# Análisis de presencias con procesos de puntos

## Tutorial básico de spatstat

Gerardo Martín

2022-06-29

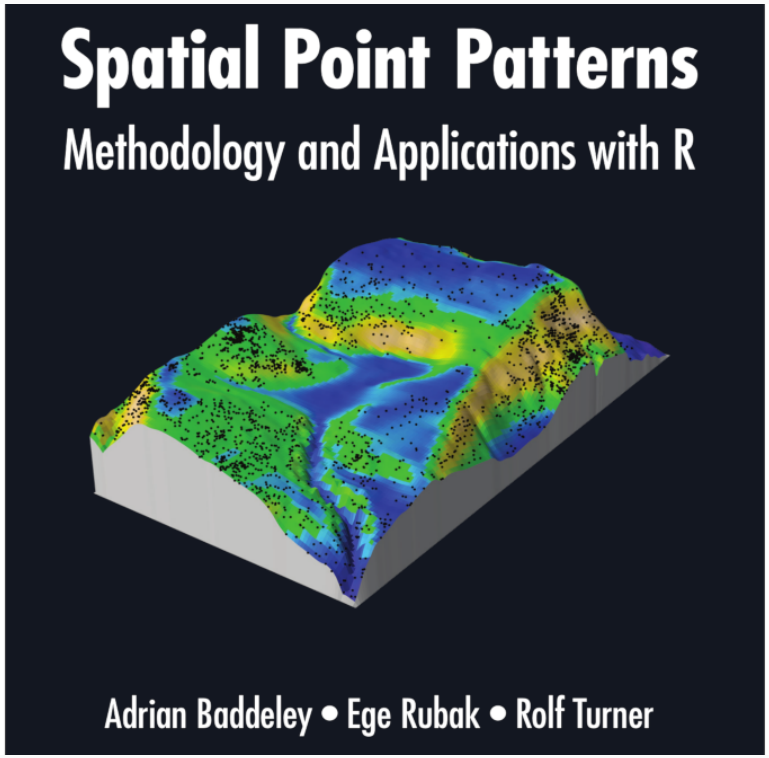
# Introducción

# ¿Qué es `spatstat`?

- `spat` = Spatial
- `stat` = Statistics
- Paquete de **R** para hacer estadística espacial de procesos de puntos
    - Sintaxis típica de R básico
    - Objetos específicos del paquete

# Libro

Por desarrolladores del paquete



# ¿Qué veremos aquí?

1. Instalación y carga
2. Formateo
3. Transformación de otros paquetes a `spatstat`
4. Análisis exploratorio

   4.1. Detección de autocorrelación 4.2. Análisis de respuesta a covariables 4.3. Propuesta de modelos en relación a covariables
5. Ajuste de modelo

# Instalación y carga

# Igual que cualquier paquete de R

Instalación

```
install.packages("spatstat")
```

Carga

```
library(spatstat)
```

# Peeero

Necesitamos también (para leer raster de covariables):

```
library(rgdal)
library(raster)
library(foreach)
```

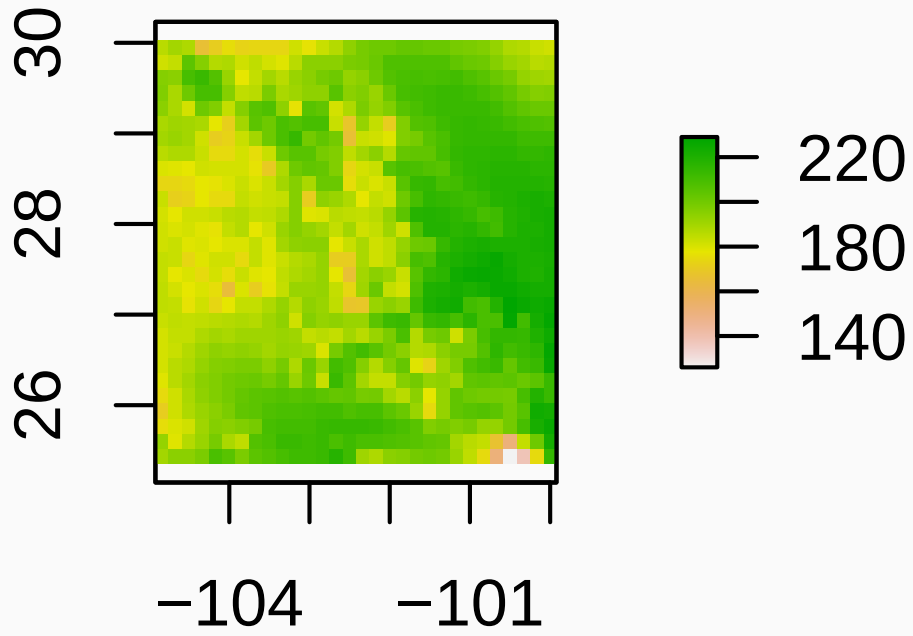
# Formateo

# Liga para los datos

Variables ambientales

# Carga de rasters

```
archivos <- list.files("Datos-ejemplos/", "tif",
                       full.names = T,
                       recursive = F)
r <- stack(archivos)

plot(r[[1]])
```



# Transformar de `raster` a `im`

Si hacemos:

```
class(r)
```

```
## [1] "RasterStack"
## attr(,"package")
## [1] "raster"
```

vemos que el tipo de objeto es `raster`, pero `spastat` utiliza `im`

Transformación en lote con `imFromStack`

# Transformar de `raster` a `im`

Para cargar `imFromStack`:

```
source("Funciones-spatstat/imFromStack.R")
```

Y se usa:

```
r.im <- imFromStack(r)
class(r.im)
```

```
## [1] "list"
```

# Ventana de trabajo

Cálculo de intensidad se hace con la “ventana”. Utilizaremos la función `winFromRaster`:

```
source("Funciones-spatstat/winFromRaster.R")
w <- winFromRaster(r)
```

Para verificar la clase del objeto:

```
class(w)
```

```
## [1] "owin"
```

# Puntos de ocurrencia

Vamos a simular las coordenadas $x$ y $y$ de un proceso de puntos:

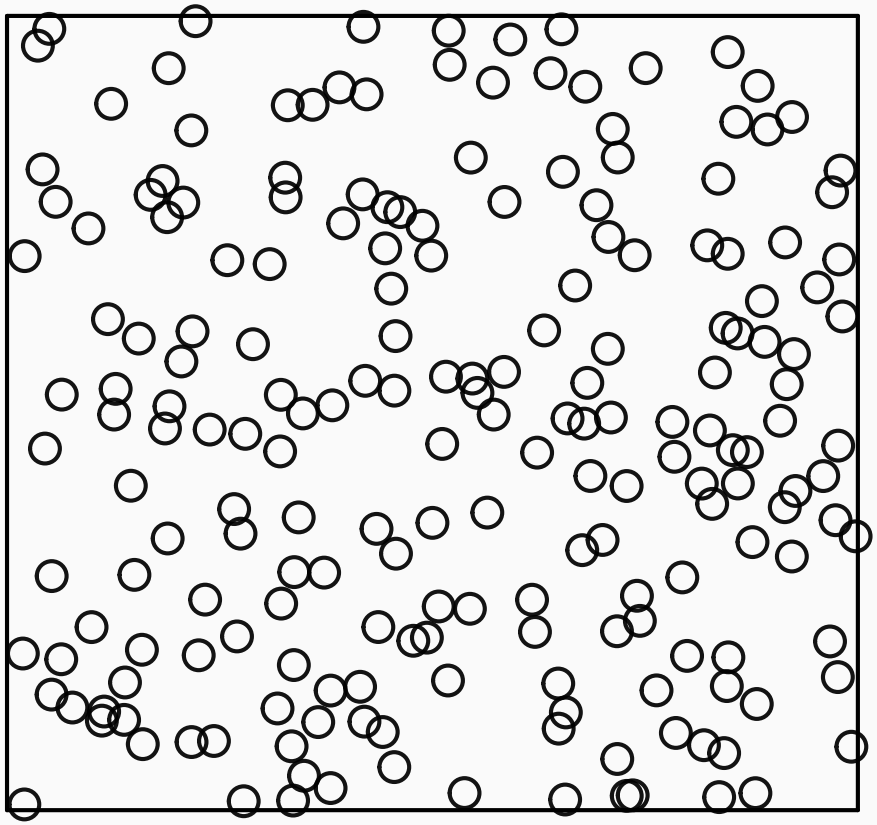
```
set.seed(984573)
puntos <- data.frame(coordinates(r)[sample(1:840, 200),])
puntos$x <- puntos$x + rnorm(200, 0, 0.05)
puntos$y <- puntos$y + rnorm(200, 0, 0.05)
```

El formato que requiere `spatstat` es `ppp`:

```
puntos.ppp <- ppp(x = puntos$x,
                  y = puntos$y,
                  window = w,
                  check = F)
class(puntos.ppp)
```

```
## [1] "ppp"
```

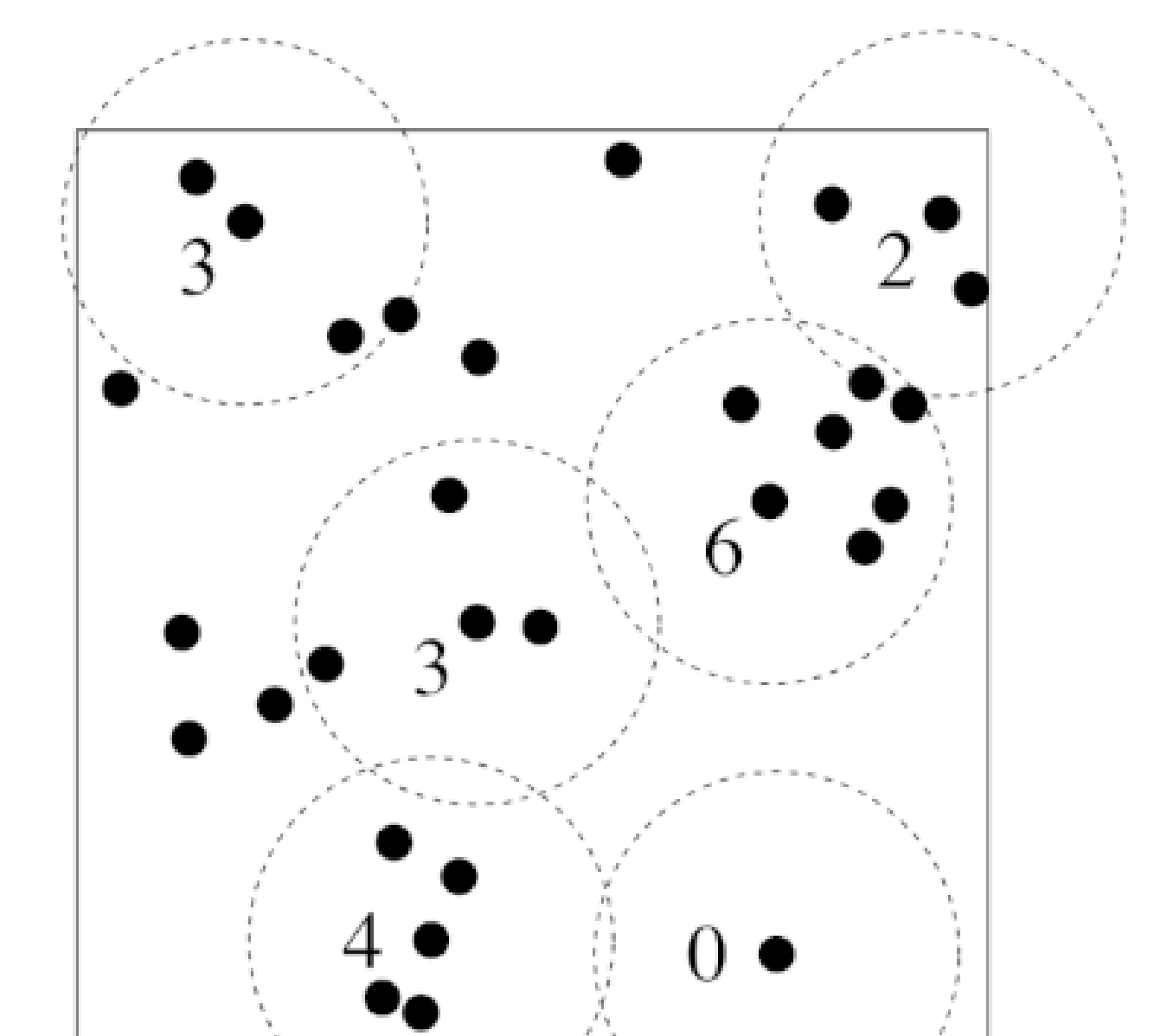
```
plot(puntos.ppp)
```



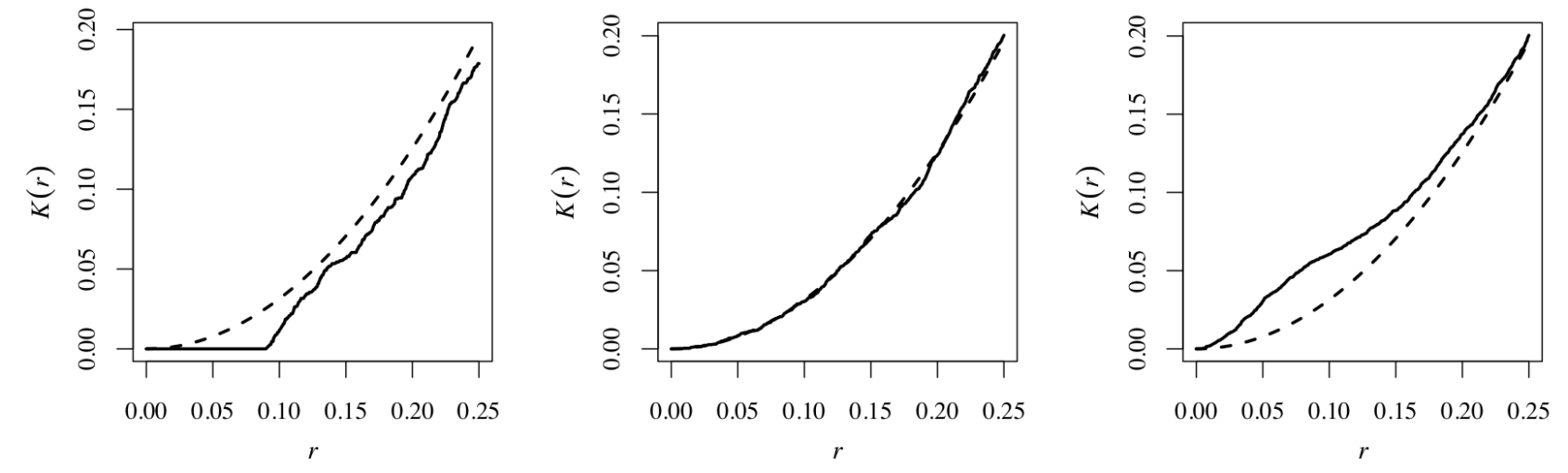
# Análisis exploratorio

# Detección/medición de autocorrelación

- Análisis visual



# Función de Ripley



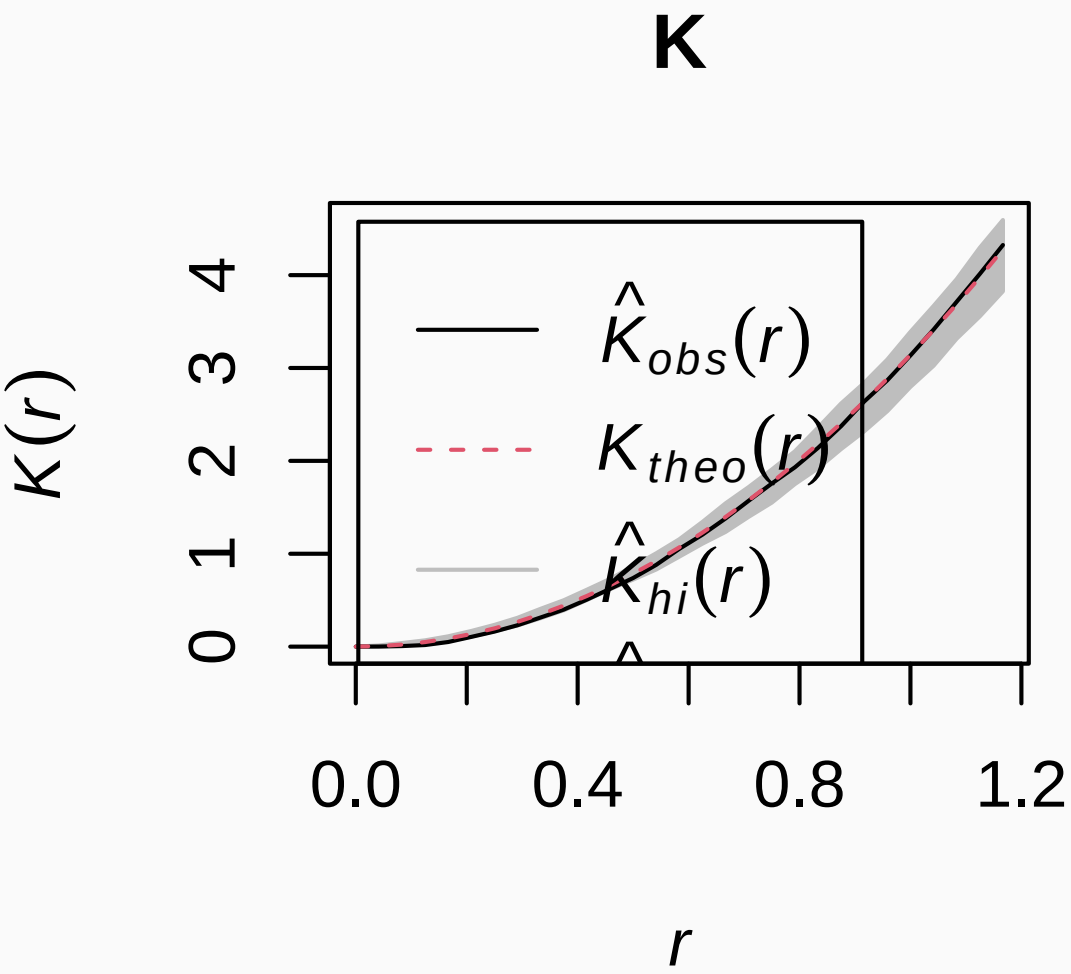
# Función de Ripley

```
K <- envelope(puntos.ppp, fun = Kest, nsim = 39)
```

- `puntos.ppp` objeto que contiene los puntos
- `fun = Kest` función que se estimará
- `nsim = 39` número de simulaciones para IC al 95%

# Función de Ripley

```
K <- envelope(puntos.ppp, fun = Kest, nsim = 39)
plot(K)
```



# Análisis de respuesta a covariables

- Función `rhohat` de `spatstat`
    - Gráfica de intensidad en relación a covariable
    - Suavizada
- Implementada en lotes con plotQuantIntens

# Análisis de respuesta a covariables

Uso de `plotQuantIntens`:

```
Q <- pixelquad(X = puntos.ppp, W = as.owin(w))

source("Funciones-spatstat/plotQuantIntens.R")
plotQuantIntens(imList = r.im,
                noCuts = 5,
                Quad = Q,
                p.pp = puntos.ppp,
                dir = "",
                name = "Respuestas")

## pdf
##   2
```

# Ajuste del modelo

# Supuestos

- Las variables ambientales no están correlacionadas
- Que la intensidad es log-lineal en relación a covariables

# Código

Se usa función `ppm`

```
m1 <- ppm(Q = puntos.ppp,
          trend = ~ Var.1,
          covariates = r.im)
```

- `Q` es el proceso de puntos
- `trend` es la fórmula del modelo
- `covariates` es la lista de variables ambientales

```
summary(m1)
```

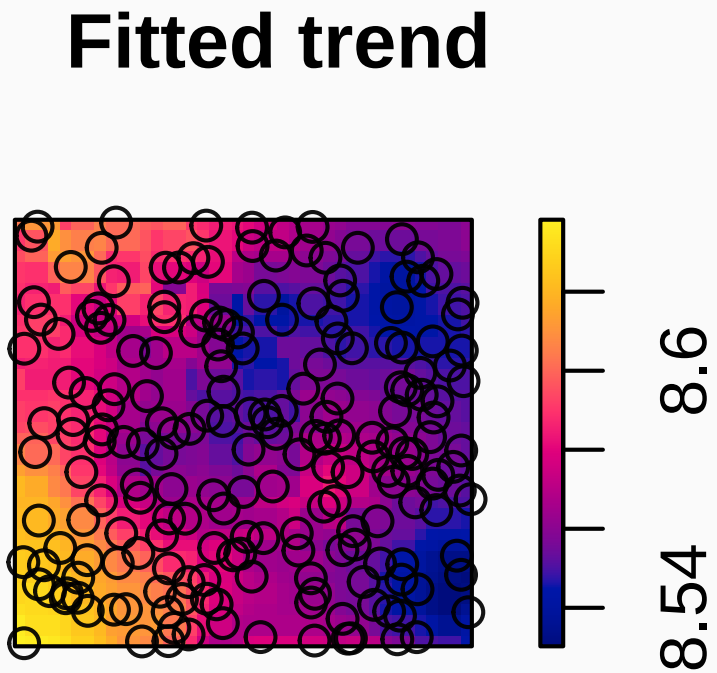
```
## Point process model
##   Fitting   method:   maximum   likelihood   (Berman-
Turner approximation)
## Model was fitted using glm()
## Algorithm converged
## Call:
## ppm.ppp(Q = puntos.ppp, trend = ~Var.1, covariates = r.im)
## Edge correction: "border"
##  [border correction distance r = 0 ]
##  ----------------------------------------------------
----------------------------
## Quadrature scheme (Berman-Turner) = data + dummy + weights
##
## Data pattern:
```

# Comparación de modelos

```
m2 <- ppm(Q = puntos.ppp,
          trend = ~ Var.2,
          covariates = r.im)

m3 <- ppm(Q = puntos.ppp,
          trend = ~ Var.3,
          covariates = r.im)

AIC(m1); AIC(m2); AIC(m3)

## [1] -458.1175

## [1] -459.3864

## [1] -455.3752
```

# El modelo “mejor”

```
plot(m3, trend = T, se = F)
```



# Para concluir

# Por hacer

- Verificar significancia de efectos
- Verificar residuales (lo que el modelo no explicó)
- Supuestos
    - Pertinencia del “área de calibración”
    - Supuesto de independencia rara vez se cumple
    - Proponer modelos de interacción o gaussianos