Assignement9

## 

#getinfo.shape("/Users/garbamoussa/Downloads/Assignment 9/Data/PS6\_data/Diseased.shp")  
  
library(raster)

## Warning: package 'raster' was built under R version 4.1.2

## Loading required package: sp

## Warning: package 'sp' was built under R version 4.1.2

library(raster)  
library(spatstat)

## Warning: package 'spatstat' was built under R version 4.1.2

## Loading required package: spatstat.data

## Loading required package: spatstat.geom

## Warning: package 'spatstat.geom' was built under R version 4.1.2

## spatstat.geom 2.4-0

##   
## Attaching package: 'spatstat.geom'

## The following objects are masked from 'package:raster':  
##   
## area, rotate, shift

## Loading required package: spatstat.random

## Warning: package 'spatstat.random' was built under R version 4.1.2

## spatstat.random 2.2-0

## Loading required package: spatstat.core

## Warning: package 'spatstat.core' was built under R version 4.1.2

## Loading required package: nlme

## Warning: package 'nlme' was built under R version 4.1.2

##   
## Attaching package: 'nlme'

## The following object is masked from 'package:raster':  
##   
## getData

## Loading required package: rpart

## Warning: package 'rpart' was built under R version 4.1.2

## spatstat.core 2.4-2

## Loading required package: spatstat.linnet

## Warning: package 'spatstat.linnet' was built under R version 4.1.2

## spatstat.linnet 2.3-2

##   
## spatstat 2.3-4 (nickname: 'Watch this space')   
## For an introduction to spatstat, type 'beginner'

library(maptools)

## Warning: package 'maptools' was built under R version 4.1.2

## Checking rgeos availability: TRUE  
## Please note that 'maptools' will be retired by the end of 2023,  
## plan transition at your earliest convenience;  
## some functionality will be moved to 'sp'.

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──

## ✓ ggplot2 3.3.5 ✓ purrr 0.3.4  
## ✓ tibble 3.1.6 ✓ dplyr 1.0.8  
## ✓ tidyr 1.2.0 ✓ stringr 1.4.0  
## ✓ readr 2.1.2 ✓ forcats 0.5.1

## Warning: package 'tidyr' was built under R version 4.1.2

## Warning: package 'readr' was built under R version 4.1.2

## Warning: package 'dplyr' was built under R version 4.1.2

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## x dplyr::collapse() masks nlme::collapse()  
## x tidyr::extract() masks raster::extract()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x dplyr::select() masks raster::select()

library(ggplot2)  
library(gridExtra)

##   
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':  
##   
## combine

library(cowplot)  
  
library(splancs)

##   
## Spatial Point Pattern Analysis Code in S-Plus  
##   
## Version 2 - Spatial and Space-Time analysis

##   
## Attaching package: 'splancs'

## The following object is masked from 'package:dplyr':  
##   
## tribble

## The following object is masked from 'package:tidyr':  
##   
## tribble

## The following object is masked from 'package:tibble':  
##   
## tribble

## The following object is masked from 'package:raster':  
##   
## zoom

library(cubature)

d <- shapefile("/Users/garbamoussa/Downloads/Assignment 9/Data/PS6\_data/Diseased.shp")

## Warning in .local(x, ...): .prj file is missing

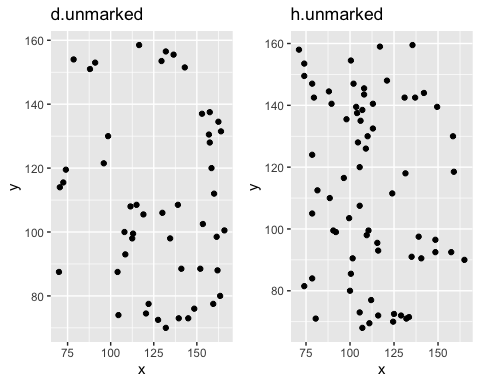
h <- shapefile("/Users/garbamoussa/Downloads/Assignment 9/Data/PS6\_data/Healthy.shp")

## Warning in .local(x, ...): .prj file is missing

d.unmarked <- unmark(as.ppp(d))  
h.unmarked <- unmark(as.ppp(h))

# Plotting the point pattern

plot1 <- ggplot(as.data.frame(d.unmarked), aes(x = x, y = y)) + geom\_point() + ggtitle("d.unmarked")  
plot2 <- ggplot(as.data.frame(h.unmarked), aes(x = x, y = y)) + geom\_point() + ggtitle("h.unmarked")  
  
  
grid.arrange(plot1, plot2, ncol=2, nrow=1)



summary(h.unmarked)

## Planar point pattern: 69 points  
## Average intensity 0.008022323 points per square unit  
##   
## Coordinates are given to 1 decimal place  
## i.e. rounded to the nearest multiple of 0.1 units  
##   
## Window: rectangle = [71, 165] x [68, 159.5] units  
## (94 x 91.5 units)  
## Window area = 8601 square units

attributes(h.unmarked)

## $names  
## [1] "window" "n" "x" "y" "markformat"  
##   
## $class  
## [1] "ppp"

summary(d.unmarked)

## Planar point pattern: 49 points  
## Average intensity 0.00576742 points per square unit  
##   
## Coordinates are given to 1 decimal place  
## i.e. rounded to the nearest multiple of 0.1 units  
##   
## Window: rectangle = [70, 166] x [70, 158.5] units  
## (96 x 88.5 units)  
## Window area = 8496 square units

## 1. Fit a point process model of a stationary Poisson process to each point pattern. What is the intensity of each point pattern? (2 points)

ppm(h.unmarked)

## Stationary Poisson process  
## Intensity: 0.008022323  
## Estimate S.E. CI95.lo CI95.hi Ztest Zval  
## log(lambda) -4.825527 0.1203859 -5.061479 -4.589575 \*\*\* -40.08384

#fit the stationary Poisson process to point pattern 'h.unmarked'  
  
 Qh <- quadscheme(h.unmarked)   
 ppm(Qh)

## Stationary Poisson process  
## Intensity: 0.008022323  
## Estimate S.E. CI95.lo CI95.hi Ztest Zval  
## log(lambda) -4.825527 0.1203859 -5.061479 -4.589575 \*\*\* -40.08384

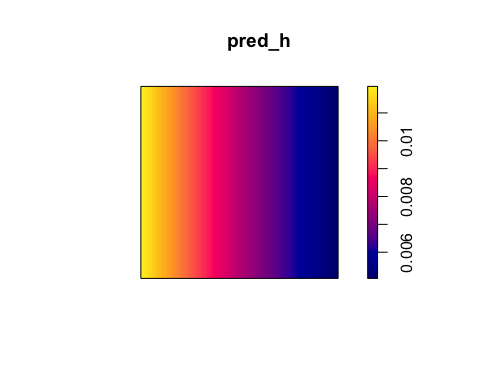
fit\_h <- ppm(h.unmarked, ~ x)  
fit\_h

## Nonstationary Poisson process  
##   
## Log intensity: ~x  
##   
## Fitted trend coefficients:  
## (Intercept) x   
## -3.769155549 -0.009215762   
##   
## Estimate S.E. CI95.lo CI95.hi Ztest Zval  
## (Intercept) -3.769155549 0.517300695 -4.78304628 -2.7552648173 \*\*\* -7.286199  
## x -0.009215762 0.004520312 -0.01807541 -0.0003561122 \* -2.038744

# spatstat predictions  
pred\_h <- predict(fit\_h)  
class(pred\_h)

## [1] "im"

plot(pred\_h)



#fit the stationary Poisson process to point pattern 'd.unmarked'  
  
 Qd <- quadscheme(d.unmarked)   
 ppm(Qd)

## Stationary Poisson process  
## Intensity: 0.00576742  
## Estimate S.E. CI95.lo CI95.hi Ztest Zval  
## log(lambda) -5.15553 0.1428571 -5.435525 -4.875536 \*\*\* -36.08871

# Evaluating Complete Spatial Randomness CSR using the G-Function

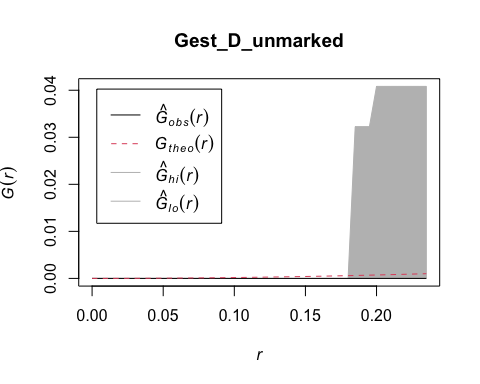
## envelope: Simulation Envelopes of Summary Function and G-Function Function that computes the desired summary statistic for a point pattern.  
set.seed(120109)  
r <- seq(0, sqrt(2)/6, by = 0.005)  
Gest\_D\_unmarked <- envelope(d.unmarked, fun = Gest, r = r, nrank = 2, nsim = 99, correction = "border")

## Generating 99 simulations of CSR ...  
## 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,  
## 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,  
## 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99.  
##   
## Done.

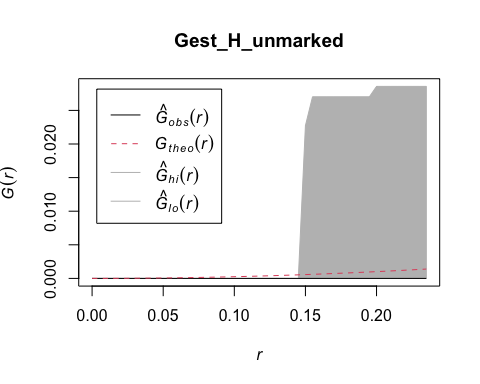
Gest\_H\_unmarked <- envelope(h.unmarked, fun = Gest, r = r, nrank = 2, nsim = 99, correction = "border")

## Generating 99 simulations of CSR ...  
## 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,  
## 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,  
## 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99.  
##   
## Done.

plot(Gest\_D\_unmarked)



plot(Gest\_H\_unmarked)

 ### Evaluating CSR using the F-Function

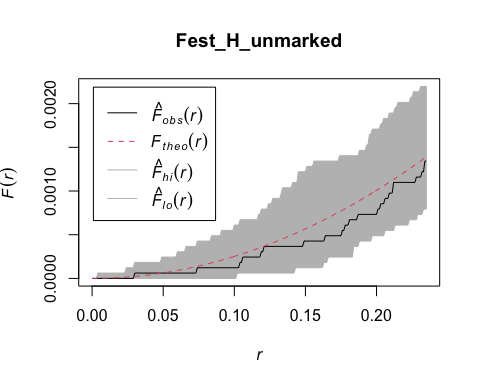
set.seed(120109)  
r <- seq(0, sqrt(2)/6, by = 0.001)  
Fest\_H\_unmarked <- envelope(h.unmarked, fun = Fest, r = r, nrank = 2, nsim = 99, correction = "border")

## Generating 99 simulations of CSR ...  
## 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,  
## 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,  
## 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99.  
##   
## Done.

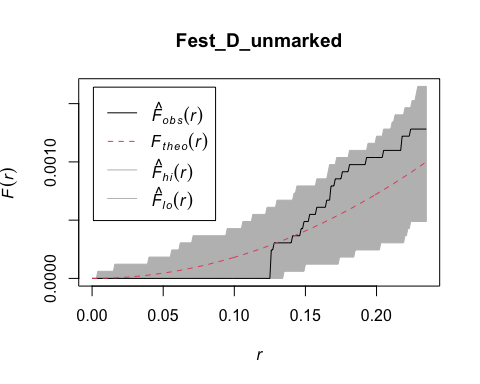
Fest\_D\_unmarked <- envelope(d.unmarked, fun = Fest, r = r, nrank = 2, nsim = 99,correction = "border")

## Generating 99 simulations of CSR ...  
## 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,  
## 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,  
## 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99.  
##   
## Done.

plot(Fest\_H\_unmarked)



plot(Fest\_D\_unmarked)



This plot shows the range of values obtained via simulation as a grey envelope. The observed k-function is much higher than the values obtained via simulation of CSR. This implies clustering, at all distances there are more points near each point than expected under CSR.

A black line inside of the grey envelope implies no significant difference from CSR. One cool thing about the K-function is that results can be scale specific. As scale varies a point pattern can contain no pattern, clustering, or dispersion. A map can be clustered at some scales but dispersed at others.

Another important thing to remember is that in the above simulation the envelope represents a homogeneous Poisson process. However, most interesting questions involve point pattern where CSR is not realistic, we need some new methods…

Assess point pattern to identify whether it is a Homogenous or Inhomogenous Poisson Process and selecting the optimal bandwidth for the h.unmarked dataset

mserwq <- mse2d(as.points(coordinates(as.data.frame(h.unmarked))), as.points(list(x = c(0, + 1, 1, 0), y = c(0, 0, 1, 1))), 100, 0.15)  
bwq <- mserwq$h[which.min(mserwq$mse)]  
bwq

## [1] 0.15

mserw <- bw.diggle(h.unmarked)  
bw <- as.numeric(mserw)  
bw

## [1] 4.968933

## Kernel smoothing using a quartic kernel

poly <- as.points(list(x = c(0, 0, 1, 1), y = c(0, 1, 1, 0)))  
sG <- Sobj\_SpatialGrid(as.SpatialPoints.ppp(h.unmarked), maxDim = 100)$SG  
grd <- slot(sG, "grid")  
summary(grd)

## Grid topology:  
## cellcentre.offset cellsize cells.dim  
## x 71.47 0.94 100  
## y 68.47 0.94 98

k0 <- spkernel2d(as.SpatialPoints.ppp(h.unmarked), poly, h0 = bw, grd)  
k1 <- spkernel2d(as.SpatialPoints.ppp(h.unmarked), poly, h0 = 0.05, grd)  
k2 <- spkernel2d(as.SpatialPoints.ppp(h.unmarked), poly, h0 = 0.1, grd)  
k3 <- spkernel2d(as.SpatialPoints.ppp(h.unmarked), poly, h0 = 0.15, grd)  
df <- data.frame(k0 = k0, k1 = k1, k2 = k2, k3 = k3)  
kernels <- SpatialGridDataFrame(grd, data = df)  
summary(kernels)

## Object of class SpatialGridDataFrame  
## Coordinates:  
## min max  
## x 71 165.00  
## y 68 160.12  
## Is projected: NA   
## proj4string : [NA]  
## Grid attributes:  
## cellcentre.offset cellsize cells.dim  
## x 71.47 0.94 100  
## y 68.47 0.94 98  
## Data attributes:  
## k0 k1 k2 k3   
## Min. : NA Min. : NA Min. : NA Min. : NA   
## 1st Qu.: NA 1st Qu.: NA 1st Qu.: NA 1st Qu.: NA   
## Median : NA Median : NA Median : NA Median : NA   
## Mean :NaN Mean :NaN Mean :NaN Mean :NaN   
## 3rd Qu.: NA 3rd Qu.: NA 3rd Qu.: NA 3rd Qu.: NA   
## Max. : NA Max. : NA Max. : NA Max. : NA   
## NA's :9800 NA's :9800 NA's :9800 NA's :9800

### Kernel smoothing using a Gaussian kernel

cc <- coordinates(kernels)  
xy <- list(x = cc[, 1], y = cc[, 2])  
k4 <- density(h.unmarked, 0.5 \* bw, dimyx = c(100, 100), xy = xy)  
kernels$k4 <- as(k4, "SpatialGridDataFrame")$v  
k5 <- density(h.unmarked, 0.5 \* 0.05, dimyx = c(100, 100), xy = xy)  
kernels$k5 <- as(k5, "SpatialGridDataFrame")$v  
k6 <- density(h.unmarked, 0.5 \* 0.1, dimyx = c(100, 100), xy = xy)  
kernels$k6 <- as(k6, "SpatialGridDataFrame")$v  
k7 <- density(h.unmarked, 0.5 \* 0.15, dimyx = c(100, 100), xy = xy)  
kernels$k7 <- as(k7, "SpatialGridDataFrame")$v  
summary(kernels)

## Object of class SpatialGridDataFrame  
## Coordinates:  
## min max  
## x 71 165.00  
## y 68 160.12  
## Is projected: NA   
## proj4string : [NA]  
## Grid attributes:  
## cellcentre.offset cellsize cells.dim  
## x 71.47 0.94 100  
## y 68.47 0.94 98  
## Data attributes:  
## k0 k1 k2 k3 k4   
## Min. : NA Min. : NA Min. : NA Min. : NA Min. :0.00000   
## 1st Qu.: NA 1st Qu.: NA 1st Qu.: NA 1st Qu.: NA 1st Qu.:0.00008   
## Median : NA Median : NA Median : NA Median : NA Median :0.00230   
## Mean :NaN Mean :NaN Mean :NaN Mean :NaN Mean :0.00800   
## 3rd Qu.: NA 3rd Qu.: NA 3rd Qu.: NA 3rd Qu.: NA 3rd Qu.:0.01262   
## Max. : NA Max. : NA Max. : NA Max. : NA Max. :0.07562   
## NA's :9800 NA's :9800 NA's :9800 NA's :9800 NA's :100   
## k5 k6 k7   
## Min. :0.00000 Min. :0.00000 Min. :0.00000   
## 1st Qu.:0.00000 1st Qu.:0.00000 1st Qu.:0.00000   
## Median :0.00000 Median :0.00000 Median :0.00000   
## Mean :0.00793 Mean :0.00793 Mean :0.00793   
## 3rd Qu.:0.00000 3rd Qu.:0.00000 3rd Qu.:0.00000   
## Max. :1.13173 Max. :1.13173 Max. :1.13173   
## NA's :100 NA's :100 NA's :100

## # Using the log-normal model to calculate intensity

loglambda <- function(x, alpha, beta) {  
 l <- alpha + sum(beta \* c(x, x \* x, prod(x)))  
 return(l)  
}  
  
L <- function(alphabeta, x) {  
 l <- apply(x, 1, loglambda, alpha = alphabeta[1], beta = alphabeta[-1])  
 l <- sum(l)  
 intL <- adaptIntegrate(lowerLimit = c(0, 0), upperLimit = c(1, 1), fDim = 1,  
 tol = 1e-08, f = function(x, alpha = alphabeta[1],  
 beta = alphabeta[-1]) {  
 exp(loglambda(x, alpha, beta))  
 })  
 l <- l - intL$integral  
 return(l)  
}  
  
x <- as.points(lansing[lansing$marks == "maple", ])  
optbeta <- optim(par = c(log(514), 0, 0, 0, 0, 0), fn = L,  
 control = list(maxit = 1000, fnscale = -1), x = x)   
  
optbeta

## $par  
## [1] 5.5568268 5.6608564 -0.9626295 -5.1417654 -1.1562095 0.9591336  
##   
## $value  
## [1] 2778.262  
##   
## $counts  
## function gradient   
## 1001 NA   
##   
## $convergence  
## [1] 1  
##   
## $message  
## NULL

# Calculating intensity using the ppm function

lmaple <- lansing[lansing$marks == "maple", ]  
ppm(Q = lmaple, trend = ~x + y + I(x^2) + I(y^2) + I(x \*y))

## Nonstationary multitype Poisson process  
##   
## Possible marks: 'blackoak', 'hickory', 'maple', 'misc', 'redoak' and 'whiteoak'  
##   
## Log intensity: ~x + y + I(x^2) + I(y^2) + I(x \* y)  
##   
## Fitted trend coefficients:  
## (Intercept) x y I(x^2) I(y^2) I(x \* y)   
## 3.7310742 5.6400643 -0.7663636 -5.0115142 -1.1983209 0.6375824   
##   
## Estimate S.E. CI95.lo CI95.hi Ztest Zval  
## (Intercept) 3.7310742 0.2542004 3.2328505 4.22929795 \*\*\* 14.6776855  
## x 5.6400643 0.7990009 4.0740514 7.20607727 \*\*\* 7.0588964  
## y -0.7663636 0.6990514 -2.1364792 0.60375200 -1.0962907  
## I(x^2) -5.0115142 0.7011631 -6.3857686 -3.63725974 \*\*\* -7.1474300  
## I(y^2) -1.1983209 0.6428053 -2.4581962 0.06155433 -1.8642051  
## I(x \* y) 0.6375824 0.6989167 -0.7322691 2.00743391 0.9122439

# 2. Consider the hypothesis that the density of trees of each type changes along an East-West gradient. For each point pattern, fit a point process model of a non-stationary Poisson process with an intensity function 𝜆 = 𝑒 a+bx. For each point pattern, is the model better than the model you fit in Q1? (4 points)

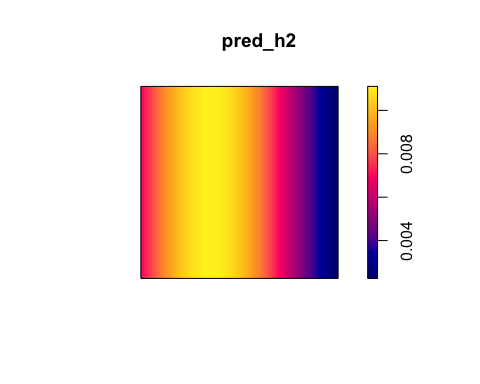
The core R density function can be used to explore the former using a spatial Kernel Density Estimation (KDE) analysis.

fit the nonstationary Poisson process with intensity function lambda(x,y) = exp(a + bx) where x,y are the Cartesian coordinates and a,b are parameters to be estimated

fit\_h2 <- ppm(h.unmarked, ~ polynom(x,2))  
fit\_h2

## Nonstationary Poisson process  
##   
## Log intensity: ~x + I(x^2)  
##   
## Fitted trend coefficients:  
## (Intercept) x I(x^2)   
## -9.2620049112 0.0913072675 -0.0004377178   
##   
## Estimate S.E. CI95.lo CI95.hi Ztest  
## (Intercept) -9.2620049112 2.6904427619 -1.453518e+01 -3.988834e+00 \*\*\*  
## x 0.0913072675 0.0482294204 -3.220660e-03 1.858352e-01   
## I(x^2) -0.0004377178 0.0002096574 -8.486387e-04 -2.679683e-05 \*  
## Zval  
## (Intercept) -3.442558  
## x 1.893186  
## I(x^2) -2.087776

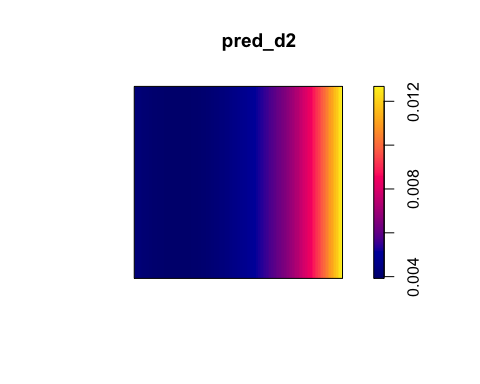
pred\_h2 <- predict(fit\_h2)  
plot(pred\_h2)



fit\_d2 <- ppm((d.unmarked), ~ polynom(x,2))  
fit\_d2

## Nonstationary Poisson process  
##   
## Log intensity: ~x + I(x^2)  
##   
## Fitted trend coefficients:  
## (Intercept) x I(x^2)   
## -3.8961815613 -0.0366529525 0.0002041089   
##   
## Estimate S.E. CI95.lo CI95.hi Ztest  
## (Intercept) -3.8961815613 2.9397156749 -9.6579184090 1.8655552863   
## x -0.0366529525 0.0503258926 -0.1352898895 0.0619839846   
## I(x^2) 0.0002041089 0.0002062412 -0.0002001163 0.0006083342   
## Zval  
## (Intercept) -1.3253600  
## x -0.7283120  
## I(x^2) 0.9896615

pred\_d2 <- predict(fit\_d2)  
plot(pred\_d2)



# 3. Consider the hypothesis that the tree density is a non-linear function of the 𝑥 coordinate. For each point pattern, fit a point process model of a non-stationary Poisson process with an intensity function 𝜆 = 𝑒 a+bx+cx 2. For each point pattern, is the model better than the model you fit in Q2? (4 points)

### Kernel Density Estimation

# First - all with Gaussian kernels but with different bandwidth selection procedures  
density.r.1 <- density(h.unmarked) # default with kernel="gaussian", bw = "nrd0"  
density.r.2 <- density(h.unmarked, kernel="gaussian",bw = "nrd")  
density.r.3 <- density(h.unmarked, kernel="gaussian",bw = "ucv")  
density.r.4 <- density(h.unmarked, kernel="gaussian",bw = "SJ-ste")  
  
# Second - all with Gaussian kernels but with different bandwidth adjustments  
density.r.5 <- density(h.unmarked, kernel="gaussian",bw = "ucv",adjust=0.2) # 20%  
density.r.6 <- density(h.unmarked, kernel="gaussian",bw = "ucv",adjust=0.4) # 40%  
density.r.7 <- density(h.unmarked, kernel="gaussian",bw = "ucv",adjust=0.6) # 60%  
density.r.8 <- density(h.unmarked, kernel="gaussian",bw = "ucv",adjust=0.8) # 80%  
  
# Third - with different kernels  
density.r.9 <- density(h.unmarked, kernel="quartic")

## Warning in density.ppp(h.unmarked, kernel = "quartic"): Bandwidth selection will  
## be based on Gaussian kernel

density.r.10 <- density(h.unmarked, kernel="disc")

## Warning in density.ppp(h.unmarked, kernel = "disc"): Bandwidth selection will be  
## based on Gaussian kernel

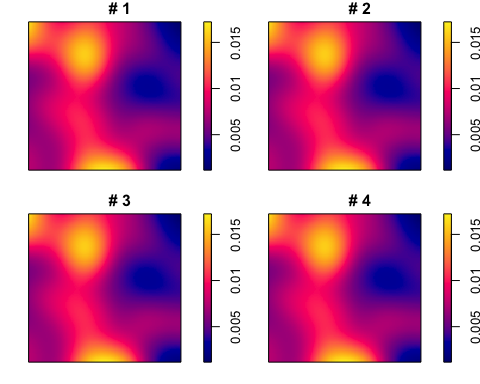
density.r.11 <- density(h.unmarked, kernel="epanechnikov")

## Warning in density.ppp(h.unmarked, kernel = "epanechnikov"): Bandwidth selection  
## will be based on Gaussian kernel

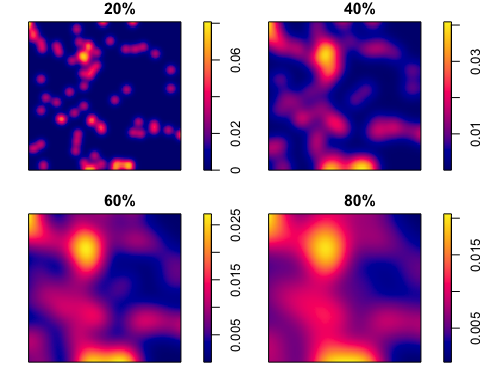
density.r.12 <- density(h.unmarked, kernel="gaussian")

.pardefault <- par(no.readonly = T)

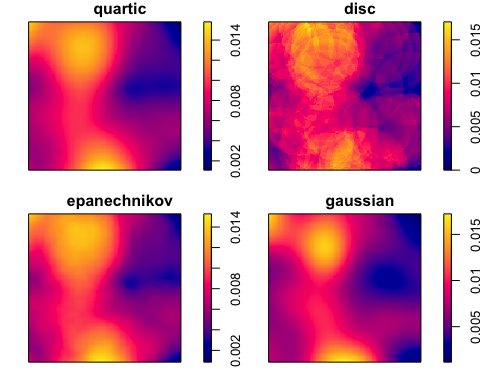
par(mfrow=c(2,2),mar=c(1,1,1,1))  
plot(density.r.1, main = "# 1")  
plot(density.r.2, main = "# 2")  
plot(density.r.3, main = "# 3")  
plot(density.r.4, main = "# 4")



par(mfrow=c(2,2),mar=c(1,1,1,1))  
plot(density.r.5, main = "20%")  
plot(density.r.6, main = "40%")  
plot(density.r.7, main = "60%")  
plot(density.r.8, main = "80%")

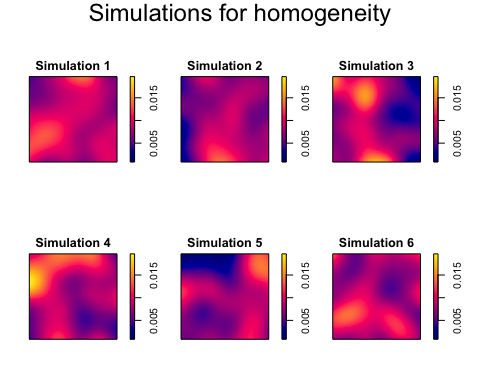


par(mfrow=c(2,2),mar=c(1,1,1,1))  
plot(density.r.9, main="quartic")  
plot(density.r.10, main="disc")  
plot(density.r.11, main="epanechnikov")  
plot(density.r.12, main="gaussian")



# Select a random position for the observed data in the figure.  
# Set the random seed for reproducibility:  
set.seed(123)   
pos <- sample(1:6,1)  
  
# Simulate 5 CSR point patterns:  
  
simp <- rpoispp(lambda = intensity(h.unmarked),win = Window(h.unmarked),nsim=5)  
  
# Replace the simulated set at the pos'th position by the observed dataset:  
tmp <- simp[[pos]]  
simp[[pos]] <- h.unmarked  
simp[[6]] <- tmp  
names(simp)[6] <- "Simulation 6"  
  
# Compute the KDEs:  
densp <- density(simp)

par(mfrow=c(2,3),mar=c(1,1,1,1))  
plot(as.listof(densp), zlim=range(unlist(lapply(densp,range))),  
 main ="Simulations for homogeneity")

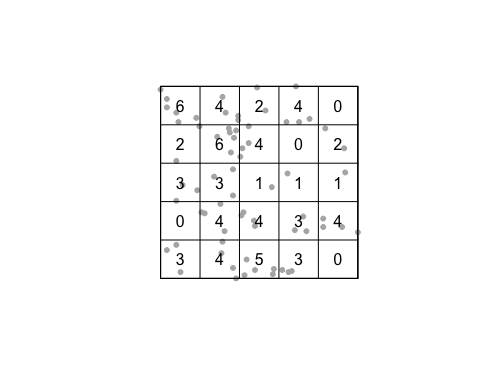


par(.pardefault)

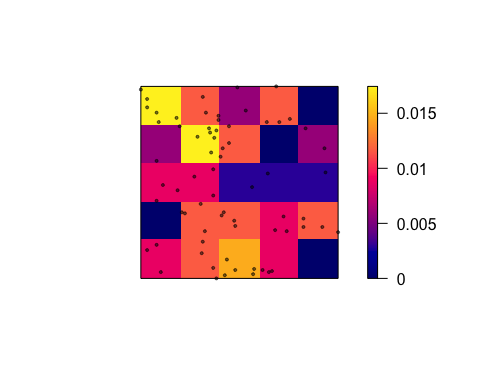
##Quadrat counts

Q <- quadratcount(h.unmarked, nx= 5, ny=5)

plot(h.unmarked, pch=20, cols="grey70", main=NULL) # Plot points  
plot(Q, add=TRUE) # Add quadrat grid



Q.d <- intensity(Q)  
plot(intensity(Q, image=TRUE), main=NULL, las=1) # Plot density raster  
plot(h.unmarked, pch=20, cex=0.6, col=rgb(0,0,0,.5), add=TRUE) # Add points



Intensity of points in quadrats

A chi-square test can then be used to infer if the h.unmarked pattern is homogenous (p > 0.05) or inhomogenous (p < 0.05):

quadrat.test(h.unmarked)

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

##   
## Chi-squared test of CSR using quadrat counts  
##   
## data: h.unmarked  
## X2 = 28.464, df = 24, p-value = 0.4819  
## alternative hypothesis: two.sided  
##   
## Quadrats: 5 by 5 grid of tiles

Thus, the null hypothesis of the h.unmarked seedlings & saplings pattern being generated by a spatial random process is rejected - i.e. strong evidence that the point pattern is inhomogenous (or non-stationary). A strong caveat to the quadrat approach is the subjective choice for the number of quadrats (where the default is 5 by 5 = 25 quadrats).

mod.intercept <- ppm(h.unmarked ~ 1)  
mod.intercept

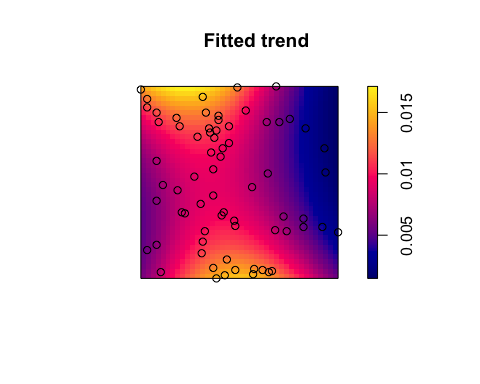
## Stationary Poisson process  
## Intensity: 0.008022323  
## Estimate S.E. CI95.lo CI95.hi Ztest Zval  
## log(lambda) -4.825527 0.1203859 -5.061479 -4.589575 \*\*\* -40.08384

mod.polynomial <- ppm(h.unmarked ~ polynom(x,y,2))  
mod.polynomial

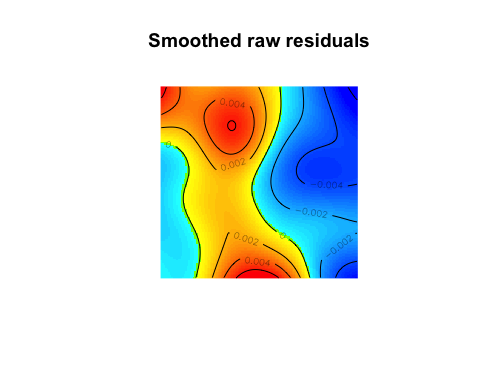
## Nonstationary Poisson process  
##   
## Log intensity: ~x + y + I(x^2) + I(x \* y) + I(y^2)  
##   
## Fitted trend coefficients:  
## (Intercept) x y I(x^2) I(x \* y)   
## -9.8330526439 0.1267273394 -0.0273047815 -0.0004646611 -0.0002593899   
## I(y^2)   
## 0.0002449732   
##   
## Estimate S.E. CI95.lo CI95.hi Ztest  
## (Intercept) -9.8330526439 4.7299377599 -1.910356e+01 -5.625450e-01 \*  
## x 0.1267273394 0.0566744248 1.564751e-02 2.378072e-01 \*  
## y -0.0273047815 0.0500114853 -1.253255e-01 7.071593e-02   
## I(x^2) -0.0004646611 0.0002129870 -8.821080e-04 -4.721424e-05 \*  
## I(x \* y) -0.0002593899 0.0001883085 -6.284678e-04 1.096880e-04   
## I(y^2) 0.0002449732 0.0001868088 -1.211654e-04 6.111118e-04   
## Zval  
## (Intercept) -2.0788968  
## x 2.2360587  
## y -0.5459702  
## I(x^2) -2.1816406  
## I(x \* y) -1.3774730  
## I(y^2) 1.3113577

The first model fitted a stationary Poisson Process, while the second model fitted a non-stationary Poisson Process. The first model assumes the intensity (the density of the redwood seedlings and saplings) is exp(-4.825527 )=0.008022325 throughout the study area (i.e. the trend is stationary and constant). Observe that the exponential arises as the models are log-linear.

plot(mod.polynomial,se=FALSE,how="image")



diagnose.ppm(mod.intercept, which = "smooth")



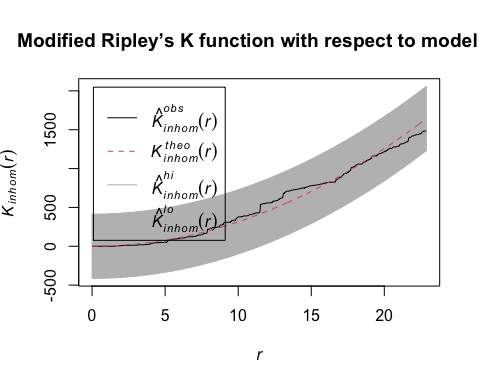
## Model diagnostics (raw residuals)  
## Diagnostics available:  
## smoothed residual field  
## range of smoothed field = [-0.006534, 0.007494]

Here, we can identify areas where the model poorly fits the observed point patterns. We can also use the fitted intensity in the Kinhom function to see if the observed point pattern is more or less clustered than expected from the model fit, again using Ripley’s K function,

K.Ripley.3 <- envelope(h.unmarked,Kinhom,funargs = list(lambda=mod.polynomial),global=TRUE)

## Generating 99 simulations of CSR ...  
## 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40,  
## 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80,  
## 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99.  
##   
## Done.

plot(K.Ripley.3, main="Modified Ripley’s K function with respect to model fit")



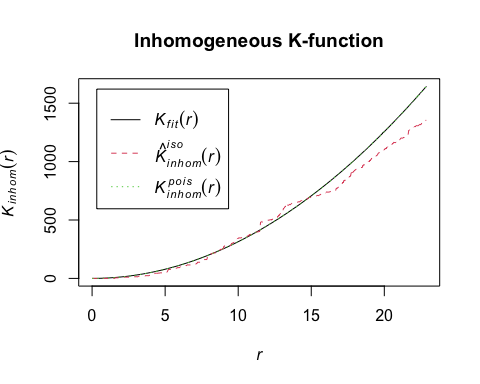
Plot for Modified Ripley’s K function with respect to model fit

We see that the observed point pattern is more clustered than expected based on the model, as the observed curve (Kobs(r)) is consistently higher than the expected curve (Ktheo(r)). One solution would be to use a clustered Poisson point process model via the function kppm:

mod.polynomial.c <- kppm(h.unmarked ~ polynom(x,y,2))  
mod.polynomial.c

## Inhomogeneous cluster point process model  
## Fitted to point pattern dataset 'h.unmarked'  
## Fitted by minimum contrast  
## Summary statistic: inhomogeneous K-function  
##   
## Log intensity: ~x + y + I(x^2) + I(x \* y) + I(y^2)  
##   
## Fitted trend coefficients:  
## (Intercept) x y I(x^2) I(x \* y)   
## -9.8330526439 0.1267273394 -0.0273047815 -0.0004646611 -0.0002593899   
## I(y^2)   
## 0.0002449732   
##   
## Cluster model: Thomas process  
## Fitted cluster parameters:  
## kappa scale   
## 191.4262 756.2934   
## Mean cluster size: [pixel image]

plot(mod.polynomial.c, what="statistic", pause=FALSE, main="Inhomogeneous K-function")

 Plot for Inhomogeneous K-function

In this plot : The dashed red line represents the (iso-corrected) observed K values. The dotted green line represents the expected K values based on Poisson model. The solid black line adds the fitted clustering process (in this case a Thomas process, via kppm) to the Poisson model predictions in (2) Adding a clustering process to the model, clearly improved the model fit (as the solid black line runs through the dashed red line).