# Final Crime

*Italo Sayan*

*11/19/2017*

## Used libraries and presets

```
library(rgdal)
library(birk)
library(data.table)
library(geosphere)
library(ggplot2)
library(mvtnorm)
library(MASS)
library(KernSmooth)
library(scales)
library(FNN)
library(fields)
library(kedd)
options(scipen=2)
tablesort<-function(x){sort(table(x, useNA = c("always")),decreasing = TRUE)}
normalize <- function(x) {return ((x - min(x)) / (max(x) - min(x)))}
```

## Import Crime Dataset

```
setwd("~/Desktop/policedata/CRIME DATA IN ARC GIS SHAPEFILE FORMAT/2016")
foo <- readOGR(dsn=".",layer="Jan1toDec31_2016")
foo.df <- as(foo, "data.frame")
#Datetime variable fix
foo.df$DATE_ON <- as.Date(foo.df$DATE_ON,format='%Y/%m/%d')
foo.df$datetime <- as.POSIXct(paste(foo.df$DATE_ON, foo.df$TIME, sep=" "))
```

## Subset and Clean Property Crime

```
burglary <- c("Burglary Habitation")
propertybur <- foo.df[foo.df$CRIME %in% burglary,]
# Clean invalid latitudes and duplicates
propertybur <- propertybur[!(propertybur$LATITUDE == 0),]
propertybur <- propertybur[!duplicated(propertybur$CASE_NUM),]
rm(foo,foo.df,burglary)
```

# Declustering algorithm

We use the following Monte-Carlo based iterative procedure:

Given point data $(t_k, x_k, y_k)_{k=1}^{N}$ and a self-exciting point process model of the form,

$$\lambda(t,x,y) = \nu(t)\mu(x,y) + \sum_{\{k:\ t_k<t\}} g(t-t_k, x-x_k, y-y_k), \qquad (13)$$

we iterate the following until convergence:

Step 1) Sample background events $\{(t_i^b, x_i^b, y_i^b)\}_{i=1}^{N_b}$ and offspring/parent inter-point distances $\{(t_i^o, x_i^o, y_i^o)\}_{i=1}^{N_o}$ from $P_{n-1}$.

Step 2) Estimate $\nu_n$, $\mu_n$ and $g_n$ from the sampled data.

Step 3) Update $P_n$ from $\nu_n$, $\mu_n$ and $g_n$ using (8) and (9).

## Sample background events and interpoint distances

```r
#Background Events. Output is 'tenevents' dataframe

total <- 400
tenevents <- propertybur[sample(1:nrow(propertybur),total),c('CASE_NUM','datetime','LONGITUDE','LATITUDE')]
tenevents <- tenevents[order(tenevents$datetime),]
#Interpoint Distances. Output is 'ginputs' dataframe
#{(ti − tj, xi − xj, yi − yj, pji)}i>j
n <- choose(total,2)
ginputs <- data.table(index=rep(0,n),t=rep(0,n), x=rep(0,n), y=rep(0,n))
ginputs$index <- as.integer(paste0(combn(1:total,2, simplify = TRUE)[1,],combn(1:total,2, simplify = TRUE)[2
,],sep=''))
ginputs$t <-as.numeric(dist(tenevents$datetime)) #In seconds
ginputs$x <-c(as.dist(distm(cbind(tenevents$LONGITUDE,0),fun=distGeo)))#x-distance LONGITUDE
ginputs$y <-c(as.dist(distm(cbind(0,tenevents$LATITUDE),fun=distGeo)))#y-distance LATITUDE
ginputs <- as.data.frame(ginputs)
```

### Normalize Data

```r
# Raw Data
interpoint<- ginputs
background<- tenevents
# Normalized
tenevents[2:4] <- as.data.frame(lapply(tenevents[2:4],function(x)normalize(as.numeric(x))))
ginputs[2:4] <- as.data.frame(lapply(ginputs[2:4],function(x)normalize(x)))
```

### Estimate gaussian kernels from sampled data

Kernel Density Estimation. To estimate $g_n$, we first scale the data $\{(t_i^o, x_i^o, y_i^o)\}_{i=1}^{N_o}$ to have unit variance in each coordinate and based upon the rescaled data compute $D_i$, the kth nearest neighbor distance (3-dimensional Euclidean distance) to data point $i$. We then transform the data back to its original scale and, letting $\sigma_x$, $\sigma_y$, and $\sigma_t$ be the sample standard deviation of each coordinate, estimate the triggering function as,

$$g_n(t,x,y) = \frac{1}{N}\sum_{i=1}^{N_o} \frac{1}{\sigma_x\sigma_y\sigma_t(2\pi)^{(3/2)}D_i^3} \exp\left(-\frac{(x-x_i^o)^2}{2\sigma_x^2 D_i^2} - \frac{(y-y_i^o)^2}{2\sigma_y^2 D_i^2} - \frac{(t-t_i^o)^2}{2\sigma_t^2 D_i^2}\right).$$

```r
# Vn:1d Fixed Gaussian.
v <- density(tenevents$datetime, bw = ucv(tenevents$datetime) )
# Un:2d Fixed Gaussian.
u <- kde2d(tenevents$LONGITUDE, tenevents$LATITUDE,n=50, h = c( ucv(tenevents$LONGITUDE), ucv(tenevents$LATI
TUDE)) )
# Gn:3d Adaptive Gaussian Self excitting part
samplesave <-1
gn <- function(t_q,x_q,y_q,sampleds){
  if(samplesave != 2){
  #Saving sampleds repeating parameters
  gmindt<<-min(sampleds$t)
  gmaxdt<<-max(sampleds$t)
  gminlo<<-min(sampleds$x)
  gmaxlo<<-max(sampleds$x)
  gminla<<-min(sampleds$y)
  gmaxla<<-max(sampleds$y)
  norm_sampleds <<- as.data.frame(lapply(sampleds[2:4],function(x)normalize(x)))
  scaled.dat<<-scale(norm_sampleds)
  varX <<- var(norm_sampleds$x)
  varY <<- var(norm_sampleds$y)
  varT <<- var(norm_sampleds$t)
  Di <<- knnx.dist(scaled.dat,matrix(scaled.dat,ncol=3), k=15,algorithm=c("kd_tree"))[,15]
  }
  samplesave <<- 2
  # Normalization for new points
  t_q <- (t_q - gmindt) / (gmaxdt - gmindt)
  x_q <- (x_q - gminlo) / (gmaxlo - gminlo)
  y_q <- (y_q - gminla) / (gmaxla - gminla)
  # Adaptive Kernel Density Estimation
  sum1 <- ((x_q-norm_sampleds$x)^2)/(2*(varX)*Di^2)
  sum2 <- ((y_q-norm_sampleds$y)^2)/(2*(varY)*Di^2)
  sum3 <- ((t_q-norm_sampleds$t)^2)/(2*(varT)*Di^2)
  element = exp(-sum1-sum2-sum3)/(sqrt(varX)*sqrt(varY)*sqrt(varT)*((2*pi)^(3/2))*Di^3)
  return(sum(element)/nrow(sampleds))
}
# Testing gn function
gn(13478400,6264.14645,12576.79269,interpoint)
```

Lambda function

## 3 A self-exciting point process model of burglary

For the purpose of modeling burglary we consider an unmarked self-exciting model for the conditional intensity of the form,

$$\lambda(t,x,y) = \nu(t)\mu(x,y) + \sum_{\{k:\ t_k<t\}} g(t-t_k, x-x_k, y-y_k). \tag{10}$$

```r
samplesave <-1
lambda <- function(t_q,x_q,y_q,sample){
  #Normalizing inputs
  mindt<<-min(as.numeric(as.POSIXct(sample$datetime)))
  maxdt<<-max(as.numeric(as.POSIXct(sample$datetime)))
  minlo<<-min(sample$LONGITUDE)
  maxlo<<-max(sample$LONGITUDE)
  minla<<-min(sample$LATITUDE)
  maxla<<-max(sample$LATITUDE)
  nt_q <- as.numeric(as.POSIXct(t_q))
  nt_q <- (nt_q - mindt) / (maxdt - mindt)
  nx_q <- (x_q - minlo) / (maxlo - minlo)
  ny_q <- (y_q - minla) / (maxla - minla)
  # Vn and Un part
  v_part <- v$y[which.closest(v$x,nt_q)]
  u_part <- interp.surface(u,matrix(c(nx_q,ny_q),nrow=1,ncol = 2))
  # Sum the g part up to time t_q
  up_to_t <- sample[sample$datetime <= t_q,]
  n <- nrow(up_to_t)
  toT <- data.frame(t=rep(t_q,n),x=rep(x_q,n),y=rep(y_q,n))
  toT$t <- as.numeric(as.POSIXct(toT$t) - as.POSIXct(up_to_t$datetime))
  toT$x <- sapply(up_to_t$LONGITUDE,function(x){distGeo(c(x_q,0),c(x,0))})
  toT$y <- sapply(up_to_t$LATITUDE,function(x){distGeo(c(0,y_q),c(0,x))})
  sumg<-sum(apply(toT, 1, function(x) gn(x[1],x[2],x[3],interpoint)))
  return((v_part*u_part)+sumg)
}
lambda("2016-12-16 07:00:00 EDT",-98.46717,29.43049,background)
```

## Matrix P Calculation

Assuming model correctness, the probability that event $i$ is a background event, $p_{ii}$, is given by,

$$p_{ii} = \frac{\mu(t_i, x_i, y_i)}{\lambda(t_i, x_i, y_i)}, \tag{8}$$

and the probability that event $j$ triggered event $i$, $p_{ji}$, is given by,

$$p_{ji} = \frac{g(t_i - t_j, x_i - x_j, y_i - y_j)}{\lambda(t_i, x_i, y_i)}, \tag{9}$$

```r
# Pii:Background events vector
Pb_u <- apply(tenevents,1,function(x)interp.surface(u,matrix(c(x[3],x[4]),nrow=1,ncol = 2)))
Pb_l <- apply(background,1,function(x)lambda(x[2],as.numeric(x[3]),as.numeric(x[4]),background))
Pii <- Pb_u/Pb_l
# Pji :Interpoint events vector
gd <- apply(interpoint,1,function(x)gn(x[2],x[3],x[4],interpoint))
Pb_lcool <- unlist(sapply(seq_along(Pb_l[-1]), function(i) Pb_l[-1][i:length(Pb_l[-1])]))
Pji <- gd/Pb_lcool
```

## Prediction

```r
# Use P and Lambda to designate crime of new locations
#u_part
#For every day k of 2005, each model assesses the risk of burglary within each of M2cells
#partitioning an 18km by 18km region of the San Fernando Valley in Los Angeles. Based on
#the data from the beginning of 2004 up through day k, the N cells with the highest risk
#(value of λ) are flagged yielding a prediction for day k + 1. The percentage of crimes falling
#within the flagged cells on day k + 1 is then recorded and used to measure the accuracy of
#each model.
```