Midterm

Jiawen Qi (jiq10)

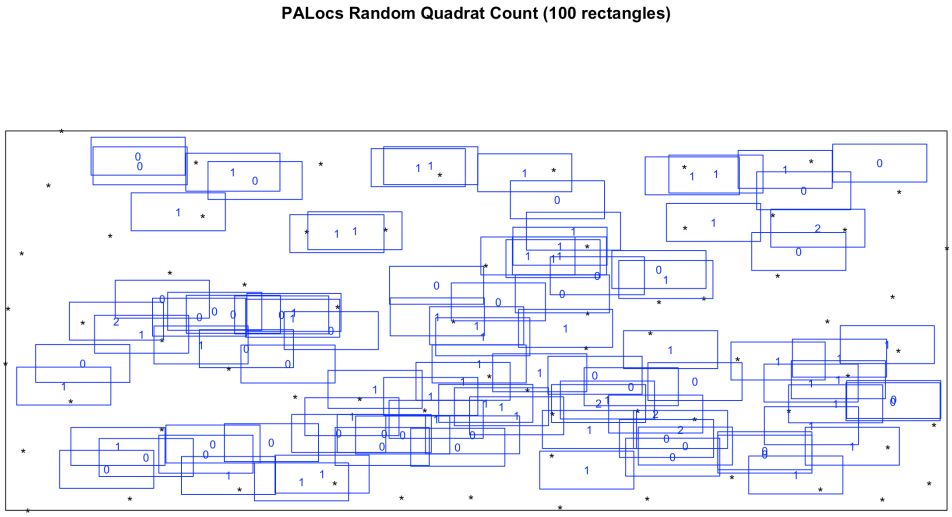
March 2, 2017

1. PART A

1.1 PALocs

1.1.1 Map shown the random quadrat count

Here is the Map with Ramdom 10x10 quadrat counts for PALocs shapefile:



1.1.2 Table shown the statistics

Regular Quadrat Count Table

No. of Events, K	No. of Quadrats, X	K - u	(K – u) ^ 2	X ((K – u) ^ 2)
0	42	-0.67	0.4489	18.8538
1	53	0.33	0.1089	5.7717
2	5	1.33	1.7689	8.8445

1.1.3 Summary of decision based on VMR

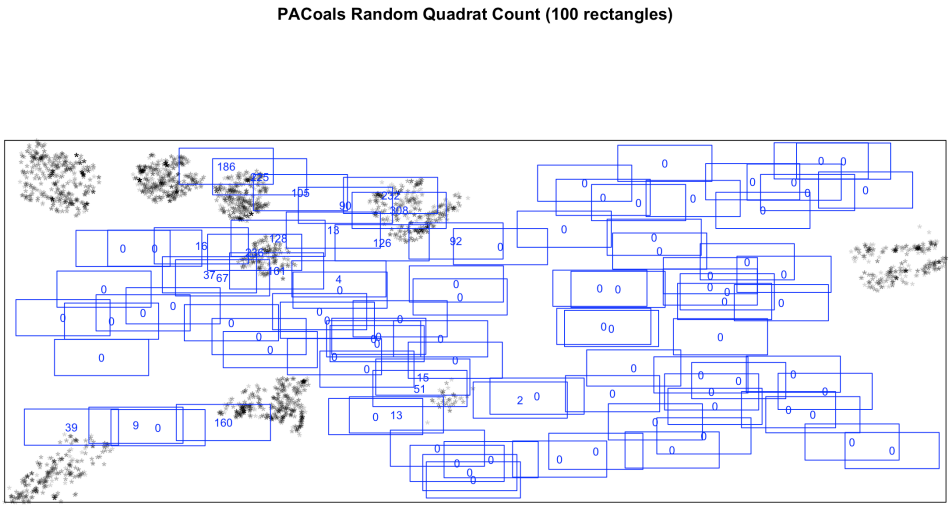
The observed variance s sqaure is `(18.8538 + 5.7717 + 8.8445) / (100 - 1) = 0.3380808`

The ratio of variance to mean `VMR = 0.3380808 / 0.67 = 0.5045982` < 1.0, which means PALocs points are evenly spaced, not clustered.

1.2 PACoals

1.2.1 Map shown the random quadrat count

Here is the Map with Ramdom 10x10 quadrat counts for PACoals shapefile:



1.2.2 Table shown the statistics

Regular Quadrat Count Table

No. of Events, K	No. of Quadrats, X	K - u	(K – u) ^ 2	X ((K – u) ^ 2)
0	77	-41.06	1685.9236	129816.1172
2	1	-39.06	1525.6836	1525.6836
4	1	-37.06	1373.4436	1373.4436
9	1	-32.06	1027.8436	1027.8436
13	2	-28.06	787.3636	1574.7272
15	1	-26.06	679.1236	679.1236
16	1	-25.06	628.0036	628.0036
37	1	-4.06	16.4836	16.4836
39	1	-2.06	4.2436	4.2436
51	1	9.94	98.8036	98.8036
67	1	25.94	672.8836	672.8836
90	1	48.94	2395.1236	2395.1236
92	1	50.94	2594.8836	2594.8836
105	1	63.94	4088.3236	4088.3236

126	1	84.94	7214.8036	7214.8036
128	1	86.94	7558.5636	7558.5636
160	1	118.94	14146.7236	14146.7236
161	1	119.94	14385.6036	14385.6036
186	1	144.94	21007.6036	21007.6036
225	1	183.94	33833.9236	33833.9236
232	1	190.94	36458.0836	36458.0836
236	1	194.94	38001.6036	38001.6036
308	1	266.94	71256.9636	71256.9636

1.2.3 Summary of decision based on VMR

The observed variance s square is `sum(X(K-u)^2) / (100 - 1) = 390359.6 / 99 = 3943.026`

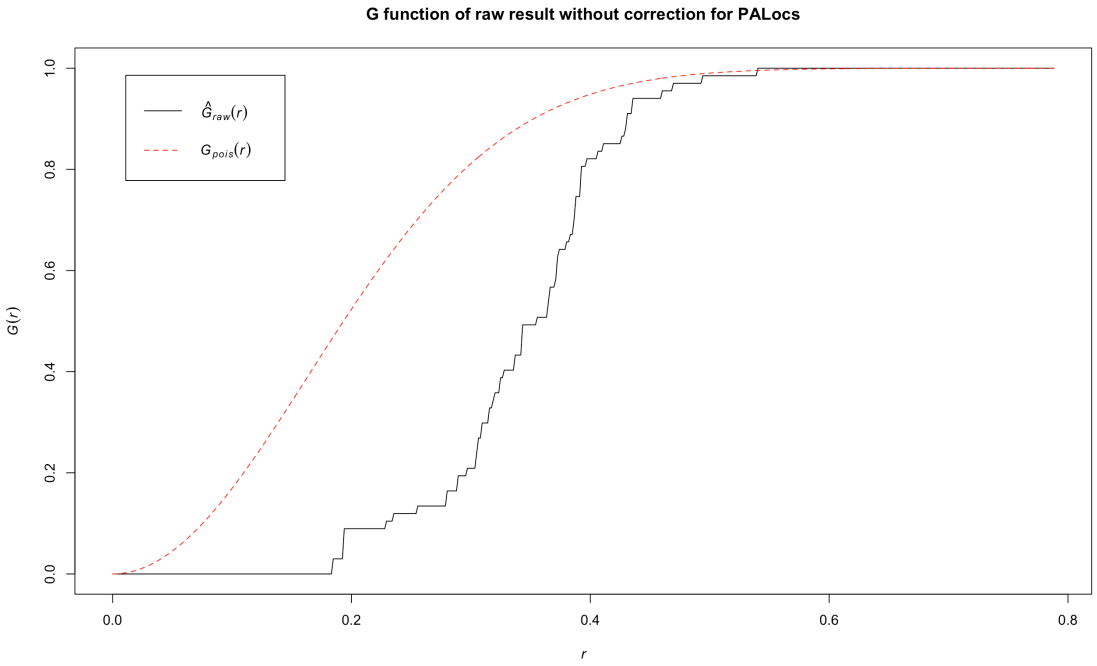
The ratio of variance to mean `VMR = 3943.026 / 41.06 = 96.03083` larger than 1.0, which means PACoals points are clustered.

2. PART B

2.1 PALocs

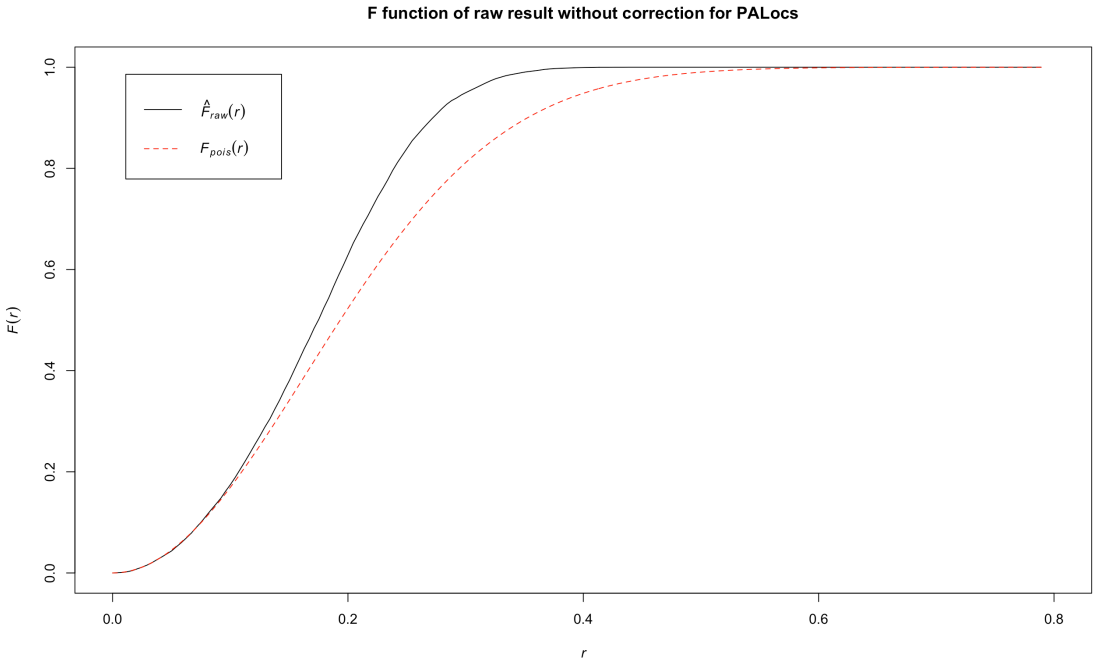
2.1.1 A plot showing the result of G function

Here is the plot of G function for PALocs shapefile:



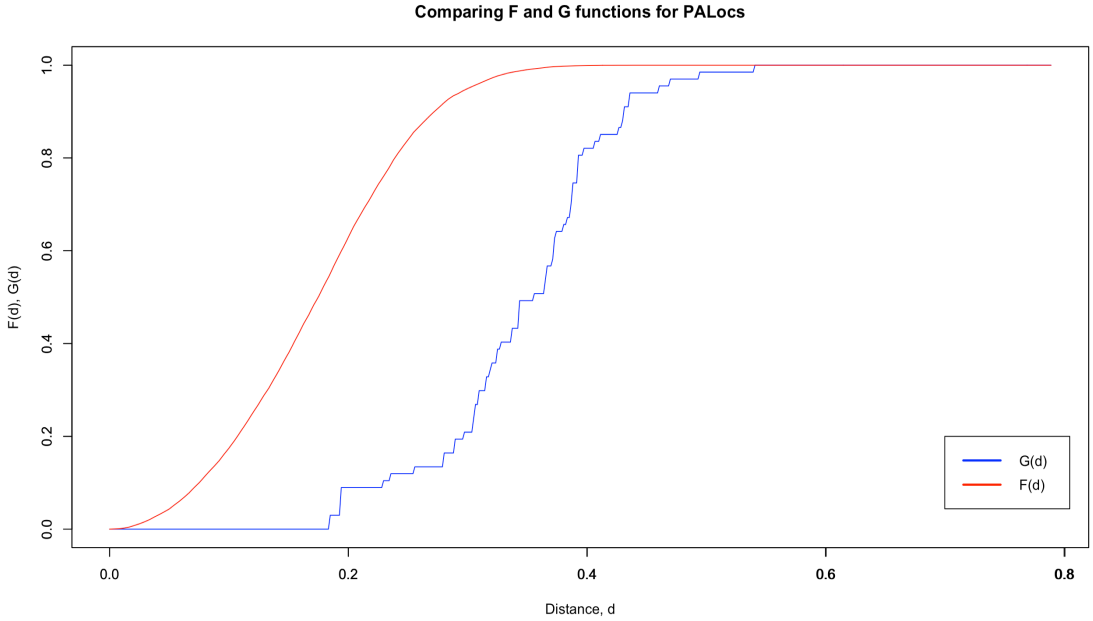
r is the distance, Graw(r) is the uncorrected estimate of G(r), that is the empirical distribution of the distance from each ppint in the pattern to the nearest other point of the pattern.Gpois(r) is the theoretical value of G(r) for a stationary Poisson process of the same estimated intensity.

2.1.2 A plot showing the result of F function



r is the distance, Fraw(r) is the uncorrected estimate of F(r), that is the empirical distribution of the distance from a random point in the window to the nearest point of the data pattern X.

2.1.3 A report comparing the results of G and F functions



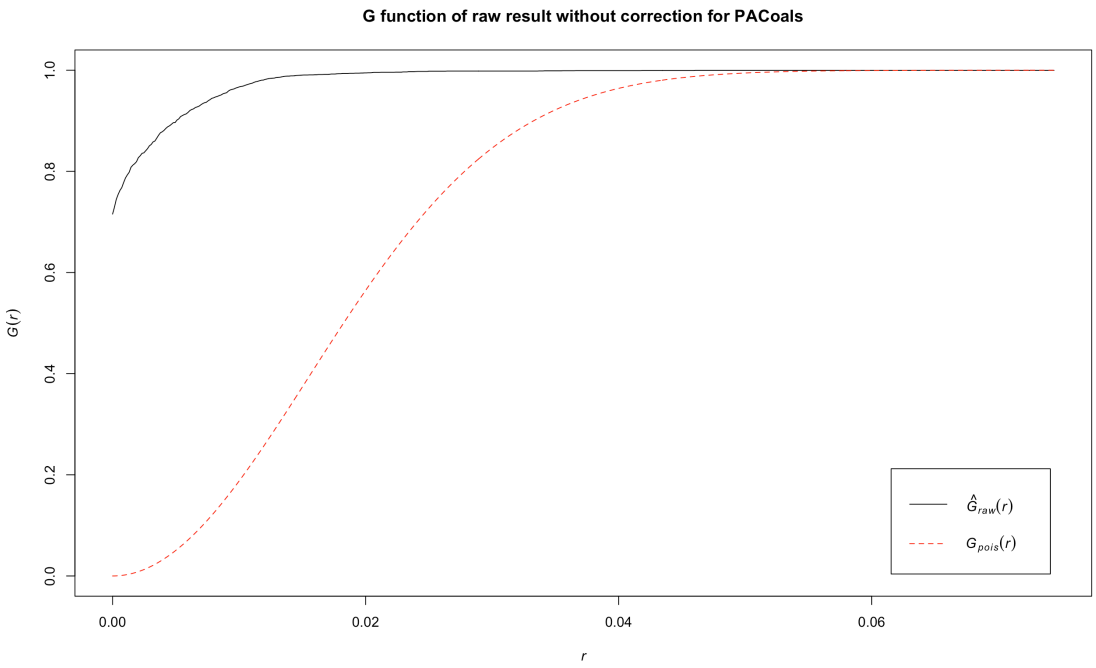
G shows how close together events in the pattern are, while F relates to how far events are from arbitrary locations in the study area.

We allready known that PALocs is in an evenly spaced pattern. Most locations in this pattern are relatively closed to an event, so F rises quickly at low d. However, events are relatively far from each other, so that G initially increases slowly and rises more quickly at longer distancs.

2.2 PACoals

2.2.1 A plot showing the result of G function

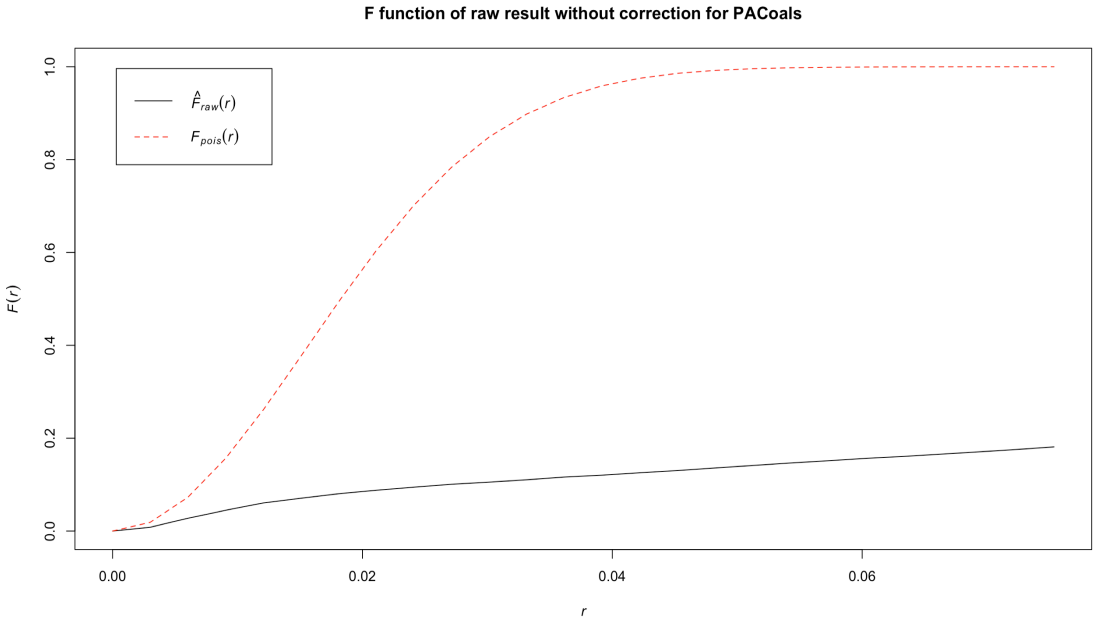
Here is the plot of G function for PACoals shapefile:



r is the distance, Graw(r) is the uncorrected estimate of G(r), that is the empirical distribution of the distance from each ppint in the pattern to the nearest other point of the pattern.Gpois(r) is the theoretical value of G(r) for a stationary Poisson process of the same estimated intensity.

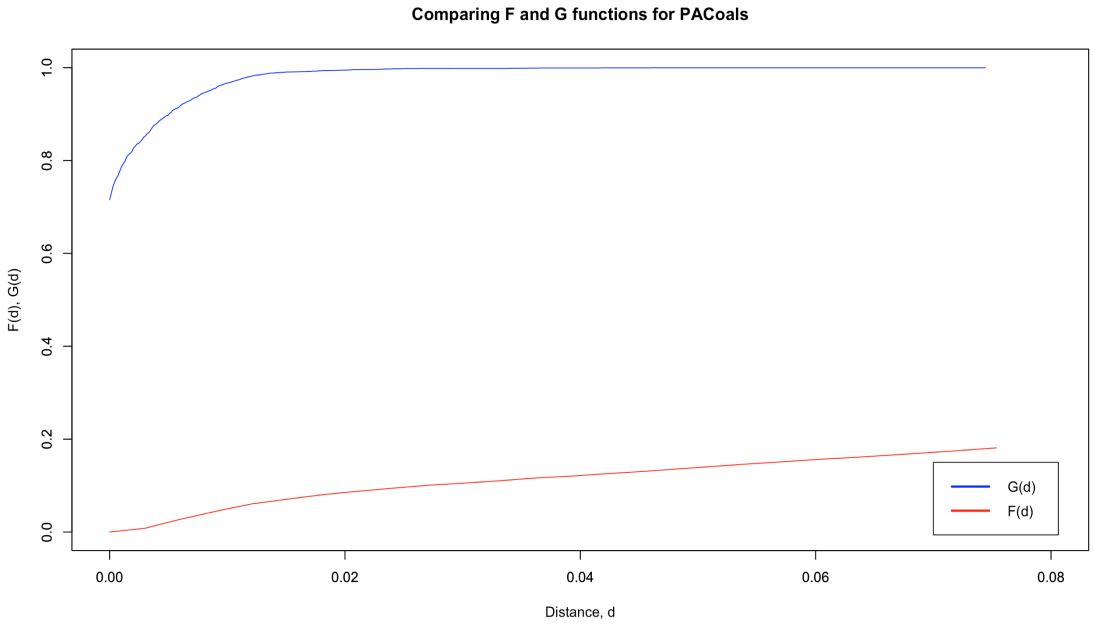
2.2.2 A plot showing the result of F function

Here is the plot of F function for PACoals shapefile:



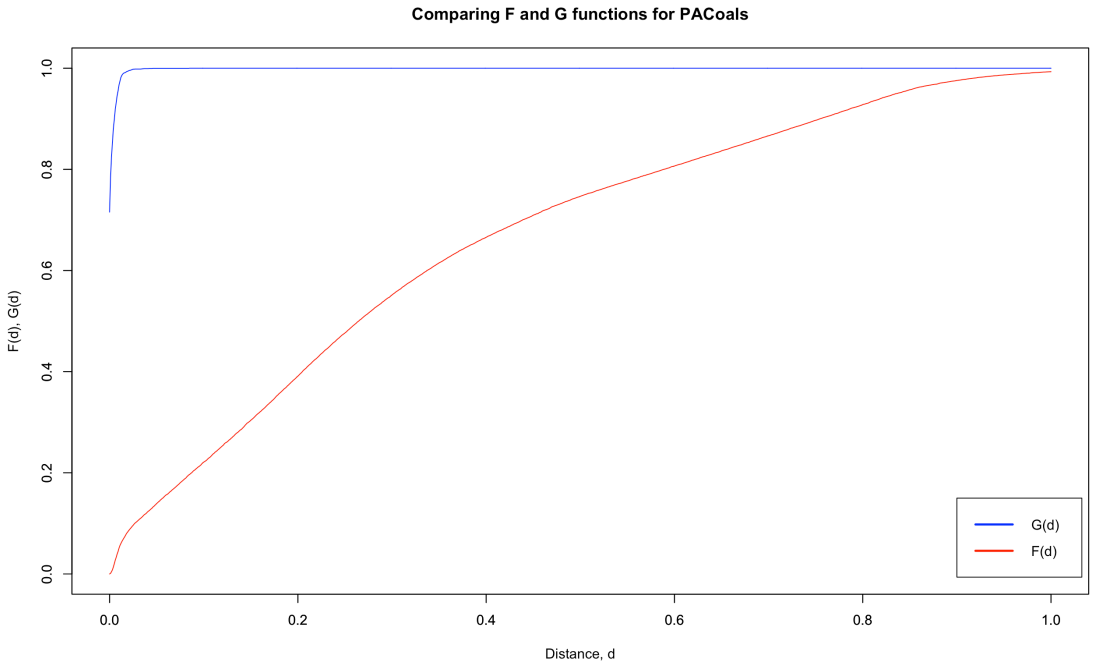
r is the distance, Fraw(r) is the uncorrected estimate of F(r), that is the empirical distribution of the distance from a random point in the window to the nearest point of the data pattern X.

2.2.3 A report comparing the results of G and F functions



Don't u think this graph looks weird? This is because in package `spatstat`, for both `Gest()` and `Fest()` function, distance r is a sensible default. We need to evaluate more distances in this problem. So I tried several sequence and finnaly decide to use seq(from = 0.000, to = 1.000, by = 0.001) for both G and F functions.

Here is the new comparing graph:



We know that PACoals is in clustered in the study area. So, if events are clustered, G rises sharply at short distances because many events have a very close nearest neighbor. The F function rises slowly at first and rapidly at longer distances because a good proportion of study area is fairly empty, so that many locations are at quite long distances from the nearest event in the pattern.

3. Code with comment

```
##### Libraries for Midterm script #####
library(rgdal)  # Bindings for the geospatial data abstraction library.
library(maptools)  # Tools for Reading and Handling Spatial Objects
library(spatstat)  # Spatial Point Pattern Analysis, Model-Fitting, Simulatio
n, Tests

##### 1. PART A #####

##### 1.1 PALocs #####
getwd()  # get the working directory
list.files()  # List the Files in a Directory
PALocs <- readOGR(dsn = "/Users/qijiawen/Desktop/2017 Spring/Spatial Data Ana
lytics/Midterm", layer = "PALocs")  # read PALocs shapefile
summary(PALocs)  # get a summary of PALocs dataset: 67 points in total with 2
0 variables
PALocs.spatialPoints <- as(PALocs, "SpatialPoints")  # change spatial points
data frame to Spatial Points
PALocs.ppp <- as(PALocs.spatialPoints, "ppp")  # change spatial points to spa
tial point pattern class ppp
coords <- as.data.frame(PALocs@coords)  # get the coords table
```

```r
x.max <- max(coords$coords.x1)  # max for x
x.min <- min(coords$coords.x1)  # min for y
y.max <- max(coords$coords.x2)  # max for y
y.min <- min(coords$coords.x2)  # max for x
# I will use 10x10 = 100 quadrats
quadratLength <- (x.max - x.min)/10  # the same length as regular
quadratWidth <- (y.max - y.min)/10  # the same width as regular
set.seed(1)  # to get same result, my favorite seed
randomX <- runif(100, min = x.min, max = x.max - quadratLength)  # generate 1
00 random x
randomY <- runif(100, min = y.min + quadratWidth, max = y.max)  # generate 10
0 random y
randomQuadratCount.PALocs <- matrix()  # create a matrix to save counts
# count the number of event for each random quadrat
for (i in 1:100) {
    # get four cornor coordinates
    x.left = randomX[i]
    x.right = randomX[i] + quadratLength
    y.top = randomY[i]
    y.bottom = randomY[i] - quadratWidth
    # do filters for four sides
    conditionLeft <- coords[coords$coords.x1 > x.left, ]
    conditionRight <- conditionLeft[conditionLeft$coords.x1 < x.right, ]
    conditionTop <- conditionRight[conditionRight$coords.x2 < y.top, ]
    conditionBottom <- conditionTop[conditionTop$coords.x2 > y.bottom, ]
    randomQuadratCount.PALocs[i] <- nrow(conditionBottom)  # count number of
points
}

##### 1.1.1 Map with random quadrat count #####
plot(PALocs.ppp, pch = "*", main = "PALocs Random Quadrat Count (100 rectangl
es)")  # plot points as background map
for (i in 1:100) {
    rect(xleft = randomX[i], ybottom = randomY[i] - quadratWidth, xright = ra
ndomX[i] + quadratLength, ytop = randomY[i], border = "blue")  # draw rectang
les
    text(randomX[i] + 0.5 * quadratLength, randomY[i] - 0.5 * quadratWidth, r
andomQuadratCount.PALocs[i], col = "blue", cex = 0.8)  # add count text
}

##### 1.1.2 Table with statistics #####
RandomCountTable <- table(randomQuadratCount.PALocs)  # aggregate and sort
RandomCountTable <- as.data.frame(RandomCountTable)  # create the table
colnames(RandomCountTable) <- c("No. of Events (K)", "No. of Quadrats(X)")
RandomCountTable$`No. of Events (K)` <- as.integer(as.character(RandomCountTa
ble$`No. of Events (K)`))  # change factor to integer
u = length(PALocs)/100  # u = 0.67
RandomCountTable$`K-u` <- RandomCountTable$`No. of Events (K)` - u  # calcula
te k-u
RandomCountTable$`(K-u)^2` <- RandomCountTable$`K-u` * RandomCountTable$`K-u`
```

```r
# calculate (K-u)^2
RandomCountTable$`X(K-u)^2` <- RandomCountTable$`No. of Quadrats(X)` * Random
CountTable$`(K-u)^2`  # calculate X(K-u)^2
write.csv(RandomCountTable, "PALocsRandom.csv", row.names = FALSE)

##### 1.1.3 Summary based on VMR #####
RandomVariance = sum(RandomCountTable$`X(K-u)^2`)/(100 - 1)  # calculate the
variance s^2 # 0.33808
RandomVMR <- RandomVariance/u  # variance mean ratio # 0.50459

##### 1.2 PACoals #####
PACoals <- readOGR(dsn = "/Users/qijiawen/Desktop/2017 Spring/Spatial Data An
alytics/Midterm", layer = "PACoals")  # read PACoals shapefile
summary(PACoals)  # get a summary of PACoals dataset: 4106 points in total wi
th 22 variables
PACoals.spatialPoints <- as(PACoals, "SpatialPoints")  # change spatial point
s data frame to Spatial Points
PACoals.ppp <- as(PACoals.spatialPoints, "ppp")  # change spatial points to s
patial point pattern class ppp
coords <- as.data.frame(PACoals@coords)  # get the coords table
x.max <- max(coords$coords.x1)  # max for x
x.min <- min(coords$coords.x1)  # min for y
y.max <- max(coords$coords.x2)  # max for y
y.min <- min(coords$coords.x2)  # max for x
# I will use 10x10 = 100 quadrats
quadratLength <- (x.max - x.min)/10  # the same length as regular
quadratWidth <- (y.max - y.min)/10  # the same width as regular
set.seed(1)  # to get same result, my favorite seed
randomX <- runif(100, min = x.min, max = x.max - quadratLength)  # generate 1
00 random x
randomY <- runif(100, min = y.min + quadratWidth, max = y.max)  # generate 10
0 random y
randomQuadratCount.PACoals <- matrix()  # create a matrix to save counts
# count the number of event for each random quadrat
for (i in 1:100) {
    # get four cornor coordinates
    x.left = randomX[i]
    x.right = randomX[i] + quadratLength
    y.top = randomY[i]
    y.bottom = randomY[i] - quadratWidth
    # do filters for four sides
    conditionLeft <- coords[coords$coords.x1 > x.left, ]
    conditionRight <- conditionLeft[conditionLeft$coords.x1 < x.right, ]
    conditionTop <- conditionRight[conditionRight$coords.x2 < y.top, ]
    conditionBottom <- conditionTop[conditionTop$coords.x2 > y.bottom, ]
    randomQuadratCount.PACoals[i] <- nrow(conditionBottom)  # count number of
points
}

##### 1.2.1 Map with random quadrat count #####
```

```r
plot(PACoals.ppp, pch = "*", main = "PACoals Random Quadrat Count (100 rectangles)")  # plot points as background map
for (i in 1:100) {
    rect(xleft = randomX[i], ybottom = randomY[i] - quadratWidth, xright = randomX[i] + quadratLength, ytop = randomY[i], border = "blue")  # draw rectangles
    text(randomX[i] + 0.5 * quadratLength, randomY[i] - 0.5 * quadratWidth, randomQuadratCount.PACoals[i], col = "blue", cex = 0.8)  # add count text
}

##### 1.2.2 Table with statistics #####
RandomCountTable <- table(randomQuadratCount.PACoals)  # aggregate and sort
RandomCountTable <- as.data.frame(RandomCountTable)  # create the table
colnames(RandomCountTable) <- c("No. of Events (K)", "No. of Quadrats(X)")
RandomCountTable$`No. of Events (K)` <- as.integer(as.character(RandomCountTable$`No. of Events (K)`))  # change factor to integer
u = length(PACoals)/100  # u = 41.06
RandomCountTable$`K-u` <- RandomCountTable$`No. of Events (K)` - u  # calculate k-u
RandomCountTable$`(K-u)^2` <- RandomCountTable$`K-u` * RandomCountTable$`K-u` # calculate (K-u)^2
RandomCountTable$`X(K-u)^2` <- RandomCountTable$`No. of Quadrats(X)` * RandomCountTable$`(K-u)^2`  # calculate X(K-u)^2
write.csv(RandomCountTable, "PACoalsRandom.csv", row.names = FALSE)

##### 1.2.3 Summary based on VMR #####
RandomVariance = sum(RandomCountTable$`X(K-u)^2`)/(100 - 1)  # calculate the variance s^2 # 3943.0258
RandomVMR <- RandomVariance/u  # variance mean ratio # 96.0308

##### 2. PART B #####

##### 2.1 PALocs #####

##### 2.1.1 A plot showing the result of G function #####
G <- Gest(PALocs.ppp, correction = "none")  # we are not required to do any correction for G function
plot(G, main = "G function of raw result without correction for PALocs")  # plot the G(r)

##### 2.1.2 A plot showing the result of F function #####
F.function <- Fest(PALocs.ppp, correction = "none")  # F function
plot(F.function, main = "F function of raw result without correction for PALocs", xlim = c(0, 0.8), ylim = c(0, 1))  # plot the F(r)

##### 2.1.3 A report comparing the results of G and F functions #####
plot(G$r, G$raw, type = "l", xlab = "Distance, d", ylab = "F(d), G(d)", main = "Comparing F and G functions for PALocs", col = "blue")
par(new = TRUE)
```

```r
plot(F.function$r, F.function$raw, type = "l", xlab = "", ylab = "", col = "red")
legend(0.7, 0.2, c("G(d)", "F(d)"), lty = c(1, 1), lwd = c(2.5, 2.5), col = c("blue", "red"))  # add a legend

##### 2.2 PACoals #####

##### 2.2.1 A plot showing the result of G function #####
G <- Gest(PACoals.ppp, correction = "none")  # we are not required to do any correction for G function
plot(G, main = "G function of raw result without correction for PACoals")  # plot the G(r)

##### 2.2.2 A plot showing the result of F function #####
F.function <- Fest(PACoals.ppp, correction = "none")  # F function
plot(F.function, main = "F function of raw result without correction for PACoals")  # plot the F(r)

##### 2.2.3 A report comparing the results of G and F functions ####
plot(G$r, G$raw, type = "l", xlab = "Distance, d", ylab = "F(d), G(d)", main = "Comparing F and G functions for PACoals", col = "blue", xlim = c(0, 0.08), ylim = c(0, 1))
par(new = TRUE)
plot(F.function$r, F.function$raw, type = "l", xlab = "", ylab = "", col = "red", xlim = c(0, 0.08), ylim = c(0, 1))
legend(0.07, 0.15, c("G(d)", "F(d)"), lty = c(1, 1), lwd = c(2.5, 2.5), col = c("blue", "red"))  # add a legend

##### Important: Check more distances #####
d = seq(from = 0, to = 1, by = 0.001)
G1 <- Gest(PACoals.ppp, r = d, correction = "none")  # add r argument
plot(G1, main = "G function of raw result without correction for PACoals")
F.function1 <- Fest(PACoals.ppp, r = d, correction = "none")  # F function
plot(F.function1, main = "F function of raw result without correction for PACoals")  # plot the F(r)
plot(G1$r, G1$raw, type = "l", xlab = "Distance, d", ylab = "F(d), G(d)", main = "Comparing F and G functions for PACoals", col = "blue", xlim = c(0, 1), ylim = c(0, 1))
par(new = TRUE)
plot(F.function1$r, F.function1$raw, type = "l", xlab = "", ylab = "", col = "red", xlim = c(0, 1), ylim = c(0, 1))
legend(0.9, 0.15, c("G(d)", "F(d)"), lty = c(1, 1), lwd = c(2.5, 2.5), col = c("blue", "red"))  # add a legend
```