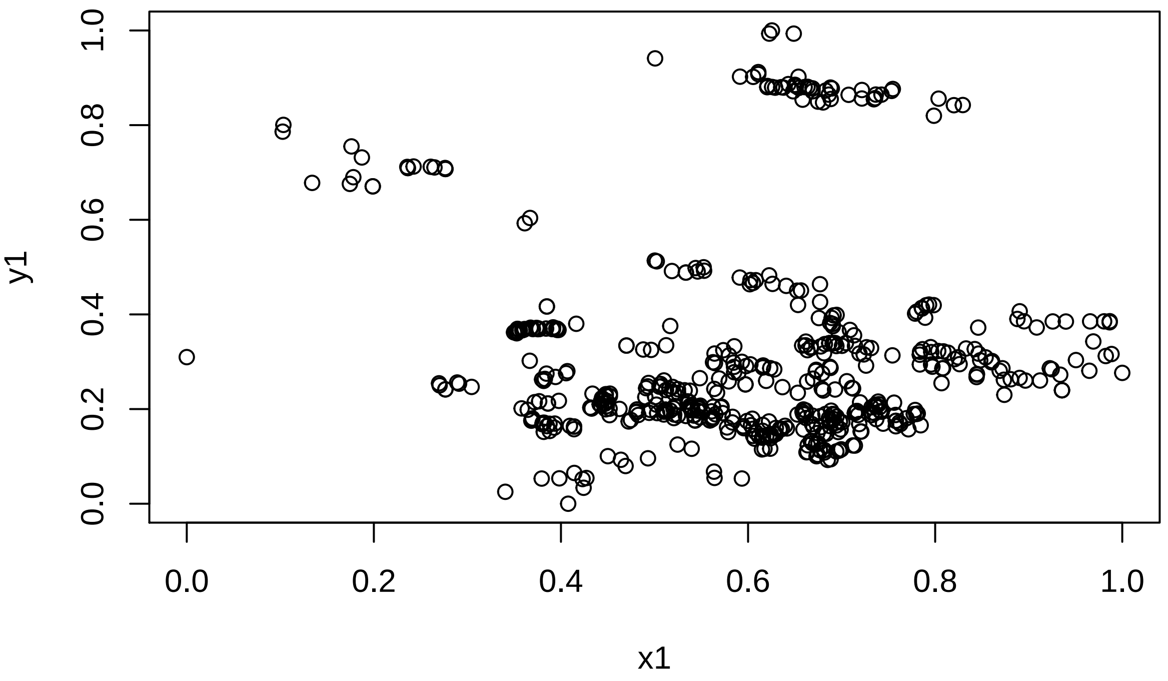
MATH 531 EXAM 3

Gustavo Esparza

6/22/2019

PROBLEM 1

Here is a plot of our original data points.



We can see that there is a larger cluster of points in the bottom right corner of our plot, so we should expect a high intensity rate for this region.

A - Kernel and Bandwidth Selection

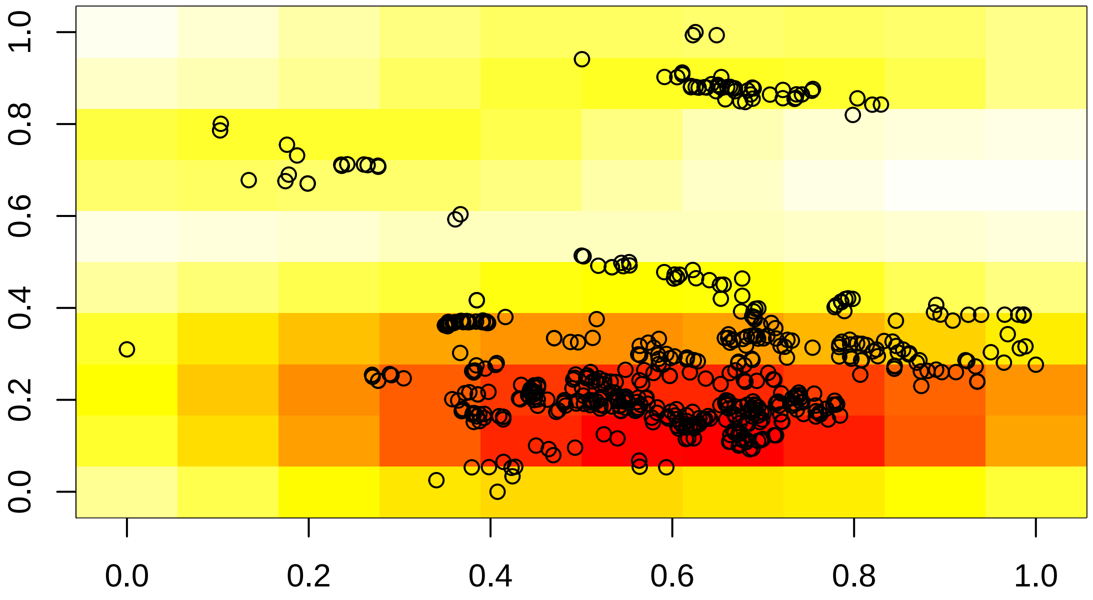
I chose a gaussian kernel because I wanted to create my intensity map with a normal assumption for the distribution of our data. It is also a kernel that I have worked with in the past, and seemed to be reliable in many situations. The bandwidth for Variances and Covairances was computed via ten fold cross validation, and will be displayed prior to the intensity rate plot. When considering bandwidth selection with our cross validated optimization, we are attempting to minimize the log-likelihood values until convergence.For the sake of this problem, we chose 20 iterations but full convergence is desired in real-world applications.

```
##  Bandwidth Selection:
##  Variance of X values = 0.06833333
##  Variance of Y values =  0.003333333
##  Covariance of X and Y =  0.006666667
```

We will be using these bandwidth values to model our intensity rates.

B - Background Rate With Data

Here is a plot of the background rate with the original data plotted over it:



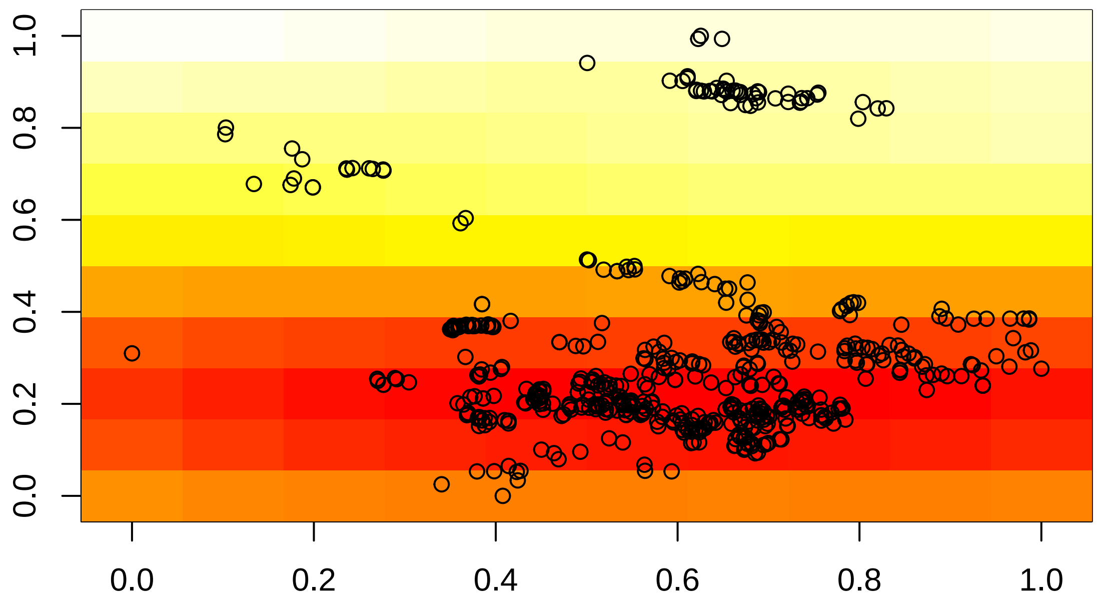
we can see that the main intensity is located in that lower right cluster, with other areas of frequent crime also having some intensity being shaded as well.

C Increase bandwidth by a magnitude of 10:

Now, we will increase the magnitude of our bandwidth and plot the data once again.

```
##  Increased Bandwidth Selection:
##  Variance of X values = 0.6833333
##  Variance of Y values =  0.03333333
##  Covariance of X and Y =  0.06666667
```

Here is the new plot:



We can see that the main intensity regions are still being shaded accordingly, but now we have adjacent areas of low crime rates still being identified as a high intensity location.

D - Comparison of two bandwiths:

When computing our kernel smoothing of the intensity rates for the LA County Crime Data, we used an optimization process that attempted to minimize the log-likelihood of our kernel smoothing. Here are the two seperate log-likelihoods for our bandwidth choices:

```
## Log-Likelihood of first bandwidth =  294.5331
```

```
## Log-Likelihood of increased bandwidth =  384.1304
```

We can see that the larger bandwidth resulted in a larger log-likelihood value. This is reflected in the background rate plot when taking into consideration the areas of low/no crime being identified with high intensity. This larger bandwidth is causing the intensity to be increased, seen in the log-likelihood values, when in adjacency to a region that does indeed have high crime rates.

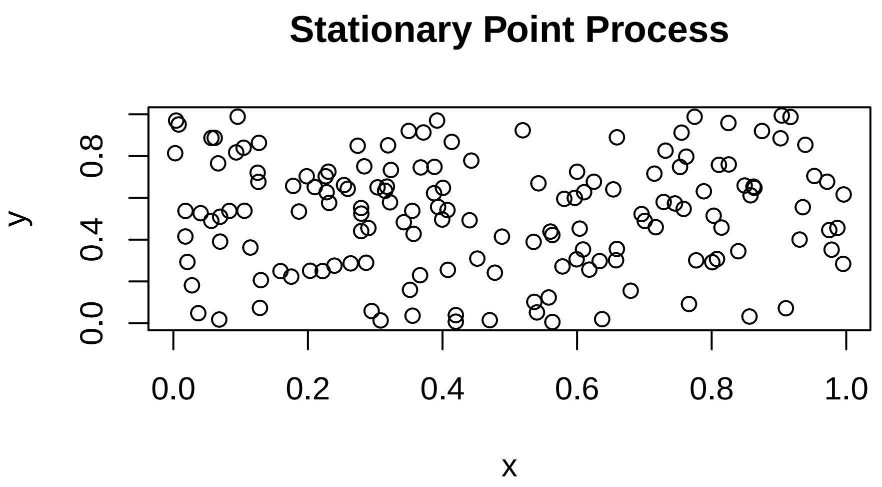
Essentially, the log-likelihood is being computed to be larger than it needs to be for a large amount of the bins that have little to no points contained within.

PROBLEM 2

DATASET 1 - STATIONARY POINT PROCESS

Here is a plot of a stationary point process.

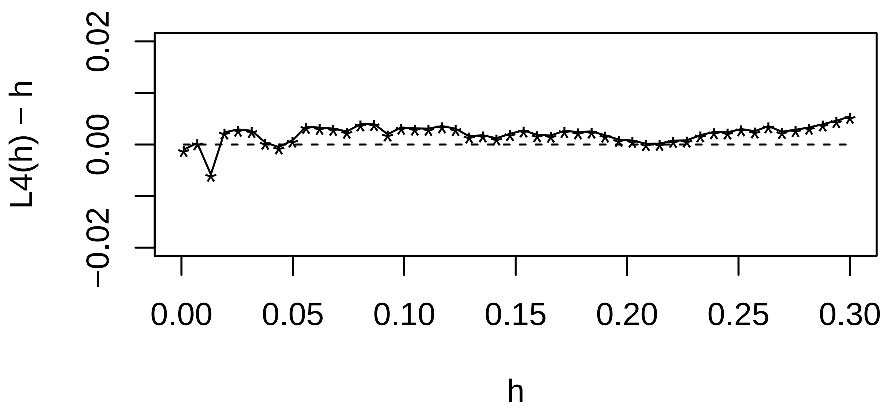
PLOT



We can see that the data is evenly distributed throughout our space, implying the desired stationarity.

L

Here is a plot of $L(\hat{h}) - h$ against h for the stationary point process:

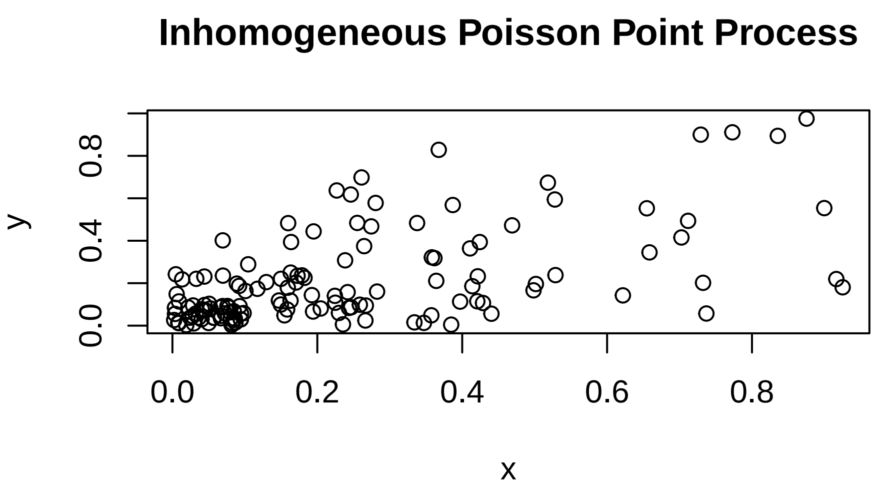


We can see that our function stays very close to the origin, fluctuating between the positive and negative values. This implies the stationarity of our process, as it never leads towards clustering or inhibition.

DATASET 2 - INHOMOGENEOUS POISSON POINT PROCESS

Here is a plot of an Inhomogeneous Poisson point process.

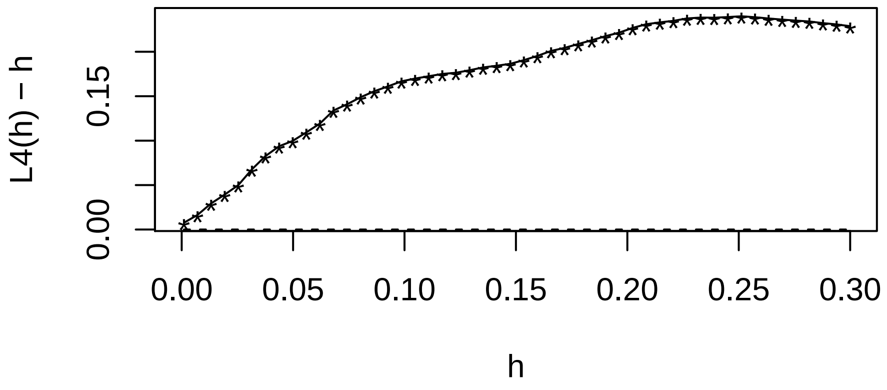
PLOT



We can see the inhomogeneous process being represented by a large cluster of data located in the bottom left corner.

L

Here is a plot of $L(\hat{h}) - h$ against h for the Inhomogeneous Poisson point process:



We can see a clear indication of clustering, denoted by increasingly large values above the x-axis.

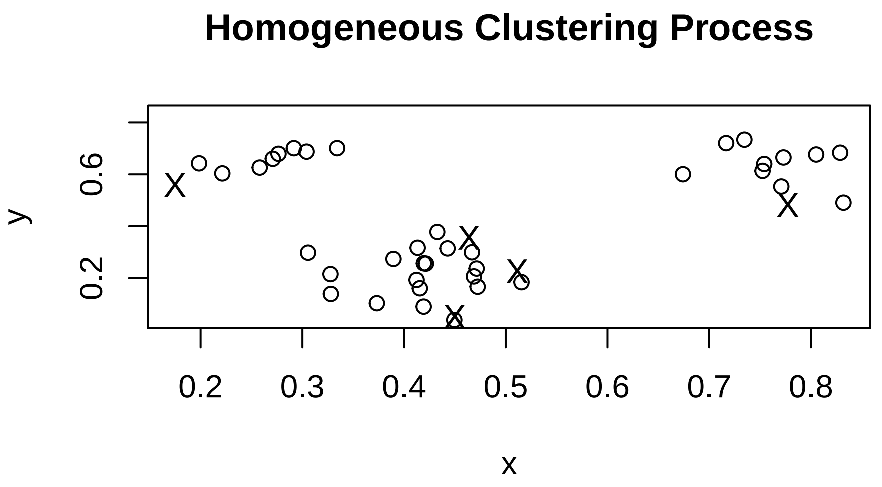
APPARENT CLUSTERING/INHIBITION

We can see that the L function plot for the Inhomogeneous Poisson Point Process appears to be the same as the Clustering Process, indicated by large values above the x-axis. Although we do have clustering evident in our inhomogeneous point process, we are modeling the L(4) function without having the neccessary assumption of a homogeneous process. We are essentially fitting a model to a process that makes no sense for an inhomogeneous process.

DATASET 3 - HOMOGENEOUS CLUSTERING PROCESS

Here is a plot of our homogeneous clustering process:

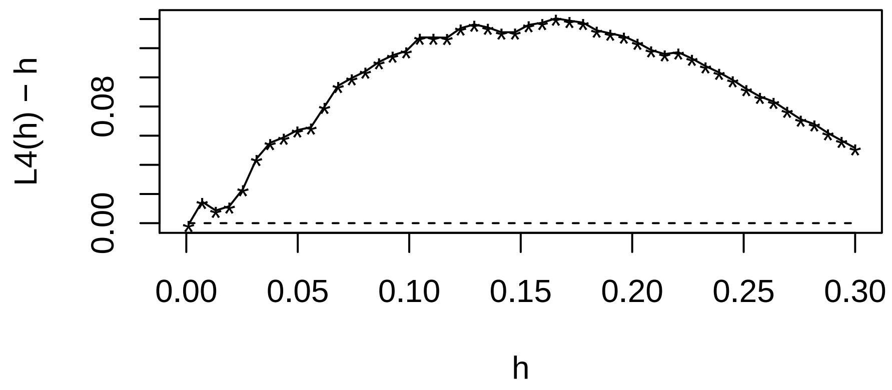
PLOT



We can clearly see the clusters located in our process, there are about five in total.

L

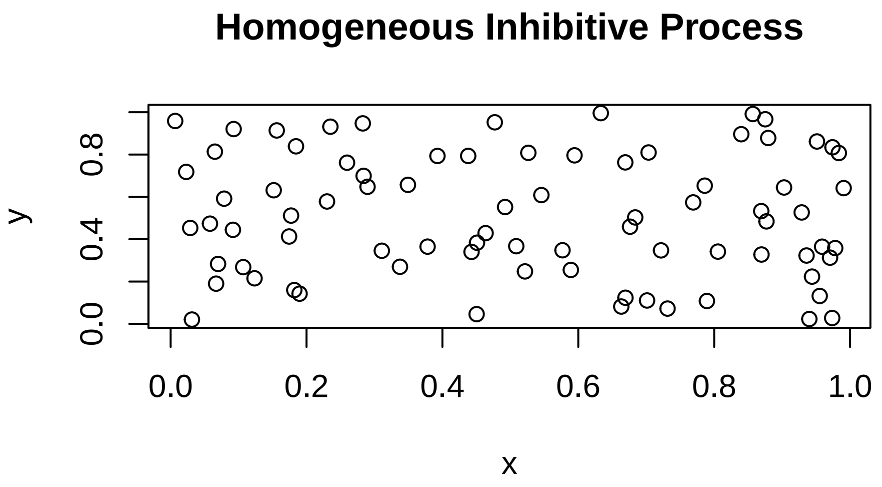
Here is a plot of $L(\hat{h}) - h$ against h for the homogeneous clustering point process:



We can clearly see the clustering effect being displayed by the large values residing above the x-axis. These values also continue to rise as the distance (h) is increased.

DATASET 4 - HOMOGENEOUS INHIBITIVE PROCESS

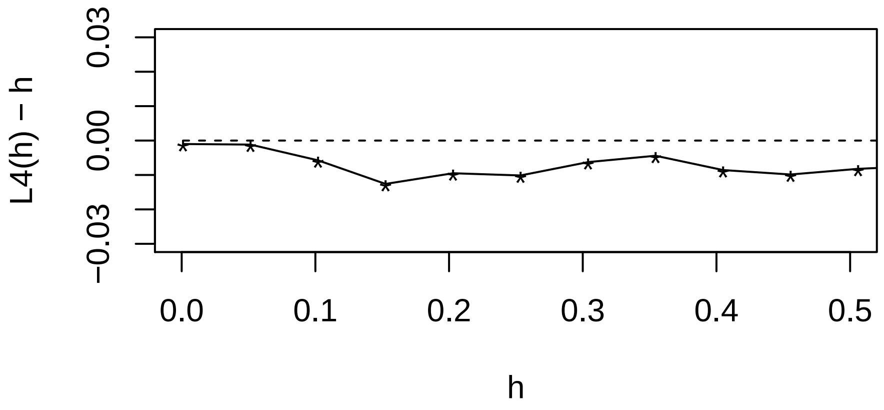
Here is a plot of our Homogeneous Inhibitive Process:



We can see that our process appears similar to the stationary point process, but there is a larger distance between the points displayed. This is a good indication of inhibition being effectively applied.

L

Here is a plot of $L(\hat{h}) - h$ against h for the homogeneous inhibitive point process:



We can see that our function is consistently below the x-axis, which implies the inhibitive effect being present in our process. It is important to note that our L(4) - h values are not as large in magnitude as the clustering process. This may be due to the amount of points or space provided not being large enough to show a greater inhibition effect, but we can still see the inhibitive effect displayed on our plots.

APPENDIX

PROBLEM 1

Here is a plot of our original data points.

```r
data = read.csv("LA County Crime Data.csv",h=T)

data = data[,c(9,10)]
data = na.omit(data)

data$x = (data$X_COORDINATE-min(data$X_COORDINATE))/(max(data$X_COORDINATE)-min(data$X_COORDINATE))
data$y = (data$Y_COORDINATE-min(data$Y_COORDINATE))/(max(data$Y_COORDINATE)-min(data$Y_COORDINATE))
n = length(data$x)
```

```r
x1=data$x
y1=data$y
plot(x1,y1)
```

A - Kernel and Bandwidth Selection

```r
#Distance Matrix
kij = matrix(0,n,n)
for(i in 1:n){
    for(j in 1:n){
        kij[i,j] = sqrt((data$x[i] - data$x[j])^2 + (data$y[i] - data$y[j])^2)
    }
  }
```

```r
#Shuffling
    shuffle.index = sample(1:n,n,replace=F)
    test.index = vector("list",10)
    for(k in 1:10){
        test.index[[k]] = shuffle.index[(1+11*(k-1)):(11*k)]
        }
```

```r
CVLL = function(h.vec){
    H = matrix(c(h.vec[1],h.vec[3],h.vec[3],h.vec[2]),ncol=2)

LL=0
for(k in 1:10){

    model.n = ceiling(n*.9)
    model.data = data[-test.index[[k]],]
    test.data = data[test.index[[k]],]
    ###STEP 2:  Do the kernel density estimation just using the model data

    Gbins = matrix(0,100,9)
    Gbins[,1] = 1:100
    Gbins[,2] = rep(seq(0,.9,.1),10)
    Gbins[,3] = Gbins[,2]+.1
    Gbins[,4] = rep(seq(0,.9,.1),rep(10,10))
    Gbins[,5] = Gbins[,4]+.1
    Gbins[,6] = .01
    for(i in 1:100){
        Gbins[i,7] = length(test.data$x[test.data$x >= Gbins[i,2] &
                                           test.data$x < Gbins[i,3] &
                                           test.data$y >= Gbins[i,4] &
                                           test.data$y < Gbins[i,5]])
    }

    Gbins.Distance.x = matrix(0,100,106)
    Gbins.Distance.y = matrix(0,100,106)

    for(i in 1:100){
        for(j in 1:106){
            Gbins.Distance.x[i,j] = abs(model.data$x[j] - (Gbins[i,2]+.05))
            Gbins.Distance.y[i,j] = abs(model.data$y[j] - (Gbins[i,4]+.05))
        }
    }

    Gbins.Dist = vector("list",100)
    for(i in 1:100){
        Gbins.Dist[[i]] = matrix(c(Gbins.Distance.x[i,],Gbins.Distance.y[i,]),ncol=2)
    }

    for(i in 1:100){
        Gbins[i,8] = (1/9)*Gbins[i,6]*sum((1/(sqrt(det(H))*2*pi))
                    *exp(-apply((Gbins.Dist[[i]]%*%solve(H)*Gbins.Dist[[i]]),1,sum)/2))
    }

    Gbins[,9] = log((Gbins[,8]^Gbins[,7])*exp(-Gbins[,8])/factorial(Gbins[,7]))
    LL = LL + sum(Gbins[,9])
}
print(-LL)
}
```

```
# optim(c(.05,.05,0),CVLL,control=list(maxit=20))
```

```
cov = c(0.068333333,0.003333333,0.006666667)
cat(" Bandwidth Selection:",
    "\n Variance of X values =",cov[1],
    "\n Variance of Y values = ",cov[2],
    "\n Covariance of X and Y = ",cov[3])
```

We will be using these bandwidth values to model our intensity rates.

B - Background Rate With Data

Here is a plot of the background rate with the original data plotted over it:

```
H = matrix(c(0.068,0.00666,0.00666,0.0033),ncol=2)
Gbins = matrix(0,100,8)
Gbins[,1] = 1:100
Gbins[,2] = rep(seq(0,.9,.1),10)
Gbins[,3] = Gbins[,2]+.1
Gbins[,4] = rep(seq(0,.9,.1),rep(10,10))
```

```r
Gbins[,5] = Gbins[,4]+.1
Gbins[,6] = .01

Gbins.Distance.x = matrix(0,100,117)
Gbins.Distance.y = matrix(0,100,117)

for(i in 1:100){
    for(j in 1:117){
        Gbins.Distance.x[i,j] = abs(data$x[j] - (Gbins[i,2]+.05))
        Gbins.Distance.y[i,j] = abs(data$y[j] - (Gbins[i,4]+.05))
    }
}

Gbins.Dist = vector("list",100)
for(i in 1:100){
    Gbins.Dist[[i]] = matrix(c(Gbins.Distance.x[i,],Gbins.Distance.y[i,]),ncol=2)
}

#Because our Bins are so small I consider a single midpoint approximation for the intergral.  \
#Note that I multiply by 1/5 (this is to get an integrated rate on a per game basis,
#our data consisted of 5 games so I discounted the rate.)

    for(i in 1:100){
        Gbins[i,8] = (1/5)*Gbins[i,6]*sum(1/(sqrt(det(H))*2*pi)
            *exp(-apply((Gbins.Dist[[i]]%*%solve(H)*Gbins.Dist[[i]]),1,sum)/2))
    }

zz = matrix(Gbins[,8],10,10,byrow=F)
image(1-zz,zlim = c(min(1-zz),max(1-zz)),col=heat.colors(100))
points(data$x,data$y)
```

we can see that the main intensity is located in that lower right cluster, with other areas of frequent crime also having some intensity being shaded as well.

C Increase bandwidth by a magnitude of 10:

Now, we will increase the magnitude of our bandwidth and plot the data once again.

```r
cat(" Increased Bandwidth Selection:",
    "\n Variance of X values =",cov[1]*10,
    "\n Variance of Y values = ",cov[2]*10,
    "\n Covariance of X and Y = ",cov[3]*10)
```

Here is the new plot:

```r
H = matrix(c(0.68,0.0666,0.0666,0.033),ncol=2)
Gbins = matrix(0,100,8)
Gbins[,1] = 1:100
Gbins[,2] = rep(seq(0,.9,.1),10)
Gbins[,3] = Gbins[,2]+.1
Gbins[,4] = rep(seq(0,.9,.1),rep(10,10))
Gbins[,5] = Gbins[,4]+.1
Gbins[,6] = .01
```

```r
Gbins.Distance.x = matrix(0,100,117)
Gbins.Distance.y = matrix(0,100,117)

for(i in 1:100){
    for(j in 1:117){
        Gbins.Distance.x[i,j] = abs(data$x[j] - (Gbins[i,2]+.05))
        Gbins.Distance.y[i,j] = abs(data$y[j] - (Gbins[i,4]+.05))
    }
}

Gbins.Dist = vector("list",100)
for(i in 1:100){
    Gbins.Dist[[i]] = matrix(c(Gbins.Distance.x[i,],Gbins.Distance.y[i,]),ncol=2)
}

    for(i in 1:100){
        Gbins[i,8] = (1/5)*Gbins[i,6]*sum(1/(sqrt(det(H))*2*pi)*
              exp(-apply((Gbins.Dist[[i]]%*%solve(H)*Gbins.Dist[[i]]),1,sum)/2))
    }
zz = matrix(Gbins[,8],10,10,byrow=F)
image(1-zz,zlim = c(min(1-zz),max(1-zz)),col=heat.colors(100))
points(data$x,data$y)
```

We can see that the main intensity regions are still being shaded accordingly, but now we have adjacent areas of low crime rates still being identified as a high intensity location.

D - Comparison of two bandwiths:

When computing our kernel smoothing of the intensity rates for the LA County Crime Data, we used an optimization process that attempted to minimize the log-likelihood of our kernel smoothing. Here are the two seperate log-likelihoods for our bandwidth choices:

```r
a =optim(c(0.068,0.00666,0.00666,0.0033),CVLL,control=list(maxit=1))
cat("Log-Likelihood of first bandwidth = ",294.5331)
```

```r
a =optim(c(0.68,0.0666,0.0666,0.033),CVLL,control=list(maxit=1))
cat("Log-Likelihood of increased bandwidth = ",384.1304)
```

PROBLEM 2

DATASET 1 - STATIONARY POINT PROCESS

Here is a plot of a stationary point process.

```r
n = rpois(1,150)
ds1 = data.frame(x=runif(n,0,1),y=runif(n,0,1))
```

PLOT

```r
plot(ds1$x,ds1$y,xlab="x",ylab="y",main="Stationary Point Process")
```

We can see that the data is evenly distributed throughout our space, implying the desired stationarity.

L

Here is a plot of $L(\hat{h}) - h$ against h for the stationary point process:

```r
x1=ds1$x
y1=ds1$y

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

```r
#par(mfrow=c(2,1))
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),c(-.02,.02),type="n",xlab="h",ylab="L4(h) - h")
points(s,L4,pch="*")
lines(s,L4)
lines(s,rep(0,50),lty=2)
```

We can see that our function stays very close to the origin, fluctuating between the positive and negative values. This implies the stationarity of our process, as it never leads towards clustering or inhibition.

DATASET 2 - INHOMOGENEOUS POISSON POINT PROCESS

Here is a plot of an Inhomogeneous Poisson point process.

```
n = round(rpois(1,150)/4)*4
ds2 = data.frame(x = c(runif(n/4,0,1),
                       runif(n/4,0,.5),runif(n/4,0,.25),runif(n/4,0,.1)),y = c(runif(n/4,0,1),runif(n/4
```

PLOT

```
plot(ds2$x,ds2$y,xlab="x",ylab="y",main="Inhomogeneous Poisson Point Process")
```

We can see the inhomogeneous process being represented by a large cluster of data located in the bottom left corner.

L

Here is a plot of $L(\hat{h}) - h$ against h for the Inhomogeneous Poisson point process:

```
x1=ds2$x
y1=ds2$y

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

```
#par(mfrow=c(2,1))
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="h",ylab="L4(h) - h")
points(s,L4,pch="*")
lines(s,L4)
lines(s,rep(0,50),lty=2)
```

DATASET 3 - HOMOGENEOUS CLUSTERING PROCESS

Here is a plot of our homogeneous clustering process:

```
clust = function(n.cluster=5,lambda=3,xmin=0,xmax=1,ymin=0,ymax=1,norm.sd=.5,plot=T){
  x.main = runif(n.cluster,xmin,xmax)
  y.main = runif(n.cluster,xmin,xmax)
  x.all = x.main
  y.all = y.main
  n.points = rpois(n.cluster,lambda)
  for(i in 1:n.cluster){
    if(n.points[i] > 0){
      for(j in 1:n.points[i]){
        x.points = cumsum(rnorm(n.points[i],0,norm.sd)) + x.main[i]
        y.points = cumsum(rnorm(n.points[i],0,norm.sd)) + y.main[i]
      }
    }
    else{
      x.points = c()
      y.points = c()
    }
    x.all = c(x.all,x.points)
    y.all = c(y.all,y.points)
  }
  x.added = x.all[(n.cluster+1):length(x.all)]
  y.added = y.all[(n.cluster+1):length(y.all)]
  plot(x.main,y.main,pch='X',cex=1.1,xlim=c(min(x.all)-.1*xmin,max(x.all)+.1*xmin),
      ylim=c(min(y.all)-.1
    *ymin,max(y.all)+.1*ymax),xlab="x",ylab="y",main="Homogeneous Clustering Process")
  points(x.added,y.added)
  ds3 = data.frame(x=x.added,y=y.added)
  ds3
}
```

PLOT

```
ds3=clust(n.cluster=5,lambda=10,xmin=0,xmax=1,ymin=0,ymax=1,norm.sd=.045)
```

We can clearly see the clusters located in our process, there are about five in total.

L

Here is a plot of $L(\hat{h}) - h$ against h for the homogeneous clustering point process:

```
x1=ds3$x
y1=ds3$y

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

```
#par(mfrow=c(2,1))
s = seq(.001,.3,length=50)
k4 = khat(b1,bdry,s)
#plot(s,k4,xlab="distance",ylab="K4(h)",pch="*")
```

```r
#lines(s,k4)
#lines(s,pi*s^2,lty=2)
L4 = sqrt(k4/pi)-s
plot(c(0,.3),range(L4),type="n",xlab="h",ylab="L4(h) - h")
points(s,L4,pch="*")
lines(s,L4)
lines(s,rep(0,50),lty=2)
```

We can clearly see the clustering effect being displayed by the large values residing above the x-axis. These values also continue to rise as the distance (h) is increased.

DATASET 4 - HOMOGENEOUS INHIBITIVE PROCESS

Here is a plot of our Homogeneous Inhibitive Process:

```
n = 80
x.temp = rep(0,n)
y.temp = rep(0,n)
x.temp[1] = runif(1,0,1)
y.temp[1] = runif(1,0,1)

keeper = function(i,r,p){
  new.x = runif(1,0,1)
  new.y = runif(1,0,1)
  dist = min(sqrt((new.x-x.temp[1:(i-1)])^2 + (new.y-y.temp[1:(i-1)])^2))
  bern = sample(c(0,1),1,prob=c(p,1-p))
  if(dist>r | bern == 1){
    new.x = new.x
    new.y = new.y
    c(new.x,new.y)
  } else{
    new.x = keeper(i,r,p)[1]
    new.y = keeper(i,r,p)[2]
    c(new.x,new.y)
  }
}

for(i in 2:n){
  x.temp[i] = keeper(i,.04,1)[1]
  y.temp[i] = keeper(i,.04,1)[2]
}

ds4 = data.frame(x=x.temp,y=y.temp)
```

```
plot(ds4$x,ds4$y,xlab="x",ylab="y",main="Homogeneous Inhibitive Process")
```

We can see that our process appears similar to the stationary point process, but there is a larger distance between the points displayed. This is a good indication of inhibition being effectively applied.

L

Here is a plot of $L(\hat{h}) - h$ against h for the homogeneous inhibitive point process:

```
x1=ds4$x
y1=ds4$y
bdw = .2
b1 = as.points(x1,y1)
bdry = matrix(c(0,0,1,0,1,1,0,1,0,0),ncol=2,byrow=T)
```

```
#par(mfrow=c(2,1))
s = seq(.001,5,length=100)
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
```

```r
plot(c(0,.5),c(-.05,.05),type="n",xlab="h",ylab="L4(h) - h")
points(s,L4,pch="*")
lines(s,L4)
lines(s,rep(0,100),lty=2)
```

```r
n = 125
x.temp = rep(0,n)
y.temp = rep(0,n)
x.temp[1] = runif(1,0,10)
y.temp[1] = runif(1,0,10)

keeper = function(i,r,p){
  new.x = runif(1,0,10)
  new.y = runif(1,0,10)
  dist = min(sqrt((new.x-x.temp[1:(i-1)])^2 + (new.y-y.temp[1:(i-1)])^2))
  bern = sample(c(0,1),1,prob=c(p,1-p))
  if(dist>r | bern == 1){
    new.x = new.x
    new.y = new.y
    c(new.x,new.y)
  } else{
    new.x = keeper(i,r,p)[1]
    new.y = keeper(i,r,p)[2]
    c(new.x,new.y)
  }
}

for(i in 2:n){
  x.temp[i] = keeper(i,.2,1)[1]
  y.temp[i] = keeper(i,.2,1)[2]
}
ds4 = data.frame(x=x.temp,y=y.temp)
```

PLOT

```r
plot(ds4$x,ds4$y,xlab="x",ylab="y",main="Homogeneous Inhibitive Process")
```

L

```r
x1=ds4$x
y1=ds4$y
bdw = .2
b1 = as.points(x1,y1)
bdry = matrix(c(0,0,1,0,1,1,0,1,0,0),ncol=2,byrow=T)
```

```r
#par(mfrow=c(2,1))
s = seq(.001,5,length=100)
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
```

```r
plot(c(0,3),c(-3,.1),type="n",xlab="lag, h",ylab="L4(h) - h")
points(s,L4,pch="*")
lines(s,L4)
lines(s,rep(0,100),lty=2)
```