*Spatial Statistics Lab 3*

**Onyedikachi J Okeke**

2024-02-04

### 0.0 To Load the Spatstat library

## Loading required package: spatstat.data

## Loading required package: spatstat.geom

## spatstat.geom 3.2-8

## Loading required package: spatstat.random

## spatstat.random 3.2-2

## Loading required package: spatstat.explore

## Loading required package: nlme

## spatstat.explore 3.2-5

## Loading required package: spatstat.model

## Loading required package: rpart

## spatstat.model 3.2-8

## Loading required package: spatstat.linnet

## spatstat.linnet 3.1-3

##   
## spatstat 3.0-7   
## For an introduction to spatstat, type 'beginner'

## Part 1: Simulation of homogeneous Poisson process

# A) To make a point pattern using one of these processes, type:  
par(mfrow = c(2, 3))  
  
pp <-rpoispp(100)  
#plot pp  
plot(pp)  
  
# B) combine generating and plotting a pattern into a single command  
plot(rpoispp(100))  
  
# C) create the pattern first before plotting  
p1 <-rpoispp(100)  
plot(p1)  
  
# D) repeatedly generate p1 and plot it, until getting an interesting realization of the process.  
  
par(mfrow = c(2, 3))  
  
p2 <- rpoispp(100)  
plot(p2)  
  
p3 <- rpoispp(100)  
plot(p3)  
  
p4 <- rpoispp(100)  
plot(p4)  
  
p5 <- rpoispp(100)  
plot(p5)  
  
p6 <- rpoispp(100)  
plot(p6)

#### Result

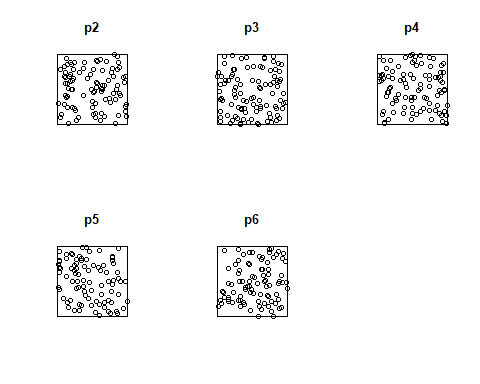
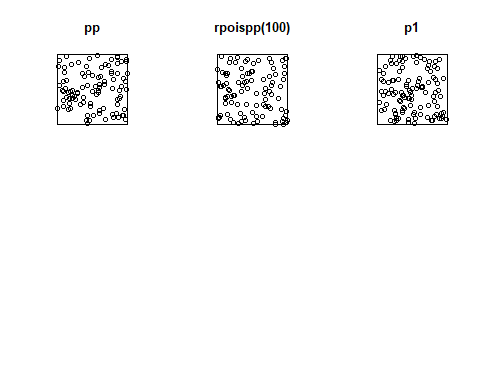


Figure 1:

Q1: from Figure 1 Do all these IRP/CSR patterns appear 'random'? If not, how do they seem to differ from 'random' patterns?  
answer  
Not all the IRP/CSR patterns appear completely random. Some patterns, like the top middle and bottom right plots, seem to have clustering of points in certain areas. Other patterns, like the top left plot, appear more evenly dispersed without clear clusters. So there is some variability in how "random" each plot looks visually.

Q2: from Figure 1 Do some exhibit clustering of events? Or, conversely, empty regions of the map? Provide an example from your explorations, and explain how it is possible for such seemingly 'non-random' patterns to be generated by a completely random process?.

Answer  
Yes, some patterns exhibit clustering while others have empty regions. For example, the bottom right plot shows clustering of points on the left side, while the right side is empty. Despite being generated by a completely random Poisson process, some realizations can end up with these seemingly non-random, clustered patterns by chance. Over many simulations, the clustering would average out, but individual realizations can vary.

# Q2 Create a sample spatial point pattern (replace with your own data)  
set.seed(123)  
n\_points <- 100  
x <- runif(n\_points)  
y <- runif(n\_points)  
data\_points <- ppp(x, y, window = owin(c(0, 1), c(0, 1)))  
  
# Plot the point pattern  
plot(data\_points, main = "Spatial Point Pattern")  
  
# Perform a quadrat test to check for clustering  
quadrat\_test\_result <- quadrat.test(data\_points)  
  
# Print the results of the quadrat test  
print(quadrat\_test\_result)  
  
# Interpret the results  
if (quadrat\_test\_result$p.value < 0.05) {  
 print("There is evidence of clustering.")  
} else {  
 print("There is no significant evidence of clustering.")  
}

#### Result

## Warning: Some expected counts are small; chi^2 approximation may be inaccurate

##   
## Chi-squared test of CSR using quadrat counts  
##   
## data: data\_points  
## X2 = 24, df = 24, p-value = 0.9232  
## alternative hypothesis: two.sided  
##   
## Quadrats: 5 by 5 grid of tiles

## [1] "There is no significant evidence of clustering."

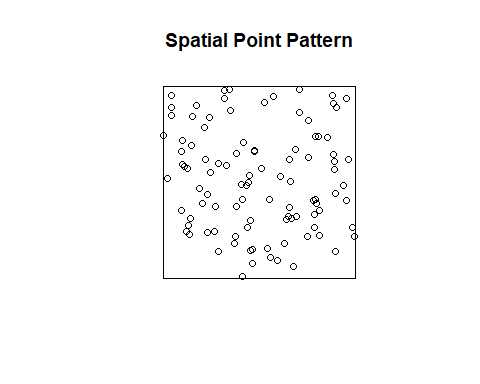
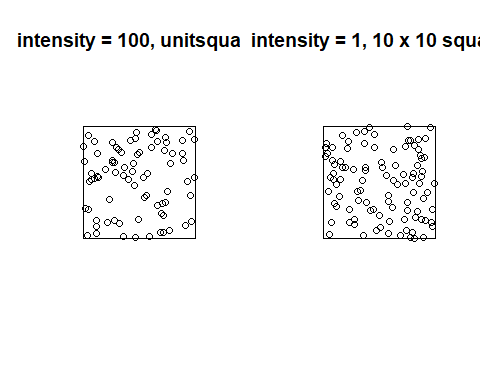
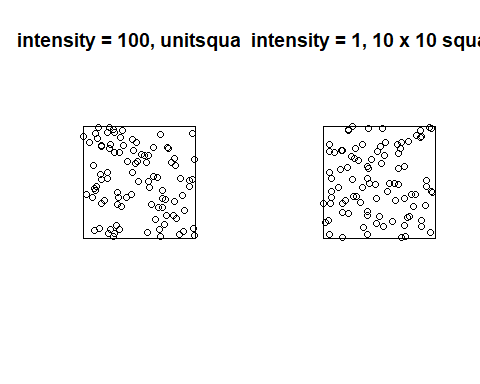


Figure 2:

Q3 Redo exactly the same commands and see the different realizations of the same process under CSR.

#Q3 generate a point pattern with intensity 1 in a 10 x 10 square  
pp.1.10 <- rpoispp(1, win=owin(c(0,10),c(0,10)))  
par(mfrow=c(1,2))  
  
plot(rpoispp(100),main = "intensity = 100, unitsquare") # uniform Poisson process with  
#intensity 100 in the unit square  
plot (pp.1.10, main = "intensity = 1, 10 x 10 square")  
  
# redoing exactly same process  
pp2.1.10 <- rpoispp(1, win=owin(c(0,10),c(0,10)))  
par(mfrow=c(1,2))  
  
plot(rpoispp(100),main = "intensity = 100, unitsquare") # uniform Poisson process with  
#intensity 100 in the unit square  
plot (pp2.1.10, main = "intensity = 1, 10 x 10 square")

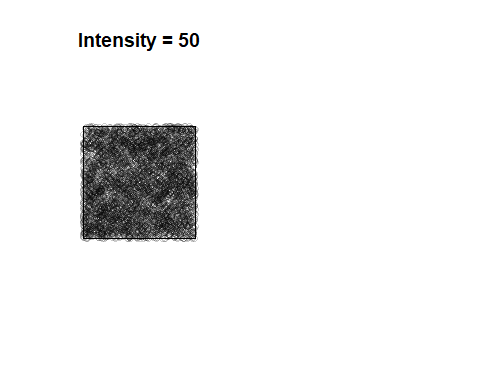
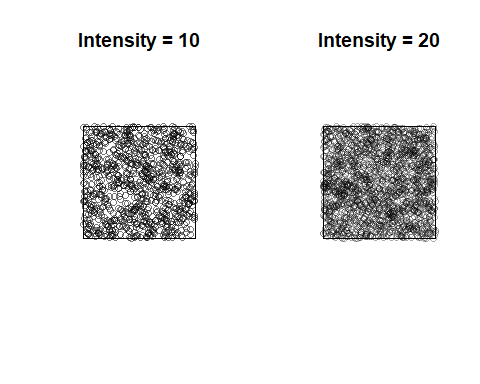
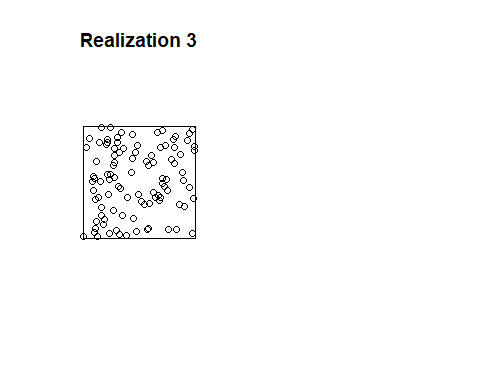
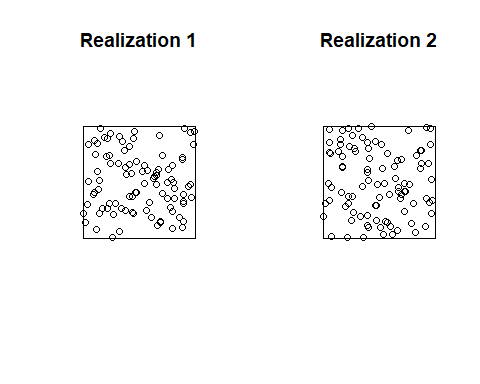
#### Result



Q4 Change the intensity parameter to 10,20, and 50 in the unit square

#Q4 Change the intensity parameter to 10,20, and 50 in the unit square  
# Set the seed for reproducibility  
set.seed(123)  
  
# Create a 10x10 square window  
window\_square <- owin(c(0, 10), c(0, 10))  
  
#Generate and visualize three realizations of the same process under CSR  
par(mfrow=c(1, 2))  
for (i in 1:3) {  
 pp\_csr <- rpoispp(lambda = 1, win = window\_square)  
 plot(pp\_csr, main = paste("Realization", i))  
}  
  
#Change the intensity parameter to 10, 20, and 50 in the unit square  
par(mfrow=c(1, 2))  
intensities <- c(10, 20, 50)  
for (i in 1:3) {  
 pp\_intensity <- rpoispp(lambda = intensities[i], win = window\_square)  
 plot(pp\_intensity, main = paste("Intensity =", intensities[i]))  
}

#### Result



Q5 No, each simulation does not show exactly 10 points because Poisson point processes are stochastic, and the number of points in each realization can vary. The intensity parameter (lambda) represents the average number of points in a unit area, but the actual number of points in a specific realization follows a Poisson distribution with mean lambda. Therefore, there will be some variability in the number of points in each simulation.

## Part 2: Simulation of inhomogeneous Poisson process

# A) define a function in place of the fixed intensity value of the standard Poisson process.  
pp <- rpoispp(100)  
  
# B) specify a function for the intensity  
pp <- rpoispp(function(x,y) {100\*x + 100\*y})  
  
par(mfrow=c(1, 2))  
# C) To plot the density of points in each realization:  
plot(density(pp))  
plot(pp, add=T)  
  
# D) Visualizing the density of each point process realization is using contours  
plot(density(pp))  
contour(density(pp), add=T)  
plot(pp, add=T)

#### Result

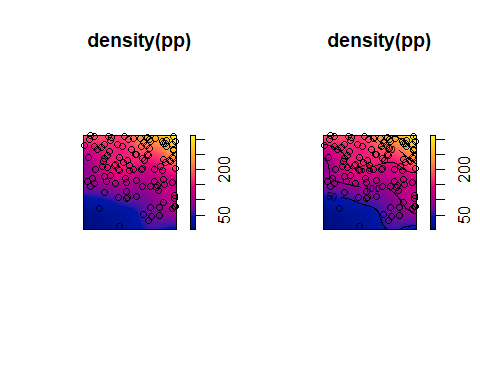


Figure 3:

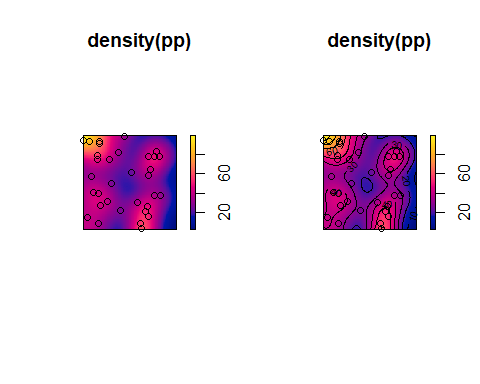
Q6. Does this pattern appear random? How does it differ from the pure random patterns you generated in the previous section, if at all? Can you tell the difference every time, just by visual inspection?

Answer  
 The pattern in Figure 3 does not appear completely spatially random. The density and intensity of points varies from left to right, with more points on the left side. This differs from a pure random pattern that would have a uniform point density throughout. The difference comes from the use of a non-homogeneous Poisson process with an intensity function that places more points on the left side. By eye, this type of trend in point density across space could likely be distinguished from a pure random pattern.

#### Now change the intensity function

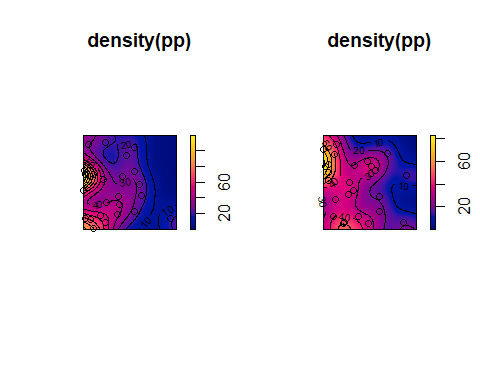
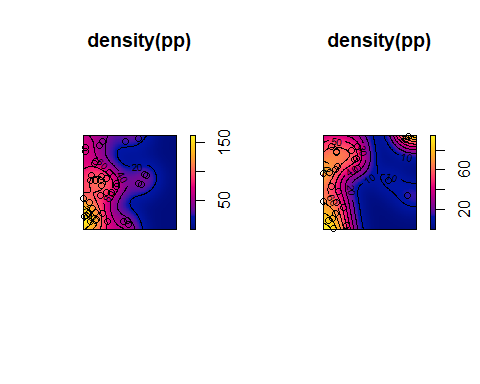
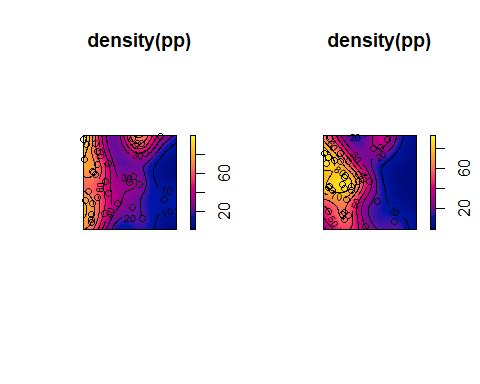
# E) with intensity lambda(x,y) = 100 \* exp(-3\*x)  
# Intensity is bounded by 100  
pp <- rpoispp(function(x,y) {100 \* exp(-3\*x)}, 100)  
par(mfrow=c(1, 2))  
# C) To plot the density of points in each realization:  
plot(density(pp))  
plot(pp, add=T)  
  
# D) Visualizing the density of each point process realization is using contours  
plot(density(pp))  
contour(density(pp), add=T)  
plot(pp, add=T)

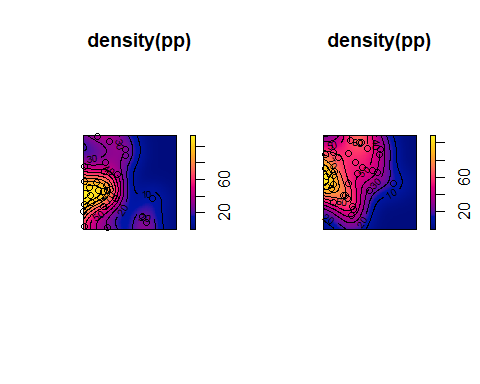
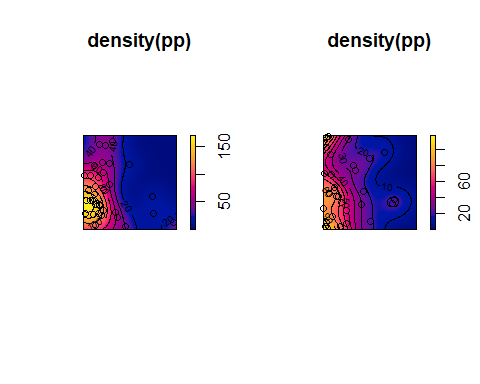
#### Result



Redo the above statement 10 times and plot each realization (including density and point pattern on top)

par(mfrow=c(1,2)) # Set up a 5x2 grid for 10 plots  
  
for (i in 1:10) {  
 # Generate a realization of the point process with a different intensity function  
 pp <- rpoispp(function(x,y) {100 \* exp(-3\*x)}, 100)  
   
 # Plot the density of points in each realization  
 plot(density(pp))  
   
 # Visualize the density using contours  
 contour(density(pp), add=TRUE)  
   
 # Plot the point process on top  
 plot(pp, add=TRUE)  
}





Q7 Generating 10 different realizations of a point process with varying intensity functions illustrates the stochastic nature of spatial patterns and highlights the intrinsic randomness in Poisson point processes. Despite using a common intensity function as a guide, each realization produces a unique spatial distribution of points. While these realizations generally follow the expected pattern defined by the intensity function, they exhibit specific variations and characteristics. Some may display clustering in certain areas, while others appear more dispersed. This variability has significant implications for fields such as spatial statistics, ecology, epidemiology, and urban planning, where understanding and accounting for randomness is crucial. It emphasizes the need for multiple realizations to assess uncertainty, make statistical inferences, and evaluate the robustness of conclusions drawn from a single point pattern.

## Part 3: Regular pattern point processes

help(spatstat)

#### Result

## starting httpd help server ... done

second-order interaction effects

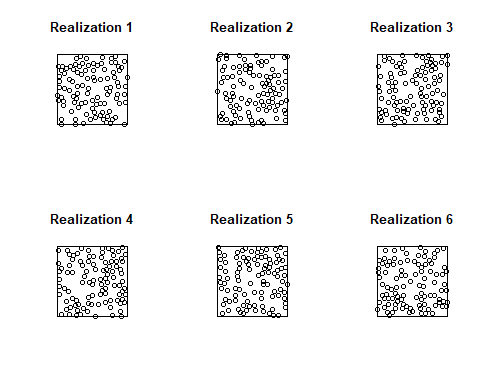
pp <- rSSI(0.05, 100)

#### Result

Q8 Redoing the above statement 6 times and plot each realization.

#Q8 Redoing the above statement 6 times and plot each realization  
# Set up a 2x3 grid for 6 plots  
par(mfrow=c(2, 3))  
  
# Generate and plot 6 realizations  
for (i in 1:6) {  
 pp <- rSSI(0.05, 100)  
 plot(pp, main = paste("Realization", i))  
}

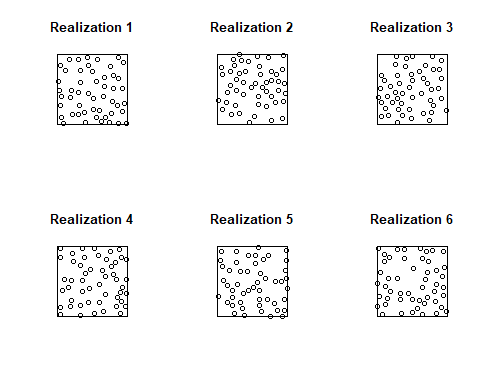
#### Result



Q9 Change the inhibition distance to 0.08 and number of events to 50

#Q9 Change the inhibition distance to 0.08 and number of events to 50.  
# Set up a 2x3 grid for 6 plots  
par(mfrow=c(2, 3))  
  
# Generate and plot 6 realizations  
for (i in 1:6) {  
 pp <- rSSI(0.08, 50)  
 plot(pp, main = paste("Realization", i))  
}

#### Result



## Part 4: Clustering pattern point processes

#contrast, a clustering process, such as the Thomas process  
# Set up a 2x3 grid for 6 plots  
par(mfrow=c(2, 3))  
  
# Generate and plot 6 realizations  
for (i in 1:6) {  
 pp <- rThomas(10, 0.1, 10)  
 plot(pp, main = paste("Realization", i))  
}

#### Result

