Homework 4

Code

Dwight Sampson

Question 1

Simulate a homogenous Poisson process on the rectange [0,100]×[0,50] with the property that the expected number of points is 650

Hide

```
library(inlabru)
```

```
Loading required package: sp
Loading required package: ggplot2
Registered S3 method overwritten by 'dplyr':
  method           from
  print.rowwise_df
```

Hide

```
library(INLA)
```

```
Loading required package: Matrix
Loading required package: parallel
This is INLA_19.09.03 built 2019-09-03 09:03:02 UTC.
See www.r-inla.org/contact-us for how to get help.
```

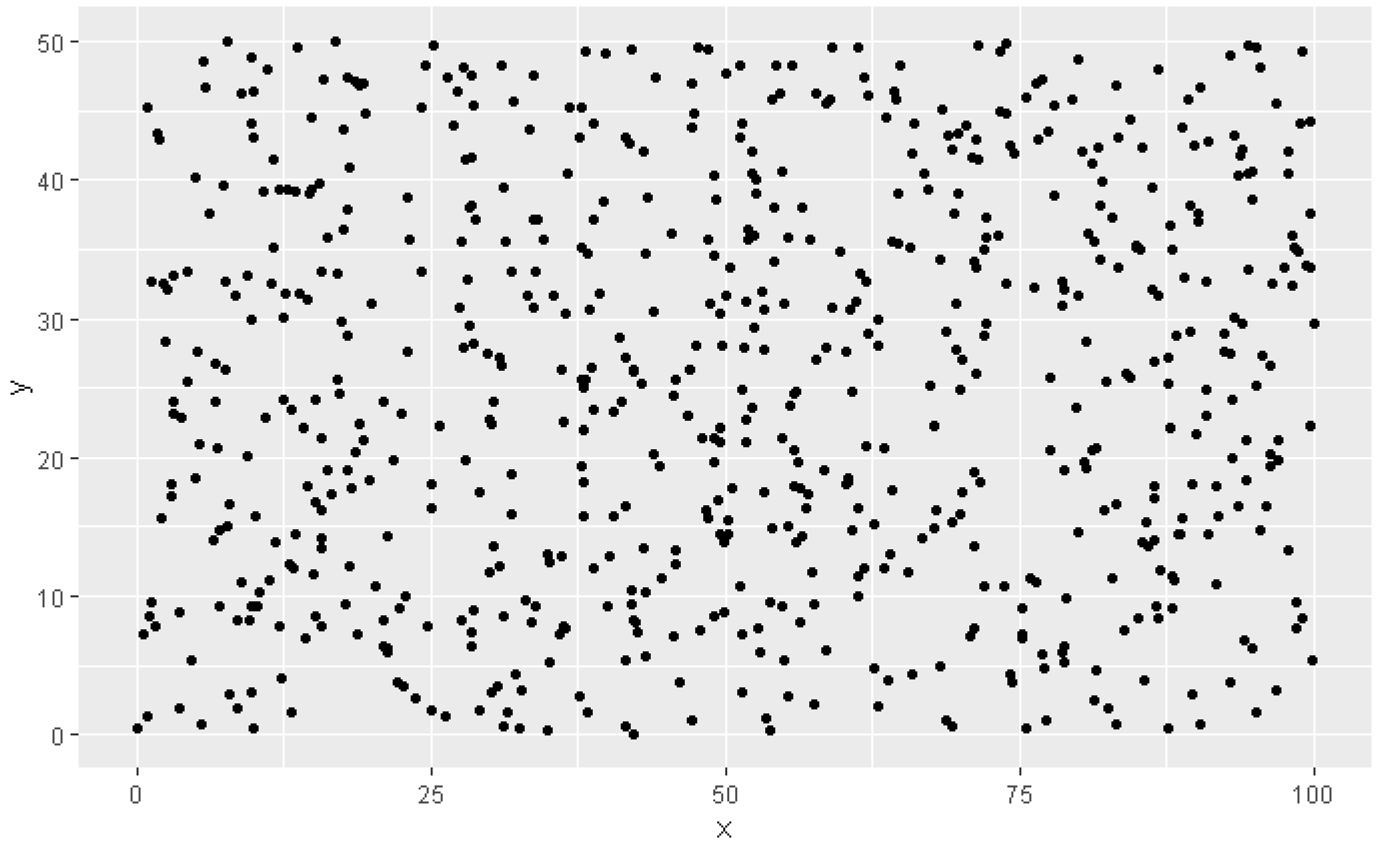
Hide

```
library(sp) #to make spatial points and spatial lines
library(tidyverse)
```

```
Registered S3 methods overwritten by 'dbplyr':
  method         from
  print.tbl_lazy
  print.tbl_sql
□[30m-- □[1mAttaching packages□[22m --------------------------------------- tidyverse 1.3.0 -
-□[39m
□[30m□[32mv□[30m □[34mtibble □[30m 2.1.3     □[32mv□[30m □[34mdplyr  □[30m 0.8.4
□[32mv□[30m □[34mtidyr  □[30m 1.0.2     □[32mv□[30m □[34mstringr□[30m 1.4.0
□[32mv□[30m □[34mreadr  □[30m 1.3.1     □[32mv□[30m □[34mforcats□[30m 0.5.0
□[32mv□[30m □[34mpurrr  □[30m 0.3.3     □[39m
□[30m-- □[1mConflicts□[22m ------------------------------------------ tidyverse_conflicts() --
□[31mx□[30m □[34mtidyr□[30m::□[32mexpand()□[30m masks □[34mMatrix□[30m::expand()
□[31mx□[30m □[34mdplyr□[30m::□[32mfilter()□[30m masks □[34mstats□[30m::filter()
□[31mx□[30m □[34mdplyr□[30m::□[32mlag()□[30m    masks □[34mstats□[30m::lag()
□[31mx□[30m □[34mtidyr□[30m::□[32mpack()□[30m   masks □[34mMatrix□[30m::pack()
□[31mx□[30m □[34mtidyr□[30m::□[32munpack()□[30m masks □[34mMatrix□[30m::unpack()□[39m
```

Hide

```
lam <- (650/5000)*(5000)
n <- rpois(1,lambda = lam) #draw a number from the poisson distn
my_data <- data.frame(x= runif(n,0,100), y= runif(n,0,50)) #simulate the data
#Turn data into a spatial object so we can analyze later
S_points <- SpatialPoints(my_data)
pointdf <- SpatialPointsDataFrame(S_points, my_data) # our spatial points dataframe
ggplot(my_data, aes(x=x, y=y)) + gg(pointdf)
```



Question 2

Using a half-normal detection function with a half-width of 2, simulate a thinned point process resulting from distance sampling on a set of 10 equally spaced parallel vertical lines.

Lets draw our transects:

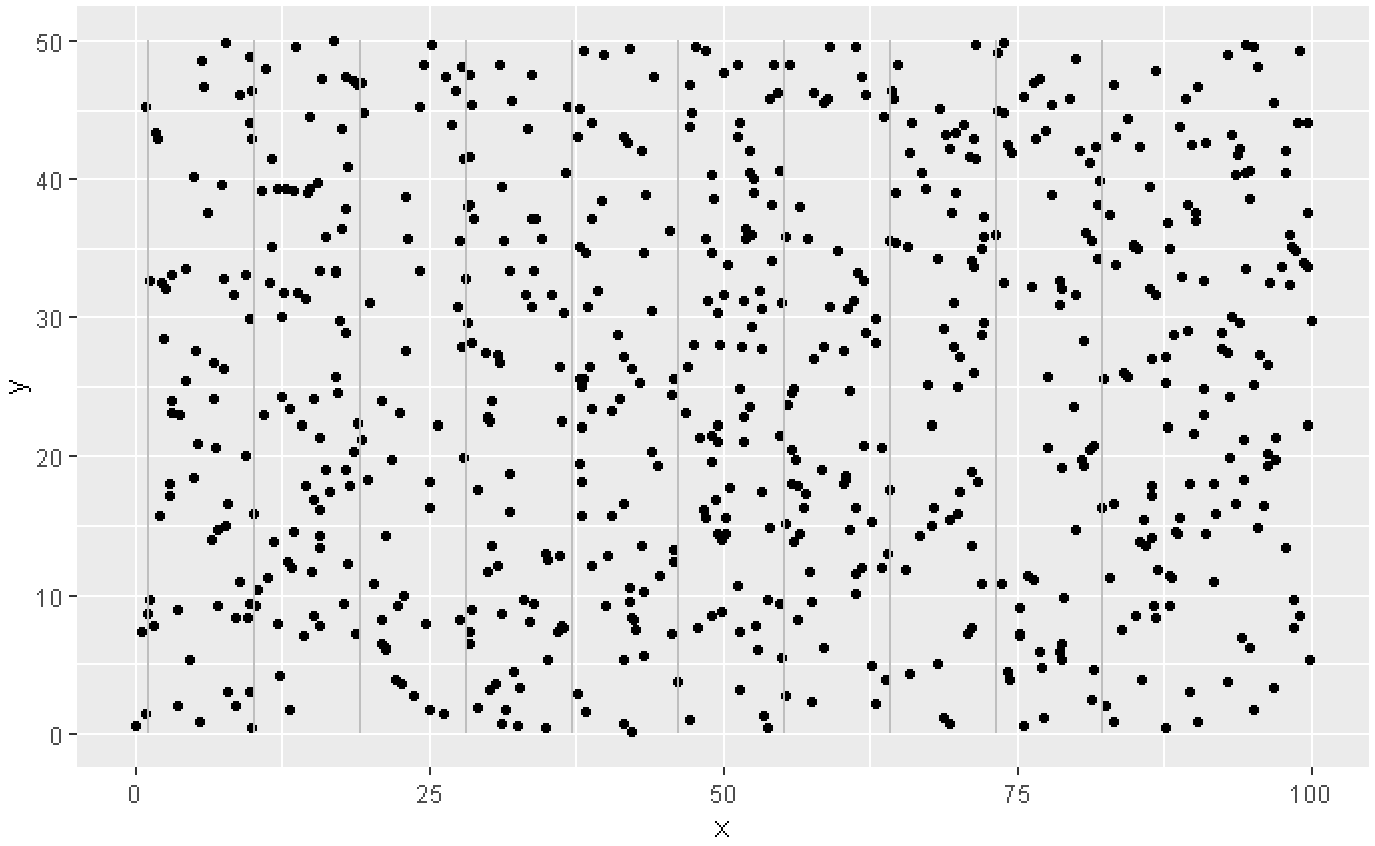
Hide

```
#our transects must be in a dataframe of class 'Spatial Lines DataFrame'
draw_lines <- function(n){
  lines <- list()
  index = 1 #index for list
  i = 0 #index for loop

  space <- sample(2:(100/n)+5,1) #the amount of space between transects
  num <- runif(1,1, (100/n)) # choose an x-value to make, the first verticle transect

  while (i != n){
    #make two points that will be the start and end of the line
    y <- c(0,50)
    x <- c(num,num)
    combo <- cbind(x,y)
    #create the line
    line0 = Line(combo)
    line_prop = Lines(list(line0), ID= index)
    #make a list of lines
    lines[index] = line_prop
    #continue the loop
    i = i+1
    index = index+1
    num = num + space
  }
  SL <- SpatialLines(lines)
  SL_dataframe <- SpatialLinesDataFrame(SL, data = tibble(id=seq(1,n,1)))
  return(SL_dataframe)
}
#generate our transects from the above function
  #because we want equal spacing, our function does "systematic sampling"
transects <- draw_lines(10)
```

```
implicit list embedding of S4 objects is deprecatedimplicit list embedding of S4 objects is deprec
atedimplicit list embedding of S4 objects is deprecatedimplicit list embedding of S4 objects is de
precatedimplicit list embedding of S4 objects is deprecatedimplicit list embedding of S4 objects i
s deprecatedimplicit list embedding of S4 objects is deprecatedimplicit list embedding of S4 objec
ts is deprecatedimplicit list embedding of S4 objects is deprecatedimplicit list embedding of S4 o
bjects is deprecated
```

Hide

```
#plot our transects
ggplot(my_data, aes(x=x, y=y)) + gg(pointdf) + gg(transects, color= "grey")
```



Lets determine the values of our transects:

```
 [1]  1.1417 10.1417 19.1417 28.1417 37.1417 46.1417 55.1417 64.1417 73.1417 82.1417
```

Lets determine the distance of points from each line. I wont display the result of the code as it populates 677 lines of warinings and messages.

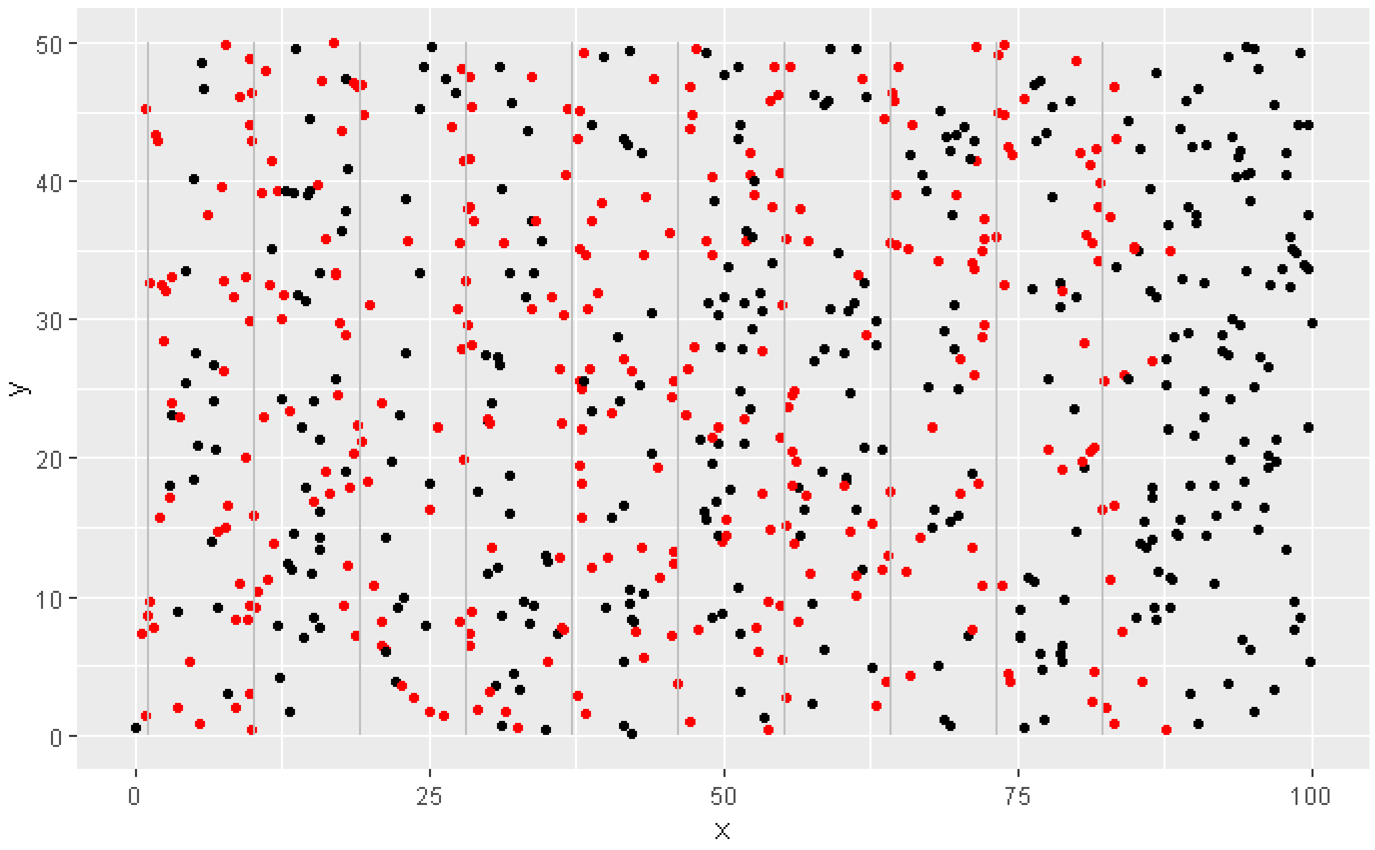
Hide

```
#the above doesnt allows the document to not show the warnings and result of this chunk
library(geosphere) #for dist2Line function
closest_line <- dist2Line(pointdf,transects)
index = 1
dist_vect <- list()
while (index < n+1){
  i <- closest_line[,4][index] #closest line by id
  dist <- abs(trans_values[i] - pointdf$x[index])
  dist_vect[index] <- dist
  index = index +1
}
q2_points <- data.frame(pointdf)%>% mutate(distance=dist_vect)
```

Here we simulate the thinning process. Red points are the points we keep in our pattern, black points are the one that have been tossed.

Hide

```
w = 2
lsig = log(2)
hn <- function(distance, logsigma){
  prob <- exp(-0.5*(distance/exp(logsigma))^2)
  return(prob)
}
thin <- function(data){
  #create a vector of colours where
  #red = keep; black= throw away
  index = 1
  in_out <- list()
  indication <- c("black","red")

  while (index < n+1){
    distance <- data$distance[index]
    prob = hn(as.numeric(distance),lsig)
    indicator <- sample(x=indication,prob = c(1-prob, prob), size = 1)
    in_out[index] <- indicator
    index <- index +1
  }
  return(in_out)
}
q2_thin <- thin(q2_points)
q2_points%>% mutate(indicator = q2_thin)%>% ggplot(aes(x=x, y=y, color= indicator)) +
  geom_point() + gg(transects, color= "grey")
```



Question 3

Using a half-normal detection function with a half-width of 2, simulate a thinned point process resulting from distance sampling on a set of 10 random lines.

Lets draw our transects:

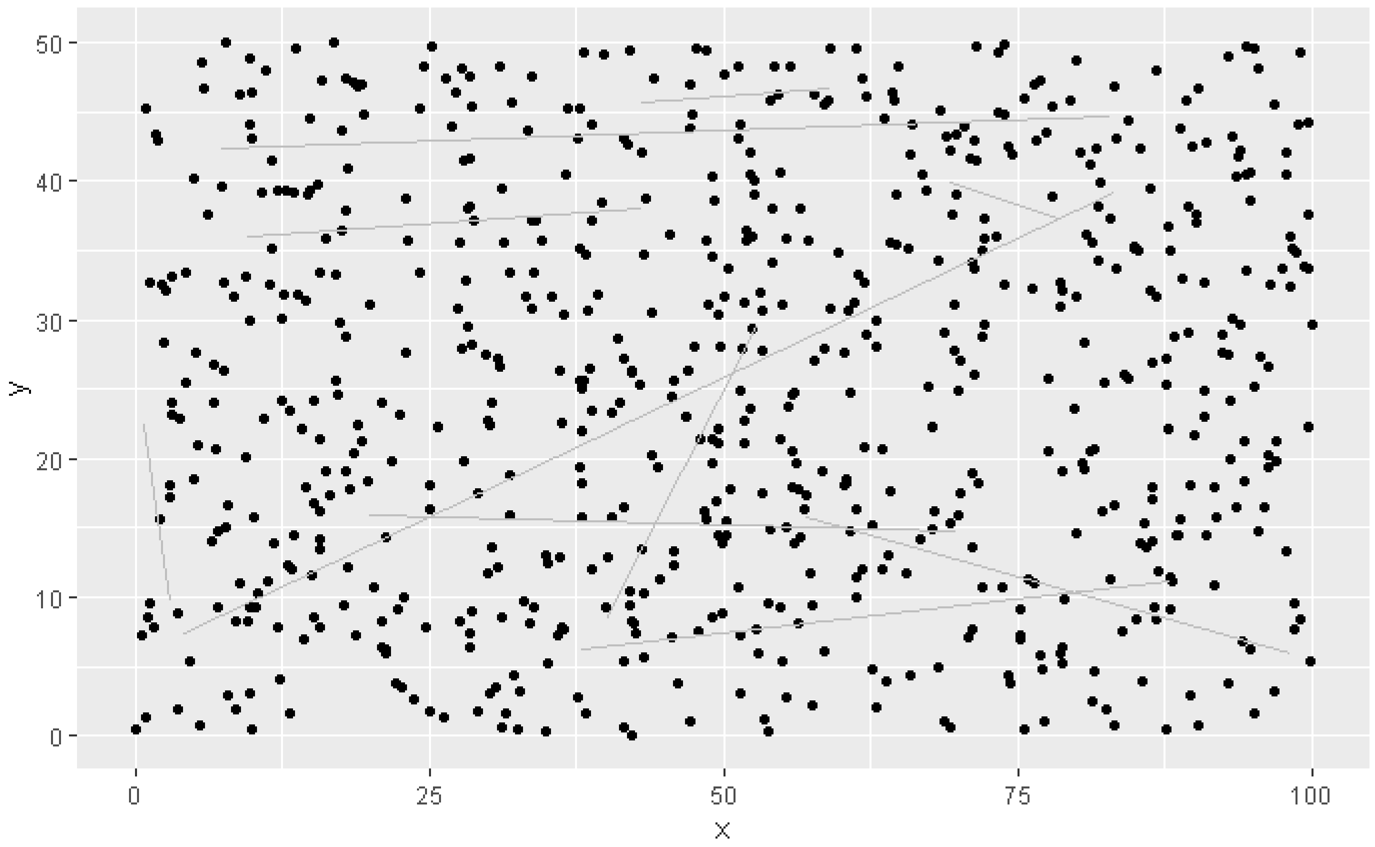
Hide

```
oneLine <- function(){
  #a function that generates a single non-parallel line
   y <- runif(2,0,50)
   x <- runif(2,0,100)
   result <- cbind(x,y)
  return(result)
}
start_stop_lines <- function(n){
  #generate 'n' non-parallel lines but, dont transform them to spatial line
  lines <- lapply(1:n,function(x) oneLine())
  return(lines)
}
#-----------------------------
#a list of our transects before we made them into spatial line
raw_lines <- start_stop_lines(10)
#-----------------------------
np_lines <- function(n){
  #function that turns transects into spatial lines
  lines <- list()
  index = 1 #index for list
  i = 0 #index for loop
  while (i != n){
    #create the line
    line0 = Line(raw_lines[index])
    line_prop = Lines(list(line0), ID= index) # cant access x and y after here
    #make a list of lines
    lines[index] = line_prop
    #continue the loop
    i = i+1
    index = index+1
  }
  SL <- SpatialLines(lines)
  SL_dataframe <- SpatialLinesDataFrame(SL, data = tibble(id=seq(1,n,1)))
  return(SL_dataframe)
}
#generate our transects from the above function
  #because we want equal spacing, our function does "systematic sampling"
transects_q3 <- np_lines(10)
```

```
implicit list embedding of S4 objects is deprecatedimplicit list embedding of S4 objects is deprec
atedimplicit list embedding of S4 objects is deprecatedimplicit list embedding of S4 objects is de
precatedimplicit list embedding of S4 objects is deprecatedimplicit list embedding of S4 objects i
s deprecatedimplicit list embedding of S4 objects is deprecatedimplicit list embedding of S4 objec
ts is deprecatedimplicit list embedding of S4 objects is deprecatedimplicit list embedding of S4 o
bjects is deprecated
```

Hide

```
#plot our transects
ggplot(my_data, aes(x=x, y=y)) + gg(pointdf) + gg(transects_q3, color= "grey")
```



Lets determine the distance of points from each line. I wont display the result of the code as it populates 647 lines of warinings and messages.

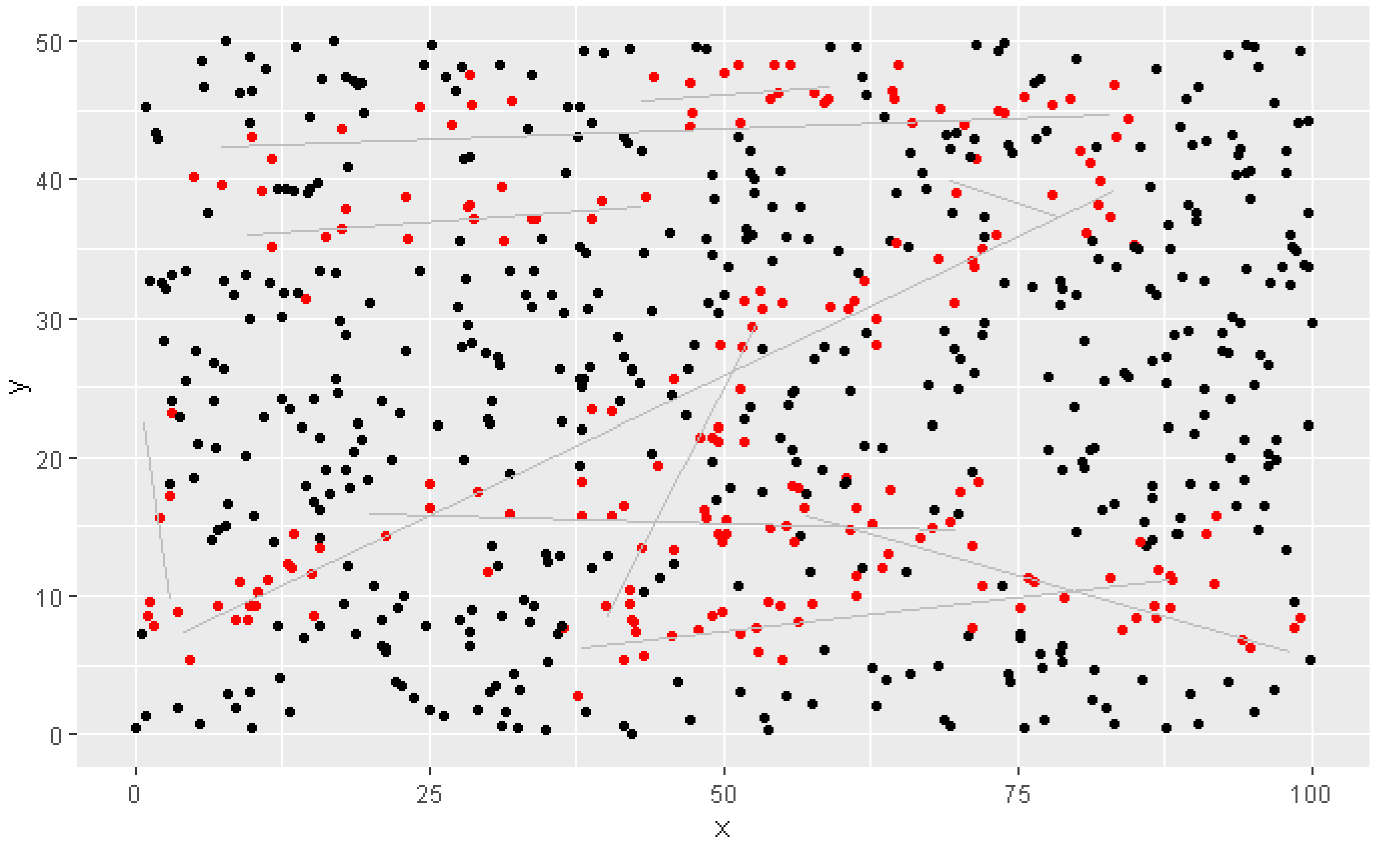
Hide

```
#the above doesnt allows the document to not show the warnings and result of this chunk
library(geosphere)
library(maptools) #for nearestPointOnSegment function
closest_line_q3 <- dist2Line(pointdf,transects_q3)
index = 1
dist_vect_q3 <- list()
while (index < n+1){
  i <- closest_line_q3[,4][index] #closest line by id
  segment = segment = cbind(raw_lines[[i]][,1], raw_lines[[i]][,2])
  dist <- nearestPointOnSegment(segment,c(my_data[index,]$x,my_data[index,]$y))[3]
  dist_vect_q3[index] <- dist
  index = index +1
}
q3_points <- data.frame(pointdf)%>% mutate(distance=dist_vect_q3)
```

Here we simulate the thinning process. Red points are the points we keep in our pattern, black points are the one that have been tossed.

Hide

```
q3_thin <- thin(q3_points)
q3_points%>% mutate(indicator = q3_thin)%>% ggplot(aes(x=x, y=y, color= indicator)) +
  geom_point() + gg(transects_q3, color= "grey")
```

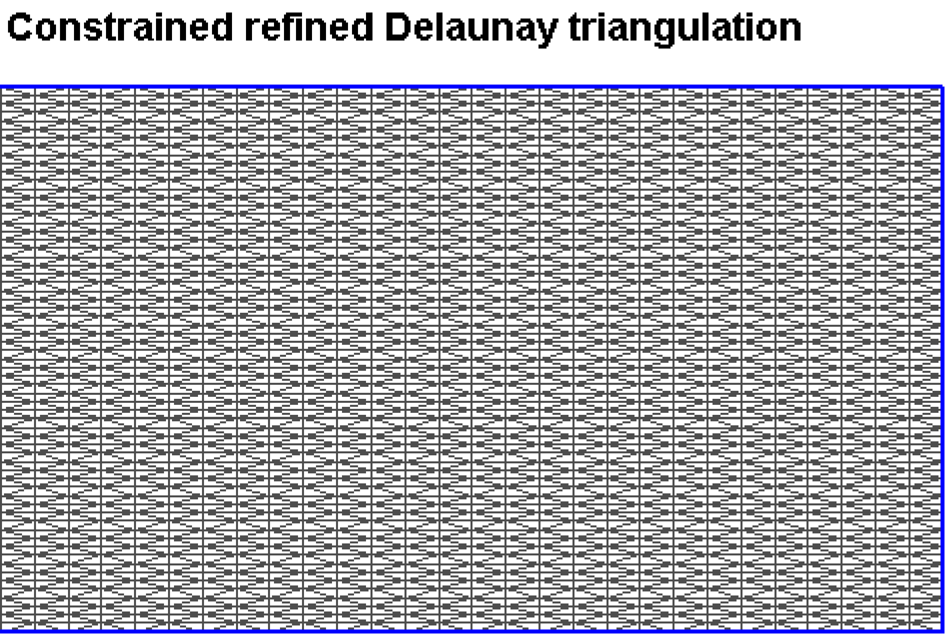


Question 4

Lets build a mesh for the data:

Hide

```
#make a mesh bigger than our area
x <- c(0,0,100,100)
y <- c(0,50,50,0)
bound <- spoly(data.frame(y,x))
mesh <- inla.mesh.2d(boundary= bound, max.edge = 2.5, cutoff = 0.01)
plot(mesh)
```



Hide

```
mesh$n
```

```
[1] 2145
```

What is the smallest range between our points in the pattern?

Hide

```
library(spatstat)
#calculate the smallest distance between points that we generated
tibble(dist_to_point = nndist(my_data)) %>% arrange(dist_to_point)%>% head(10)
```

dist_to_point <dbl>
0.04579873
0.04579873
0.07757762
0.07757762
0.08120933
0.08120933
0.12735118

dist_to_point <dbl>
0.12735118
0.12776495
0.12776495

1-10 of 10 rows

Lets specify the SPDE model:

Hide

```
matern <- inla.spde2.pcmatern(mesh, prior.sigma = c(2,0.01),
                              prior.range = c(0.5, 0.1))
#we also have to define the components of the model
# we dnt know observed, SPDE, logsigma or intercpet but we want to know them
cmp <- ~ mySPDE(map= coordinates, model= matern) + logsigma + Intercept
formula = coordinates  ~ mySPDE + log(hn(dist_vect_q3, logsigma)) + log(1/w) + Intercept
mod <- lgcp(components = cmp, pointdf, samplers = transects_q3, formula = formula)
```

```
The integration points provided have no weight column. Setting weights to 1.Found spatial lines wi
th start or end point ouside of the mesh. Omitting.Error in spTransform(sp3d, CRSobj = CRS("+proj=
geocent +ellps=sphere +R=1.00")) :
  No transformation possible from NA reference system
```

Question 5

Repeat the exercise for 20 different sets of random transects (same set up as part 3) and comment on the frequentist properties of the estimate for the total number of points.

Hide

```
process_line <- function(num_transects){
  #generate transects
  raw <- start_stop_lines(10)
  transects_q5 <- np_lines(10)
  #find the closest line to each point
  closest_line_q5 <- dist2Line(pointdf,transects_q5)
  #determine the given point to the closest line
  index = 1
  dist_vect_q5 <- list()
  while (index < n+1){
    i <- closest_line_q5[,4][index] #closest line by id
    segment = segment = cbind(raw_lines[[i]][,1], raw[[i]][,2])
    dist <- nearestPointOnSegment(segment,c(my_data[index,]$x,my_data[index,]$y))[3]
    dist_vect_q5[index] <- dist
    index = index +1
  }
  #save a table of points with distances from the closest line
  q5_points <- data.frame(pointdf)%>% mutate(distance=dist_vect_q5)
  #model the data
  mod <- lgcp(components = cmp, pointdf, samplers = transects_q3, formula = formula)
  #generate a posterior
  #get the expected number of points
  spde.range <- spde.posterior(mod, "mySPDE", what = "range")
  return(spde.range)
}

#repeat 20 times
rerun(20, process_line(10))
```