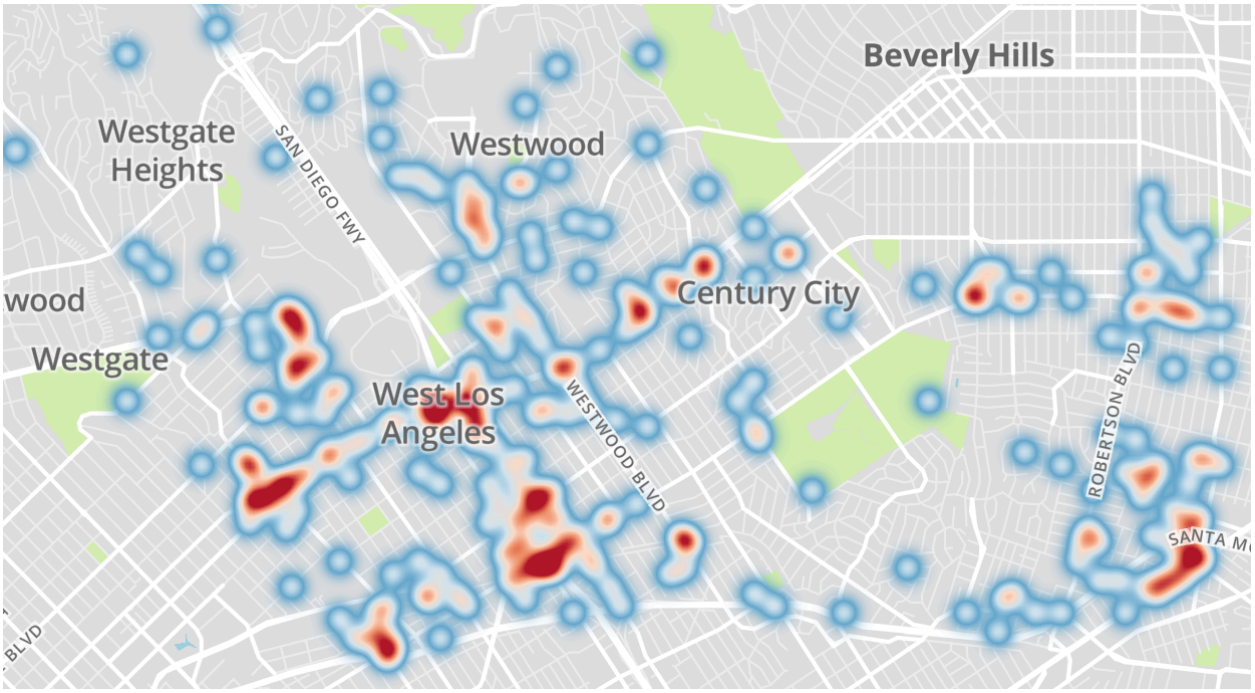
Stat 222 Final Project: Effect of Social Distancing Measures on Crime

Spatial-Temporal Analysis on West Los Angeles Arrest Dataset



Spring 2020

Introduction

One of most primary influence to be taken into account for the causes that regulate crime is its environment. This is demonstrated by Richard L. Dugdale in his famous book "Origin of Crime in Society" with examples from the yellow-fever epidemic in Memphis: there were a great many robberies committed in that dreadful time[2]. Indeed, the disturbance of social order which leaves property unprotected promotes unlawful appropriation [3].

In response to the ongoing COVID-19 pandemic, governments across the states have implemented social distancing regulations. Social distancing measures are powerful public health tools to reduce transmission speed of an infection agent, which include instructions that individuals maintain a distance from one another, limitations of gatherings, business operation constrains[1]. In city of Los Angeles, government announced the "shelter in place" order in March 20th, 2020 and the social distancing measure continue to today (June 6th, 2020). Most of unnecessary businesses had remained closed until May 15th, 2020.

This certainly changes the environment of crime in various aspects. First, when fully implemented, social distancing measure drives a massive disruption of daily routines, significantly altering and disrupting the conditions under which crime may occur. For example, it reduces the possibility of criminal acts that require in-person contact. Second, social distancing is built upon economy strain. The economic pressure might prompt the incentives for some people to conduct crime. Therefore, it is not obvious how the social distancing measure will shape the crime patterns.

Investigation of spatial-temporal pattern of crime may provide valuable insights into 1) whether social distancing measures are effectively implemented and whether individuals and communities are more meaningful for critical public health measure; (2) what policy need to be adopt to compensate the harm from social distancing measure on public safety.

In this study, we will use spatial-temporal analysis tools to investigate **the arrest data** in **west Los Angeles area** in **April 2019** and **April 2020**. The main questions to be addressed in this study are:

1. is there any clustering or inhibition within event points;
2. what can be a suitable model for the dataset;
3. is there any effect from social distancing measures on spatial-temporal distribution of criminal events.

I will first introduce the data and some preprocessing steps. The methodology adopted will be covered in the following section. I will then diagnose the dataset to see clustering effect. Later, two different point process models are fit to the data in order to explain the possible patterns in the arrest data. Finally, I will summarize our findings and future directions.

Dataset and preprocessing

Our data comes from the open data source of City of Los Angeles (https://data.lacity.org/A-Safe-City/), which contains more than one million records of arrest incidents from 2010 to present in the area of Los Angeles city. Due to privacy issue, location fields are only provided to the nearest hundred block. I specifically look at arrests took place in west Los Angeles area (area code 08) in April 2019 and April 2020 (Fig.1). The major area is located in a square window with latitude/longitude ((-118.48, -118.37)x(34.02,34.08)). For the time period, April 2020 is special because it is right after the shutting down of Los Angeles city and before the reopening of economy, corresponding to a period under social distancing measures. To compare it with usual case, I chose data from April 2019, a period without social distancing measures. This reduced the size of dataset to around 100.

In order to apply the spatial-temporal models, we need to preprocess the data. First, I normalize the latitude and longitude data into a unit square. Due to the high level of coarse graining, some records have the same date and the same location. To let the dataset fit into a *simple* point process, I add a small noise to the date and location record: $t = t_0 + 0.1(\delta_t - 0.5)$, $x = x_0 + 0.01(\delta_x - 0.5)$ and $y = y_0 + 0.01(\delta_y - 0.5)$ where $\delta_t, \delta_x, \delta_y$ are uniform random noises. This makes all the times and locations distinct.

Method

Our data is distributed over the space $S = [0,1] \times [0,1] \times [0,31]$, in which I will be conducting all of our analysis. The first thing I looked at was the kernel smoothing that calculated the density of features in a neighborhood around those features. The bandwidth used is a constant calculated from standard distance and distance to the mean. This helps to identify the hot spots. Next I looked at the estimated F, G, and J functions of our data, which indicate possible clustering behavior. After that, I tried to fit two different point process models on our sample data, which are the inhomogeneous Poisson process and Hawkes process model.

To find the best fit from the inhomogeneous Poisson process, I used the Stoyan method to estimate the parameters of this model. The conditional intensity function I used on our data was

$$\lambda(z \mid z_1, \ldots, z_k) = \mu + \alpha x + \beta y + \gamma \sum_{i=1}^{k} \frac{a_1 e^{-a_1 D(z_i, z)}}{2\pi D(z_i, z)}$$

where $z = (x, y)$ and $D(z_1, z_2) = |z_1 - z_2|$ is the distance.

As for fitting an Hawkes process, the conditional intensity function takes the following form:

$$\lambda(t, x, y) = \mu\rho(x, y) + K \sum_{t_i, x_i, y_i; t_i < t} g(t - t_i) g(x - x_i, y - y_i)$$

where, $\rho(x, y) = \frac{1}{X_1 Y_1}$, $g(t) = \beta e^{-\beta t}$, $g(x, y) = \frac{\alpha}{\pi} e^{-\alpha r^2}$, $r^2 = x^2 + y^2$,

which contains parameter μ, K, β, α. Then I use maximum-likelihood estimation to estimate parameter values. To evaluate the fitting performance, super-thinning method was adopted to make super-thinned points plot as well as estimate the F, G, and J function and compare to pure Poisson process.

Results

1. Raw data and diagnostic plots

Figure 1 shows the plot of original data obtained from website of Los Angeles open data source. There is obvious inhomogeneity in the distribution of arrest point, with a lot of points locate in main streets or

central business zone corresponding to high population density area. To further illustrate this, the data was visualized in a density plot after kernel smoothing (Fig. 2), which seems to indicate some clustering around districts such as Sawtelle, Beverly Wood for both period of time. The data from 2020 seems to be more clustered at north Sawtelle area (west Wilshire blvd. and west Santa Monic blvd.)

To further confirm the clustering behavior, I looked at the estimated F, G, and J functions in Figure 3. F function plot indicates the distance from a fixed point to the nearest point. The estimated F function appears to show more clustering of arrest al all distances than that generated by a stationary Poisson process. Similarly, the G function plot, which implies the average distance from a typical point to its nearest neighbor, indicates possible clustering behavior for both dataset. Besides, the J function, which is a combination of F and G function also shows the same conclusion.

2. Fitting an inhomogeneous Poisson process

I next fit the data using an inhomogeneous model by maximum likelihood estimation. The parameters and corresponding std are listed below in the form of mean ± std.

Table 1: estimate of parameters and STDs for inhomogeneous Poisson process

dataset	μ	α	β	γ	a1
2019 April	19.7±25.0	2.17±18.1	-9.46±31.1	0.857±0.1	52.3±9.62
2020 April	42.1±16.7	-41.6±29.4	-0.540±24.3	0.828±0.099	36.5±6.4

The background rate and total density rate for the space is shown in Fig.5. Density plots of the fitting model agree relatively well with the original data. Both of the models are successfully capture those hot spots. The background rate with social distancing measures (2020) is more inhomogeneous compared to the estimation for data without social distancing measures (2019). The west region has higher background rate compared to the east region in 2020. Since districts in the east are mostly single house residential area whereas many apartments and commercial businesses are located in the west, this suggests that crime events are less likely to happen in the the residential area, as expected due to “shelter in place” order. In

addition, similar positive γ coefficients in both models indicate clustering for both datasets. a_1 values are large, indicating correlation decreases quickly over distance. However, the standard errors appear to be extremely high, indicating this model might not be suitable.

3. Fitting a Hawkes process

Another model fitted in this analysis was the Hawkes model. The form we took can be found in the method section. In the table below we show fitting results.

Table 2: estimate of parameters and STDs for Hawkes process model

dataset	μ	K	α	β
2019 April	1.86±0.31	0.484±0.084	40.2±17.9	40.1±6.0
2020 April	1.30±0.23	0.594±0.084	11.7±2.2	45.0±6.9

This model turns out to be a good fit for both datasets, which can be seen from the relatively small standard deviations compared to the estimated means. Model for 2019 data has a bit higher background rate than that of 2020. This is expected because the total number of cases decreases from 112 in April 2019 to 99 in April 2020. The values of the triggering function are approximately 0.5. This implies that on average each point is likely to directly trigger 0.5 point. So each background point is expected to generate 1 triggered point in total. Besides, we have relatively high value of α and β compared to the scale of length ($X_1 = Y_1 = 1$) and time ($T = 31$), suggesting short correlation length in both time and space. This might be from group crime so that offender and accomplices were arrested in the near location and time.

To see how well this Hawkes model can fit the data, we further evaluate the model using super-thinning. In Fig.6 we show the original points as well as super-thinned points. One can see the super-thinned points appear to follow a simple homogeneous Poisson process, which suggested that Hawkes model could be a good fit. The F, G and J functions for super-thinned points are displayed in Fig. 7 together with the original curves. For all those functions, at least in short distance, the super-thinned lines collapse with pure stationary Poisson process results. This provides evidence that the Hawkes model was suitable.

Conclusion and Future Direction

To conclude, we studied the spatial-temporal arrest dataset in April, 2019 and April 2020, in order to understand the effect of social distancing in crime. Kernel smoothing and simple diagnostic plots suggest clustering in arrest data. We then fitted the data using inhomogeneous Poisson process model as well as Hawkes model. The later one provides a good fit to both datasets, which is evaluated by diagnostic functions. Our results suggest that social distancing and shelter in place order have had some impact on crime in terms of the background rate and spatial distribution, even though the effect is marginal.

We note several limitations of the present study. First, the results are specific to west Los Angeles area. Second, the arrest data couldn't reflect all criminal events which might not result in an arrest record if it is light. Besides, the police departments have undertaken dramatic changes in response to the pandemic(New York city is a good example). Officers who contract COVID-19 or must accommodate children home from school, leads to reduced workforce for crime arrest.

In addition to looking at larger dataset, it might be helpful to introduce covariates. Intuitively the crime frequency is related to population density or poverty. Adding those additional information might make our model more accurate. Besides, some quantitive evaluation of the difference between fitted models for data in 2019 and 2020 would be helpful to demonstrate the impact of social distancing measure more clearly. For example, we can simulate the fitted model from 2019 and compare it to dataset in 2020.

Reference

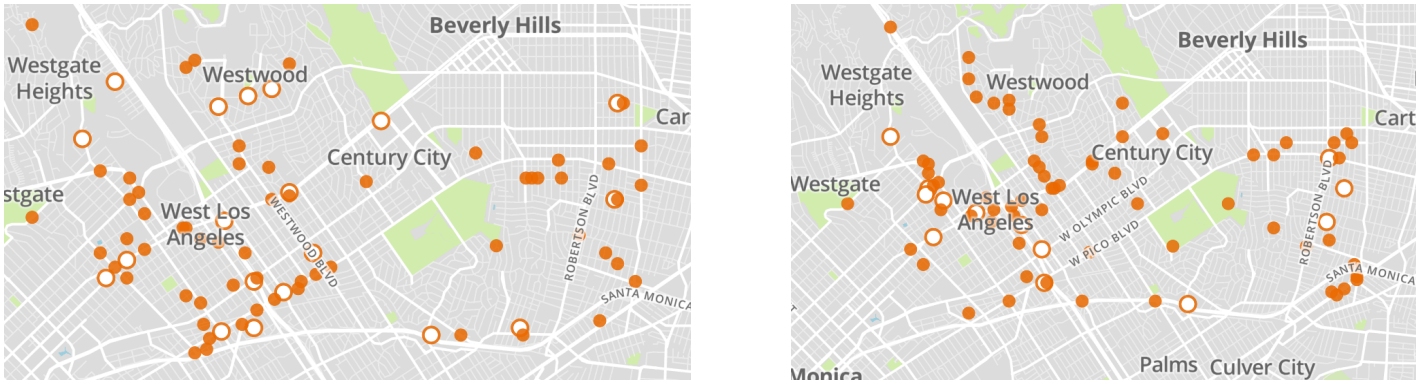
[1] Mohler, et al. Impact of social distancing during COVID-19 pandemic in Los Angeles and Indianapolis, *Journal of Criminal Justice* **68**, 101692 (2020)

[2] Richard L. Dugdale, Origin of Crime in Society, *The Atlantic*, (1881)

[3] Wilson, James Q Kelling, George L. Broken Windows, *The Atlantic*, (1982)

[4] David Weisburd, The Law of Crime Concentration and Criminology in Place, Criminology 53, 133 (2015)

Figure 1. Map of west Los Angeles and distribution of arrest data.



left: data from April 2019, 112 points in total. Right: data from April 2020, 99 points in total.

Figure 2. Kernel smoothing plot for both datasets. Top: 2019; Bottom: 2020

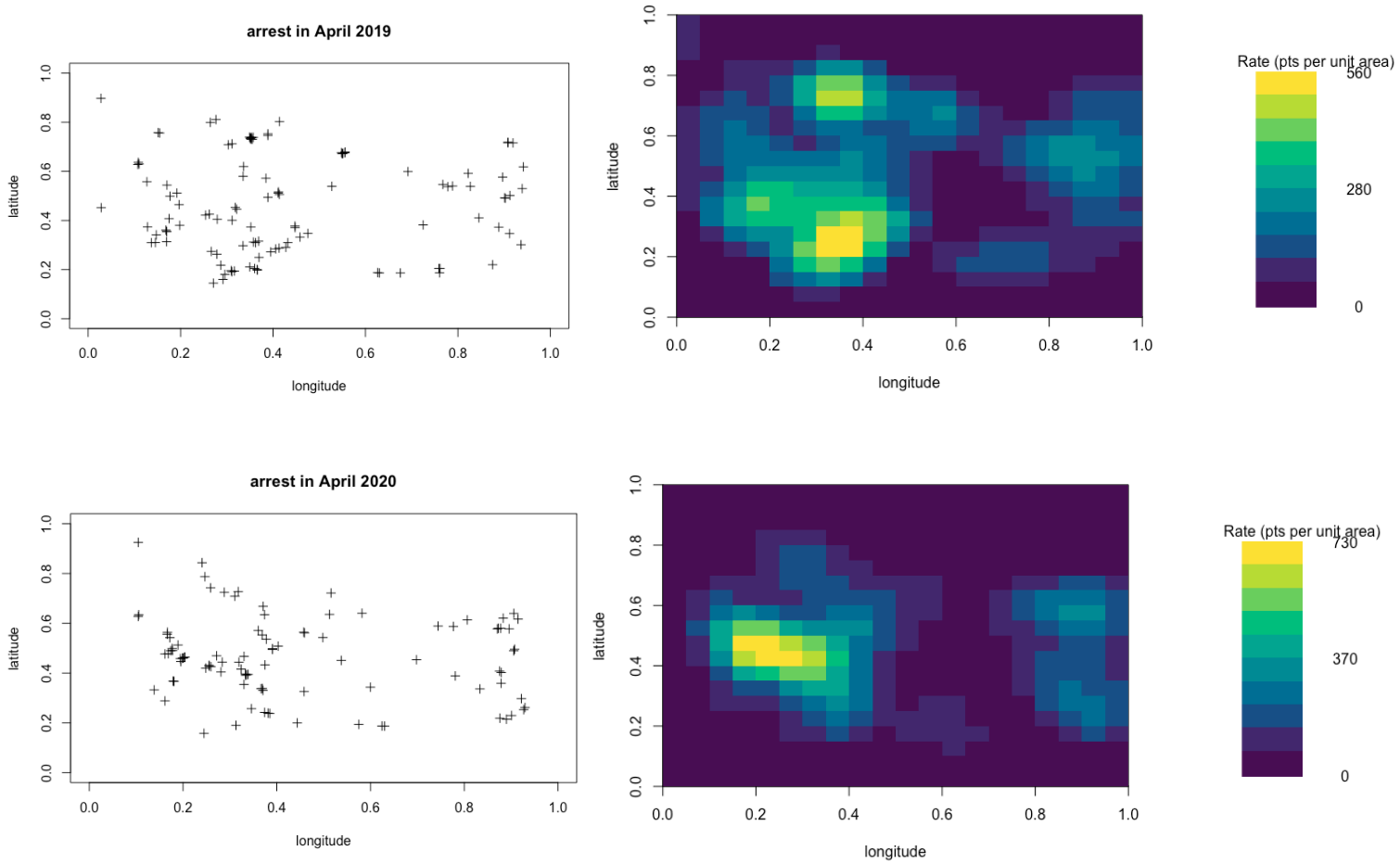


Figure 3. F, G, J function plot for original data. Left: 2019 data; Right 2020 data

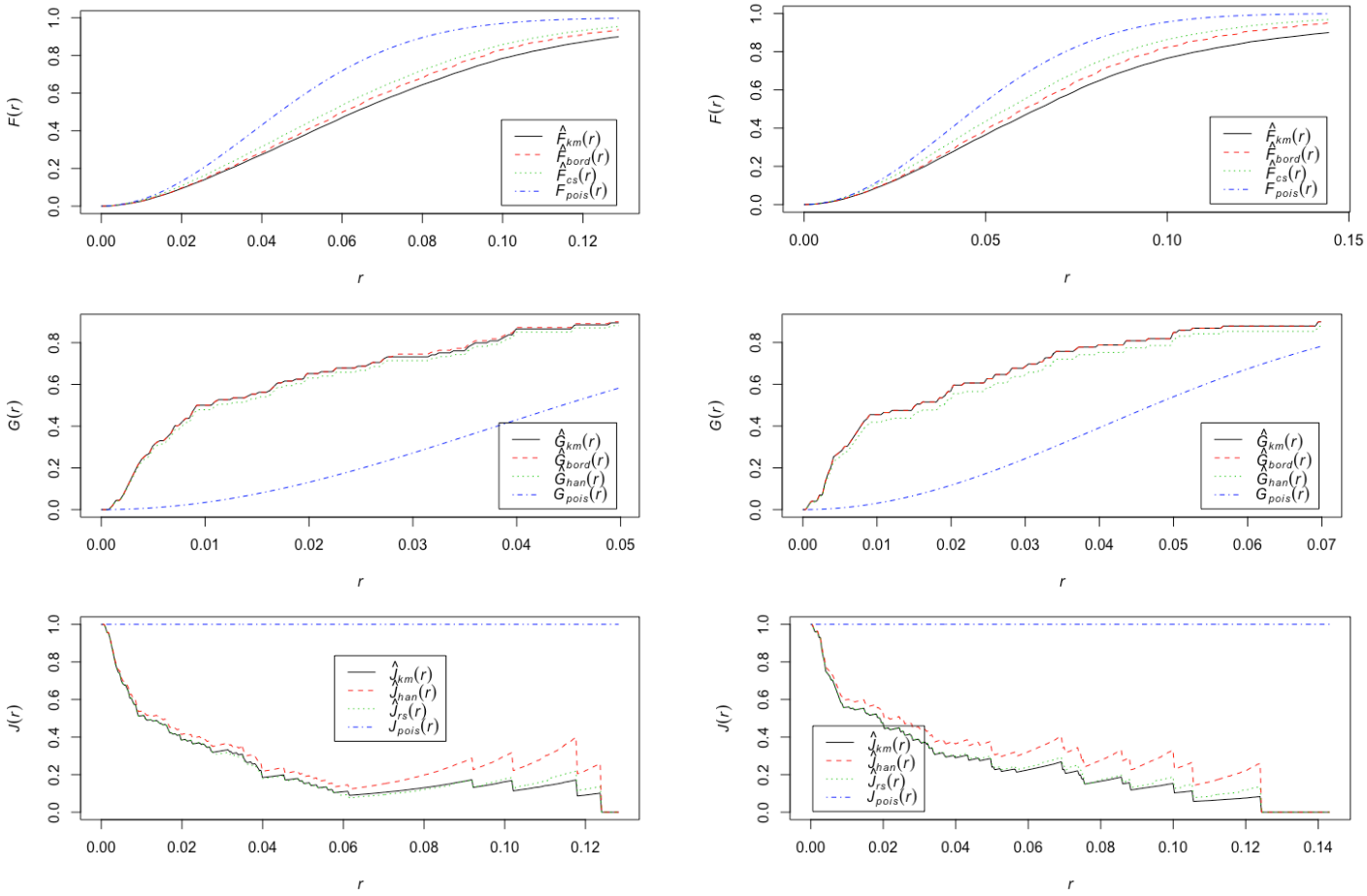


Figure 4. K and L functions for original data. Left: 2019 data. Right: 2020 data

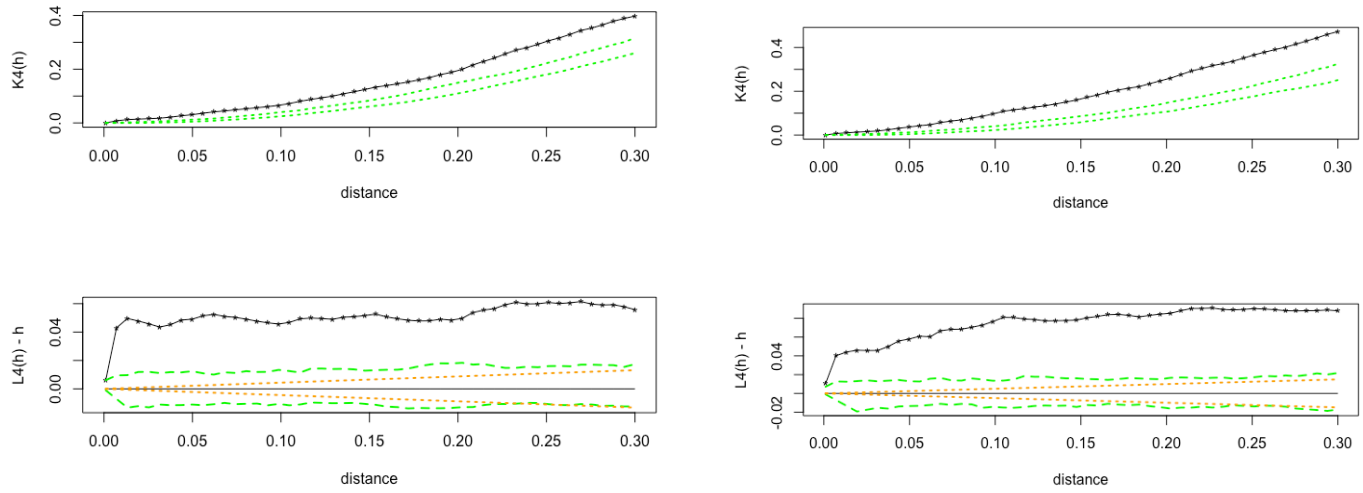


Figure 5. Fitted inhomogeneous background rate (left) and lambda (right) for 2019 data (top) and 2020 data (bottom)

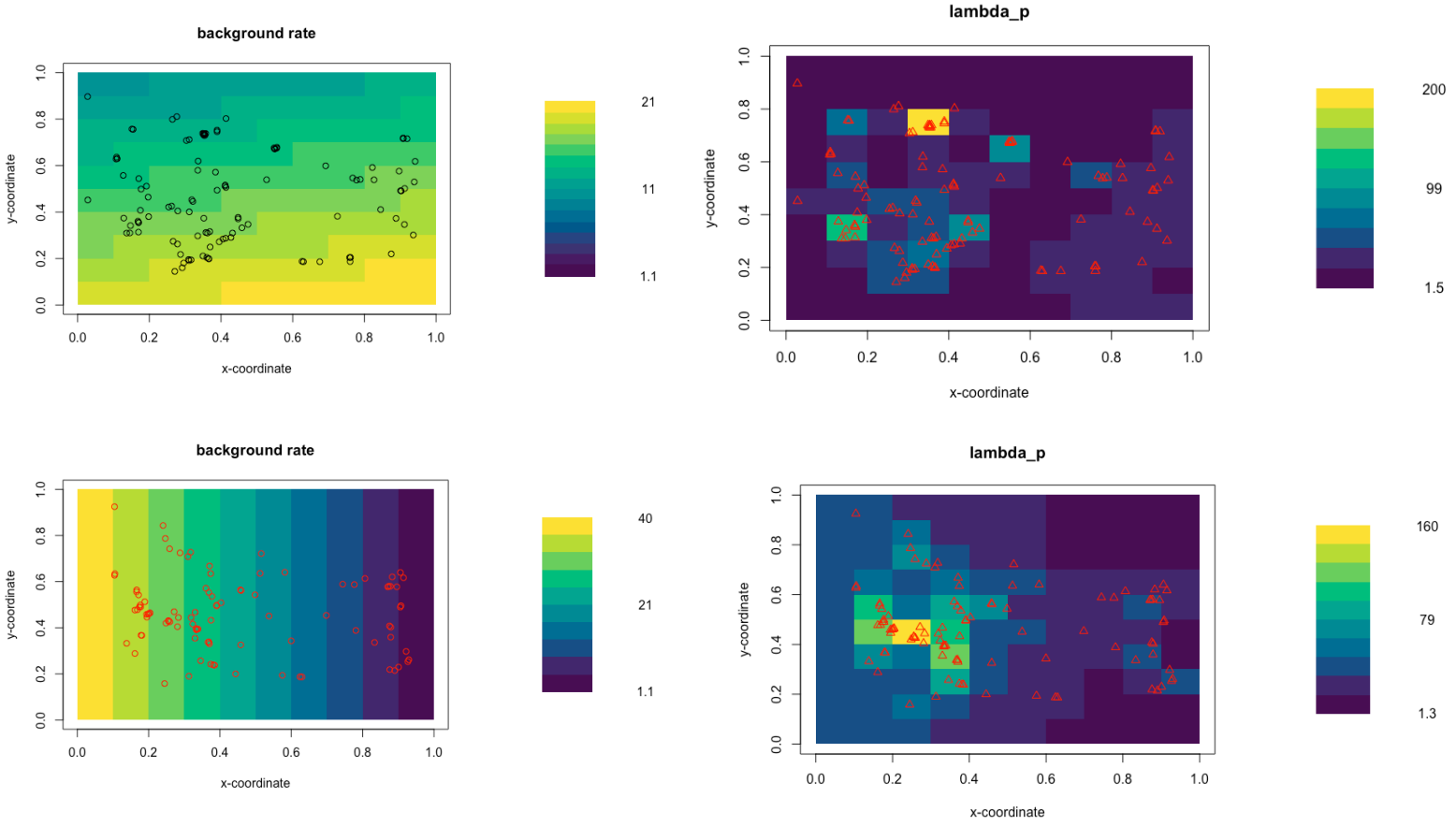


Figure 6. Overlay of original points and super-thinned points. Left: 2019 data; Right 2020 data

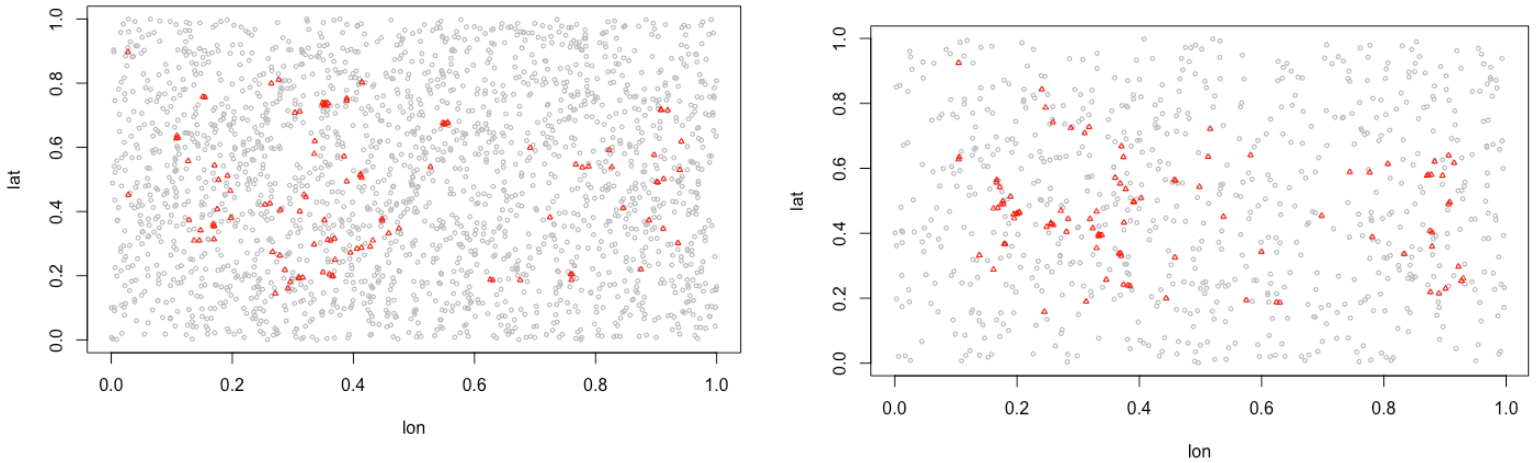
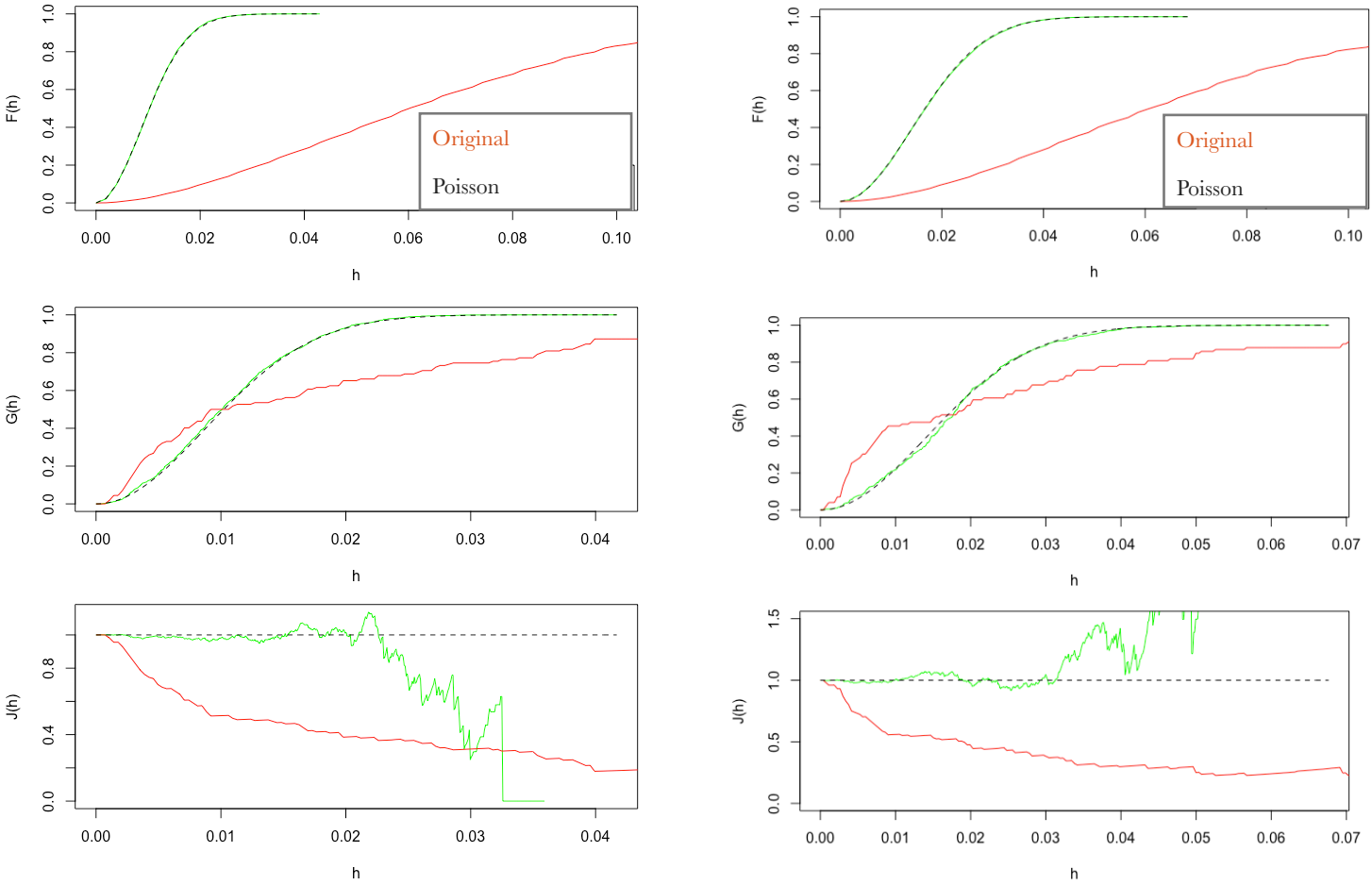


Figure 7. F, G, and J function for super-thinned points



```
######## Code
##### import data from April 2020
mydata20 = read.csv("dataApril2020.csv")
par(mfrow=c(1,1))
#plot(c(-118.48, -118.37), c(34.02,34.08),type="n",xlab="x-coordinate",ylab="y-coordinate",
plot(c(0, 1), c(0, 1), type="n", xlab="longitude", ylab="latitude", main="arrest in April 2020")
points(mydata20[, "lon"], mydata20[, "lat"],pch=3)

##### import data from April 2019
mydata19 = read.csv("dataApril2019.csv")
par(mfrow=c(1, 1))
plot(c(0, 1), c(0, 1), type="n", xlab="longitude", ylab="latitude", main="arrest in April 2019")
points(mydata19[, "lon"], mydata19[, "lat"], pch=3)


############### kernel smoothing for 2019 data
x1 = mydata19[, "lon"]
y1 = mydata19[, "lat"]
n = length(x1)
b1 = as.points(x1,y1)

############### kernel smoothing for 2020 data
x1 = mydata20[, "lon"]
y1 = mydata20[, "lat"]
n = length(x1)
b1 = as.points(x1,y1)

###  Kernel smoothing
stddist = sqrt(1/n*(sum((x1-mean(x1))^2)+sum((y1-mean(y1))^2))) ## standard distance
ds = sqrt((x1-mean(x1))^2+(y1-mean(y1))^2) ## distances to mean
dm = median(ds)
bdw = .9*min(stddist,sqrt(1/log(2))*dm)*n^-.2
## this is the suggestion in
## https://desktop.arcgis.com/en/arcmap/10.3/tools/spatial-analyst-toolbox/how-kernel-density-works.htm

## bdw = sqrt(bw.nrd(x1)^2+bw.nrd(y1)^2)  ## another option for a default bandwidth
# bdw = .2 ## or just pick a reasonable bandwidth yourself, like this.
bdry = matrix(c(0,0,1,0,1,1,0,1,0,0),ncol=2,byrow=T)
z = kernel2d(b1,bdry,bdw)
par(mfrow=c(1,2))
image(z,col=hcl.colors(10),xlab="longitude",ylab="latitude")
# points(b1)
x4 = (0:100)/100*(max(z$z)-min(z$z))+min(z$z)
plot(c(0,10),c(.8*min(x4),1.2*max(x4)),type="n",axes=F,xlab="",ylab="")
image(c(-1:1),x4,matrix(rep(x4,2),ncol=101,byrow=T),add=T,col=hcl.colors(10))
text(2,min(x4),as.character(signif(min(x4),2)),cex=1)
text(2,(max(x4)+min(x4))/2,as.character(signif((max(x4)+min(x4))/2,2)),cex=1)
text(2,max(x4),as.character(signif(max(x4),2)),cex=1)
mtext(s=3,l=-3,at=1,"Rate (pts per unit area)")
```

```
######### K-function & L-function:
## par(mfrow=c(2,1)) ## if you want to make a 2x1 grid of plots
s = seq(.001,.3,length=50)
k4 = khat(b1,bdry,s)
plot(s,k4,xlab="distance",ylab="K4(h)",pch="*")
lines(s,k4)
lines(s,pi*s^2,lty=2)
L4 = sqrt(k4/pi)-s
plot(c(0,.3),range(L4),type="n",xlab="lag, h",ylab="L4(h) - h")
points(s,L4,pch="*")
lines(s,L4)
lines(s,rep(0,50),lty=2)

### CONFIDENCE BOUNDS FOR K-FUNCTION via simulation
k4conf = Kenv.csr(npts(b1), bdry, 1000, s)
plot(c(0,max(s)),c(0,max(k4conf$upper,k4)), type="n",xlab="distance",ylab="K4(h)")
points(s,k4,pch="*")
lines(s,k4)
lines(s,k4conf$upper,lty=3,col="green",lwd=2)
lines(s,k4conf$lower,lty=3,col="green",lwd=2)
L4upper = sqrt(k4conf$upper/pi) - s
L4lower = sqrt(k4conf$lower/pi) - s

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

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

#### F-function (empty-space function):
## The cumulative distribution function (cdf), F,
## of the distance from a fixed location to the nearest point of X.
## Lower F indicates clustering.
## If F(0.2) = 0.4, for instance, then
## 40% of locations are within distance 0.2 of a point of the process.
```

```
b2 = as.ppp(b1, W = c(0,1,0,1))
## the above convert the points into a "ppp" object,
## using as a window [0,1] x [0,1]
par(mfrow=c(1,1))
f4 = Fest(b2)
plot(f4)

#### G-function:
## the cdf, G, of the distance from a typical point to its nearest neighbor.
## Higher G indicates clustering.
## If G(0.2) = 0.9, then 90% of points have another point within 0.2 of them.
g4 = Gest(b2)
plot(g4)


#### J-function:
## J(r) = (1-G(r))/(1-F(r)).
## J = 1 corresponds to a stationary Poisson process.
## J < 1 indicates clustering. J > 1 indicates inhibition.
j4 = Jest(b2)
plot(j4)


#### load data 2020 to z
mydata20 = read.csv("dataApril2020.csv")
data20 = list()
data20$t = c()
data20$n=0
data20$lat = c()
data20$lon = c()
data20$t = mydata20$t
data20$lon = mydata20$lon
data20$lat = mydata20$lat
data20$n = length(mydata20$t)
View(data20)
z = data20

#### load data 2019 to z
mydata19 = read.csv("dataApril2019.csv")
data19 = list()
data19$t = c()
data19$n=0
data19$lat = c()
data19$lon = c()
data19$t = mydata19$t
data19$lon = mydata19$lon
data19$lat = mydata19$lat
data19$n = length(mydata19$t)
View(data19)
```

```
z = data19

#################################################################
### Fitting a Pseudo-Likelihood model.
## I'm using the model lambda_p ( z | z_1, ..., z_k) =
## mu + alpha x + beta y + gamma SUM_{i = 1 to k} a1 exp{-a1 D(z_i,z)}/(2piD(z_i,z)),
## where z = (x,y), and where D means distance.
## So, if gamma is positive, then there is clustering; otherwise inhibition

x1 = z$lon
y1 = z$lat
n = length(x1)
n1 = length(x1)
n2 = length(y1)
d1 = as.matrix(dist(cbind(x1,y1))) ## matrix of distances between pts

f = function(p){
## returns the negative pseudo log-likelihood
## p = (mu,alpha,beta,gamma,a1)
if(p[1] < 0) return(99999)
if(p[1] + p[2] < 0) return(99999)
if(p[1] + p[3] < 0) return(99999)
if(p[1] + p[2] + p[3] < 0) return(99999)
if(p[4] < 0) return(99999)
if(p[4] > 1) return(99999)
if(p[5] < 0) return(99999)
lam = p[1] + p[2] * x1 + p[3] * y1
for(i in 1:n1){
for(j in c(1:n1)[-i]){
lam[i] = lam[i] + p[4] * p[5] * exp(-p[5] * d1[i,j]) / (2*pi*d1[i,j])
}
}
if (min(lam) < 0) return (99999)
int2 = p[1] + p[2]/2 + p[3]/2 + p[4]*n1
## Note that this above is for a window of [0,1] x [0,1]
cat("integral = ",int2," negative loglikelihood = ",
int2-sum(log(lam)), "\n"," p = ",p,"\n")
## integral should be roughly n when it's done
return(int2-sum(log(lam)))
}
pstart = c(1, .13, 1, .1, 15)
fit1 = optim(pstart,f,control=list(maxit=500),hessian=T)
pend = fit1$par
f(pstart)  ## -109.52.
f(pend)    ## -221.96.
pend
sqrt(diag(solve(fit1$hess)))
```

```
### Plot the Model's Background Rate
par(mfrow=c(1,1))
plot(c(0,1),c(0,1),type="n",xlab="x-coordinate",ylab="y-coordinate",
main="background rate")
x2 = seq(0.05,0.95,length=10)
y2 = seq(0.05,0.95,length=10)
z2 = matrix(rep(0,(10*10)),ncol=10)
z3 = matrix(rep(0,(10*10)),ncol=10)
for(i in 1:10){
for(j in 1:10){
z2[i,j] = pend[1] + pend[2]*x2[i] + pend[3]*y2[j]
z3[i,j] = pstart[1] + pstart[2]*x2[i] + pstart[3]*y2[j]
}}
zmin = min(c(z2,z3))
zmax = max(c(z2,z3))
image(x2,y2,z2,col=hcl.colors(10),zlim=c(zmin,zmax),add=T)
points(x1,y1)
par(mfrow=c(1,2))
plot(c(0,1),c(0,1),type="n",xlab="x-coordinate",ylab="y-coordinate",
main="background rate")
x2 = seq(0.05,0.95,length=10)
y2 = seq(0.05,0.95,length=10)
z2 = matrix(rep(0,(10*10)),ncol=10)
z3 = matrix(rep(0,(10*10)),ncol=10)
for(i in 1:10){
for(j in 1:10){
z2[i,j] = pend[1] + pend[2]*x2[i] + pend[3]*y2[j]
z3[i,j] = pstart[1] + pstart[2]*x2[i] + pstart[3]*y2[j]
}}
zmin = min(c(z2,z3))
zmax = max(c(z2,z3))
image(x2,y2,z2,col=hcl.colors(10),zlim=c(zmin,zmax),add=T)
points(x1,y1)
######### LEGEND:
zrng = zmax - zmin
zmid = zmin + zrng/2
plot(c(0,10),c(zmid-2*zrng/3,zmid+2*zrng/3),type="n",axes=F,xlab="",ylab="")
zgrid = seq(zmin,zmax,length=100)
## zgrid = vector of 100 equally-spaced numbers spanning range of the values.
image(c(-1:1),zgrid,matrix(rep(zgrid,2),ncol=100,byrow=T),add=T,col=hcl.colors(10))
text(2.5,zmin,as.character(signif(zmin,2)),cex=1)
text(2.5,zmax,as.character(signif(zmax,2)),cex=1)
text(2.5,zmid,as.character(signif(zmid,2)),cex=1)
text(4.5,zmid,"pts/unit area",srt=-90)

### PLOT LAMBDA_p on a 10 x 10 grid.
par(mfrow=c(1,2)) ## change this 3 to 2 for your projects.
plot(c(0,1),c(0,1),type="n",xlab="x-coordinate",ylab="y-coordinate",
```

```
main="lambda_p")
x2 = seq(0.05,0.95,length=10)
y2 = seq(0.05,0.95,length=10)
zz2 = matrix(rep(0,(10*10)),ncol=10)
zz3 = matrix(rep(0,(10*10)),ncol=10)
for(i in 1:10){
for(j in 1:10){
zz2[i,j] = pend[1] + pend[2] * x2[i] + pend[3] * y2[j]
zz3[i,j] = pstart[1] + pstart[2] * x2[i] + pstart[3] * y2[j]
for(k in c(1:n1)){
zz2[i,j] = zz2[i,j] + pend[4] * pend[5] * exp(-pend[5] *
sqrt((x2[i]-x1[k])^2+(y2[j]-y1[k])^2))
zz3[i,j] = zz3[i,j] + pstart[4] * pstart[5] * exp(-pstart[5] *
sqrt((x2[i]-x1[k])^2+(y2[j]-y1[k])^2))
}
}
}
zmin = min(c(zz2,zz3))
zmax = max(c(zz2,zz3))
image(x2,y2,zz2,col=hcl.colors(10),zlim=c(zmin,zmax),add=T)
points(x1,y1, col="red", pch=2)
######### LEGEND:
zrng = zmax - zmin
zmid = zmin + zrng/2
plot(c(0,10),c(zmid-2*zrng/3,zmid+2*zrng/3),type="n",axes=F,xlab="",ylab="")
zgrid = seq(zmin,zmax,length=100)
## zgrid = vector of 100 equally-spaced numbers spanning range of the values.
image(c(-1:1),zgrid,matrix(rep(zgrid,2),ncol=100,byrow=T),add=T,col=hcl.colors(10))
text(2.5,zmin,as.character(signif(zmin,2)),cex=1)
text(2.5,zmax,as.character(signif(zmax,2)),cex=1)
text(2.5,zmid,as.character(signif(zmid,2)),cex=1)
text(4.5,zmid,"pts/unit area",srt=-90)


#################################################################
############ fit Hawkes
T = 31
X1 = 1
Y1 = 1
M0 = 3.5

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

## Make sure the data are stored in z, and you define T,X1,Y1, and M0 externally.
## First we will write the loglikelihood function in R.
loglhawk = function(theta,draw=0){
  mu = theta[1]; K = theta[2]; alpha = theta[3]; beta = theta[4]
  cat("\n mu = ",m3(mu),", K = ",m3(K),", alpha = ",m3(alpha),", beta = ",m3(beta),".\n")
  if(min(mu,K,alpha,beta)<0.000000001) return(99999)
```

```
 if(K>.99999) return(99999)
 if(draw){
  r = seq(0,3,length=100)
  t = alpha/pi * exp(-alpha * r^2)
  lines(r,t,col="orange",lty=2)
 }
 sumlog = log(mu/X1/Y1)
 intlam = mu*T + K*z$n
 const = K*alpha/pi*beta
 for(j in 2:(z$n)){
  gij = 0
  for(i in 1:(j-1)){
   r2 = (z$lon[j]-z$lon[i])^2+(z$lat[j]-z$lat[i])^2
   gij = gij + exp(-beta*(z$t[j]-z$t[i])-alpha*r2)
  }
  lamj = mu / X1 / Y1 + const*gij
  if(lamj < 0){
   cat("lambda ",j," is less than 0.")
   return(99999)
  }
  sumlog = sumlog + log(lamj)
 }
 loglik = sumlog - intlam
 cat("loglike is ", loglik, ". sumlog = ", sumlog,". integral = ", intlam,".\n")
 if(draw) lines(r,t,col="white",lty=2)
 return(-1.0*loglik)
}


theta1 = c(2,0.5, 40, 40)/2
b1 = optim(theta1,loglhawk)
b2 = optim(b1$par,loglhawk,hessian=T)
theta2 = b2$par
sqrt(diag(solve(b2$hess))) ## for SEs


mu = theta2[1]; K = theta2[2]; alpha = theta2[3]; beta=theta2[4]
lambda = rep(mu/X1/Y1,z$n)
const = K*alpha/pi*beta
for(j in 2:(z$n)){
 gij = 0
 for(i in 1:(j-1)){
  r2 = (z$lon[j]-z$lon[i])^2+(z$lat[j]-z$lat[i])^2
  gij = gij + exp(-beta*(z$t[j]-z$t[i])-alpha*r2)
 }
 lambda[j] = mu / X1 / Y1 + const*gij
}
mean(lambda)
```

```
###### f function and super-thin
f = function(t,x,y,z){
  ## compute lambda(t,x,y) given data, z.
  const = K*alpha/pi*beta
  gij = 0
  j = 0
  if(t > z$t[1]) j = max(c(1:z$n[1])[z$t<t])
  if(j>0) for(i in 1:j){
    r2 = (x-z$lon[i])^2+(y-z$lat[i])^2
    gij = gij + exp(-beta*(t-z$t[i])-alpha*r2)
  }
  mu / X1 / Y1 + const*gij
}


s = supthin(z,lambda,f)

####### superthin func ###
supthin = function(z,lambda,f,b=mean(lambda)){
  ## z = data, lambda = conditional intensity at pts, f = function to compute lambda,
  ## and b = resulting rate.
  ## First thin, then superpose
  keepz = list()
  for(i in 1:z$n){
    if(runif(1) < b/lambda[i]){
      keepz$t = c(keepz$t,z$t[i])
      keepz$lon = c(keepz$lon,z$lon[i])
      keepz$lat = c(keepz$lat,z$lat[i])
    }
  }
  candn = rpois(1,b*X1*Y1*T)
  candt = sort(runif(candn)*T)
  candx = runif(candn)*X1
  candy = runif(candn)*Y1
  for(i in 1:candn){
    v = f(candt[i],candx[i],candy[i],z)
    if(v < b){
      if(runif(1) < (b-v)/b){
        keepz$t = c(keepz$t,candt[i])
        keepz$lon = c(keepz$lon,candx[i])
        keepz$lat = c(keepz$lat,candy[i])
      }}
  }
  keepz$lon = keepz$lon[order(keepz$t)]
  keepz$lat = keepz$lat[order(keepz$t)]
  keepz$t = sort(keepz$t)
  keepz$n = length(keepz$t)
  keepz
}
```

```
par(mfrow=c(1,2))
plot(z$lon,z$lat,pch=3,cex=.5,xlab="lon",ylab="lat",main="original pts.")
plot(s$lon,s$lat,pch=1,cex=.1,xlab="lon",ylab="lat",main="superthinned points")
par(mfrow=c(1,1))
plot(s$lon,s$lat,pch=1,cex=.5, col="gray", xlab="lon",ylab="lat")
points(z$lon,z$lat,pch=2,cex=.5,col="red")


s2 = as.ppp(as.points(s$lat, s$lon), W = c(0,1,0,1))
d2 = as.ppp(as.points(z$lat, z$lon), W=c(0, 1,0,1))
s2 = as.ppp(as.points(s$lat, s$lon), W = c(0,1,0,1))
d2 = as.ppp(as.points(z$lat, z$lon), W=c(0, 1,0,1))

f4 = Fest(s2)
f40 = Fest(d2)
par(mfrow=c(1,1))
plot(c(0,0.1), c(0, 1),type="n",xlab="h",ylab="F(h)",
    main="F function")
lines(f4$r[f4$r<.5],f4$rs[f4$r<.5],type="l", lty=1,col="green")
lines(f40$r[f40$r<.5],f40$rs[f40$r<.5],lty=1,col="red")
lines(f4$r[f4$r<.5],f4$theo[f4$r<.5],type="l",lty=2)
legend(.07,.2,lty=c(1,2),legend=c("original","Poisson","superthined data"))

g4 = Gest(s2)
g40=Gest(d2)
plot(g4$r[g4$r<.3],g4$rs[g4$r<.3],xlab="h",ylab="G(h)",type="l",lty=1, col="green")
lines(g40$r[g40$r<.3],g40$rs[g40$r<.3],lty=1, col="red")
lines(g4$r[g4$r<.3],g4$theo[g4$r<.3],lty=2)
legend(.2,.2,lty=c(1,2),legend=c("superthined data","Poisson"))

j4 = Jest(s2)
j40 = Jest(d2)
plot(j4$r[j4$r<.3],j4$rs[j4$r<.3],xlab="h",ylab="J(h)",type="l",lty=1, col="green")
lines(j40$r[j40$r<.3],j40$rs[j40$r<.3],lty=1, col="red")
lines(j4$r[j4$r<.3],j4$theo[j4$r<.3],lty=2)
legend(.2,.2,lty=c(1,2),legend=c("data","Poisson"))
```