location
c = 1-np.exp(-gridDat)
```

[ ]:
```
# Choose figure size
figure(figsize=(15, 7), dpi=80)

## Plot Grid
plt.plot(usa[:,0], usa[:,1],color="black",linewidth=3)
plt.plot(usa[:,0], usa[:,1],color="white",linewidth=1.5)

plt.imshow(c.reshape(gridShape),
           origin="lower",
           extent=extent,
           aspect=1.4,
           cmap="hot")

plt.clim(0, 0.1)
plt.axis('off')
plt.show()
```

Prob $Z_{t+1} > z$

[ ]:
```
xi, sigma = 4.311206401720284e-05, 0.9623093445252835    # Set to optimal␣
 ↪parameters
alpha = 1 - min(c[~np.isnan(c)])
#u = 0.01
u = np.log((0.01/(1-0.01))) # reverse sigma transformed threshold
```

[ ]:
```
# Compute VaR in all locations
z = sigma/xi *(  ((1-alpha)/c)**(-xi) - 1  )+ u
z = 1/(1+np.exp(-z))
```

[ ]:
```
## Prob. that we get a fire of size prop_fire in a grid cell

prop_fire = 0.02    # proportion of burnt area
p = ( ( 1 + xi*(prop_fire-u)/sigma)**(-1/xi) *c)    # probability
```

[ ]:
```
# Choose figure size
figure(figsize=(15, 7), dpi=80)

## Plot Grid
plt.plot(usa[:,0], usa[:,1],color="black",linewidth=3)
plt.plot(usa[:,0], usa[:,1],color="white",linewidth=1.5)
plt.imshow(z.reshape(gridShape),
           origin="lower",
           extent=extent,
           aspect=1.4,
           cmap="RdBu_r")
#plt.clim(0, 1)
plt.colorbar()
plt.show()
```

Actual fires at time i

[ ]:
```
def actual(x):
    '''
    Returns I=1 if input parameter x has a wildfires at time i
    '''

    idx = (T ==np.unique(T)[i])
    Z = X[idx,1:3]

    I=0
    if (x == Z).all(1).any():
        I=1

    return I
```

[ ]:
```
## For each point in "points", search the nearest point in the grid
## Return distannce to grid point and index
dist,idx = tree.query(points,k=1)


# Choose a time point
```

```
i = 152
print(np.unique(T)[i])

## Initialize grid of nan
gridDat_actual=np.zeros(Ngrid)+np.nan
## Fill the US points with values
gridDat_actual[idx]=np.array([actual(x) for x in points])
```

[ ]:
```
# Choose a figure size
figure(figsize=(15, 7), dpi=80)

## Plot Grid
plt.plot(usa[:,0], usa[:,1],color="white",linewidth=1.5)
plt.imshow(gridDat_actual.reshape(gridShape),
           origin="lower",
           extent=extent,
           aspect=1.4,
           cmap="hot")

plt.clim(0, 2)
plt.colorbar()
plt.show()
```

Functions:

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib
matplotlib.use("TkAgg")
from matplotlib import pyplot as plt
import scipy.stats as stats


def mean_res_plot(data, thresholds):
    """
    Mean residual life plot

    :param data: (np.ndarray) the data to be plotted
    :param thresholds: (np.ndarray) the thresholds to investigate
    """
    mean_res = np.zeros([len(thresholds), 1])

    i = 0
    for u in thresholds:
        # add the exceedance over u to event
        mask = (data - u > 0)
        event = data[mask] - u
        # compute the mean of the exceedances
        mean_res[i] = np.mean(event)
        i += 1

    # Plot the MRL
    plt.plot(thresholds, mean_res, linewidth=4)
    plt.grid()
    plt.xlabel("$u$", size=18)
    plt.ylabel("Mean Excess", size=18)
    plt.xticks(fontsize=16)
    plt.yticks(fontsize=16)

    #plt.title("Mean Residual Life Plot")
    plt.show()



def exceedances(data, u):
    """
    Plots the exceedances (W) over threshold u and returns a list of only the␣
 ↪exceedances.
    :param data: (np.ndarray)
    :param u: (float) threshold
```

```
    """
    # if the event exceeds u set to data-u, otherwise set to zero
    exceedances = np.zeros(len(data))
    # only the exceedances at days where data-u > 0
    only_exceedances = []

    for i in range(len(data)):
        if data[i] - u <= 0:
            exceedances[i] = 0
        else:
            exceedances[i] = data[i] - u
            only_exceedances.append(exceedances[i])    # add only the exceedances

    percent = (data - u > 0)
    print("Approximately 10% of the data should be exceedances: ", sum(percent) /
 ↪ len(data) * 100)
    #plt.plot(exceedances, linewidth=0.3)
    #plt.xlabel("Time")
    #plt.ylabel("Exceedances over threshold")
    #plt.title("u=2.5")
    #plt.show()
    return exceedances, only_exceedances


def days_betw_events(data, u):
    """
    Distribution of months BETWEEN extreme events
    :param data: (np.ndarray) negative log returns
    :param u: (float) chosen threshold
    :return: list of distribution of days between events
    """

    # find exceedances in data
    mask = (data - u > 0)
    days_dist = []

    count = 0
    for i in range(len(data)):

        if mask[i] == True:
            days_dist.append(count)     # save months between two events
            count = 0     # restart count
        else:
            count += 1     # increase the number of months between events
    return days_dist
```

```
def time_of_events(data, u):
    """
    Computes the TIMES AT which events occur (T), that is when (data - u) > 0.
    :param data: the marked point process values
    :param u: threshold value
    :return: List of times of exceedances
    """

    mask = (data - u > 0)    # find exceedances in data
    T = np.array(range(len(data)))
    T_events = T[mask]    # save the times of exceedances
    return T_events

def qq_plot(data, dist, rate):
    """
    QQ-plot of the data
    :param data: (np.ndarray)
    :param dist: e.g. 'expon'
    :param rate: mean of distribution
    """
    stats.probplot(data, sparams = (rate), dist=dist, plot=plt)
    plt.show()

def define_data(name_of_set):
    """
    Reads the bayn negative log returns data into a dataframe, multiplies this␣
 ↪by 100 and sets threshold value.
    :return: data (np.array), threshold (float)
    """
    if name_of_set == 'dji':
        df = pd.read_csv('dji_log_returns.csv')
        name = 'Djind index'
        u = 1.4
    elif name_of_set == 'bayn':
        df = pd.read_csv('bayn_log_returns.csv')
        name = 'Bayn data'
        u = 2.5

    data = pd.DataFrame.to_numpy(df['log_ret']) *100


    print('Data: ', name)
    print('Threshold: u =', u)
    return data, u
```