M1683. Intensificación en Saberes Técnicos: Diseño y Gestión de Bases de Datos Territoriales. Patrones espaciales puntuales

Domingo Rasilla Álvarez

16 febrero, 2022

Contents

Los datos puntuales proporcionan la ubicación de objetos o eventos dentro de un área. Los objetos pueden ser árboles o el lugar donde se detectó un animal en un rastreo; también pueden ser casas, cementerios o incluso personas. Los eventos pueden ser cosas como el epicentro de un terremoto o donde se inició un incendio forestal. También pueden ser delitos o cuando alguien tuiteó algo.

El análisis de patrones de puntos debe responder a preguntas como

- ¿Cuál es la densidad de un proceso? Si trabaja con incendios forestales, ¿dónde es más alta su densidad?.
- ¿Cómo se distribuyen los puntos en el espacio? Un biólogo necesita saber si una especie invasora está dispersa o agrupada aleatoriamente en un área de estudio.
- ¿Los eventos u objetos influyen en la ubicación de otros eventos u objetos similares? Esto se conoce como 'efecto de contagio.

- ¿Se diferencian (1) - (3) en alguna característica? Es posible que le interese saber si ciertos eventos están más agrupados que otros.
- ¿La densidad de puntos está influenciada por otros procesos (covaría), como el suelo, la altitud etc? .

El análisis de patrones de puntos se centra en la distribución espacial de los eventos observados y hace inferencias sobre el proceso subyacente que los generó. En particular, hay dos cuestiones principales de interés (Bivand et al. 2008):

- La distribución de eventos en el espacio, también denotada como propiedad de primer orden, medida por la intensidad del patrón de puntos,
- Las posibles interacciones entre eventos, también denotada como propiedad de segundo orden, medida por la tendencia de los eventos a aparecer agrupados, independientes o espaciados regularmente.

En esta sesión realizaremos algunos análisis relacionados con patrones espaciales puntuales. Para ello, cubriremos tanto la construcción de objetos espaciales de carácter puntual en R, la exploración básica de estos datos y la construcción de modelos simples.

Esta actividad requiere la instalación y activación de dos paquetes de R: **sf** and **spatstat**.Todos los análisis los realizaremos con funciones que se encuentran en el paquete **spatstat**.

```
# pkgs <- c("spatstat","sf")
# if (!require(package)) install.packages(pkgs)
library(sf)
library(maptools)
library(spatstat)
library(tidyverse)
```

Los ficheros a utilizar son los siguientes:

- Localización de los árboles de una selva tropical (*arboles.csv*) y los valores correspondientes a la pendiente del terreno en la que se asientan (*pendiente.csv*)
- Localización de los secuoyas en un sector del Parque Nacional de Secuoyas en el N de California: *secuoyas*
- Localización de seis especies de árboles en un arboles y el perímetro de la zona de estudio: *arboles*

2. DATOS PUNTUALES

El fichero *arboles.csv* comprende la ubicación de 3605 árboles dentro de una selva tropical. Antes de comenzar el procedimiento de análisis, este fichero debe ser leido por R y convertido en un objeto (`ppp`). Este debe contener, como mínimo: + La localización (**coordenadas espaciales**) de los puntos. + Una ventana espacial que contiene el área de estudio (que puede ser tanto una caja como un polígono). + Opcionalmente, las **marcas** adjuntas a los puntos (atributos, si los hay), + Opcionalmente, el nombre de la unidad de longitud para las coordenadas espaciales.

En el caso del citado conjunto de datos, realizaremos sucesivamente + La importación de los datos en formato csv a R. + Comprobaremos el rango de las coordenadas espaciales (función `summary()`).
+ Crearemos la venta espacial del objeto usando la función `owin()`. + Crearemos el objeto `ppp` con las coordenadas X e Y, acompañado de la ventana espacial generada en el punto anterior.

```
arboles <- read.csv("D:/Docencia_Master_2021/M1683/sesion03/arboles.csv")
str(arboles)
```

```
## 'data.frame':    3604 obs. of  2 variables:
##  $ x: num  11.7 998.9 980.1 986.5 944.1 ...
##  $ y: num  151 430 434 426 415 ...
```

```
summary(arboles)                                                    # Rango de coordena
```

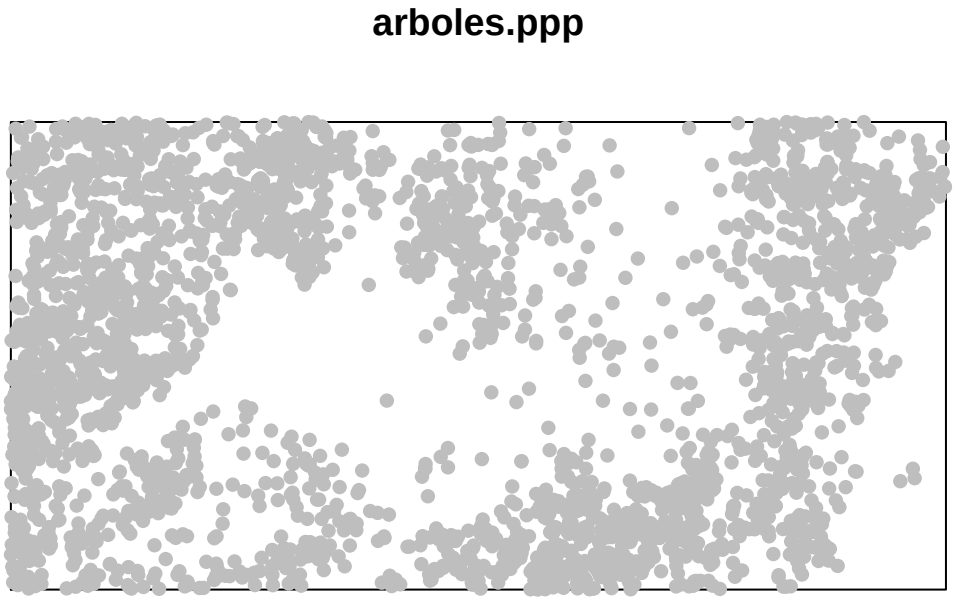
```
##        x               y
##  Min.   :  0.1   Min.   :  0.1
##  1st Qu.:151.9   1st Qu.:103.5
##  Median :342.3   Median :282.6
##  Mean   :433.8   Mean   :263.5
##  3rd Qu.:719.0   3rd Qu.:416.1
##  Max.   :998.9   Max.   :499.9
```

Una vez conocido el rango de coordenadas, podemos crear la ventana espacial y transformar el fichero original en un objeto `ppp`.

```
arboles.owin <- owin(xrange = c(0,1000),                            # Creación
                     yrange = c(0,500))
arboles.ppp <- ppp(arboles$x, arboles$y,                            # Creación
                   window = arboles.owin)
```

Con las funciones `plot()` y `summary()` se obtiene información sobre el contenido del objeto que acabamos de crear.

```
plot(arboles.ppp, pch = 16, cols = 'gray')
```



```r
summary(arboles.ppp)
```

```
## Planar point pattern:  3604 points
## Average intensity 0.007208 points per square unit
##
## Coordinates are given to 1 decimal place
## i.e. rounded to the nearest multiple of 0.1 units
##
## Window: rectangle = [0, 1000] x [0, 500] units
## Window area = 5e+05 square units
```

La función `summary()` además devuelve el número de objetos (en este caso árboles) por unidad de superficie.

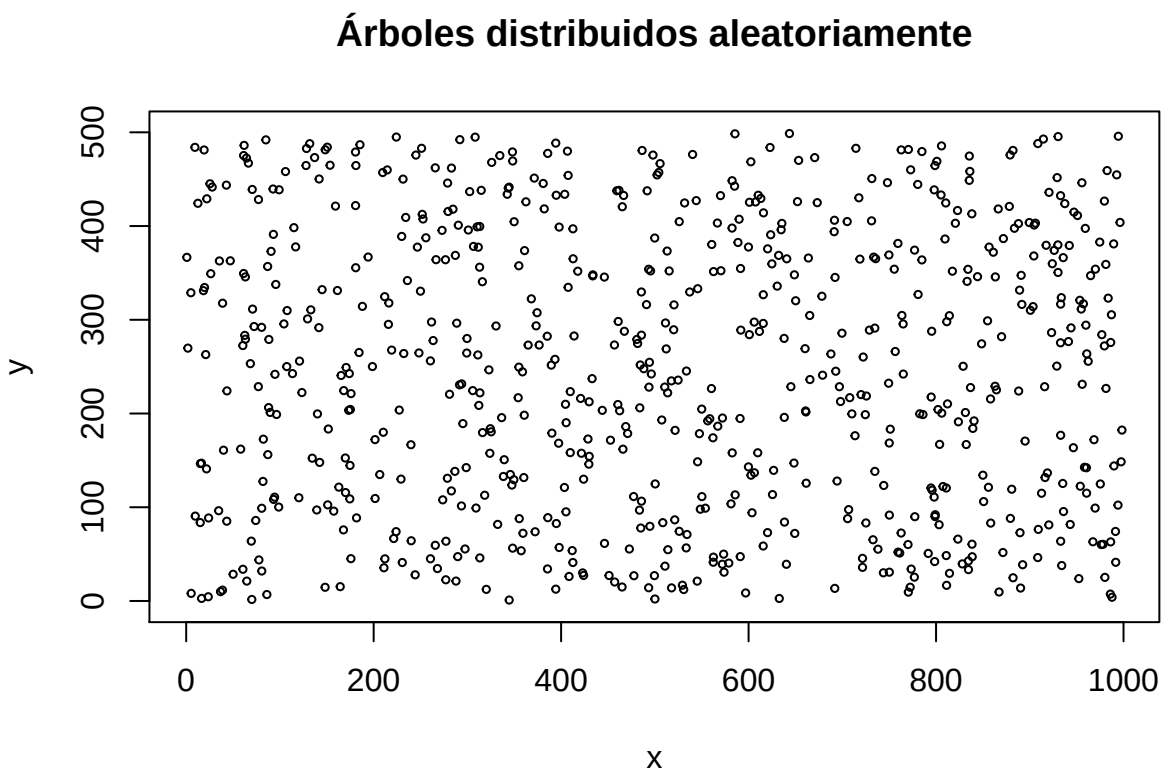
Con el fin de contrastar métodos estadísticos particulares para el análisis de patrones de puntos espaciales, se van a añadir dos conjuntos de datos artificiales. Uno de ellos, **random.ppp**, incluirá puntos distribuidos aleatoriamente. El otro conjunto de datos, **regular.ppp**, constará de puntos distribuidos de forma casi regular.

Para crear un objeto compuesto de puntos distribuidos aleatoriamente sobre el espacio, se generará un secuencia de números aleatorios con la función `runif()`, aunque es necesario fijar la semilla de aleatorización para que se obtengan los mismos resultados iniciales

```r
set.seed(1111)                                                    # Semilla para la repro
x.random <- runif(n = nrow(arboles)*0.2,                              # Número de filas (
                  min = min(arboles$x),                               # Valor mínimo.
                  max = max(arboles$x))                               # Valor máximo.
y.random <- runif(n = nrow(arboles)*0.2,
                  min = min(arboles$y),
                  max= max(arboles$y))
```

Una vez generados los vectores con números aleatorios, podemos agruparlos en un único dataframe y dibujarlos

```r
random.ppp  <- data.frame('x' = x.random, 'y' = y.random)
plot(x=random.ppp$x,
     y=random.ppp$y,
     main="Árboles distribuidos aleatoriamente",
     xlab="x", ylab="y", cex=.5,
     asp=T)
```

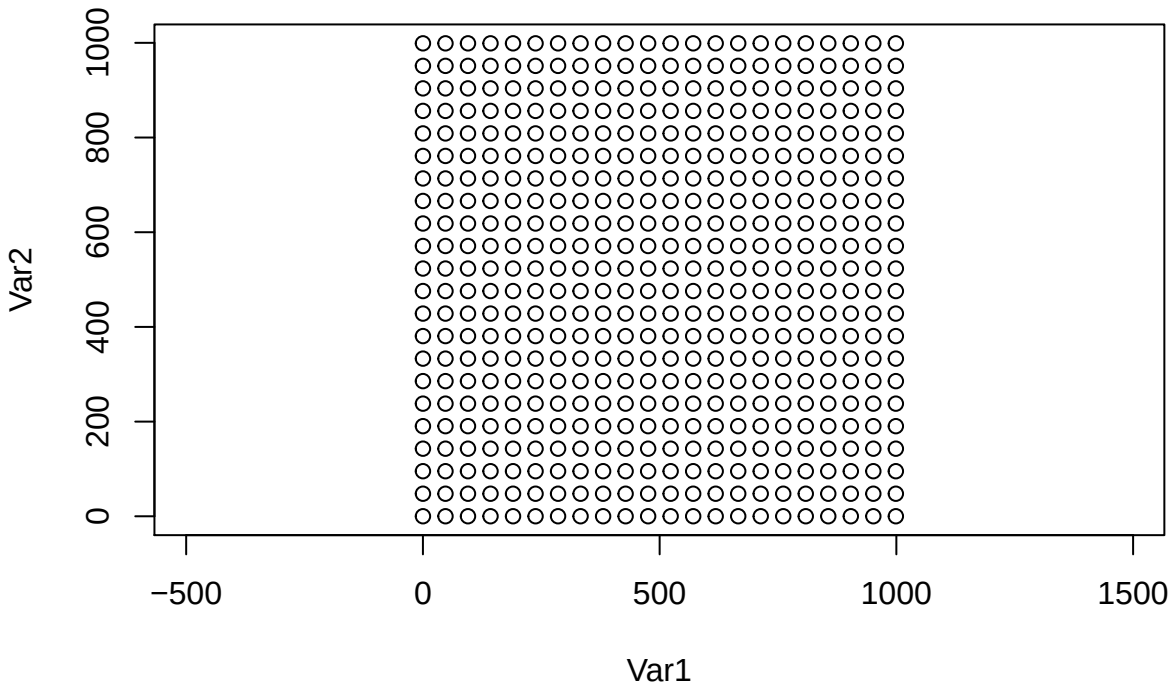


la creación de un objeto con puntos espaciados regularmente sigue los mismos pasos, pero a diferencia de los puntos aleatorios utiliza la función `seq()` y añade la función `expand.grid()`.

```
x.regular <- seq(from=min(arboles$x), to=max(arboles$x),
                 length.out = sqrt(nrow(arboles))*0.35)
y.regular <- seq(from=min(arboles$x),to=max(arboles$x),
                 length=sqrt(nrow(arboles))*0.35)
```

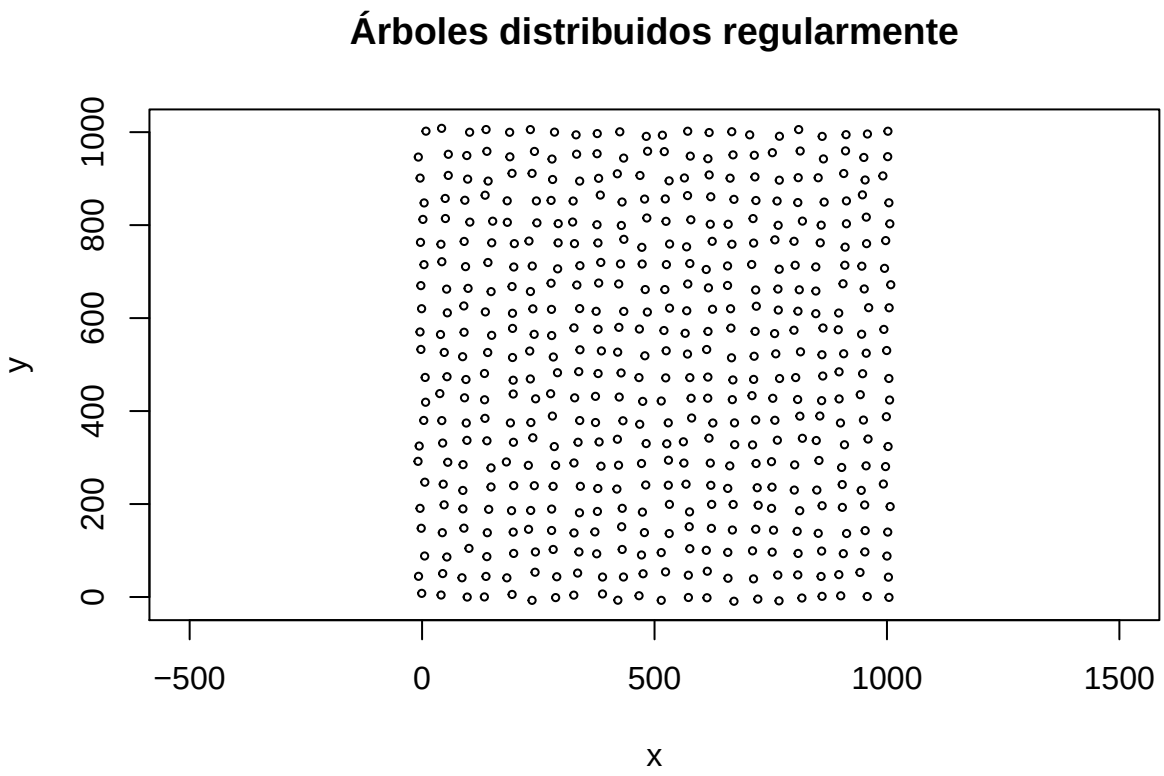
El segundo paso es crear el dataframe, a partir de todas las combinaciones de variables usando la función `expand.grid()`. Veremos posteriormente que es necesario modificar los resultados de esta función.

```
regular <- expand.grid(x.regular, y.regular)
plot(regular, asp=T)
```



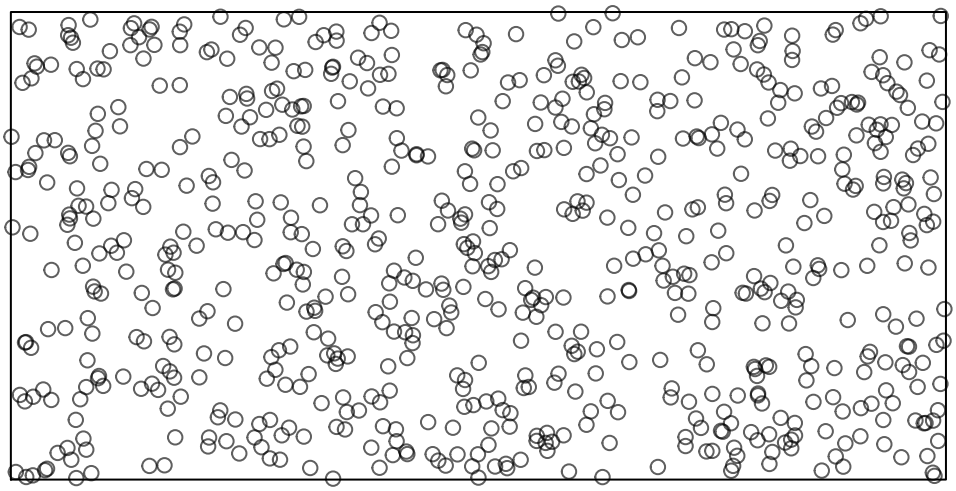
Finalmente, se añade algún ruido aleatorio a los puntos usando la función `jitter()` y se combinan en un dataframe.

```
x.regular <- jitter(regular[,1], factor = 1)
y.regular <- jitter(regular[,2], factor = 1)
regular.ppp <- data.frame('x' = x.regular, 'y' = y.regular)
plot(x=regular.ppp$x,
     y=regular.ppp$y,
     main="Árboles distribuidos regularmente",
     xlab="x",ylab="y", cex=.5,
     asp=T)
```



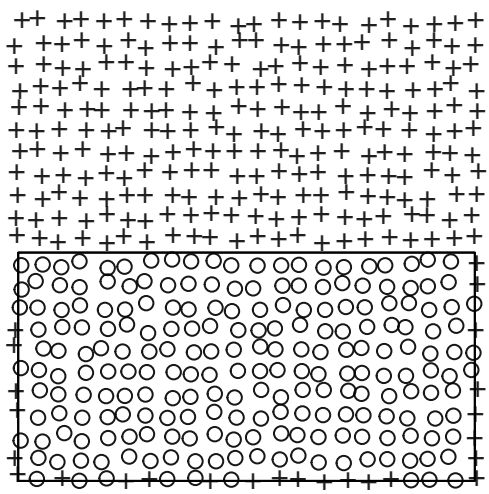
Finalmente, convertiremos ambos conjuntos de datos en objetos `ppp`. Primero, el objeto **ppp.random**

```
ppp.random <- ppp(x = random.ppp$x,
                  y = random.ppp$y,
                  window = arboles.owin)
plot(ppp.random)
```



En segundo lugar, el objeto.ppp.regular.

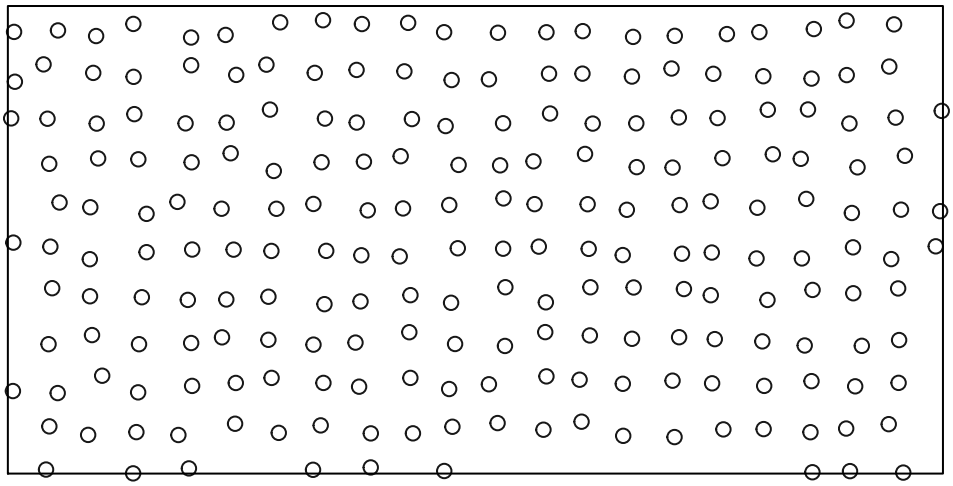
```
ppp.regular <- ppp(x = regular.ppp$x,
                   y = regular.ppp$y,
                   window = arboles.owin)
plot(ppp.regular)
```



Obsérvese, que tenemos un gran número de puntos fuera de la ventana de trabajo, debido a que la función `expand.grid()`utiliza todas las posibles combinaciones. Para eliminar los puntos sobrantes, recurriremos a forzar esos puntos dentro de la ventana `owin`de los datos originales

```
ppp.regular  <- as.ppp(ppp.regular)                                    # Forzamos la expulsión de pu
plot(ppp.regular)
```

ppp.regular



3. ESTADÍSTICOS BÁSICOS.

Antes de considerar otros estadísticos más sofisticados, iniciaremos nuestro análisis calculando el centro medio y la distancia estándar. Para calcular estos valores, podemos actuar de dos maneras: + Extraer las coordenadas x e y del objeto *ppp* usando la función `coords()` correspondiente del paquete **spatstat**. + Recurrir al objeto original *arboles*.

Utilizaremos la primera opción

```
xy <- coords(arboles.ppp)
```

El centro medio se calcula de la siguiente manera.

```
mc <- summarize(xy, xmean = mean(x), ymean = mean(y)) # Centro medio
mc
```

```
##      xmean    ymean
## 1 433.7792 263.5103
```

```
# Alternativamente
mc2 <- apply(xy, 2, mean)
mc2
```

```
##        x        y
## 433.7792 263.5103
```

A continuación se calcula la distancia estándar.

```
sd <- sqrt(sum((xy[,1] - mc[1])^2 + (xy[,2] - mc[2])^2) / nrow(xy))
sd
```

```
## [1] 7.275819
```

4. INTENSIDAD DE EVENTOS EN EL ESPACIO

Una primer cuestión a determinar es la intensidad del patrón de puntos en el espacio (propiedad de primer orden), esto es, la densidad promedio de puntos o, en otras palabras, el número esperado de puntos por unidad de área.

La intensidad puede ser constante (uniforme) o puede variar de un lugar a otro (no uniforme o no homogéneo).

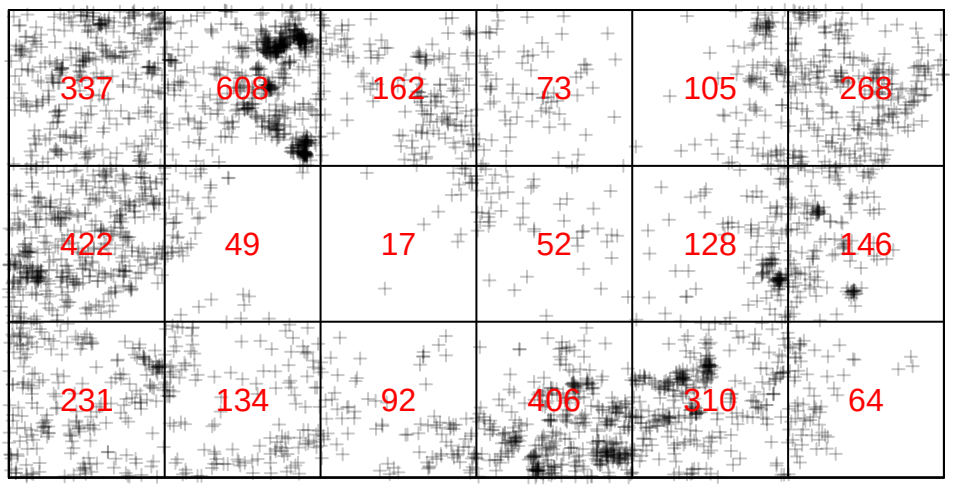
4.1 Distribución en cuadrículas

Un enfoque para evaluar la intensidad es dividir el área de estudio en cuadrantes y contar el número de puntos que caen en cada cuadrante. Si los puntos tienen una intensidad uniforme y son completamente aleatorios, entonces los recuentos de cuadrantes deben ser números aleatorios de Poisson con media constante. Podemos probar esa hipótesis usando el estadístico Chi cuadrado de prueba de bondad de ajuste

La imagen resultante de la función `plot()` muestra que la distribución espacial de árboles en nuestra zona de trabajo no es uniforme ni homogénea. Para comprobarlo, existen diferentes procedimientos para evaluar cómo varía en el espacio esta distribución.

Probablemente, el más simple es dividir el área de trabajo en *cuadrantes* y contar el número de objetos (árboles) que incluye cada cuadrante. En el paquete **spatstat** esto se realiza con la función `quadratcount` que incluye dos parámetros, `nx` y `ny`, para que el investigador defina el número de cuadrantes.

```
arboles.qc <- quadratcount(arboles.ppp,                          # Objeto 'ppp'
                           nx = 6,                             # Número de cuadrantes según la componen
                           ny = 3)                             # Número de cuadrantes según la componen
plot(arboles.ppp, pch=3, cex=0.6)
plot(arboles.qc, add=T, textargs = list(col='red'))
```



Es conveniente comprobar cuán sensibles son los resultados a rejillas de diferentes tamaños, cambiando el tamaño de filas y columnas.

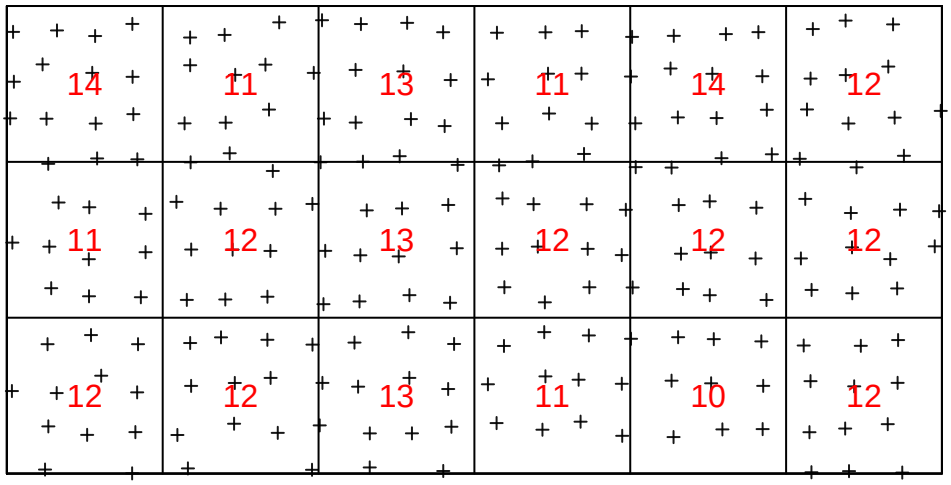
```
arboles.qc2 <- quadratcount(arboles.ppp, nx= 20, ny=8)
plot(arboles.qc2)
```

arboles.qc2

32	63	53	46	22	148	119	18	13	17	11	8	4	0	2	11	50	34	13	13
31	32	23	32	34	102	22	9	19	39	17	15	6	0	1	17	30	36	50	33
22	35	40	16	18	21	103	1	17	23	14	3	4	2	2	9	32	32	31	3
43	44	50	25	5	0	0	0	1	5	17	7	8	3	6	12	25	39	11	0
100	50	46	21	1	2	0	0	1	0	2	1	4	3	5	4	80	26	20	0
44	27	15	16	4	5	11	2	3	3	3	10	5	1	20	12	26	6	3	1
23	14	22	42	6	7	17	11	3	2	10	33	57	47	113	25	38	14	0	2
39	30	13	14	17	22	20	7	14	35	53	91	92	61	32	13	17	11	0	0

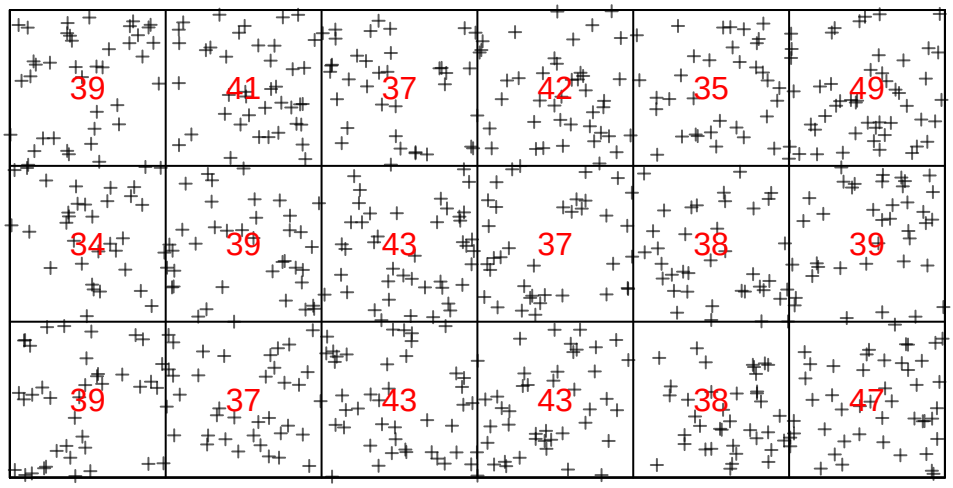
En el caso de los puntos distribuidos regularmente

```
regular.qc <- quadratcount(ppp.regular,                                    # Objeto 'ppp'
                           nx = 6,                                         # Número de cuadrantes según la co
                           ny = 3)                                         # Número de cuadrantes según la co
plot(ppp.regular, pch=3, cex=0.6)
plot(regular.qc, add=T, textargs = list(col='red'))
```



Finalmente, para los puntos distribuidos aleatoriamente

```
random.qc <- quadratcount(ppp.random,                                    # Objeto 'ppp'
                          nx = 6,                                        # Número de cuadrantes según la co
                          ny = 3)                                        # Número de cuadrantes según la co
plot(ppp.random, pch=3, cex=0.6)
plot(random.qc, add=T, textargs = list(col='red'))
```



4.2 Estadístico VMR

Adicionalmente, es posible calcular el valor del estadístico VMR (variance mean ratio), es decir, la relación entre la varianza y la media. *Valores superiores a 1* indicarían una tendencia al agrupamiento espacial de los árboles. Para su cálculo es necesario primero convertir el objeto `arboles.qc`en un dataframe

```
Qcount <- data.frame(arboles.qc)
```

Posteriormente, podemos calcular VMR como

```
var(Qcount$Freq)/mean(Qcount$Freq)
```

```
## [1] 129.869
```

Para el caso de los puntos aleatorios y los regulares

```
Qcount.random <- data.frame(random.qc)
Qcount.regular <- data.frame(regular.qc)
```

Posteriormente, podemos calcular VMR como

```r
var(Qcount.random$Freq)/mean(Qcount.random$Freq)
```

```
## [1] 0.3852941
```

```r
var(Qcount.regular$Freq)/mean(Qcount.regular$Freq)
```

```
## [1] 0.09243697
```

4.3 Análisis con Chi cuadrado.

A continuación, podemos probar si la distribución en cuadrantes difiere significativamente de una distribución uniforme u homogénea, utilizando para ello la función `quadrat.test()`. Esta función compara el número de objetos (árboles) existente en cada cuadrícula con el número de árboles esperados si éstos se distribuyeran uniformemente (número de árboles / número de cuadrantes). Las diferencias entre ambas distribuciones son sometidas a un test de Chi-cuadrado, consistiendo la hipótesis nula en un homogeneidad espacial o distribución equitativa de los árboles según cuadrantes.

```r
arboles.qt <- quadrat.test(arboles.ppp,
                       nx = 6,
                       ny = 3)
arboles.qt
```

```
##
##  Chi-squared test of CSR using quadrat counts
##
## data:  arboles.ppp
## X2 = 2207.8, df = 17, p-value < 2.2e-16
## alternative hypothesis: two.sided
##
## Quadrats: 6 by 3 grid of tiles
```

```r
plot(arboles.qt, cex=0.8)
```

arboles.qt

337 200.2 9.7	608 200.2 29	162 200.2 −2.7	73 200.2 −9	105 200.2 −6.7	268 200.2 4.8
422 200.2 16	49 200.2 −11	17 200.2 −13	52 200.2 −10	128 200.2 −5.1	146 200.2 −3.8
231 200.2 2.2	134 200.2 −4.7	92 200.2 −7.6	406 200.2 15	310 200.2 7.8	64 200.2 −9.6

Los p-values calculados sugieren la posibilidad de rechazar la hipótesis nula y aceptar la alternativa, es decir, que existe una distribución no homogénea. De esto se deriva la posibilidad de que haya algún tipo de proceso espacial que controla la distribución de los árboles sobre el espacio de trabajo.

```
random.qt <- quadrat.test(ppp.random,
                          nx = 6,
                          ny = 3)
random.qt
```

```
##
##  Chi-squared test of CSR using quadrat counts
##
## data:  ppp.random
## X2 = 6.55, df = 17, p-value = 0.02268
## alternative hypothesis: two.sided
##
## Quadrats: 6 by 3 grid of tiles
```

```
plot(random.qt, cex=0.8)
```

random.qt

39 40 −0.16	41 40 0.16	37 40 −0.47	42 40 0.32	35 40 −0.79	49 40 1.4
34 40 −0.95	39 40 −0.16	43 40 0.47	37 40 −0.47	38 40 −0.32	39 40 −0.16
39 40 −0.16	37 40 −0.47	43 40 0.47	43 40 0.47	38 40 −0.32	47 40 1.1

También con una distribución espacial regular.

```
regular.qt <- quadrat.test(ppp.regular,
                           nx = 6,
                           ny = 3)
regular.qt
```

```
##
##  Chi-squared test of CSR using quadrat counts
##
## data:  ppp.regular
## X2 = 1.5714, df = 17, p-value = 1.072e-06
## alternative hypothesis: two.sided
##
## Quadrats: 6 by 3 grid of tiles
```

```
plot(regular.qt, cex=0.8)
```

regular.qt

14 12.1 0.56	11 12.1 −0.3	13 12.1 0.27	11 12.1 −0.3	14 12.1 0.56	12 12.1 −0.016
11 12.1 −0.3	12 12.1 −0.016	13 12.1 0.27	12 12.1 −0.016	12 12.1 −0.016	12 12.1 −0.016
12 12.1 −0.016	12 12.1 −0.016	13 12.1 0.27	11 12.1 −0.3	10 12.1 −0.59	12 12.1 −0.016

4.4 Análisis con covariables.

Como alternativa a la aproximación basada en cuadrículas, se puede utilizar una covariable para evaluar si existe asociación entre los valores de esa covariable y la distribución espacial del objeto en cuestión. En este caso, se trata de comprobar si hay algún tipo de relación entre los árboles y los valores de la pendiente de la zona de trabajo. Estos valores de pendiente se hallan en el fichero `pendiente.csv`, en formato rejilla.

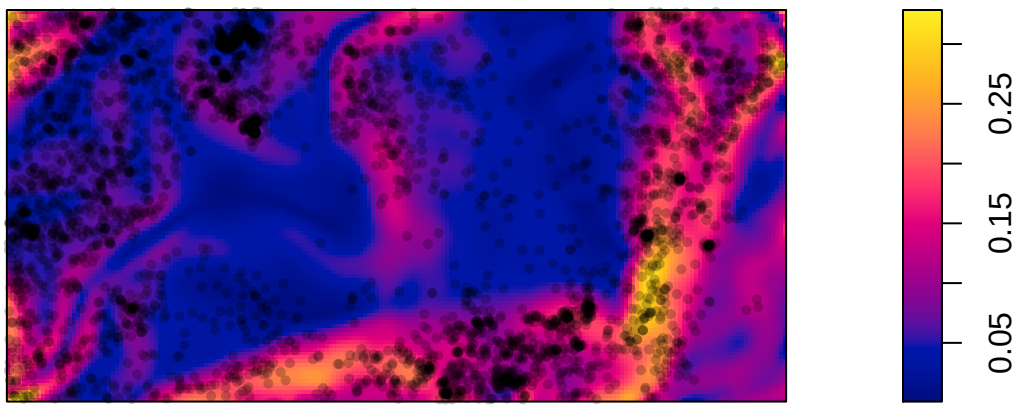
Como en el caso anterior, hay que proceder en varias fases: 1. Leer el conjunto de datos en rejilla; 2. Crear una nueva ventana para los datos en rejilla (con una rejilla de 10m); 3. Crear un objeto en formato raster (`pixel image`) usando la función `as.im()`.

Además, definiremos el tamaño de la ventana espacial apropiada para estos datos de forma manual, especificando los valores máximos y mínimos de las coordenadas `x` e `y` del área de trabajo.

```
arboles.pendiente <- read.csv("D:/Docencia_Master_2021/M1683/sesion03/pendiente.csv", header = FALSE)
```

Transformación en objeto `im`, estableciendo primero las dimensiones de la ventana (un poco más grande)

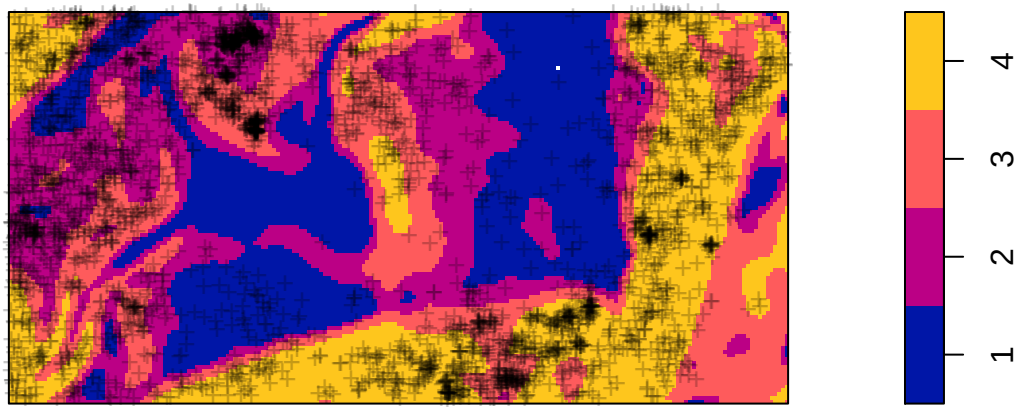
```
arboles.pendiente.owin <- owin(xrange = c(-2.5, 1002.5),
                               yrange = c(-2.5, 502.5))

arboles.pendiente <- as.im(as.matrix(arboles.pendiente),
                  W = arboles.pendiente.owin)

plot(arboles.pendiente)
plot(arboles.ppp, pch = 16, cex = 0.7, add = TRUE)
```



Es posible utilizar el método de la cuadrícula con estos nuevos datos. Para ello, es necesario convertir los valores (variable continua) de la pendiente en categorías (variable discreta), y posteriormente verificar la asociación de cada categoría.

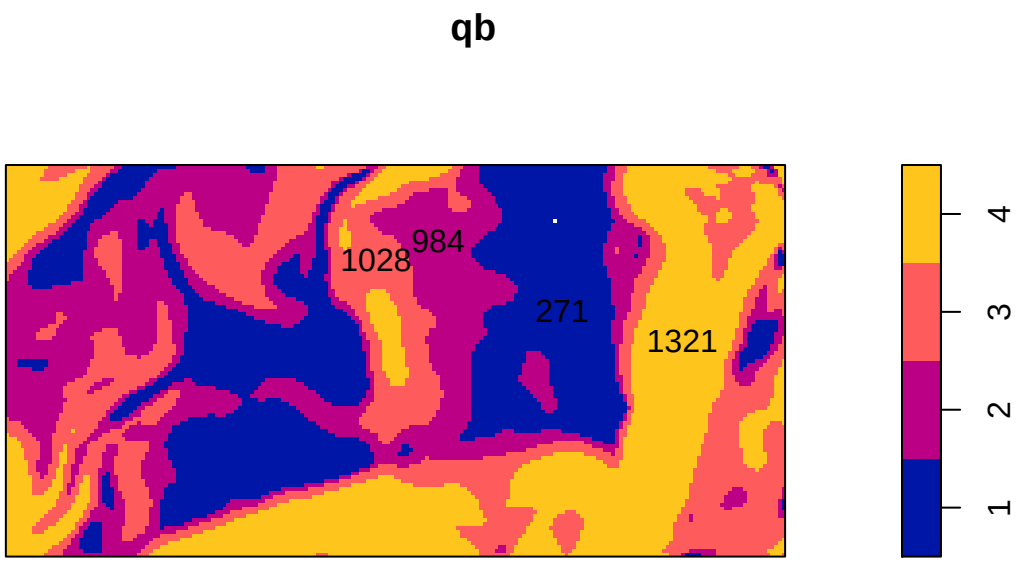
El primer paso es la creación de las categorías: 1. Calcular los cuartiles de la distribución de la pendiente, que serán utilizados como límites de cada categoría; 2. La función `cut()` asignará cada uno de los valores de pendiente a una de las 4 categorías creadas con los cuartiles; 3. Creación de una `tessellation` basada en las categorías, que se usará para identificar a qué clase pertenece cada objeto, usando la función `tess()`.

```
b <- quantile(arboles.pendiente,                                    # Cálculo de cuan
              probs = seq(0,1, by = 0.25))

arboles.pendiente.cut <- cut(arboles.pendiente,                     # Asignación de v
                     breaks = b,
                     labels = 1:4)

arboles.pendiente.tess <- tess(image = arboles.pendiente.cut)       # "Tesselation"

plot(arboles.pendiente.tess, valuesAreColours=FALSE)

plot(arboles.ppp, add = TRUE, pch = "+")
```



Es a partir de este momento cuando podemos utilizar la función `quadratcount()`, pero en vez de una cuadrícula regular usaremos la `tessellation`.

```
qb <- quadratcount(arboles.ppp, tess = arboles.pendiente.tess)
plot(qb, valuesAreColours=FALSE)
```



Es posible mostrar en pantalla la composición de cada una de las categorías

```
as.table(qb)
```

```
## tile
##    1    2    3    4
##  271  984 1028 1321
```

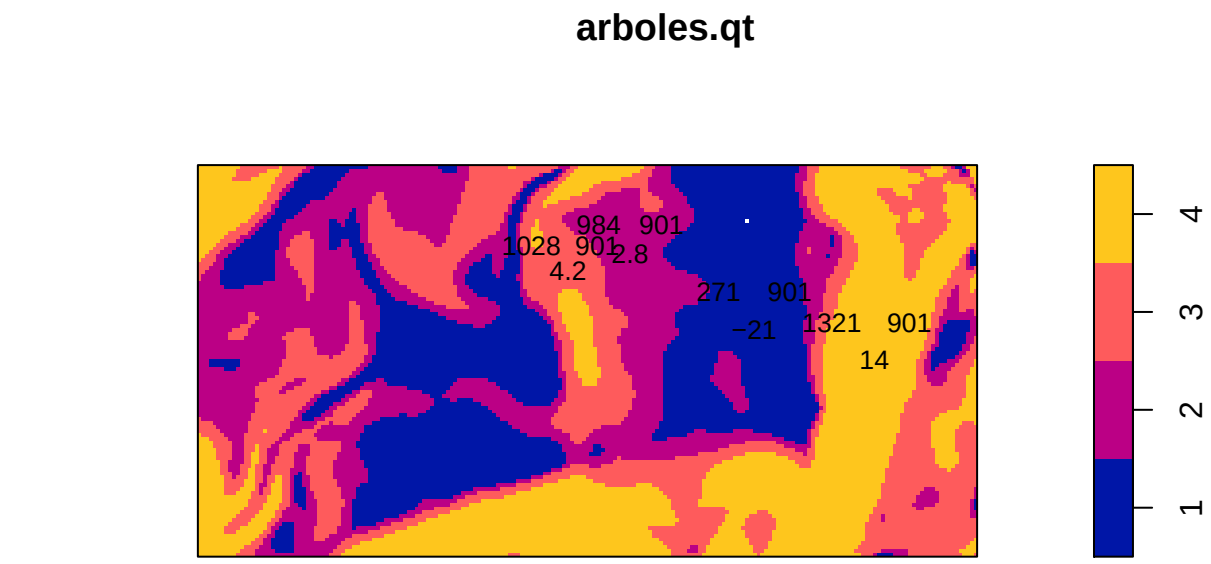
De esta manera, parece que la mayoría de los árboles se concentra en zonas con cierta pendiente (¿las zonas bajas ocupadas por tierras de labor?)

Finalmente, podemos evaluar si los árboles se agrupan preferentemente en alguna de las categorías de la pendiente, usando para ello la función `quadrat.test()`. De nuevo, el test se realiza contra la hipótesis nula de que la distribución debe ser igual en todas las categorías de la pendiente:

```
arboles.qt <- quadrat.test(arboles.ppp, tess = arboles.pendiente.tess)
arboles.qt
```

```
##
##  Chi-squared test of CSR using quadrat counts
##
## data:  arboles.ppp
## X2 = 661.84, df = 3, p-value < 2.2e-16
## alternative hypothesis: two.sided
##
## Quadrats: 4 tiles (levels of a pixel image)
```

```
plot(arboles.qt, cex = 0.8, valuesAreColours = FALSE)
```



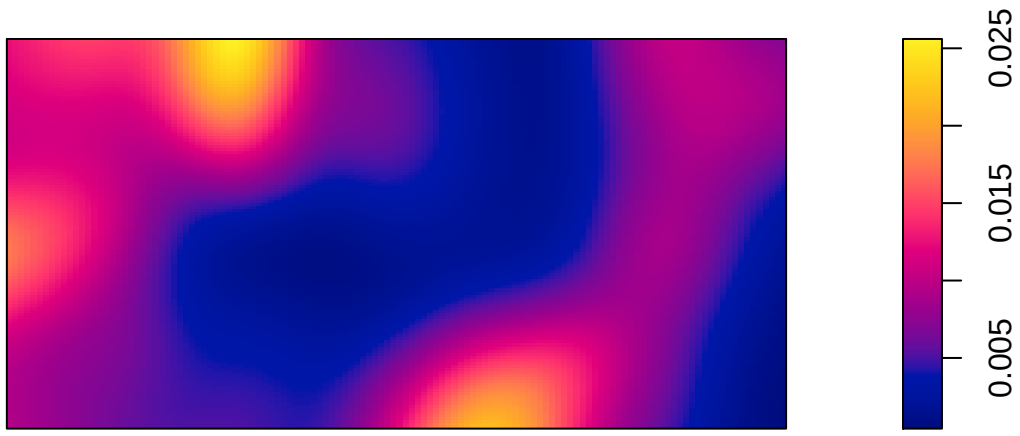
De nuevo, los bajos valores de p-value sugieren que es posible rechazar la hipótesis nula y afirmar que los árboles no se distribuyen de manera uniforme en cada una de las categorías en las que hemos dividido la pendiente del ámbito de trabajo.

4.5 Función de densidad Kernel

Otro método para estudiar las variaciones en la intensidad de un proceso espacial puntual es el uso de un método de densidad Kernel. Este método ajusta un "kernel" (o ventana) bidimensional a los objetos puntuales (los árboles), y los suma. Áreas con densidades elevadas de objetos mostrarán sumas más elevadas que las zonas con bajas densidades. Estas funciones de densidad proporcionan un resumen útil de las variaciones de intensidad de ese proceso puntual y una buena visualización para examinar si un conjunto de datos muestra una distribución aleatorio o no aleatoria.

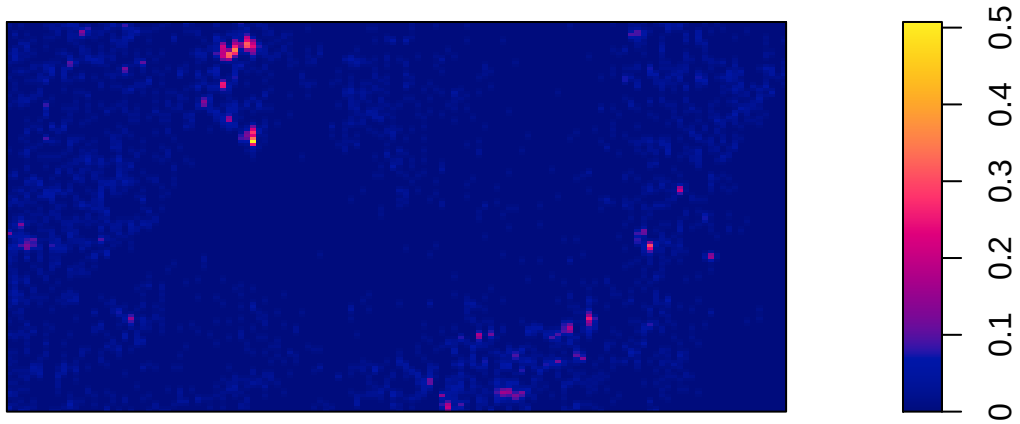
Las densidades son calculadas usando la función `density()` adaptada a un objeto `ppp`. El parámetro más importante de la función es denominado `sigma`, que controla el ancho de banda o tamaño de la ventana que se ajusta a cada punto. `sigma` es la desviación estándar del kernel gaussiano isotrópico: una desviación estándar pequeña da como resultado un gráfico de densidad menos suavizado, meintras que una desviación estándar grande da como resultado un gráfico de densidad más uniforme.

```
plot(density(arboles.ppp, sigma = 60))
```



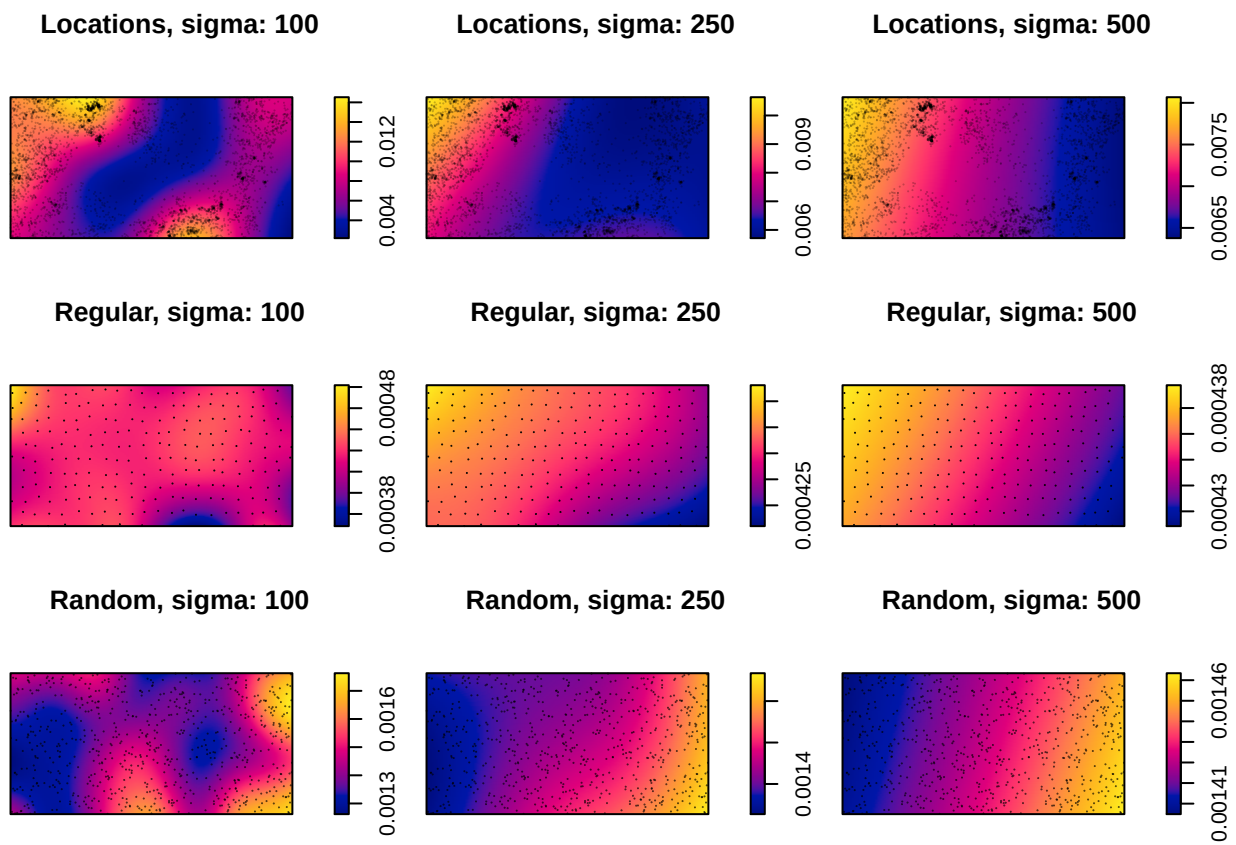
- ATENCIÓN: es conveniente repetir el procedimiento con diferentes valores de `sigma` para evaluar su efecto. En algunas ocasiones, es posible seleccionar ese valor usando validación cruzada. Este método puede llevarse a cabo siguiendo un proceso en dos pasos, en el que primero se selecciona el ancho de banda usando `bw.diggle`, y luego aplicándolo a la función de densidad. Este procedimiento proporciona estimaciones bastante "conservadoras" del ancho de banda:

```
arboles.bw <- bw.diggle(arboles.ppp)
plot(density(arboles.ppp, sigma = arboles.bw))
```



Otra posibilidad es dibujar simultáneamente diferentes valores de `sigma` (100,250,300) mediante este `loop´para aplicar la funcióndensity.ppp()‘` a los 3 conjuntos de datos

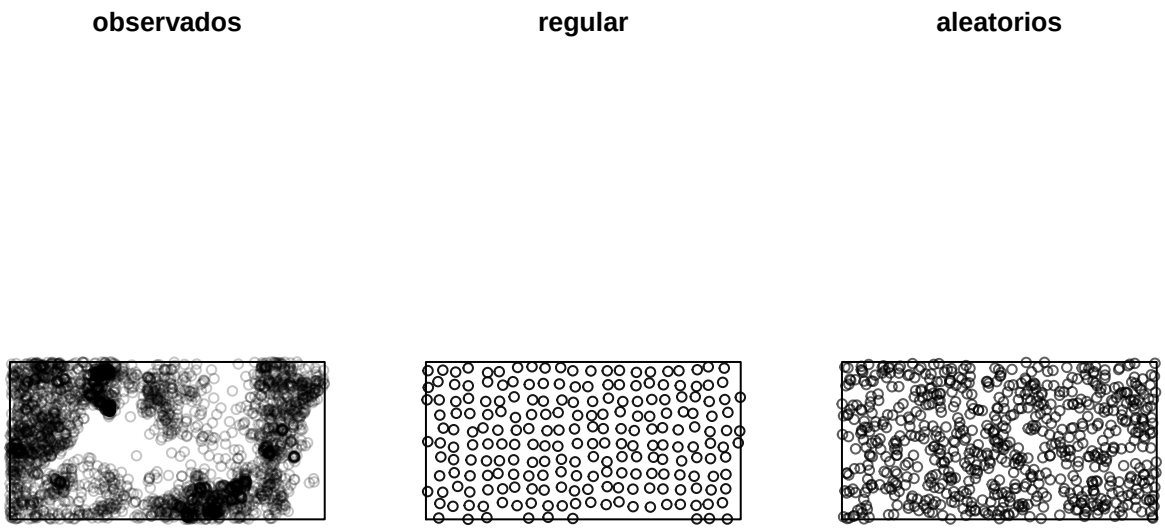
```
par(mfrow=c(3,3), mar=c(0,0,1,2))
sigma <- c(100, 250, 500)
data <- list(arboles.ppp, ppp.regular, ppp.random)
main <- c('Locations', 'Regular', 'Random')
for (i in 1:3){
  for (j in 1:3){
    ds <- density.ppp(data[[i]], sigma=sigma[j])
    plot(ds,
         main = paste0(main[i], ', sigma: ', sigma[j]))
    plot(data[[i]], add=T, cex=0.01, regular=F)
    }
}
```



5. INTERACCIONES ENTRE PUNTOS

La interacción entre dos puntos se mide a través de las propiedades de segundo orden, que reflejan cualquier tendencia de los puntos a agruparse (los puntos tienden a estar juntos), ser independientes (un proceso de Poisson) o estar espaciados regularmente (los puntos tienden a evitarse entre sí) .

```
par(mar=c(1,1,2,2), mfrow=c(1,3))
plot(arboles.ppp, main='observados')
plot(ppp.regular, main='regular')
plot(ppp.random, main='aleatorios')
```



5.1 FRY-PLOT

Una opción inicial de diagnóstico del grado de dependiencia entre puntos es denominado *Fryplot*. Es un diagrama de puntos en el que se representa el vector de diferencias entre todos los pares de puntos distintos. En el paquete **spatstat** este *Fryplot* es calculado mediante el comando `fryplot()`.

```
# par(mar=c(1,1,2,2), mfrow=c(1,3))
# fryplot(arboles.ppp, main='independant', pch=16, cex=0.2)
# fryplot(ppp.regular, main='regular', pch=16, cex=0.2)
# fryplot(ppp.random, main='clustered', pch=16, cex=0.2)
```

5.2 FUNCIONES DE DENSIDAD

Las funciones de distancia pueden usarse para investigar la interacción entre puntos en el espacio. Para ello existen varios métodos, basados todos ellos en la idea de calcular distancias entre unos puntos y otros, o entre puntos fijos en el área de estudio.

El método más conocido es la función K de Ripley, que describe la distribución como el conjunto de distancias entre todos los puntos.

Vamos a estudiar los resultados de esta función mediante los datos incluidos en el fichero *secuoyas.shp*. Para ello, primero importaremos este fichero en R, y luego lo convertiremos en un objeto `ppp`. Como el formato inicial de los datos es un shapefile (.shp), es necesario acudir a la función `st_read()` del paquete **sf**. Este paquete incluye la función `as.ppp()` para convertir los shapefiles en un objeto `ppp`.

```
secuoyas.sf <- st_read("D:/Docencia_Master_2021/M1683/sesion03/secuoyas.shp",
                       quiet = TRUE)
secuoyas.sf
```

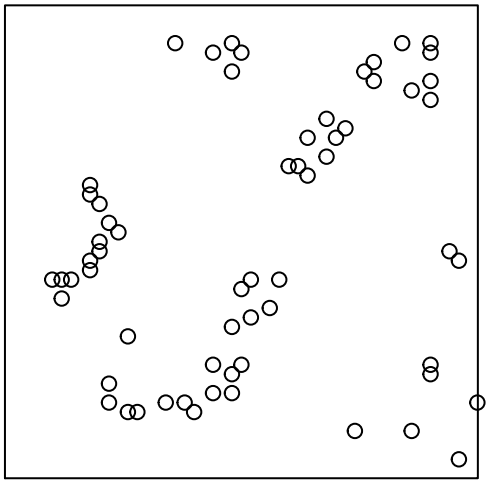
```
## Simple feature collection with 62 features and 4 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 0.1 ymin: 0.04 xmax: 0.999 ymax: 0.92
## CRS:           NA
## First 10 features:
##       x    y   mx   my          geometry
## 1  0.36 0.92 0.36 0.92 POINT (0.36 0.92)
## 2  0.44 0.90 0.44 0.90  POINT (0.44 0.9)
## 3  0.48 0.92 0.48 0.92 POINT (0.48 0.92)
## 4  0.48 0.86 0.48 0.86 POINT (0.48 0.86)
## 5  0.50 0.90 0.50 0.90   POINT (0.5 0.9)
## 6  0.76 0.86 0.76 0.86 POINT (0.76 0.86)
## 7  0.78 0.88 0.78 0.88 POINT (0.78 0.88)
## 8  0.78 0.84 0.78 0.84 POINT (0.78 0.84)
## 9  0.84 0.92 0.84 0.92 POINT (0.84 0.92)
## 10 0.86 0.82 0.86 0.82 POINT (0.86 0.82)
```

```
secuoyas.ppp <- as.ppp(secuoyas.sf)
secuoyas.ppp
```

```
## Marked planar point pattern: 62 points
## marks are numeric, of storage type  'double'
## window: rectangle = [0.1, 0.999] x [0.04, 0.92] units
```

Por defecto, la función `as.ppp` crea un objeto `ppp`en el que, por defecto, usa la primera columna del dataframe `sf` como *mark*, es decir, una etiqueta de cada punto. Aunque volveremos a ello más tarde, inicialmente ignoraremos esto estableciendo un valor NULL. El otro aspecto que debemos corregir es el tamaño de la ventana, estableciendo 0 y 1 como límites máximo y mínimo tanto en el eje x como en el eje y.

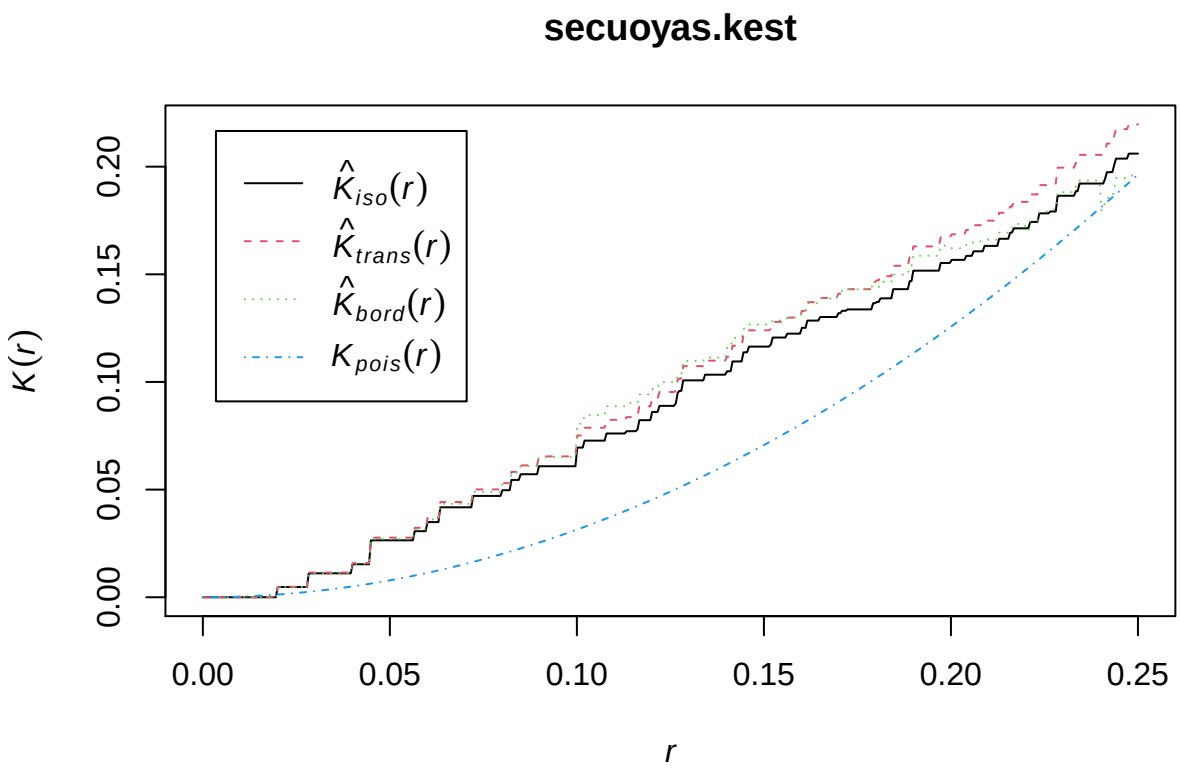
```
marks(secuoyas.ppp) <- NULL
Window(secuoyas.ppp) <- owin(x = c(0, 1), y = c(0, 1))
plot(secuoyas.ppp)
```



5.2.1 K de Ripley

El estadístico K de Ripley se calcula usando la función `Kest()`; existen funciones similares para los estadísticos F y G. Una vez calculado el estadístico, se puede representar los resultados, incluyendo los valores observados de K y una curva teórica basada en un proceso de Poisson homogéneo con una intensidad igual a nuestro proceso de interés.

```
secuoyas.kest <- Kest(secuoyas.ppp)
plot(secuoyas.kest)
```



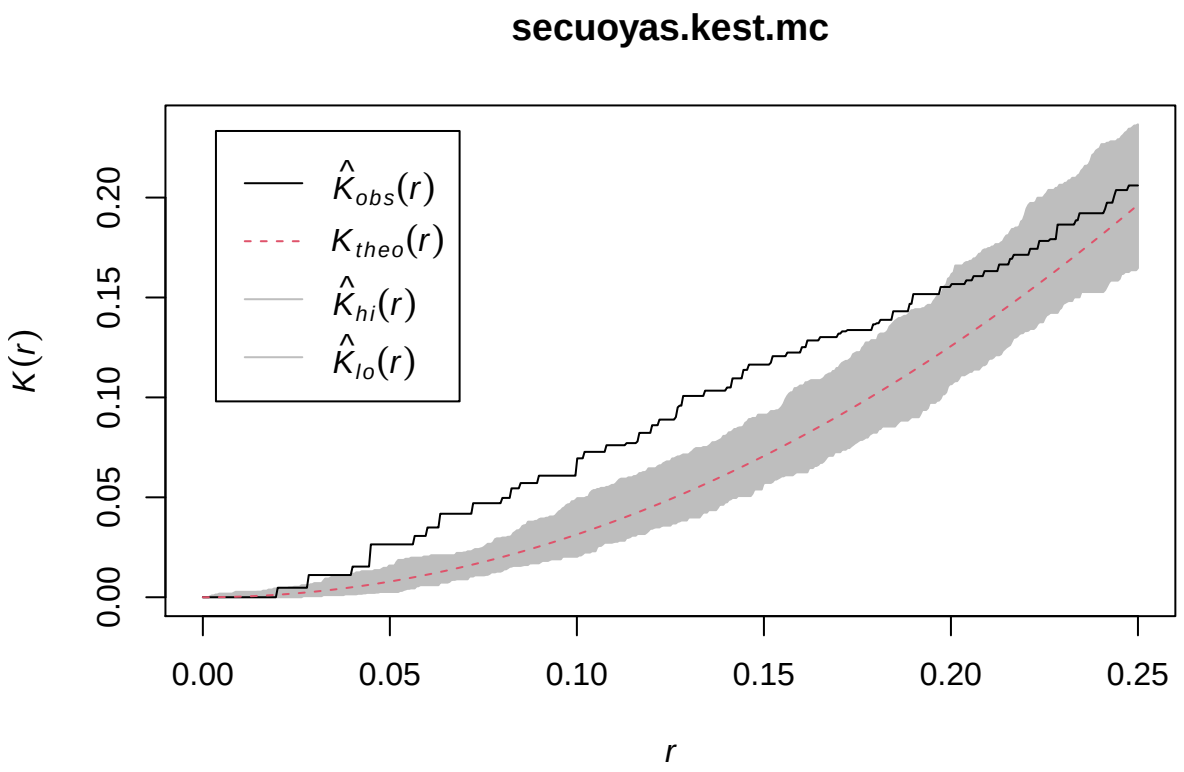
Si el proceso espacial que estamos estudiando (la distribución de Secuoyas) es puramente aleatoria (es decir, sigue una distribución de Poisson), la línea que representa los valores observados (línea negra) debería aproximarse a la línea teórica (en color azul).

Si la curva basada en las observaciones se sitúa por encima de la curva teórica, estaríamos ante un distribución espacial organizada en `clusters`o agrupaciones de objetos; si se encuentra por debajo indica una distribución regular u ordenada. En el gráfico aparecen otras dos líneas, una verde y otra roja, que representan valores de K calculados con diferentes correcciones para atenuar el efecto de borde o límite. De acuerdo con el gráfico, la distribución espacial de secuoyas tiende a formar agrupamientos.

Para comprobar si estos agrupamientos difieren significativamente de un distribución aleatoria, se suele utilizar una serie de simulaciones aleatorias (Montecarlo) de procesos de Poisson homogéneos, usando la función `envelope()`. Esta función nos proporciona un conjunto de posibles valores de la K de Ripley, que dan cuenta de simples diferencias estocásticas en distribuciones aleatorias. Si los datos están realmente agrupados, la K de Ripley observada debería situarse fuera de ese conjunto de valores simulardos. La función `envelope()` requiere:

1. Un objeto `ppp`
2. La función a comprobar (en este caso la K de Ripley; `Kest`)
3. El número de simulaciones aleatorias que requerimos (en este caso 99)

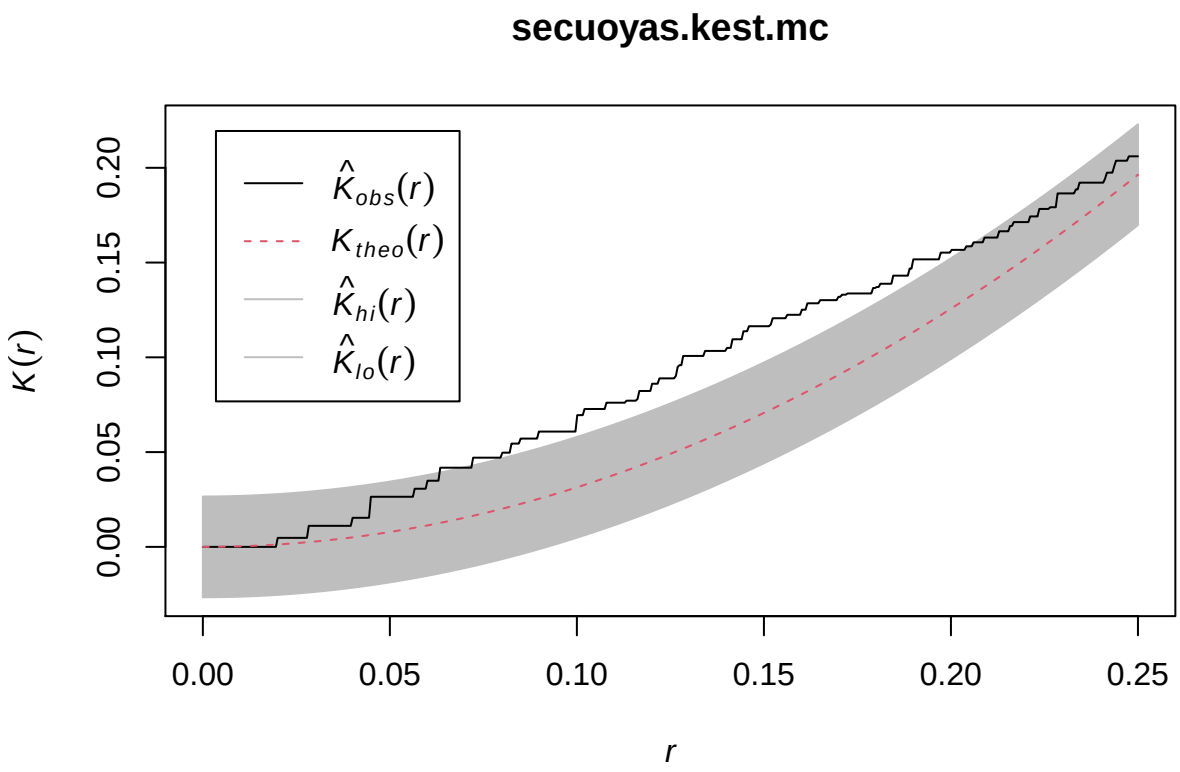
```
secuoyas.kest.mc <- envelope(secuoyas.ppp,
                             fun = 'Kest',
                             nsim = 99,
                             verbose = FALSE)
plot(secuoyas.kest.mc,
     shade = c("hi", "lo"))
```



Note that this uses point-wise estimates of uncertainty, and cannot be used as a post-hoc test (las prueba de rango post hoc identifica subconjuntos homogéneos de medias que no se diferencian entre sí.).

Una aproximación mejor consiste en calcular la incertidumbre global, entendida como la mayor desviación entre los valores de K generados aleatoriamente y el valor teórico.

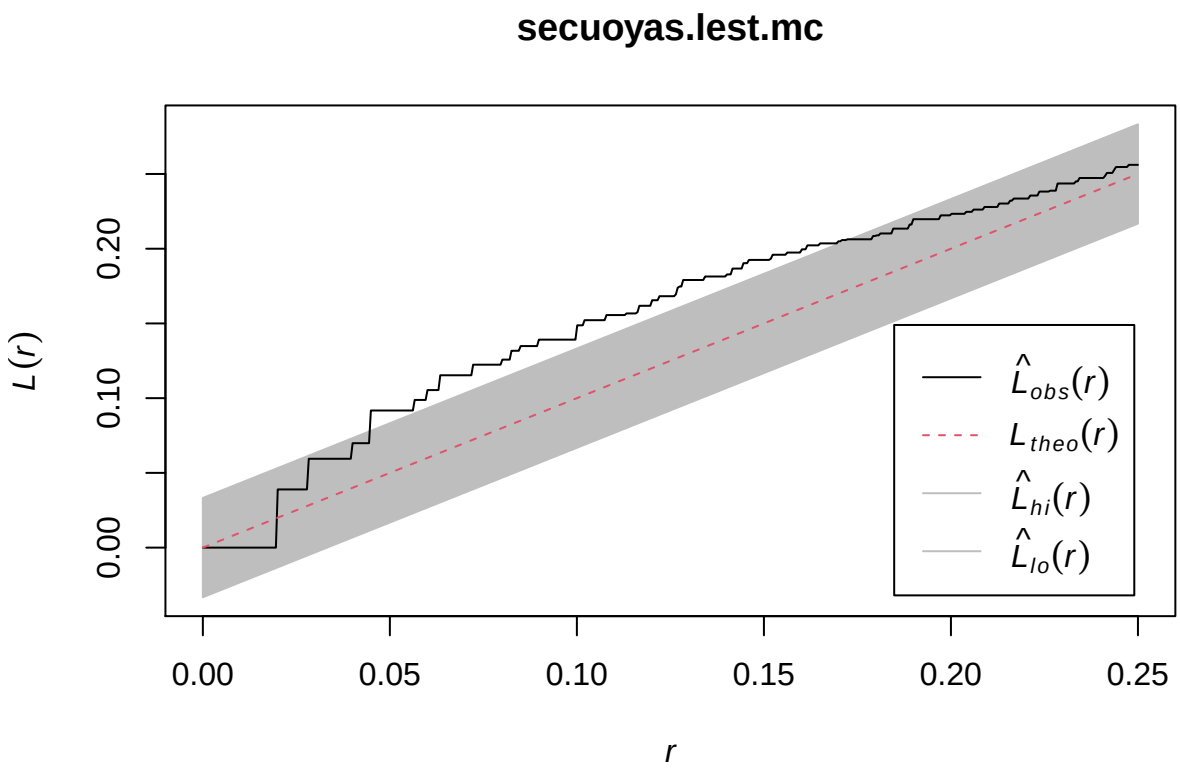
```
secuoyas.kest.mc <- envelope(secuoyas.ppp,
                             fun = 'Kest',
                             nsim = 99,
                             verbose = FALSE,
                             global = TRUE)
plot(secuoyas.kest.mc, shade = c("hi", "lo"))
```



5.2.2 La L de Besag

La función L fue propuesta por Besag como una manera de estibilizar la varianza en el estadístico K de Ripley y mejorar su interpretabilidad. Este estadístico puede calcularse incorporando el argumento `Lesten` la función `envelope()`:

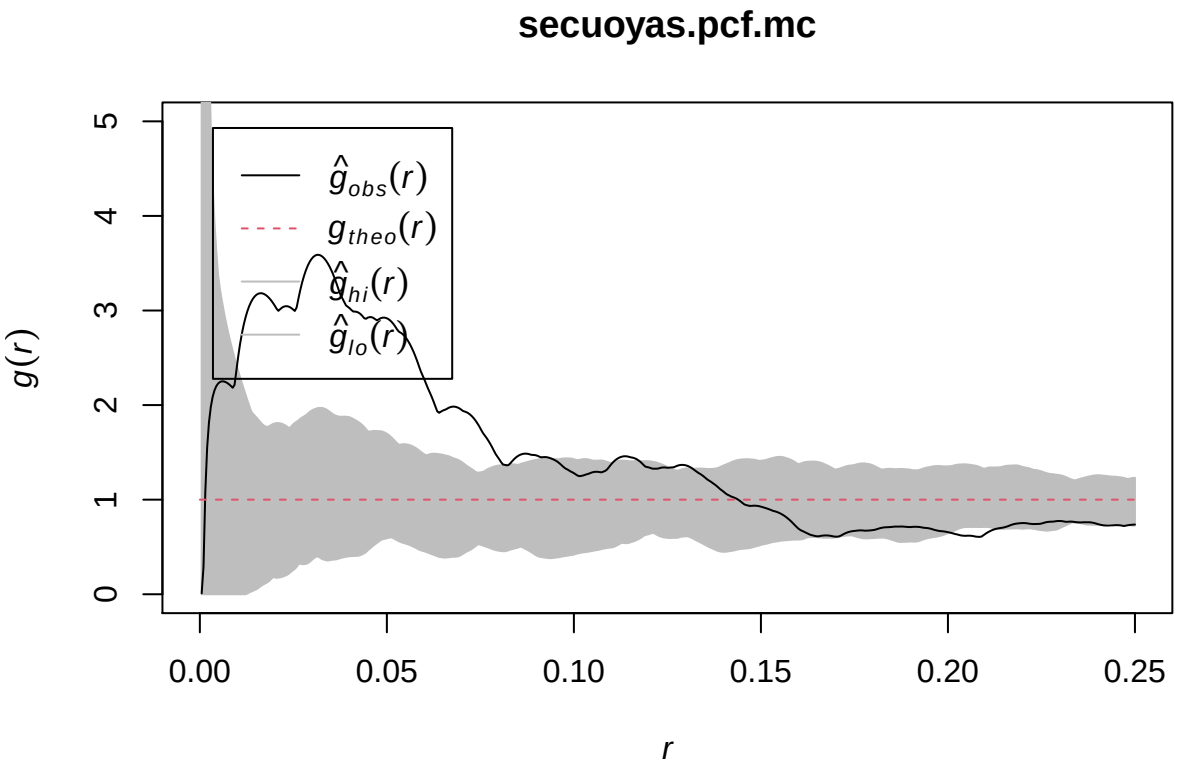
```
secuoyas.lest.mc <- envelope(secuoyas.ppp,
                             fun = 'Lest',
                             nsim = 99,
                             verbose = FALSE,
                             global = TRUE)
plot(secuoyas.lest.mc, shade = c("hi", "lo"))
```



5.2.3 Función de correlación por pares

La función de correlación por pares es la última que revisaremos en este capítulo. En vez de usar pares de distancias acumulados, esta función está basada en el número de pares de puntos separados por una banda de distancia. Su ventaja es que proporciona una idea más clara del rango de interacciones - como la K de Ripley está basada en un conjunto de distancias acumuladas. Se implementa también mediante la función `envelope()` pero eliminando la opción `global` y reemplazándola por FALSE.

```
secuoyas.pcf.mc <- envelope(secuoyas.ppp,
                            fun = 'pcf',
                            nsim = 99,
                            verbose = FALSE)
plot(secuoyas.pcf.mc, shade=c("hi", "lo"), ylim = c(0,5))
```




```
## Error in rebound.owin(X[[i]], ...) :
##   The new rectangle 'rect' does not contain the window 'win'
```

El gráfico muestra interacciones positivas hasta un rango de alrededor de 0.07 unidades de distancia (dependen de las unidades del mapa), mucho menor que en la función K.

6. PROCESOS PUNTUALES CON MARCAS

En las secciones precedentes, los procesos puntuales han sido considerados como objetos singulares. Los datos que corresponden a un proceso puntual *con marcas* incluyen información adicional que distingue los objetos en diferentes clases, permitiendo el estudio de su co-ocurrencia (tanto positiva como negativa) entre esas diferentes categorías de objetos. Los datos incluidos en el fichero `bosque` contienen la localización de un conjunto de árboles en un arboles, pero incluyendo como variable adicional la especie.

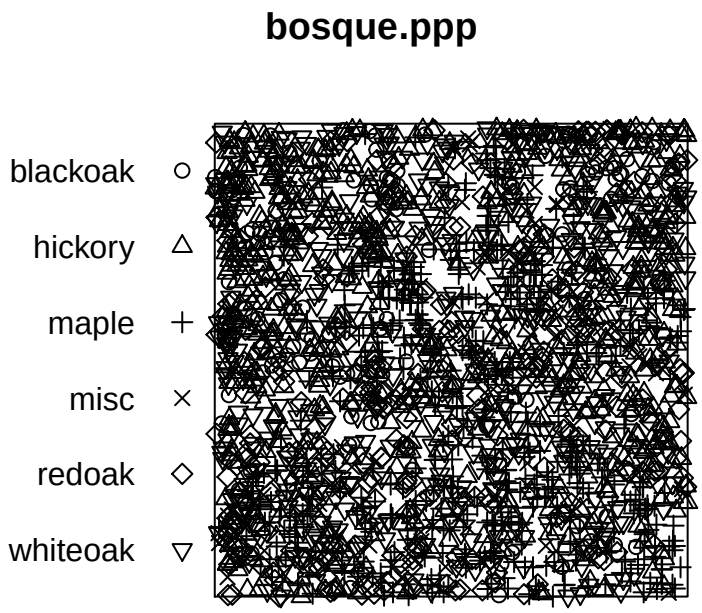
```
bosque <- read.csv("D:/Docencia_Master_2021/M1683/sesion03/bosque.csv")
str(bosque)
```

```
## 'data.frame':    2250 obs. of  3 variables:
##  $ x      : num  0.078 0.076 0.051 0.015 0.03 0.102 0.135 0.121 0.04 0.065 ...
##  $ y      : num  0.091 0.266 0.225 0.366 0.426 0.474 0.498 0.489 0.596 0.608 ...
##  $ species: chr  "blackoak" "blackoak" "blackoak" "blackoak" ...
```

A continuación se creará un objeto `ppp` usando la variable `species` para definir las *marks* o etiquetas de cada punto. Como novedades respecto a anteriores ejercicios se encuentran:

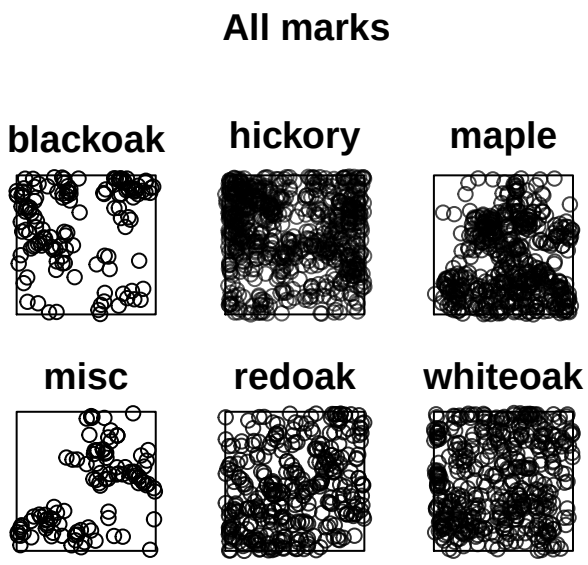
1. La ventana describiendo el área de estudio es un fichero shape (*bosque.shp*) que debe ser importado y transformado en un objeto `owin`;
2. Además, es necesario especificar la información categórica (los nombres de las especies) al crear el objeto `ppp` usando el argumento `marks`. Para ello, es necesario convertir esta columna en un `factor` para que R lo reconozca como una variable categórica (etiqueta).

```
bosque$species <- as.factor(bosque$species)
bosque.win <- st_read("D:/Docencia_Master_2021/M1683/sesion03/bosque.shp", quiet = TRUE)
bosque.ppp <- ppp(x = bosque$x,
                  y = bosque$y,
                  win = bosque.win,
                  marks = bosque$species)
plot(bosque.ppp)
```



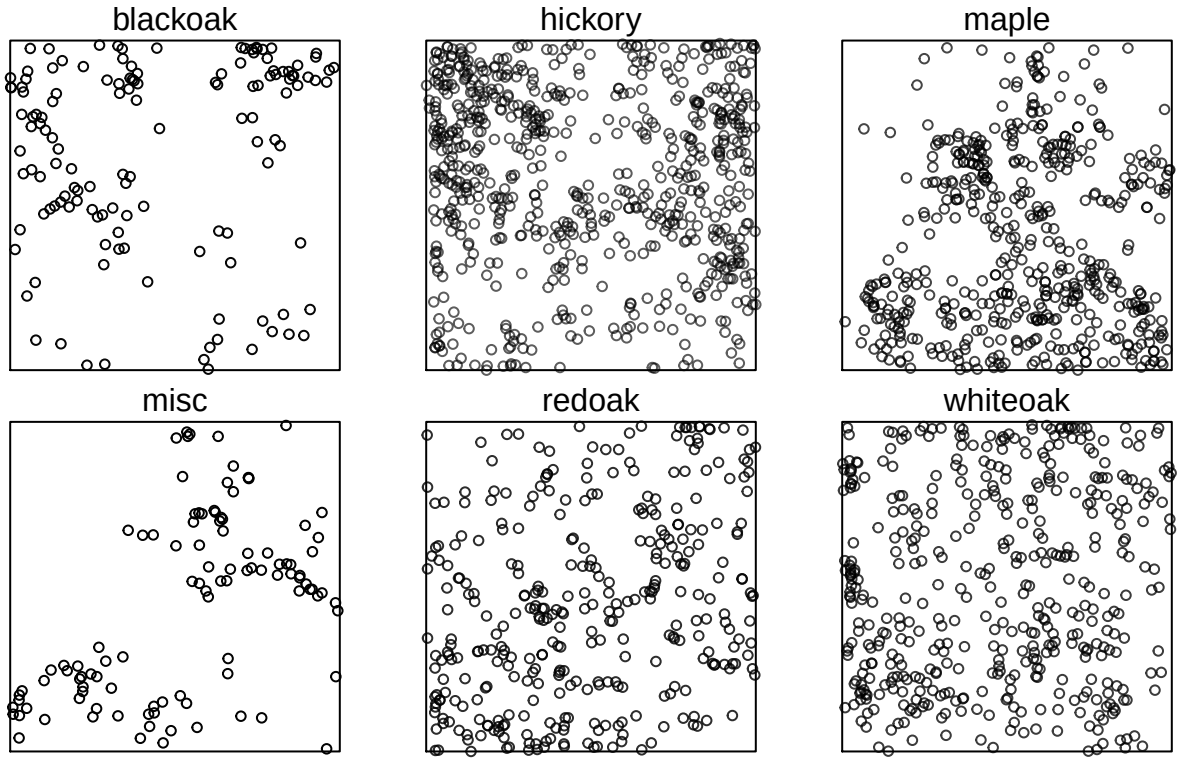
En los gráficos anteriores hemos dibujado todas las *marcas* (especies de árboles). Podemos aislar cada una de las marcas usando la función `split()`, para analizar posteriormente la distribución de cada especie por separado:

```
plot(split(bosque.ppp),
     main = "All marks")
```

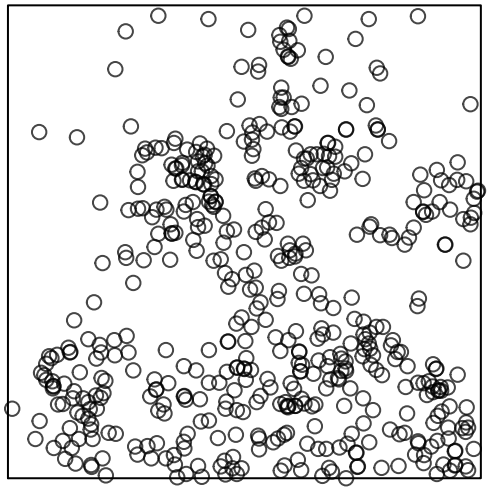


En las siguientes líneas se crea y aplica una función para dibujar por separado cada una de las especies

```
bob <- split(bosque.ppp)
mrk_names <- names(bob)
bob <- unclass(bob)
par(mfrow = c(2, 3),
    mar = c(0, 0, 1, 0),
    oma = c(1, 0, 3, 0))

for(i in seq_along(mrk_names)){
  mrk_name <- mrk_names[[i]]
  plot(bob[[i]], main = "")
  mtext(text = mrk_name,
        side = 3,
        line = -0.3,
        adj = 0.5)
}

mtext("All Marks", cex = 1.75, side = 3, line = 1, outer = TRUE)
```

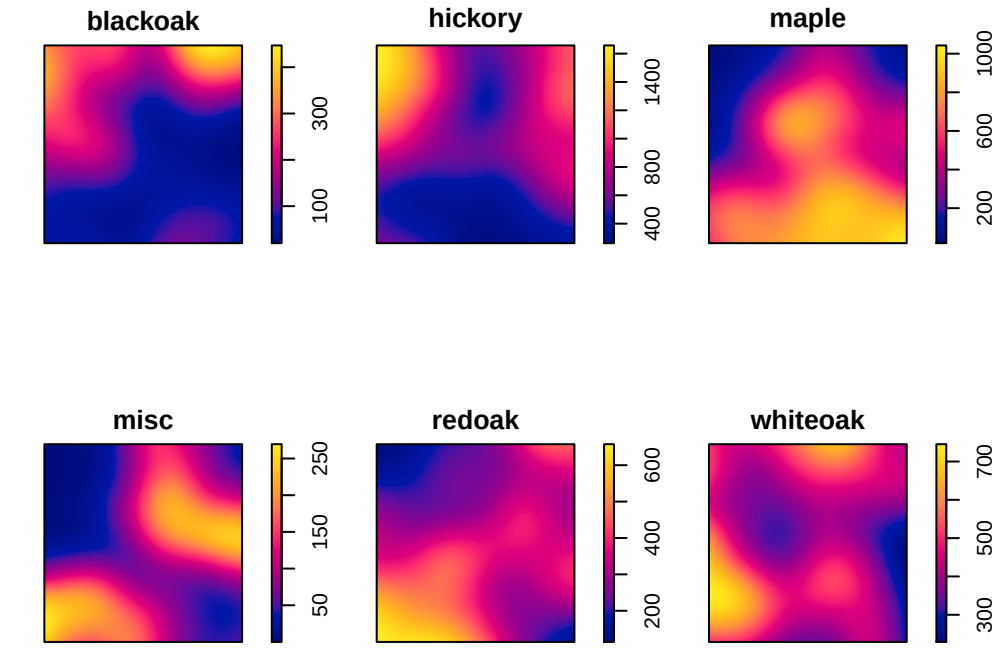



```
plot(split(bosque.ppp)$maple, "Maple trees")
```

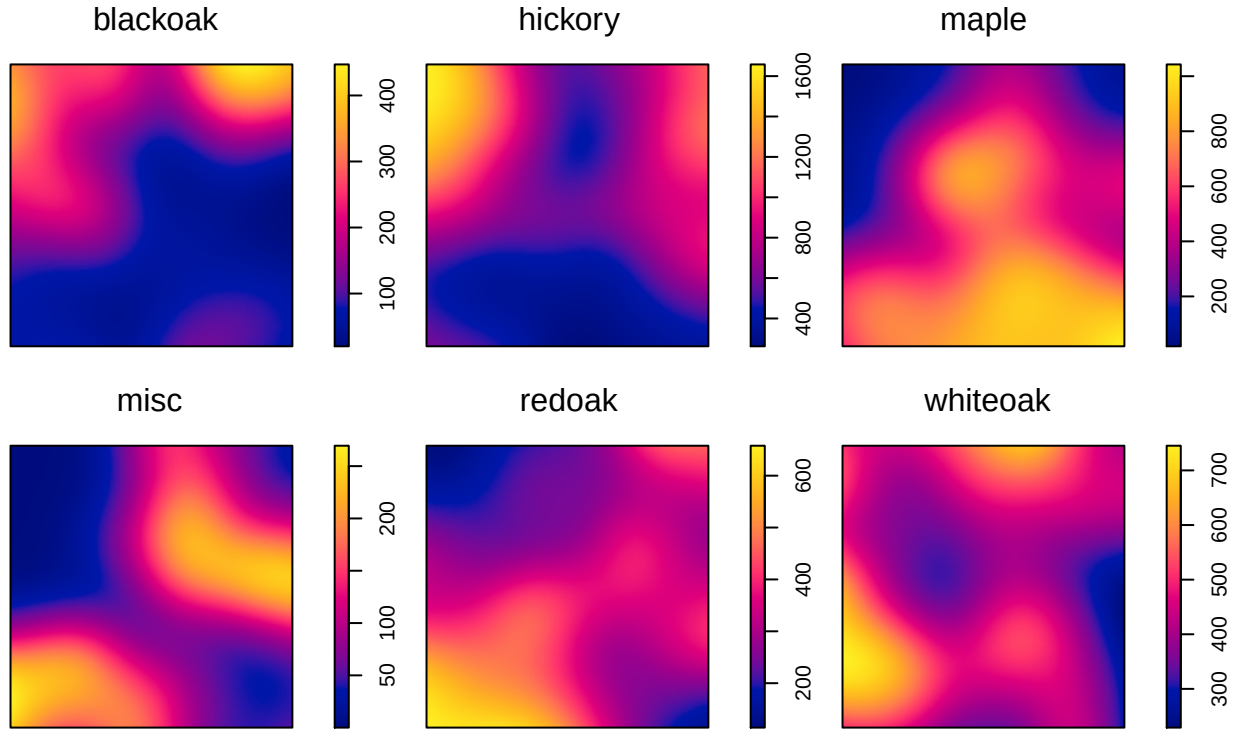


De manera visual podemos comprobar que las diferentes especies tienen distribuciones espacial bastante contrastadas, aunque se puede utilizar la función `density()` para clarificar esta característica:

```
plot(density(split(bosque.ppp)),
     main = "Distribución espacial de las especies del bosque")
```




```
par(mfrow = c(2, 3),
    mar = c(0, 0, 1, 2),
    oma = c(1, 0, 3, 0))

for(i in seq_along(mrk_names)){
  mrk_name <- mrk_names[[i]]
  dns <- density(bob[[i]])
  plot(dns, main = "")
  mtext(text = mrk_name,
        side = 3,
        line = -0.3,
        adj = 0.4)
}
mtext("Distribución espacial de las especies del bosque", cex = 1.75, side = 3, line = 1, outer = TRUE)
```

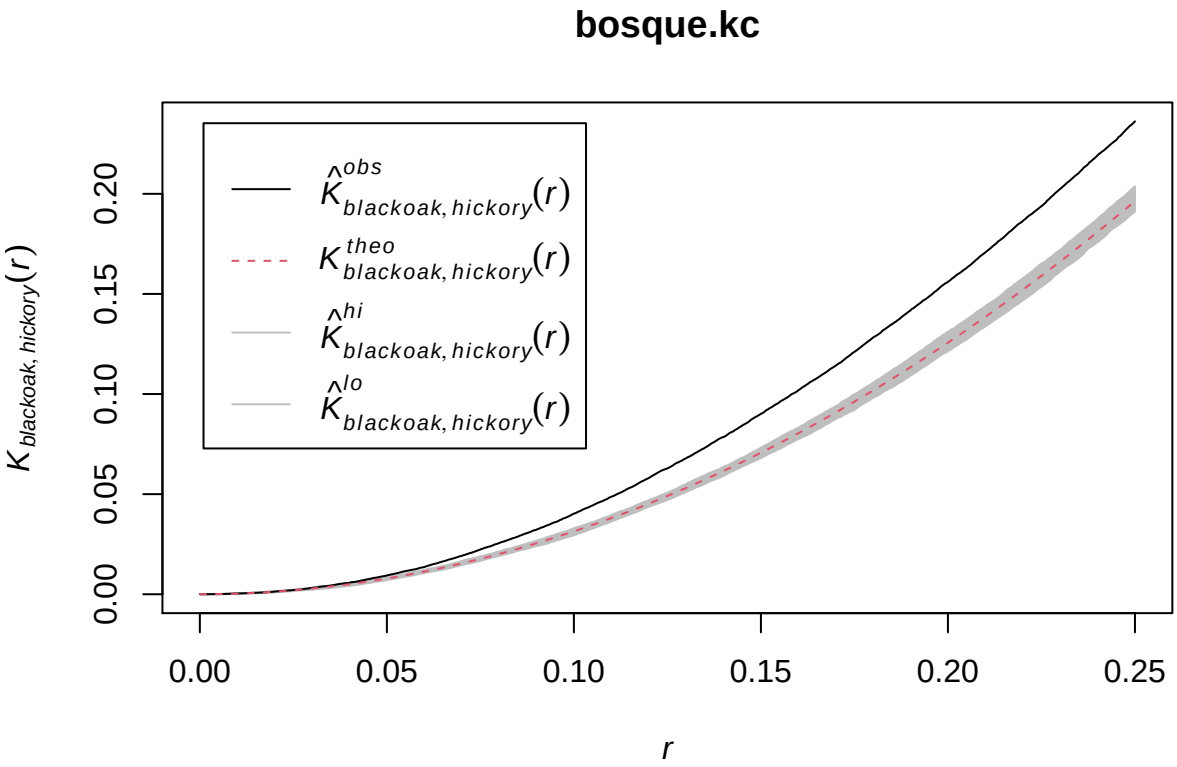


Para examinar la ocurrencia simultánea de dos especies, se puede acudir inicialmente a la función K de Ripley. En este caso, se incluye el argumento `Kcross()`, que examina las diferencias por parejas entre los objetos que forman ambas categorías. Además, se usa la función `envelope()` para simular distribuciones aleatorias, especificando las dos especies que los argumentos i y j:

```
bosque.kc <- envelope(bosque.ppp,
                      Kcross,
                      i = "blackoak",
                      j = "hickory",
                      nsim = 99,
                      verbose = FALSE)
```

Como en casos anteriores, es conveniente representar las curvas observadas y la envolvente que muestra las distribuciones aleatorias simuladas. Las posibilidades son: 1. Si la curva observada se sitúa por encima de la envolvente, ambas especies tienen a coincidir en el espacio; 2. Si la curva se sitúa por debajo, las especies aparecen en diferentes áreas, sugiriendo algún tipo de interacción competitiva. 3. Si la curva se sitúan dentro de la envolvente, la distribución combinada de ambas especies es aleatoria.

```
plot(bosque.kc)
```



Modelos de procesos puntuales

Es posible "ajustar" modelos de procesos puntuales a cualquier objeto `ppp` usando la función `ppm()`. Esta función usa el conjunto de puntos observados para modelizar las variaciones en la intensidad de ese proceso puntual, basándose en un cojunto de covariables. El primer argumento de la función es un objeto `ppp`, mientras que el segundo argumento identifica las covariables. Este segundo argumento utiliza la misma sintaxis que sirve para el cálculo de un modelo lineal en R. Comenzaremos construyendo um modelo sencillo correspondiente a un proceso de Posson homogéneo (sin covariables):

```
fit0 <- ppm(arboles.ppp ~ 1)
fit0
```

```
## Stationary Poisson process
## Intensity: 0.007208
##              Estimate       S.E.    CI95.lo    CI95.hi Ztest      Zval
## log(lambda) -4.932564 0.01665742 -4.965212 -4.899916   *** -296.1182
```

Este modelo devuelve un único parámetro, la intensidad media para la región de análisis. Como aparece en unidades logarítmicas, se puede "traducir". El resultado nos informa que, cada metro cuadrado está ocupado por un 0.007 de un árbol

```
exp(coef(fit0))
```

```
## log(lambda)
##    0.007208
```

Para comprobar este número, se puede calcular directamente su intensidad a partir del objeto `arboles.ppp`:

```
summary(arboles.ppp)
```

```
## Planar point pattern:  3604 points
## Average intensity 0.007208 points per square unit
##
## Coordinates are given to 1 decimal place
## i.e. rounded to the nearest multiple of 0.1 units
##
## Window: rectangle = [0, 1000] x [0, 500] units
## Window area = 5e+05 square units
```

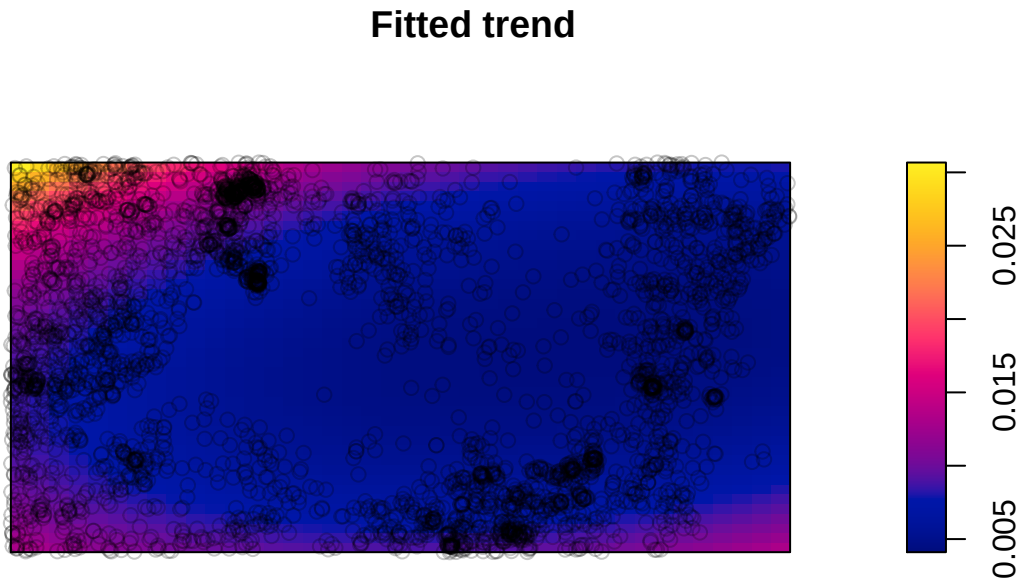
Ahora vamos a modelizar la intensidad como una función polinómica de segundo orden calculada a partir de las coordenadas x e y de los objetos. La función `polynom()` expande un conjunto de variables en su forma de segundo (o n^{th}) orden (por ejemplo $x + y + x^2 + y^2 + x * y$ para las coordenadas de segundo orden).

```
fit1 <- ppm(arboles.ppp ~ polynom(x, y, 2))
fit1
```

```
## 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)
## -4.275762e+00 -1.609187e-03 -4.895166e-03  1.625968e-06 -2.836387e-06
##        I(y^2)
##  1.331331e-05
##
##                  Estimate         S.E.       CI95.lo       CI95.hi Ztest
## (Intercept) -4.275762e+00 7.811138e-02 -4.428857e+00 -4.122666e+00   ***
## x           -1.609187e-03 2.440907e-04 -2.087596e-03 -1.130778e-03   ***
## y           -4.895166e-03 4.838993e-04 -5.843591e-03 -3.946741e-03   ***
## I(x^2)       1.625968e-06 2.197200e-07  1.195325e-06  2.056611e-06   ***
## I(x * y)    -2.836387e-06 3.511163e-07 -3.524562e-06 -2.148212e-06   ***
## I(y^2)       1.331331e-05 8.487506e-07  1.164979e-05  1.497683e-05   ***
##                   Zval
## (Intercept) -54.739290
## x            -6.592577
## y           -10.116084
## I(x^2)        7.400185
## I(x * y)     -8.078197
## I(y^2)       15.685769
```

A partir de los resultados anteriores se puede representar la tendencia ajustada de la superficie:

```
plot(fit1,
     how = 'image',
     se = FALSE,
     pause = FALSE)
```



Anteriormente, se planteó la posibilidad de una relación entre la ubicación de los árboles y el valor de la pendiente. Por lo tanto, podemos usar la función `ppm()` para modelizar la intensidad de la distribución usando como covariable la pendiente. Además de alrededor de los puntos en los que se sitúan los objetos, es conveniente disponer de valores de la covariable más lejos. En nuestro caso, utlizaremos el fichero image (`arboles.pendiente`).

```
fit2 <- ppm(arboles.ppp ~ arboles.pendiente)
fit2
```

```
## Nonstationary Poisson process
##
## Log intensity:  ~arboles.pendiente
##
## Fitted trend coefficients:
##       (Intercept) arboles.pendiente
##         -5.391053          5.026710
##
##                     Estimate       S.E.   CI95.lo   CI95.hi Ztest      Zval
## (Intercept)       -5.391053 0.03001787 -5.449887 -5.332219   *** -179.5948
## arboles.pendiente  5.026710 0.24534296  4.545847  5.507573   ***   20.4885
```

El coeficiente de la pendiente es 5, pero hay que tener en cuenta que en modelos de regresión basados en el logarítmo de la variable respuesta ese valor es un multiplicador, y refleja el incremento de la intensidad de la variable dependiente según cada incremento en la variable independiente, en este caso la pendiente topográfica. A continuación, podemos representar la tendencia de la superficie ajustada

```
plot(fit2,
     how = 'image',
     se = FALSE,
     pause = FALSE)
```