TIPE: Modélisation et Prévision du risque de cambriolage en ville

MUNIER Maxime - Candidat 38530

Épreuve de TIPE

Session 2023

Plan de l'exposé

Plan de l'exposé

Présentation théorique des Processus de Hawkes

Définition d'un processus de Hawkes à une dimension

Processus de Hawkes

Un processus de Hawkes à une dimension est un processus stochastique ponctuel qui modélise une série d'événements unidimensionnels dans le temps. Il est défini par sa fonction d'intensité conditionnelle, qui décrit la probabilité d'occurrence d'un événement à un instant donné, en fonction de l'historique des événements précédents.

Présentation théorique des Processus de Hawkes

Application aux cambriolages

Pourquoi choisir les processus de Hawkes pour modéliser les risques de cambriolages en ville ?

- Modélisation de la dépendance temporelle
- Effet auto-excitateur
- Adaptation aux caractéristiques locales
- Prise en compte des événements antérieurs
- Flexibilité et adaptation aux données
- Applications pratiques

Présentation théorique des Processus de Hawkes

Définition d'un processus de Hawkes à une dimension

Représentation mathématique

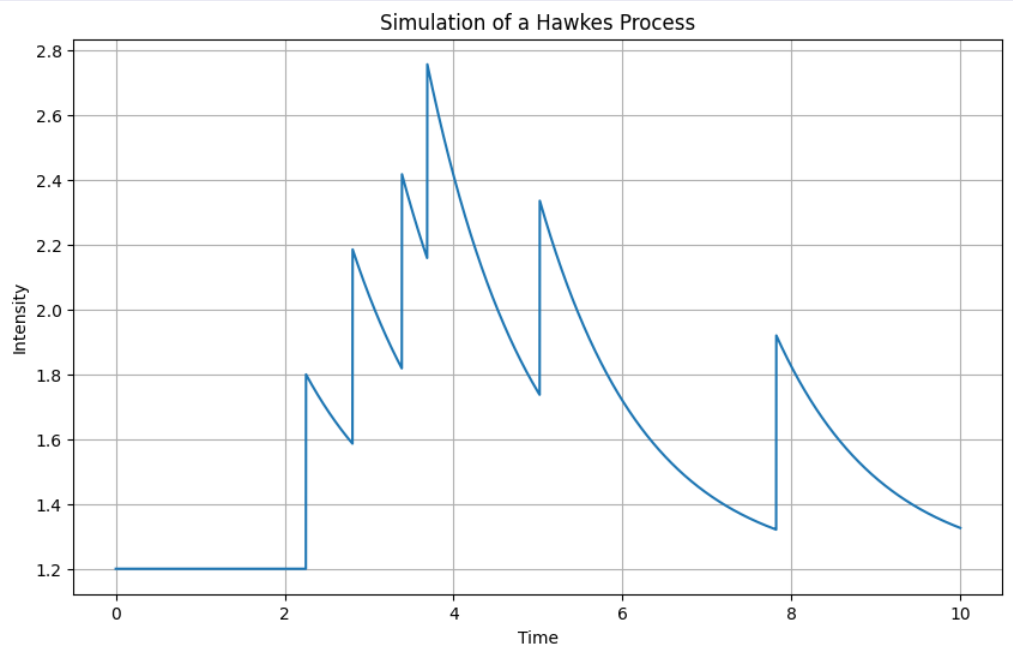
$$N(t) = \sum_i \delta(t - t_i)$$

- $N(t)$: nombre d'événements qui se sont produits jusqu'à l'instant t
- t_i : instants auxquels les événements se sont produits
- $\delta(t - t_i)$: fonction delta de Dirac qui vaut 1 si $t = t_i$ et 0 sinon.

Objectif

L'objet de notre étude est de déterminer le nombre de cambriolages $N(t)$ au cours d'une période $[0, T]$, où $N(t)$ est un processus de Hawkes d'intensité $\lambda(t)$, $t \geq 0$.

Exemple



Présentation théorique des Processus de Hawkes

Interprétation de la forme du modèle

Représentation mathématique

$$\lambda(t) = \lambda_0 + \sum_i \alpha_i \cdot e^{-\beta_i \cdot (t - t_i)}$$

- $\lambda(t)$: fonction d'intensité conditionnelle à l'instant t
- λ_0 : taux de base (taux d'événements en l'absence d'influence des événements passés)
- α_i : coefficient d'excitation correspondant à l'événement i
- β_i : coefficient de décroissance correspondant à l'événement i
- t_i : temps de l'événement i

Présentation théorique des Processus de Hawkes

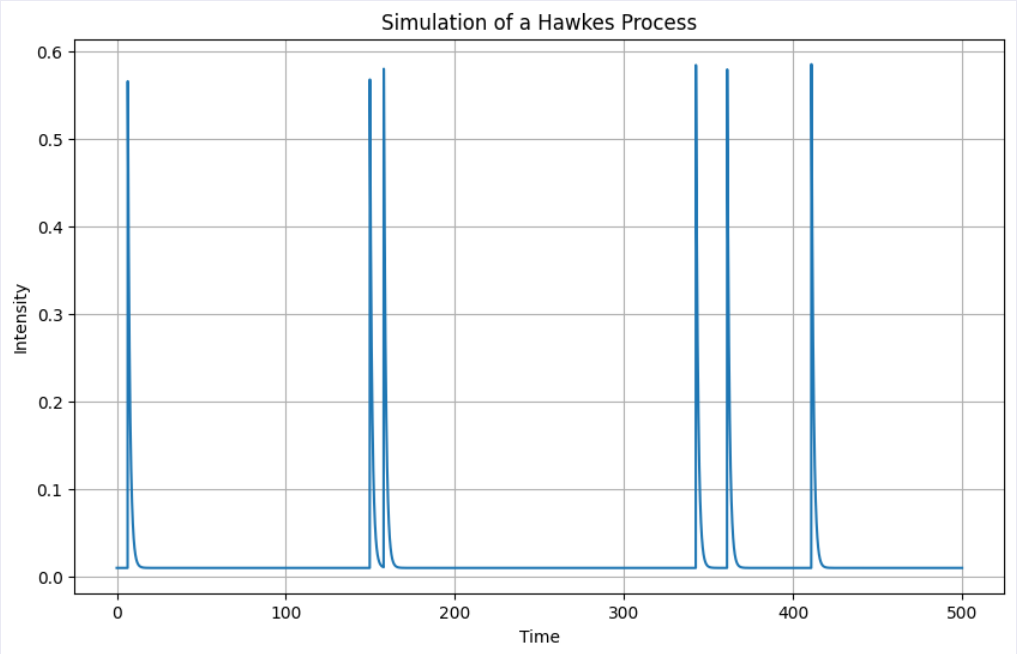
Interprétation des paramètres

Influence du paramètre λ_0

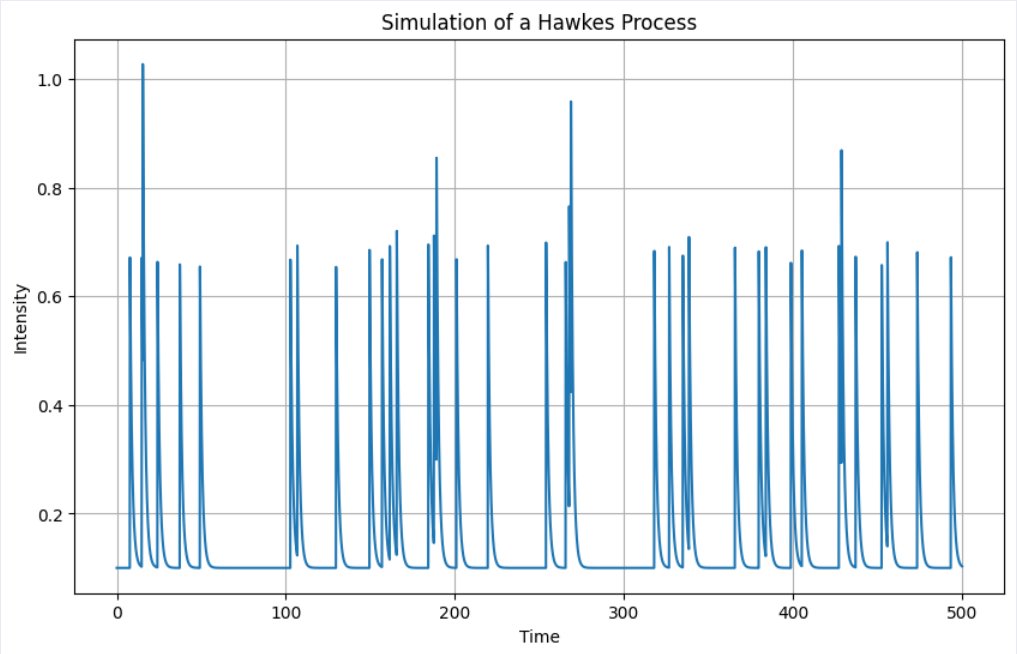
$$\lambda(t) = \lambda_0 + \sum_i \alpha_i \cdot e^{-\beta_i \cdot (t - t_i)}$$

- $\lambda(t)$: fonction d'intensité conditionnelle à l'instant t
- α_i : coefficient d'excitation correspondant à l'événement i fixé à 0.6
- β_i : coefficient de décroissance correspondant à l'événement i fixé à 0.8
- t_i : temps de l'événement i

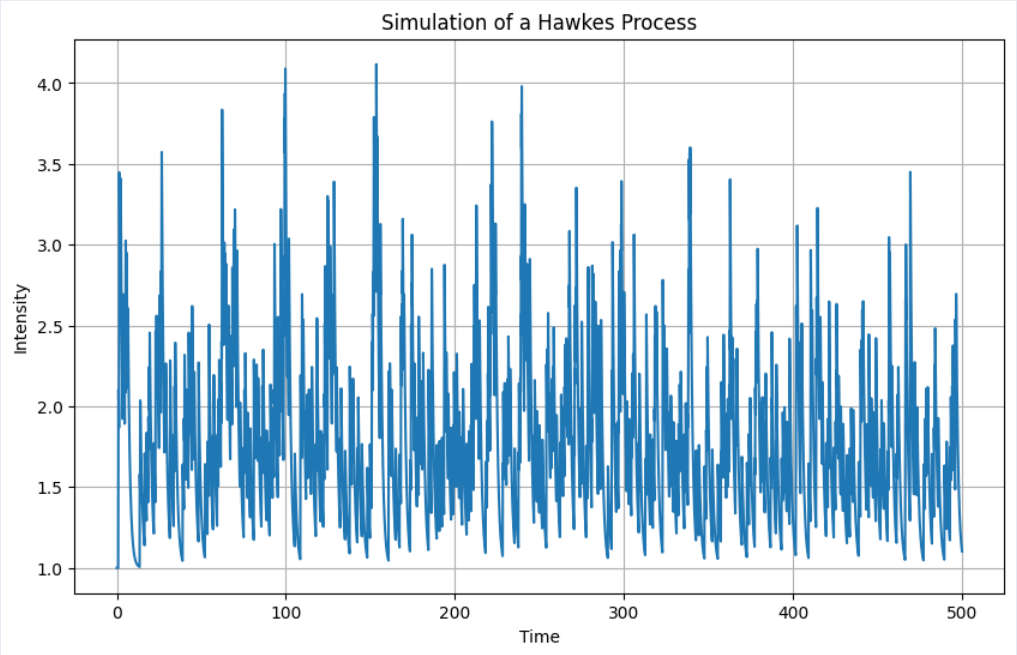
$\lambda_0 = 0.01$


Simulation of a Hawkes Process
Intensity
Time
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
100
200
300
400
500

$\lambda_0 = 0.1$


Simulation of a Hawkes Process
Intensity
Time
1.0
0.8
0.6
0.4
0.2
0
100
200
300
400
500

$\lambda_0 = 1$


Simulation of a Hawkes Process
Intensity
Time
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0
100
200
300
400
500

Présentation théorique des Processus de Hawkes

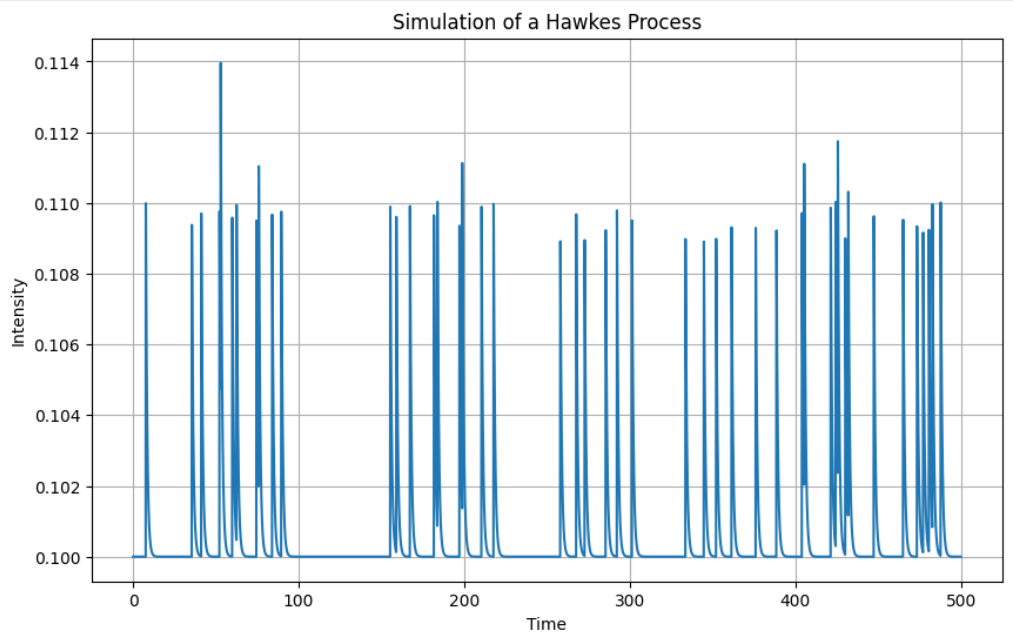
Interprétation des paramètres

Influence du paramètre α

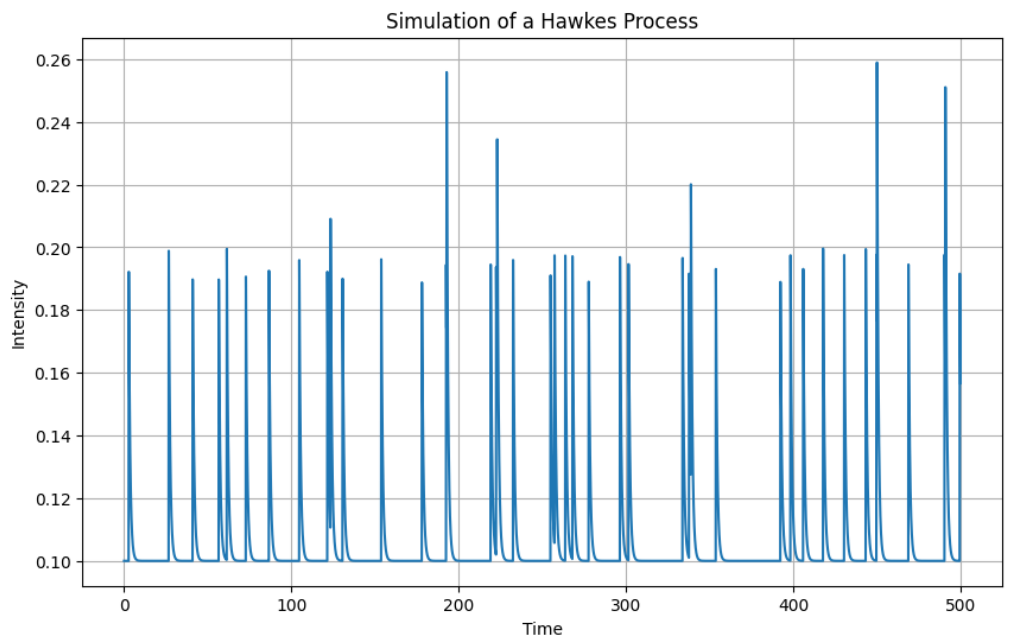
$$\lambda(t) = \lambda_0 + \sum_i \alpha_i \cdot e^{-\beta_i \cdot (t - t_i)}$$

- $\lambda(t)$: fonction d'intensité conditionnelle à l'instant t
- λ_0 : taux de base (taux d'événements en l'absence d'influence des événements passés) fixé à 0.1
- β_i : coefficient de décroissance correspondant à l'événement i fixé à 1.2
- t_i : temps de l'événement i

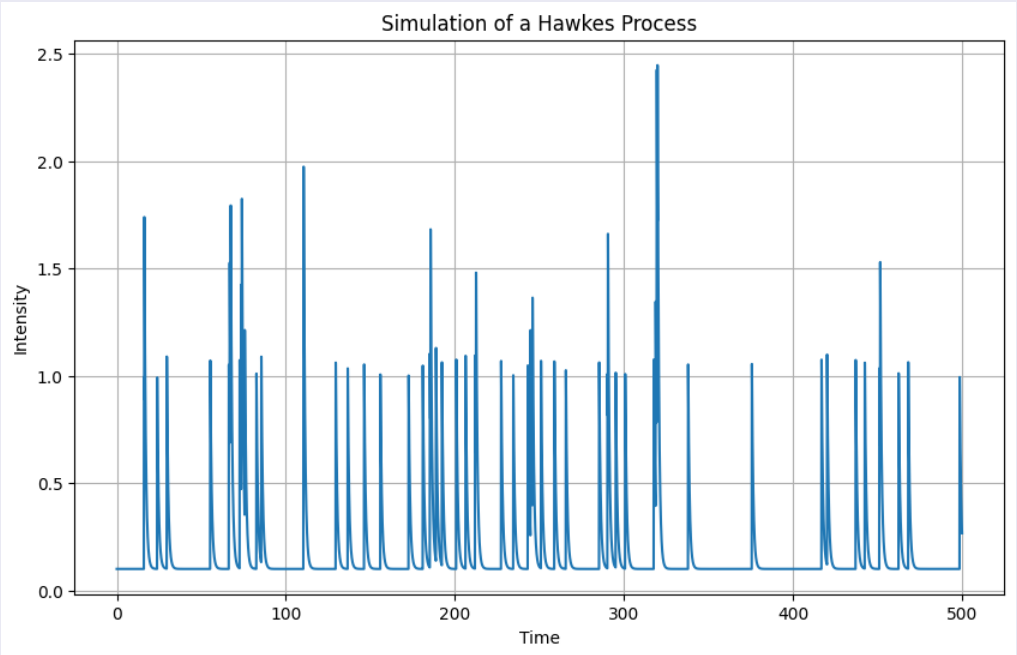
$\alpha = 0.01$


Simulation of a Hawkes Process
Intensity
Time
0.114
0.112
0.110
0.108
0.106
0.104
0.102
0.100
0
100
200
300
400
500

$\alpha = 0.1$


Simulation of a Hawkes Process
Intensity
Time
0.10
0.12
0.14
0.16
0.18
0.20
0.22
0.24
0.26
0
100
200
300
400
500

$\alpha = 1$


Simulation of a Hawkes Process
Intensity
Time
0.0
0.5
1.0
1.5
2.0
2.5
0
100
200
300
400
500

Présentation théorique des Processus de Hawkes

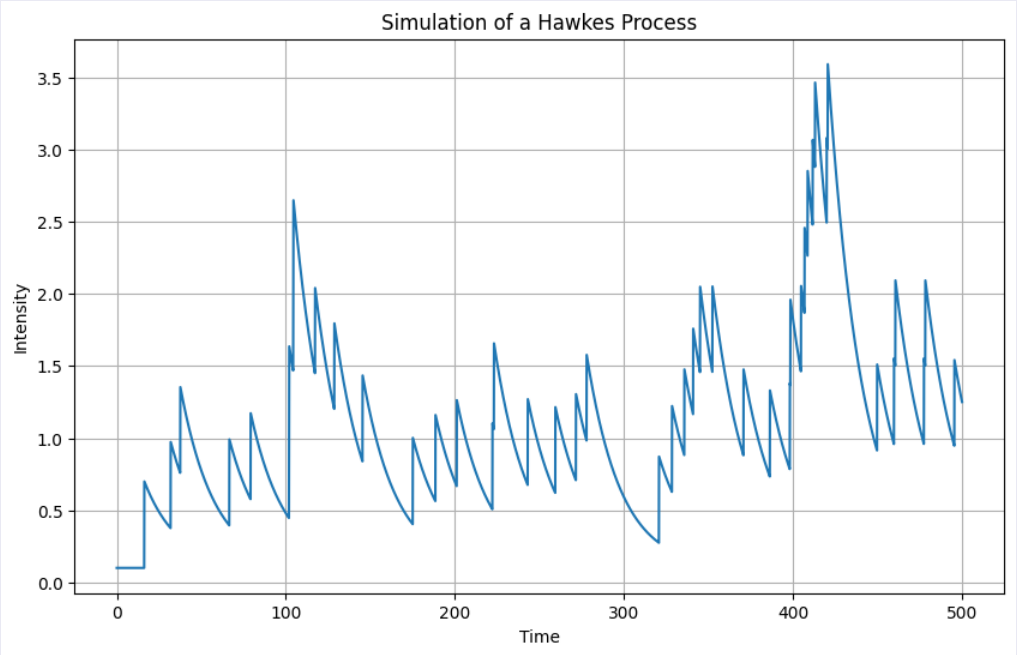
Interprétation des paramètres

Influence du paramètre β

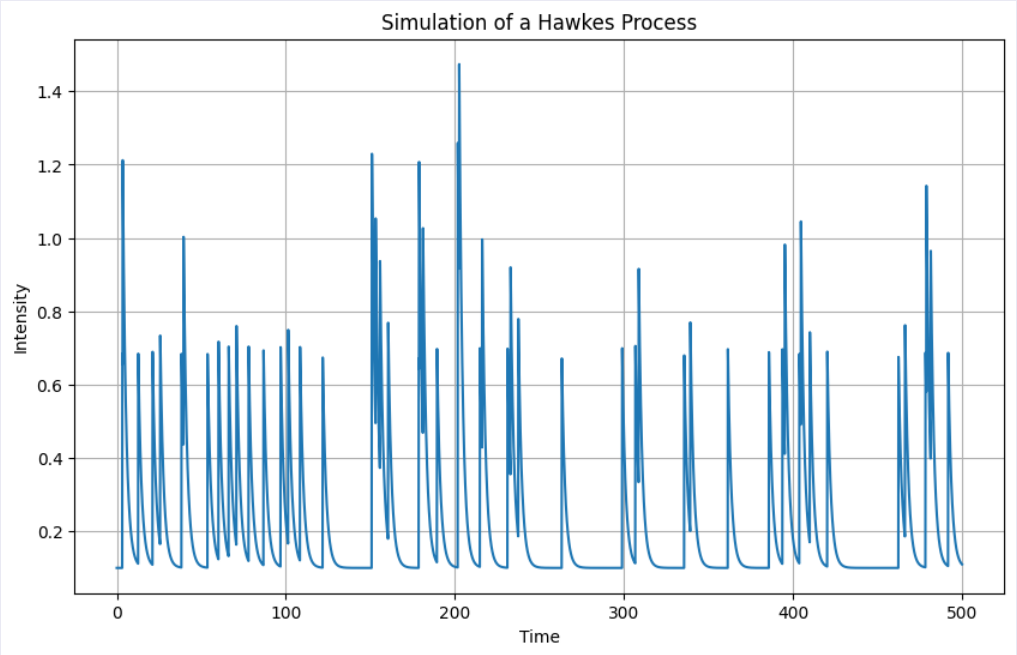
$$\lambda(t) = \lambda_0 + \sum_i \alpha_i \cdot e^{-\beta_i \cdot (t - t_i)}$$

- $\lambda(t)$: fonction d'intensité conditionnelle à l'instant t
- λ_0 : taux de base (taux d'événements en l'absence d'influence des événements passés) fixé à 0.1
- α_i : coefficient d'excitation correspondant à l'événement i fixé à 0.6
- t_i : temps de l'événement i

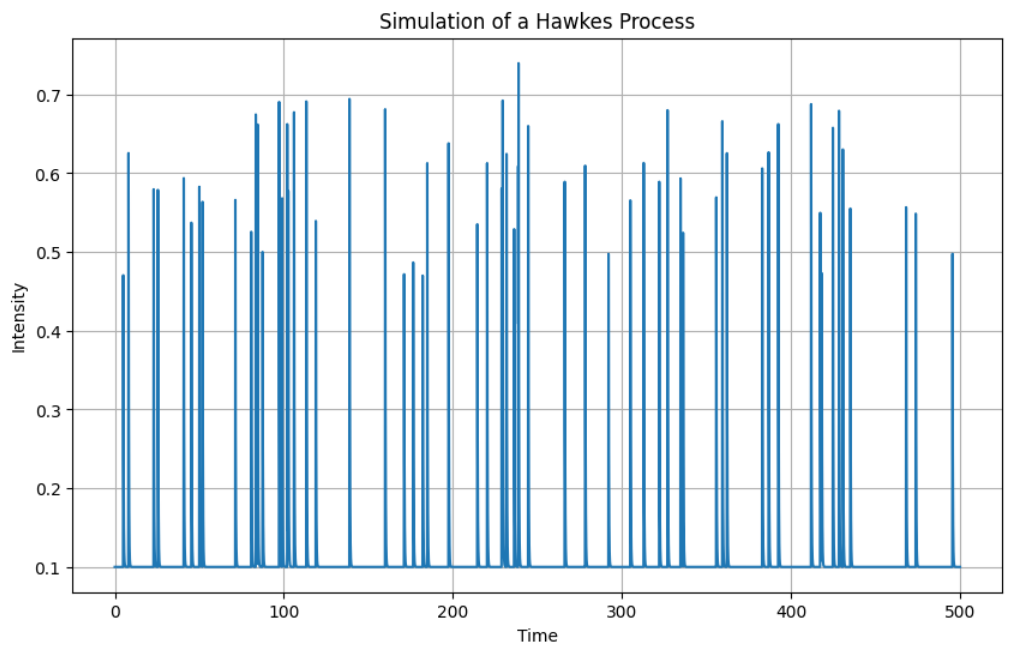
$\beta = 0.05$


Simulation of a Hawkes Process
Intensity
Time
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0
100
200
300
400
500

$\beta = 0.5$


Simulation of a Hawkes Process
Intensity
Time
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0
100
200
300
400
500

$\beta = 5$


Simulation of a Hawkes Process
Intensity
Time
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
100
200
300
400
500

Présentation théorique des Processus de Hawkes

Simulation d'un processus de Hawkes

Algorithme d'Ogata

L'algorithme d'Ogata (1981) génère des temps de survenance suivant un processus de Hawkes sur un intervalle de temps [0,T] en utilisant les paramètres λ_0, α et β, ainsi que la fonction d'intensité λ qui dépend de ces trois paramètres.

Présentation théorique des Processus de Hawkes

Estimation des paramètres

Méthode du maximum de vraissemblance

La méthode du maximum de log-vraisemblance (ML pour Maximum Likelihood) est une technique statistique couramment utilisée pour estimer les paramètres d'un modèle statistique. L'idée est de trouver les paramètres du modèle qui maximisent la fonction de log-vraisemblance.

Présentation théorique des Processus de Hawkes

Estimation des paramètres

	n=10	n=100	n=200
T=50	1.53099	1.533462	1.526972
T=100	1.410459	1.388181	1.384644
T=200	1.593578	1.355312	1.330897
T=500	1.220407	1.2360074	
		Lambda0	
	n=10	n=100	n=200
T=50	0.5227487	0.5473307	0.583421
T=100	0.5135027	0.6097493	0.5752622
T=200	0.5869971	0.6091595	0.5986176
T=500	0.5984739	0.5976343	
		Alpha	
	n=10	n=100	n=200
T=50	0.7393073	0.8620208	1.023031
T=100	0.7254344	0.8804755	0.8343176
T=200	0.8668946	0.8617870	0.8363573
T=500	0.7919300	0.8086969	
		Beta	

Plan de l'exposé

Présentation de la base de données

Chicago Crimes Database 2020

ID	Case Number	Date	Block	IUCR	Primary Type	Description	Location Des	Arrest	Domestic
12014684	JD189901	03/17/2020	039XX N LECL	820	THEFT	$500 AND UN	STREET	false	false
12012127	JD189186	03/18/2020	039XX W JAC	910	MOTOR VEHI	AUTOMOBILE	APARTMENT	false	true
12012330	JD189367	03/18/2020	023XX N KEEL	560	ASSAULT	SIMPLE	RESIDENCE	false	false
12014760	JD192130	03/18/2020	047XX W MC	1150	DECEPTIVE PF	CREDIT CARD	OTHER (SPEC	false	false
12012667	JD189808	03/18/2020	003XX S CICE	2017	NARCOTICS	MANUFACTUF	SIDEWALK	true	false
12015216	JD192637	03/16/2020	049XX S CALL	820	THEFT	$500 AND UN	STREET	false	false

Beat	District	Ward	Community A	FBI Code	X Coordinate	Y Coordinate	Year	Updated On	Latitude	Longitude	Location
1634	16	45	15	6	1141659	1925649	2020	03/25/2020	41.95205194	-87.7546603	(41.952051946, -87.754660372)
1132	11	28	26	7	1150196	1898398	2020	03/25/2020	41.87711018	-87.7239897	(41.877110187, -87.723989719)
2525	25	35	20	08A	1147996	1915240	2020	03/25/2020	41.92336897	-87.7316338	(41.923368973, -87.731633833)
1113	11	28	25	11	1144749	1899145	2020	03/25/2020	41.87926442	-87.7439708	(41.879264422, -87.743970898)
1533	15	28	25	18	1144446	1898000	2020	03/25/2020	41.8761281C	-87.7451122	(41.876128106, -87.745112291)
224	2	3	38	6	1179270	1872264	2020	03/25/2020	41.80478062	-87.6180383	(41.804780628, -87.618038332)

Présentation de la base de données

Filtrage de la base de données

Filtrage

- Supprimer les lignes avec des données manquantes
- Conserver uniquement les cambriolages
- Conserver uniquement les colonnes suivantes : Date, Primary Type, Latitude et Longitude

Présentation de la base de données

Formatage de la base de données

Formatage

- Passage du format date US au format EU
- Trier les cambriolages par ordre chronologique
- Définition de l'instant t_0=0 correspondant à l'instant du 01/01/2020 à 00H00

Présentation de la base de données

Première analyse

Problèmes rencontrés

- Liste trop longue
- Valeurs des instants trop grandes

Solutions exploitées

- Restriction spatiale
- Changement échelle temporelle

Présentation de la base de données

Chicago Crimes Database 2020

Affichage Chicago Map

Présentation de la base de données

Premières estimations

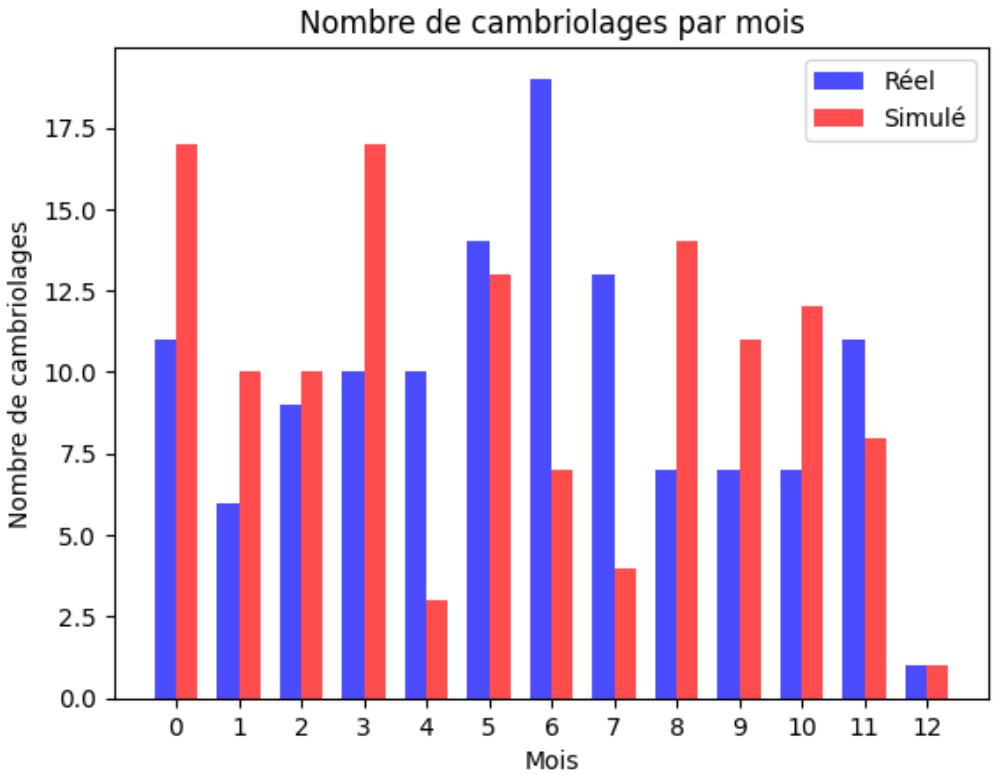
Estimations des paramètres

- λ_0=0.3067893
- α=0.11263505
- β=0.03089197

Estimations des cambriolages

- Nombre réel : 125
- Nombre simulé : 125

Application du modèle



Application du modèle

Exemple Fonction Calibrage(instants, 50)

Périodes (en jours	[0, 0+50]	[50, 100]	[100, 150]
Nombre réel	16	17	13
Nombre simulé	16,14	17,6	12,91
Erreur relative	0,875	3,52941176	0,69230769
IC 95%	[15.37483648, 16.90516352]	[16.18282377, 19.01717623]	[11.91286457, 13.90713543]

[150, 200]	[200, 250]	[250, 300]	[300, 350]
28	21	11	15
27,54	20,72	11,17	15,95
1,64285714	1,333333333	1,54545455	6,333333333
[26.18393494, 28.89606506]	[19.80567313, 21.63432687]	[10.48145392, 11.85854608]	[15.18065319, 16.71934681]

Application du modèle

Calibrage

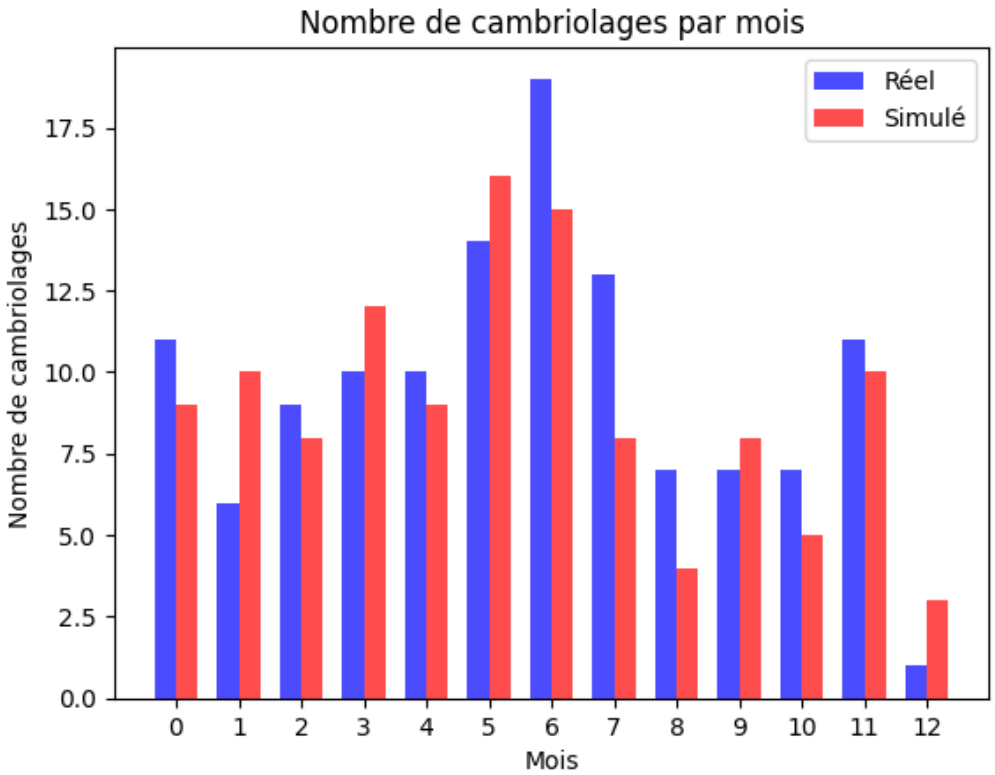
Estimations des paramètres

- λ_0=0.30990218
- α=0.13504193
- β=0.02762651

Estimations des cambriolages

- Nombre réel : 125
- Nombre simulé : 123

Application du modèle



Plan de l'exposé

Validation du modèle

Résultats

- Correspondance avec les données réelles
- Qualité du modèle (erreur relative faible de 2.44%)
- Pertinence du processus de Hawkes
- Utilité pour la prévision ?

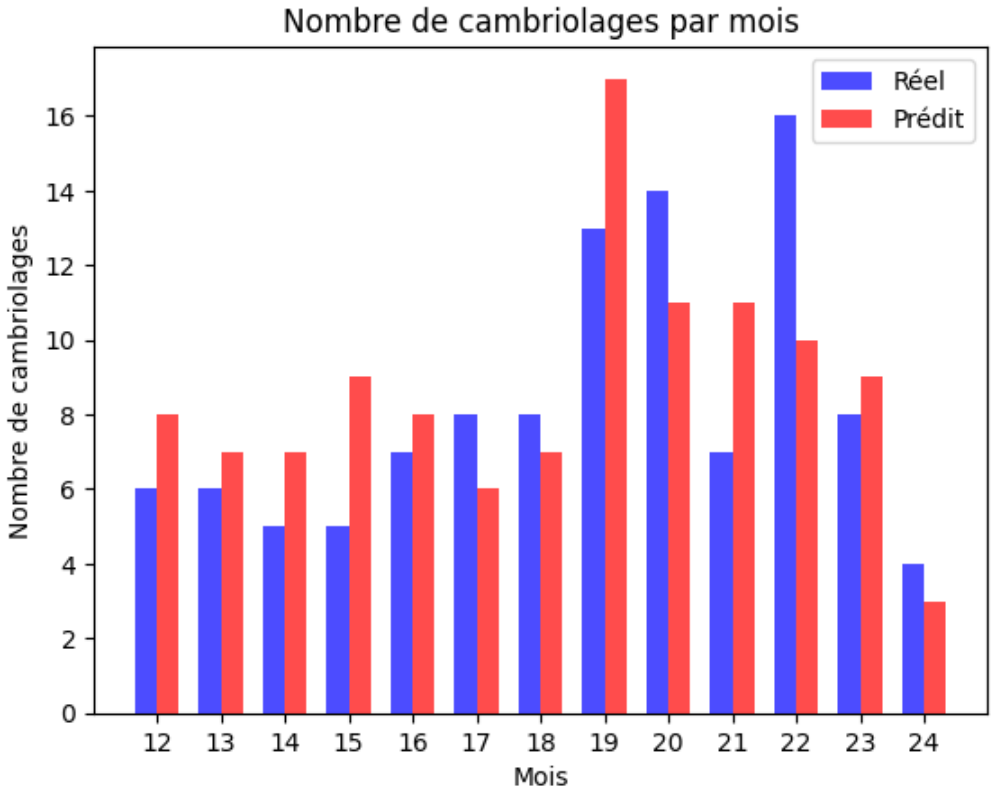
Utilisation du modèle

Prévisions du nombre de cambriolages en 2021

Résultats

- Nombre réel : 107
- Nombre prédit : 109

Application du modèle



Utilisation du modèle

Limites du modèle

Limites du modèle

- Simplification du processus réel
- Hypothèses du modèle
- Prédictions à long terme
- Sensibilité aux paramètres
- Difficulté de calibrage
- Manque d'explications causales

Utilisation du modèle

Limites du modèle

Peut-on envisager l'utilisation d'un tel modèle dans la vie courante ? L'exemple de PredPol

Figure 1 – www.predpol.com

Utilisation du modèle

Limites du modèle

Peut-on envisager l'utilisation d'un tel modèle dans la vie courante ? L'exemple de PredPol

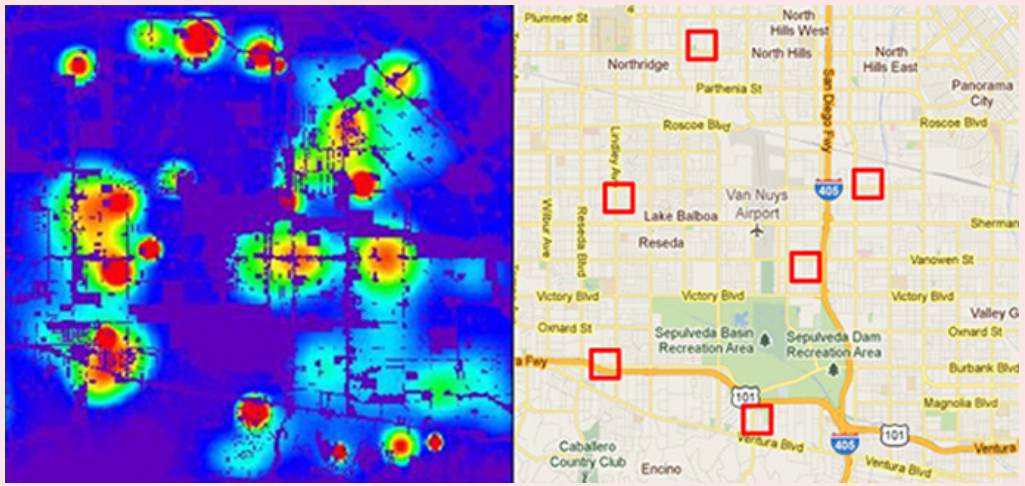


Figure 2 – www.predpol.com

Fin du TIPE

Merci pour votre attention !

Annexe I

Code Python

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import folium
from scipy.optimize import minimize
from tick.hawkes import SimuHawkesExpKernels
from collections import Counter


def simulate_hawkes_process_ogata(lmbda0, alpha, beta, T):
    # Initialisation du processus
    t = 0
    event_times = []

    while True:
        # Calcul de la borne superieure de l'intensite
        lmbda_bar = lmbda0 + alpha * len(event_times)

        # Generation d'un temps d'attente
        w = -np.log(np.random.uniform()) / lmbda_bar
        t = t + w

        if t > T:
            break

        # Calcul de l'intensite a l'instant t
        lmbda_t = lmbda0 + alpha * sum(np.exp(-beta * (t - ti)) for ti in event_times)

        # Generation d'une valeur aleatoire uniforme
        u = np.random.uniform()

        # Si u est inferieur ou egal au ratio de l'intensite sur sa borne superieure,
```

Annexe II

Code Python

```
33          # on accepte le temps d'attente comme le prochain temps d'evenement du processus de
                Hawkes
34          if u <= lmbda_t / lmbda_bar:
35              event_times.append(t)
36
37      # Etape 2 : Calcul de l'intensite a chaque instant
38      t_values = np.linspace(0, T, 5000)
39      intensity_values = []
40      for t in t_values:
41          lmbda = lmbda0 + alpha * sum(np.exp(-beta * (t - ti)) for ti in event_times if ti < t)
42          intensity_values.append(lmbda)
43
44      return t_values, intensity_values, event_times
45
46
47  def filter_burglary(filename):
48      # lire le fichier csv
49      df = pd.read_csv(filename, delimiter=';', error_bad_lines=False)
50
51      # filtrer le dataframe pour ne garder que les lignes ou le 'Primary Type' est 'BURGLARY'
52      df = df[df['Primary␣Type'] == 'BURGLARY']
53
54      # ne garder que les colonnes Date, Primary Type, Latitude, Longitude
55      df = df[['Date', 'Primary␣Type', 'Latitude', 'Longitude']]
56
57      # supprimer les lignes avec des valeurs manquantes
58      df = df.dropna()
59
60      # Convertir la colonne "Date" en format datetime
61      df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y␣%I:%M:%S␣%p')
62
63      # Modifier le format de la colonne "Date" pour afficher les dates avec l'heure en format 24h
```

Annexe III

Code Python

```
64        df['Date'] = df['Date'].dt.strftime('%m/%d/%Y␣%H:%M:%S')
65
66        # Trier le dataframe par la colonne "Date"
67        df = df.sort_values(by='Date')
68
69        # Reinitialiser les index du dataframe trie
70        df = df.reset_index(drop=True)
71
72        # enregistrer le nouveau dataframe dans un nouveau fichier csv
73        df.to_csv('Crimes2020_trie.csv', index=False)
74
75
76    def filter_and_display_burglaries(filename, map_bounds):
77        # lire le fichier csv
78        df = pd.read_csv(filename, delimiter=',', error_bad_lines=False)
79
80        # filtrer pour ne garder que les cambriolages dans la zone
81        df = df[(df['Latitude'] >= map_bounds[0][0]) & (df['Latitude'] <= map_bounds[1][0]) &
82                (df['Longitude'] >= map_bounds[0][1]) & (df['Longitude'] <= map_bounds[1][1])]
83
84        # enregistrer le nouveau dataframe dans un nouveau fichier csv
85        df.to_csv('Crimes2020_resized.csv', index=False)
86
87
88    def get_burglary_times(filename, start_time='01/01/2020␣00:00:00'):
89
90        # Charger le nouveau fichier CSV en tant que DataFrame
91        df = pd.read_csv(filename)
92
93        # Convertir la colonne de date en datetime
94        df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y␣%H:%M:%S')
95
```

Annexe IV

Code Python

```
96      # Definir le temps de reference
97      ref_time = pd.to_datetime(start_time, format='%m/%d/%Y %H:%M:%S')
98
99      # Convertir les dates de cambriolage en secondes par rapport au temps de reference
100     burglary_times = round((df['Date'] - ref_time).dt.total_seconds() / 86400)
101
102     return burglary_times.values.astype(int)
103
104
105 def display_burglaries_on_map(filename, map_bounds):
106     # lire le fichier csv
107     df = pd.read_csv(filename, delimiter=',', error_bad_lines=False)
108
109     # filtrer le dataframe pour ne garder que les lignes ou le 'Primary Type' est 'BURGLARY'
110     df = df[df['Primary Type'] == 'BURGLARY']
111
112     # ne garder que les colonnes Date, Primary Type, Latitude, Longitude
113     df = df[['Date', 'Primary Type', 'Latitude', 'Longitude']]
114
115     # supprimer les lignes avec des valeurs manquantes
116     df = df.dropna()
117
118     # definir le centre de la carte comme la moyenne des latitudes et longitudes
119     map_center = [df['Latitude'].mean(), df['Longitude'].mean()]
120
121     # creer la carte avec folium
122     map_burglaries = folium.Map(location=map_center, zoom_start=13)
123
124     # ajouter des marqueurs pour chaque cambriolage
125     for _, row in df.iterrows():
```

Annexe V

Code Python

```
126             folium.Marker(location=[row['Latitude'], row['Longitude']],
                              popup=row['Date']).add_to(map_burglaries)
127
128         # definir les limites de la carte
129         map_burglaries.fit_bounds(map_bounds)
130
131         # afficher la carte
132         return map_burglaries
133
134
135     # Definition d'une fonction recursive pour le calcul de r, un terme utilise dans la fonction de
            vraisemblance
136     def _recursive(timestamps, beta):
137         r_array = np.zeros(len(timestamps))
138         for i in range(1, len(timestamps)):
139             r_array[i] = np.exp(-beta * (timestamps[i] - timestamps[i - 1])) * (1 + r_array[i - 1])
140         return r_array
141
142     # Definition de la fonction de log-vraisemblance specifiant les differents parametres :
143     def log_likelihood(timestamps, mu, alpha, beta, runtime):
144         r = _recursive(timestamps, beta)
145         return -runtime * mu + alpha * np.sum(np.exp(-beta * (runtime - timestamps)) - 1) + \
146                np.sum(np.log(mu + alpha * beta * r))
147
148     # Simulation de donnees Hawkes en utilisant la bibliotheque tick :
149     mu = 1.2
150     alpha = 0.6
151     beta = 0.8
152     rt = 365
153
154     simu = SimuHawkesExpKernels([[alpha]], beta, [mu], rt)
155     simu.simulate()
```

Annexe VI

Code Python

```
156   t = simu.timestamps[0]
157
158   # Definition d'une nouvelle fonction a utiliser par la fonction minimize et qui renvoie la
          log-vraisemblance negative :
159   def crit(params, *args):
160       mu, alpha, beta = params
161       timestamps, runtime = args
162       return -log_likelihood(timestamps, mu, alpha, beta, runtime)
163
164   # Minimisation de la fonction crit :
165   minimize(crit, [0.5, 0.5, 0.5], args=(burglary_times, rt), bounds = ((1e-10, None), (1e-10,
          None), (1e-10, None)), method = 'Nelder-Mead')
166
167
168   def estimate_hawkes_parameters(temps_survenance, duree):
169       result = minimize(crit, [0.1, 0.1, 0.1], args=(temps_survenance, duree), bounds = ((1e-10,
              None), (1e-10, None), (1e-10, None)), method = 'Nelder-Mead')
170       return result.x
171
172
173   def plot_burglaries(real_times, simulated_times):
174       # Convertir les jours en mois
175       real_times_month = np.floor(real_times / 30).astype(int)
176       simulated_times_month = np.floor(simulated_times / 30).astype(int)
177
178       # Compter le nombre de cambriolages par mois
179       real_counts = Counter(real_times_month)
180       simulated_counts = Counter(simulated_times_month)
181
182       # Trouver les mois pour lesquels nous avons des donnees
183       all_months = sorted(set(real_counts.keys()).union(set(simulated_counts.keys())))
184
```

Annexe VII

Code Python

```
185         # Creer les positions des barres sur l'axe des x
186         bar_width = 0.35
187         real_positions = np.arange(len(all_months))
188         simulated_positions = [x + bar_width for x in real_positions]
189 
190         # Creer les histogrammes
191         plt.bar(real_positions, [real_counts[month] for month in all_months], width=bar_width,
                alpha=0.7, label='Reel', color='blue')
192         plt.bar(simulated_positions, [simulated_counts[month] for month in all_months],
                width=bar_width, alpha=0.7, label='Simule', color='red')
193 
194         # Ajouter une legende
195         plt.legend()
196 
197         # Etiquettes des axes
198         plt.xlabel('Mois')
199         plt.ylabel('Nombre de cambriolages')
200 
201         # Ajouter les etiquettes des mois sur l'axe des x
202         plt.xticks([r + bar_width / 2 for r in range(len(all_months))], all_months)
203 
204         # Titre du graphe
205         plt.title('Nombre de cambriolages par mois')
206 
207         # Afficher le graphe
208         plt.show()
209 
210     plot_burglaries(burglary_times, timestamps)
211 
212 
213     def Calibrage(temps, p, n_simulations=100):
```

Annexe VIII

Code Python

```
214         # Calculer le nombre de periodes
215         k = int(np.max(temps) // p)
216 
217         R = np.zeros(k)     # Nombre reel d'evenements
218         S = np.zeros((k, n_simulations))     # Nombre theorique d'evenements
219         Diff = np.zeros(k)  # Pourcentage d'erreur
220 
221         # Partitionner les donnees en differentes periodes et calibrer un processus de Hawkes pour
                chaque periode
222         for i in range(k):
223             # Obtenir les temps d'evenements pour cette periode
224             temps_periode = temps[(temps >= i*p) & (temps < (i+1)*p)] - i*p
225 
226             # Compter le nombre reel d'evenements
227             R[i] = len(temps_periode)
228 
229             # Estimer les parametres du processus de Hawkes
230             params = estimate_hawkes_parameters(temps_periode, p)
231 
232             # Simuler le processus de Hawkes plusieurs fois avec les parametres estimes et compter
                    le nombre d'evenements
233             for j in range(n_simulations):
234                 timestamps = simulate_hawkes_process(params[0], params[1], params[2], p)
235                 S[i, j] = len(timestamps)
236 
237             # Calculer le pourcentage d'erreur
238             Diff[i] = np.abs(np.mean(S[i]) - R[i]) / R[i] * 100
239 
240         # Calculer l'esperance et l'ecart-type du nombre de cambriolages simules
241         S_mean = np.mean(S, axis=1)
242         S_std = np.std(S, axis=1)
```

Annexe IX

Code Python

```
243
244         # Calculer l'intervalle de confiance a 95% pour l'esperance du nombre de cambriolages
                simules
245         z = 1.96  # z-score pour un intervalle de confiance a 95%
246         CI_lower = S_mean - z * S_std / np.sqrt(n_simulations)
247         CI_upper = S_mean + z * S_std / np.sqrt(n_simulations)
248
249         return R, S_mean, Diff, (CI_lower, CI_upper)
250
251
252     def optimiser_periodes(temps, p_max=365):
253         # Initialiser l'erreur minimale et le nombre optimal de periodes
254         erreur_min = float('inf')
255         p_optimal = 1
256
257         # Parcourir chaque nombre possible de periodes
258         for p in range(1, p_max+1):
259             # Calibrer le processus de Hawkes pour le nombre actuel de periodes
260             R, S, Diff, (CI_lower, CI_upper) = Calibrage(temps, p)
261
262             # Calculer l'erreur moyenne
263             erreur_moyenne = np.mean(Diff)
264
265             # Si l'erreur moyenne est inferieure a l'erreur minimale actuelle, mettre a jour
                    l'erreur minimale et le nombre optimal de periodes
266             if erreur_moyenne < erreur_min:
267                 erreur_min = erreur_moyenne
268                 p_optimal = p
269
270         return p_optimal, erreur_min
271
```

Annexe X

Code Python

```
272
273   def predict_hawkes_process(mu, alpha, beta, start_time, end_time):
274       # Creation de l'objet de simulation du processus de Hawkes
275       simu = SimuHawkesExpKernels([[alpha]], beta, [mu], end_time)
276
277       # Simulation du processus de Hawkes
278       simu.simulate()
279
280       # Filtration des predictions pour ne garder que les evenements futurs
281       future_events = [t for t in simu.timestamps[0] if t >= start_time]
282
283       # Conversion en tableau numpy et arrondi a l'unite la plus proche
284       future_events = np.rint(np.array(future_events)).astype(int)
285
286       return future_event
```