Analysis of point patterns with R

V. Gómez-Rubio

Departamento de MatemÃ¡ticas
Escuela de Ingenieros Industriales de Albacete
U. de Castilla-La Mancha

Imperial College London, 16 May 2014

based on work by Roger S. Bivand, Edzer Pebesma and H. Rue

Introduction

- What we see on a map is a pattern, or perhaps some patterns mixed together.
- It is not easy to work back from map pattern to the process or processes that generated it/them.
- Using a variety of approaches, we can explore and analyse point patterns, also reviewing an important chapter in the development of quantitative geography.
- Practically, we will also see how we can try out different approaches, and how their assumptions affect our conclusions.

Intro to point patterns

- A point pattern is made of the location where some events ocurred
- Possibly, we may have some extra data attached to each event (i.e., covariates)
- The aim is to disentangle the process that generated the events
  - Distribution of events in the study areas
  - Areas of high/low presence of events
  - Do events appear in clusters or equally spaced?
  - Parametric models can be proposed to account for different effects
- We will focus on a descriptive summary

Tornado Data 2009

- We will use some tornado data to show the analysis of point patterns
- These data have been obtained from the Storm Prediction Center:
- Tornado data from 1955 until 2009 are available
- In addition to the coordinates, we have a wealth of related information

Load and display tornado data

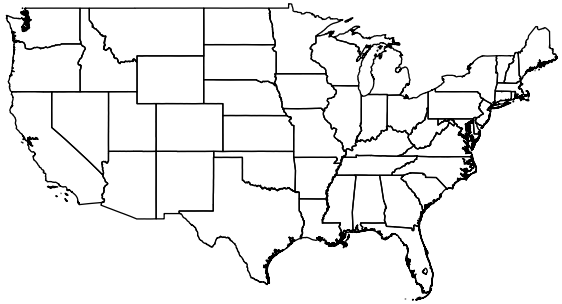
- Tornado data are avaiable in file `2009_torn.csv`.
- We will load the data and assign names to the columns
- Then we will create an **sp**-object of type SpatialPointsDataFrame
- Finally, we will set the Coordinate Reference System (CRS) of the data

```
> d<-read.csv(file="datasets/2009_torn.csv", header=FALSE)
> #Names obtanied from accompanying description file
> names(d)<-c("Number", "Year", "Month", "Day", "Date",
+     "Time", "TimeZone","State", "FIPS", "StateNumber",
+     "EFscale", "Injuries", "Fatalities", "Loss", "CLoss",
+     "SLat", "SLon", "ELat", "ELon", "Length",
+     "Width","NStates", "SNumber", "SG", "1FIPS",
+     "2FIPS", "3FIPS", "4FIPS")
> storn<-d
> #Convert data into sp-objects and assign CRS
> library(sp)
> coordinates(storn)<-~SLon+SLat
> proj4string(storn)<-"+proj=longlat"
>
```

US States Boundaries

```
> library(maptools)
> states<-readShapePoly(fn="datasets/s_01au07")
> proj4string(states)<-"+proj=longlat"
> #Remove some states...
> states2<-states[-which(states$STATE %in% c("AK", "AS", "HI", "UM", "GU", "PR", "VI")),]
> states2<-states2[!is.na(states2$STATE),]
>

> plot(states2)
```



- Displaying the points alone is almost meaningless
- In order to add some contextual information we could draw US states' boundiares
- Available from the *Storm Prediction Center* website

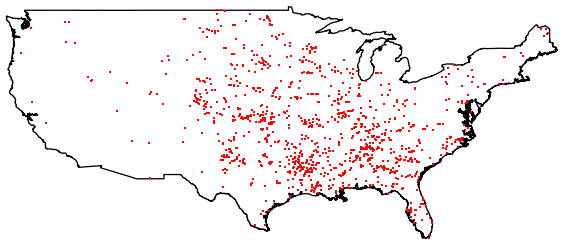
Data checking and overlay

```
> #Thin lines. THis is useful for displaying results
> statesth<-thinnedSpatialPoly(states2,  tolerance=0.05, minarea=0.001)
> save(file="statesth.RData", "statesth")

> #Create continental boundary
> USboundary<-unionSpatialPolygons(statesth, rep(1, nrow(statesth)))
> save(file="USboundary.RData", "USboundary")

> #Overlay
> sidx<-overlay(storn, states2)
> storn2<-storn[!is.na(sidx),]

> #plot(states2)
> plot(USboundary)
> points(coordinates(storn2), col="red", pch=".")
```

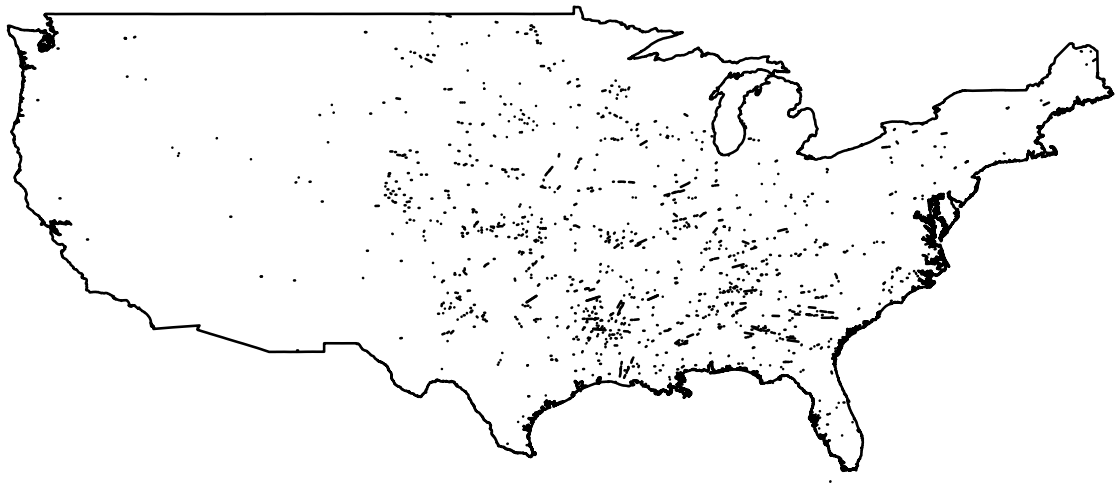


- Data may have errors
- We may want to constrain the analysis to continental US
- We can remove several regions (Alaska, Puerto Rico, Hawaii, Virgin Islands, ...)
- Create a boundary of continental US by dissolving the states boundaries
- Then use only the tornados located within this boundary
- In addition, we may want to thin the states boundaries to

Creating lines

As we have previously seen, we can produce a set of lines representing the path followed by the tornados:

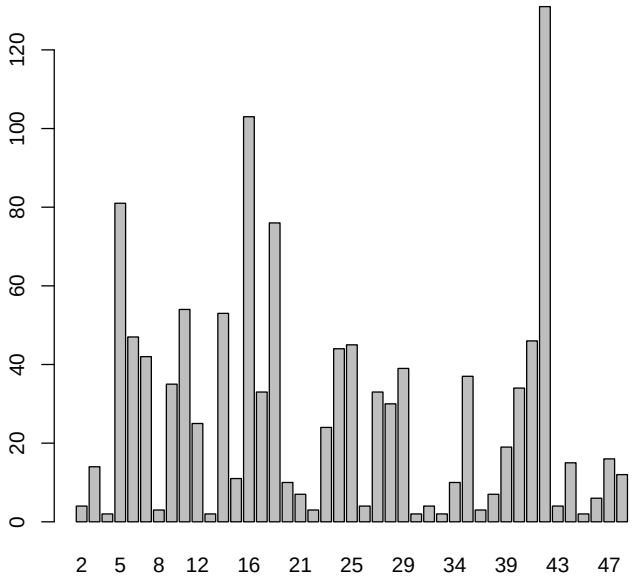
```
> sl<-lapply(unique(d$Number), function(X){
+  dd<-d[which(d$Number==X),c("SLon", "SLat", "ELon", "ELat")]
+
+
+  L<-lapply(1:nrow(dd), function(i){
+     Line(matrix(as.numeric(dd[i,]),ncol=2, byrow=TRUE))
+  })
+  Lines(L, ID=as.character(X))
+ })
> Tl<-SpatialLines(sl)
```



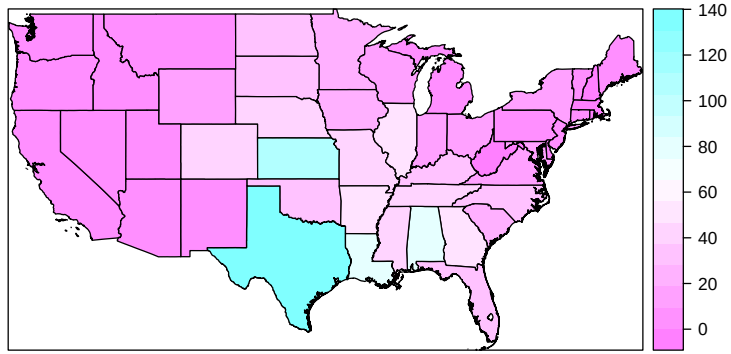
Tornados per state

From the results of the `overlay()` method we can compute the number of tornados occurred in each state:

```
> tab<-(table(sidx))
> barplot(tab)
>
```




```
> states2$NTorn<-0
> states2$NTorn[as.numeric(names(tab))]<-tab
> print(spplot(states2, "NTorn"))
>
```



Using spatstat with sp

Although spatstat and the sp classes have developed independently, they have a good deal in common, and point patterns, images and polygon windows can be exchanged

```
> library(maptools)
> library(spatstat)
> #Use data form Kansas state
> kansas<-states2[which(states2$NAME=="Kansas"),]
> kidx<-overlay(storn2, kansas)
> kstorn<-storn2[!is.na(kidx), ]
> kstorn_ppp <- as(kstorn, "ppp")
> kstorn_ppp
> kstorn_ppp2<-ppp(kstorn_ppp$x, kstorn_ppp$y, owin=as(kansas, "owin"))
>
```

Complete Spatial Randomness (CSR)

Complete Spatial Randomness occurs when events appear as follows:

- Independent of each other
- Distributed all over the study region (i.e., there are no regions with higher/lower probabilities of finding an event)

A point process like this is called a *Homogeneous Poisson Process* in a study region A.
It is defined by its intensity λ (the average number of points per unit square). It can be estimated as:

$$\hat{\lambda} = \frac{\#points}{|A|}$$

Inhomogeneous Poisson Processes (IPP)

Intensity

The intensity $\lambda(x)$ provides the average number of events at location x. The total number of events in the study region A is Poisson with mean

$$\int_A \lambda(x)dx$$

Spatial correlation

Events occur independently of each other (which is a rather strong assumption)

Modelling

- Non-parametric methods (kernel smoothing, etc.)
- Parametric methods (log-intensity as polynomials, etc.)
- Semi-parametric

Kernel Density Plots

- Kernel smoothing is a popular non-parametric way of estimating the intensity at a given point x:

$$\hat{\lambda}(x) = \sum_{i=1}^{n} \frac{1}{h^2} \kappa\left(\frac{|x - x_i|}{h}\right); \kappa(\cdot) \text{ is a kernel function}$$

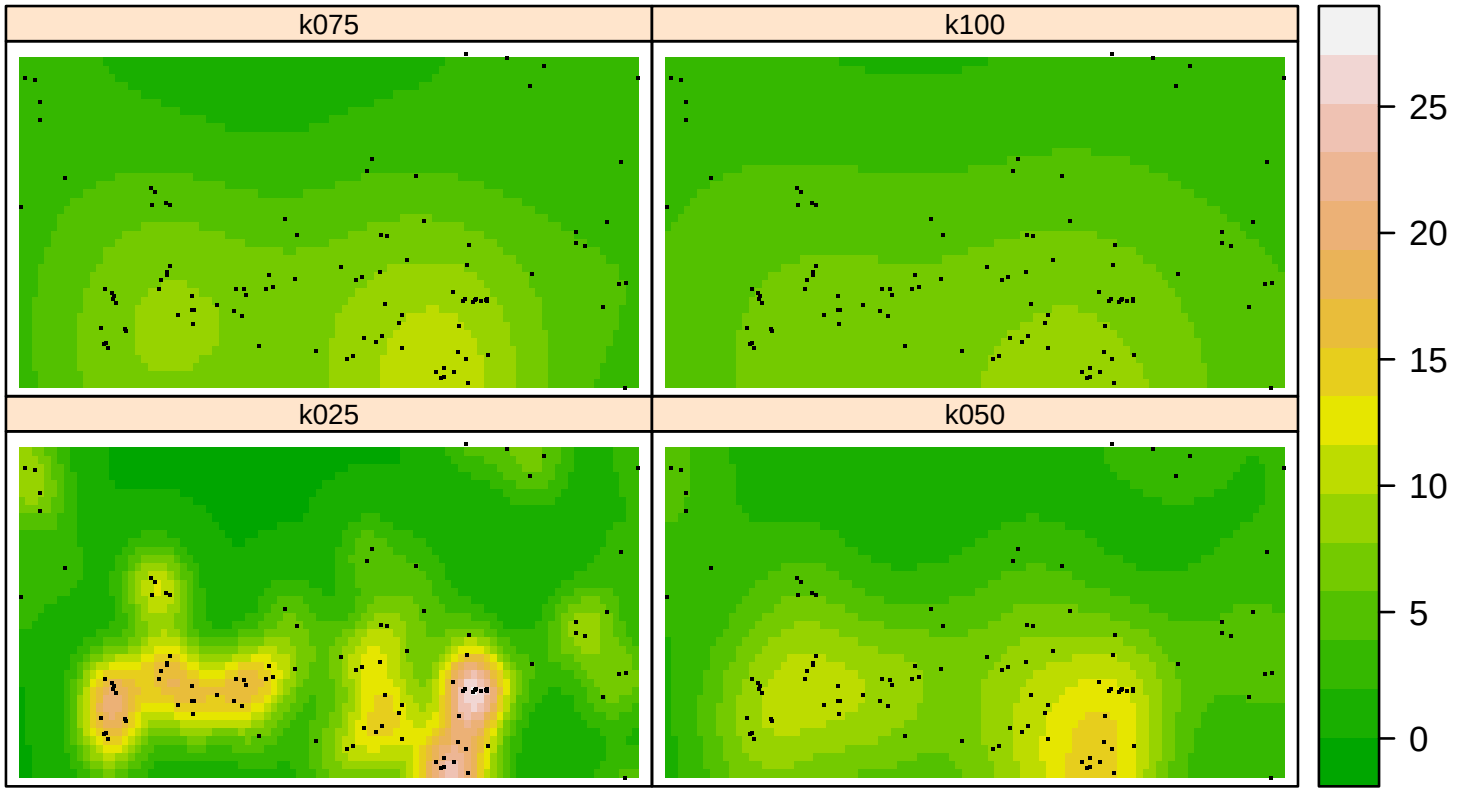
Density plots

Density plots use a 2D kernel, in spatstat a Gaussian kernel, to create smoothed histograms avoiding the problems of quadrat counts. The key argument to pass to the density method for point patterm objects is sigma=, which determines the bandwidth of the kernel. Since we can coerce the image objects output by the method to an sp class, we use this to cumulate density values for different values of sigma.

```
> k025 <- density(kstorn_ppp, sigma = 0.25, xy = crds)
> SG <- as(k025, "SpatialGridDataFrame")
> proj4string(SG)<-"+proj=longlat"
> k050 <- density(kstorn_ppp, sigma = 0.50, xy = crds)
> SG <- cbind(SG, as(k050, "SpatialGridDataFrame"))
> k075 <- density(kstorn_ppp, sigma = 0.75, xy = crds)
> SG <- cbind(SG, as(k075, "SpatialGridDataFrame"))
> k100 <- density(kstorn_ppp, sigma = 1.00, xy = crds)
> SG <- cbind(SG, as(k100, "SpatialGridDataFrame"))
> names(SG) <- c("k025", "k050", "k075", "k100")
>
```

Kernel density plots

```
> spl <- list("sp.points", kstorn, pch=".", col=1)
> print(spplot(SG, c("k025", "k050", "k075", "k100"), col.regions=terrain.colors(16), cuts=15, sp.layout=list(spl
>
```



Impact of bandwidth in the surface

Narrower bandwidths yield more extreme values, broader bandwidths narrow the interquartile range. From this table, we can see how the change in the bandwidth is affecting the relative differences in our view of the local estimates of intensity.

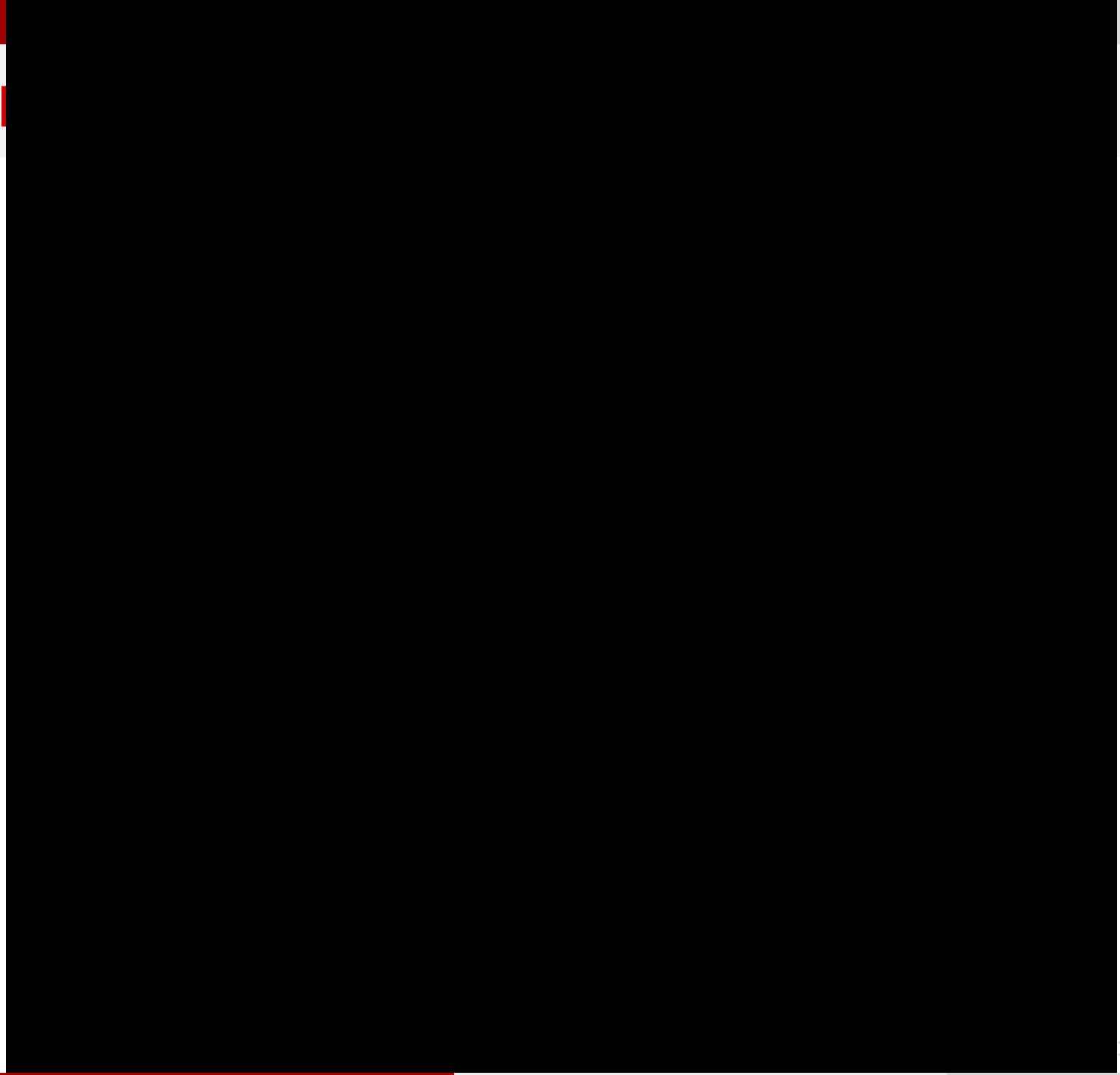
```
> summary(as(SG, "data.frame")[, 1:4])
```

```
      k025              k050              k075
 Min.   : 0.0000   Min.   : 0.06472   Min.   : 0.7076
 1st Qu.: 0.7117   1st Qu.: 2.09415   1st Qu.: 2.7809
 Median : 2.9960   Median : 3.84699   Median : 4.2155
 Mean   : 4.7789   Mean   : 4.78610   Mean   : 4.7773
 3rd Qu.: 7.2184   3rd Qu.: 7.19586   3rd Qu.: 6.8392
 Max.   :27.0827   Max.   :14.72675   Max.   :11.2575
      k100
 Min.   :1.741
 1st Qu.:3.310
 Median :4.546
 Mean   :4.768
 3rd Qu.:6.128
 Max.   :8.999
```

Interactive perspective plots

We can also display the local estimates of point intensity as a quasi-3D surface using the rgl package, and examine it interactively. This can be manipulated by changing the viewpoint with and , and equivalent parameters for the lighting source. The vertical scale is that of the local intensity.

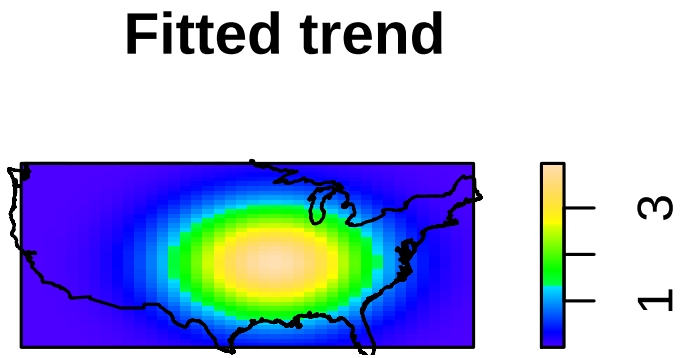
```
> library(rgl)
> z <- as(SG["k025"], "matrix")
> z[is.na(z)] <- 0
> x <- 1:nrow(z)
> y <- 1:ncol(z)
> rgl.bg(color = "white")
> rgl.clear()
> rgl.viewpoint(theta = 350, phi = 35)
> rgl.surface(x, y, z)
> rgl.snapshot("images/k025.png")
```



Parametric Estimation

- Parametric models can be used to model the log-intensity. For example:

$$log(\lambda(x)) = 1 + x + y + x^2 + y^2$$



Do Tornados appear at random?

- A starting point in the analysis of point patterns is assessing Complete Spatial Randomness
- This may seem an odd starting point in most cases
- But still, we may want to do it
- CSR can be replaced by other patterns
- For example, we may propose a model for our data and we would like to see whether it fits the data well (by simulation, for example)
- So, do tornados appear at random all over the US?

Second-order properties

Sometimes, events do not appear to be independent. Hence, we need to measure interaction between points.
This is often done by using some measure of the distance between points on the observed point pattern and for a similar point pattern (for example, with the same intensity) under CSR.
Then the two results are compared.
Usually, many different point patterns under CSR are simulated and the measure computed on them and then compared to the one for the observed point pattern

Nearest-neighbour distances: G function

A very simple function is based on using the nearest neighbour.
d_{ij} is the distance from point i to point j
$d_i = min_j\{d_{ij}, \forall i \neq j\}$ is the distance to the nearest neighbour
The G function is computed at different ranges and it computes the number of nearest neighbours within this range:

$$\hat{G}(r) = \frac{\#\{d_i : d_i < r\}}{n}$$

Under CSR this function is:

$$G(r) = 1 - \exp\{-\lambda\pi r^2\}$$

So, $\hat{G}(r)$ and $G(r)$ can be compared

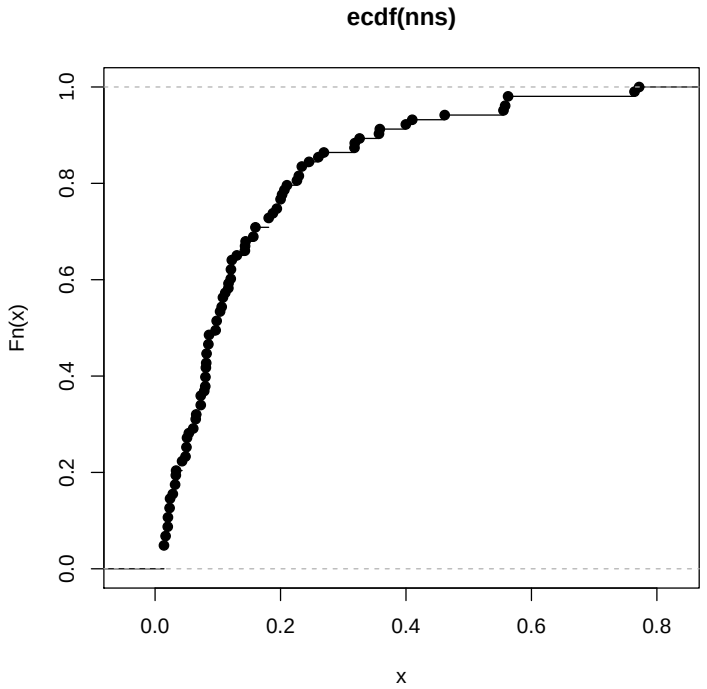
Nearest-neighbour distances

We can find and plot nearest neighbour distances, finding them with nndist – plotting the empirical cumulative distribution function of the nearest neighbour distances is interesting:

```
> nns <- nndist(kstorn_ppp)
> summary(nns)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01414 0.05045 0.09817 0.15020 0.19710 0.77170
```

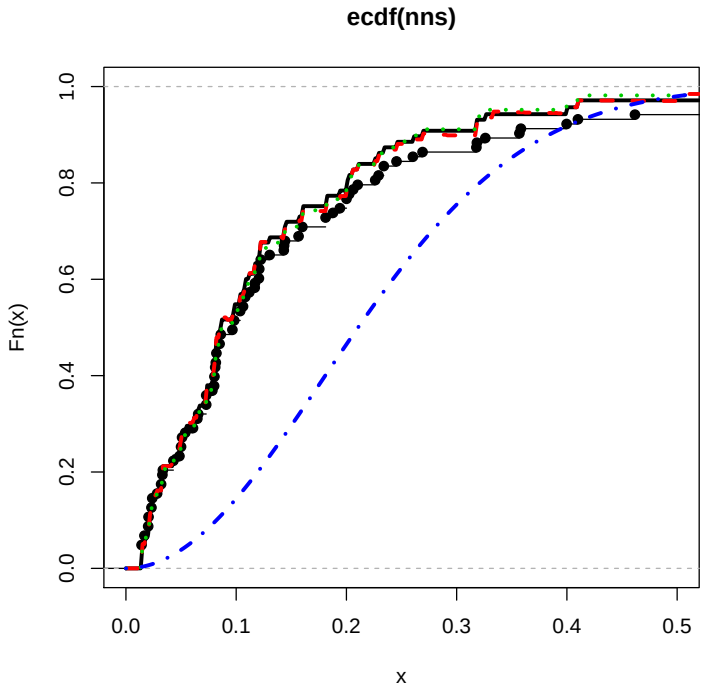
```
> plot(ecdf(nns))
```



Using $\hat{G}$ – empirical cumulative distribution function

The $\hat{G}$ measure turns out to be just the ECDF of the nearest neighbour distances, plotted by default with the expected CSR line; Gest returns binned values for a range of distance bins best chosen by the function:

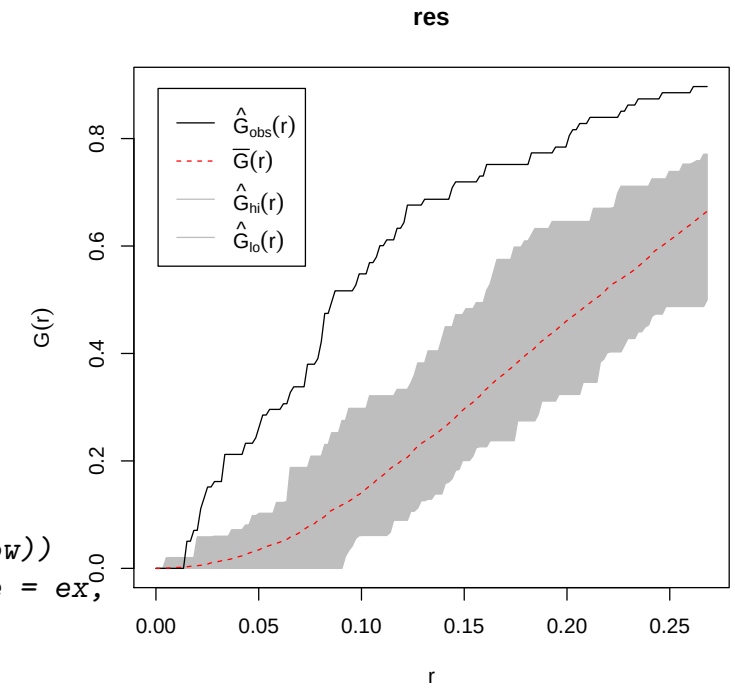
```
> plot(ecdf(nns), xlim = c(0, 0.5))
> plot(Gest(kstorn_ppp), add = TRUE,  lwd = 3)
```



Simulating $\hat{G}$ for CSR

If we generate many simulated CSR point patterns for the current window, we can use the envelope method to explore whether the observed $\hat{G}$ measures lie in relation to the simulated ones:

```
> n <- kstorn_ppp$n
> ex <- expression(runifpoint(n, win = kstorn_ppp$window))
> res <- envelope(kstorn_ppp, Gest, nsim = 99, simulate = ex,
+     verbose = FALSE, saveall = TRUE)
>
```



Ripley's K-function

Ripley's K-function is based on computing the average number of points within a range r of any given point.
For a HPP it is estimated as:

$$\hat{K}(s) = \frac{1}{\hat{\lambda}} \frac{1}{n-1} \sum_{i=1}^{n} \sum_{j \neq i} w_{ij}^{-1} |\{x_j : d(x_i, x_j) \leq s\}|$$

For a HPP $K(s)$ is

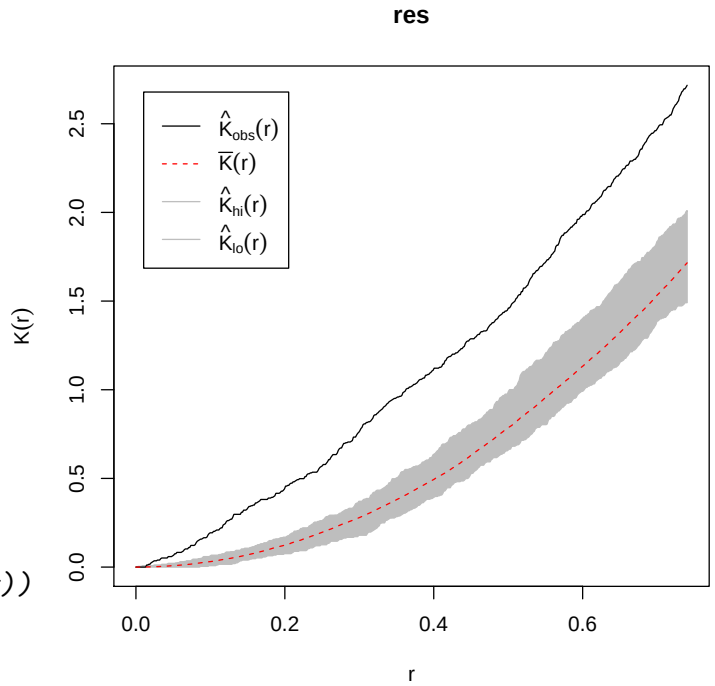
$$K(s) = \pi s^2$$

CLustering occurs at ranges where $K(s) > \pi s^2$ whilst repulsion occurs if $K(s) < \pi s^2$

$\hat{K}$ with CSR simulation

As we know, $\hat{G}$ uses nearest neighbour distances to express summary features of a point pattern. The $\hat{K}$ function uses point intensities in rings spreading out from the points, and so uses more of the data to examine what is driving the process:

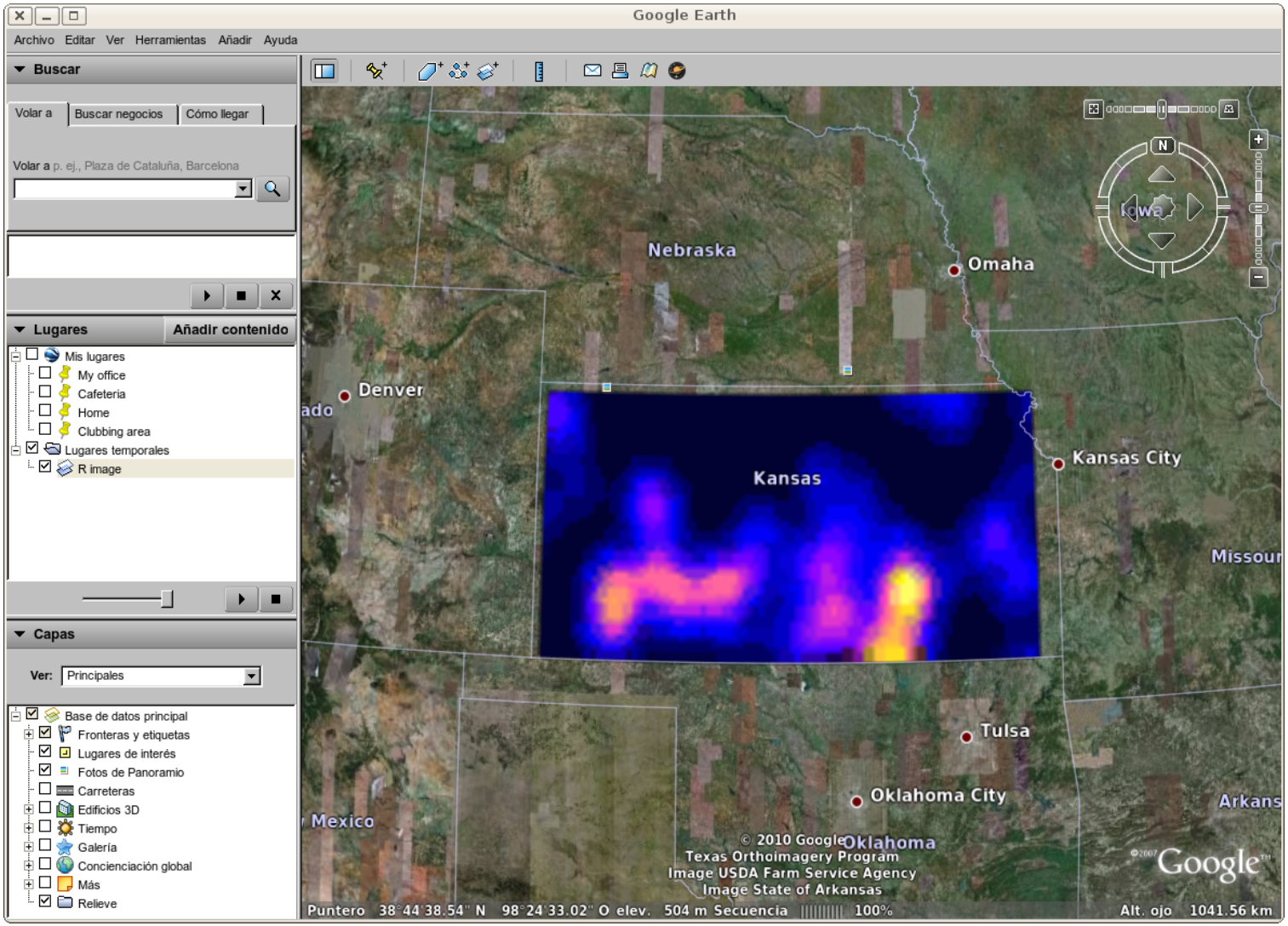
```
> ex <- expression(runifpoint(n, win = kstorn_ppp$window))
> res <- envelope(kstorn_ppp,
+     Kest, nsim = 99, simulate = ex,
+     verbose = FALSE, saveall = TRUE)
>
```



Exporting Tornado density to GE

```
> library(maptools)
> library(rgdal)
> grd1 <- as(as(SG, "SpatialPixels"), "SpatialPolygons")
> proj4string(grd1) <- CRS(proj4string(SG))
> grd.union <- unionSpatialPolygons(grd1, rep("x", length(slot(grd1, "polygons"))))
> grd.union.ll <- spTransform(grd.union, CRS("+proj=longlat"))
> llGRD <- GE_SpatialGrid(grd.union.ll, maxPixels = 100)
> llGRD_in <- overlay(llGRD$SG, grd.union.ll)
> llSPix <- as(SpatialGridDataFrame(grid=slot(llGRD$SG, "grid"),
+    proj4string=CRS(proj4string(llGRD$SG)),
+    data=data.frame(in0=llGRD_in)), "SpatialPixelsDataFrame")
> #SPix <- spTransform(llSPix, CRS("+init=epsg:28992"))
> #z <- predict(OK_fit, newdata=SPix, debug.level=0)
> #llSPix$pred <- z$OK_fit.pred
>
>
> png(file="tornado_kansas.png", width=llGRD$width, height=llGRD$height,
+    bg="transparent")
> par(mar=c(0,0,0,0), xaxs="i", yaxs="i")
> image(SG, "k025", col=bpy.colors(20))
> dev.off()
> kmlOverlay(llGRD, "tornado_kansas.kml", "tornado_kansas.png")
>
>
```

GE


Google Earth
Archivo Editar Ver Herramientas Añadir Ayuda
Buscar
Volar a
Buscar negocios
Cómo llegar
Volar a p. ej., Plaza de Cataluña, Barcelona
Lugares
Añadir contenido
Mis lugares
My office
Cafeteria
Home
Clubbing area
Lugares temporales
R image
Capas
Ver: Principales
Base de datos principal
Fronteras y etiquetas
Lugares de interés
Fotos de Panoramio
Carreteras
Edificios 3D
Tiempo
Galería
Concienciación global
Más
Relieve
Nebraska
Omaha
Denver
Kansas
Kansas City
Missour
Tulsa
Oklahoma City
Arkans
Mexico
© 2010 Google
Texas Orthoimagery Program
Image USDA Farm Service Agency
Image State of Arkansas
Google
Puntero 38°44'38.54" N 98°24'33.02" O elev. 504 m Secuencia 100%
Alt. ojo 1041.56 km

Conclusions about tornados (in Kansas)

- The number of tornados is uneven across the states
- Kansas has one of the highest occurrances of tornados (as we already knew from the *Wizard from Oz*)
- In kansas it is more likely to see a tornado in the south (is there where Dorothy's house was placed?)
- Tornados appear in clusters, i.e., if you have seen one it is more likely that you will see another one soon!

Analysis of case-control data

- So far we have studied the case of a point pattern
- Sometimes we may be interested in comparing two point patterns:
  - Cases and non-cases (controls) of a disease
  - Two species of trees
- We need to take into account the spatial distribution of both point processes and some additional information (i.e., covariates)
- This can be extended to the case of more than two point processes

Asthma in children

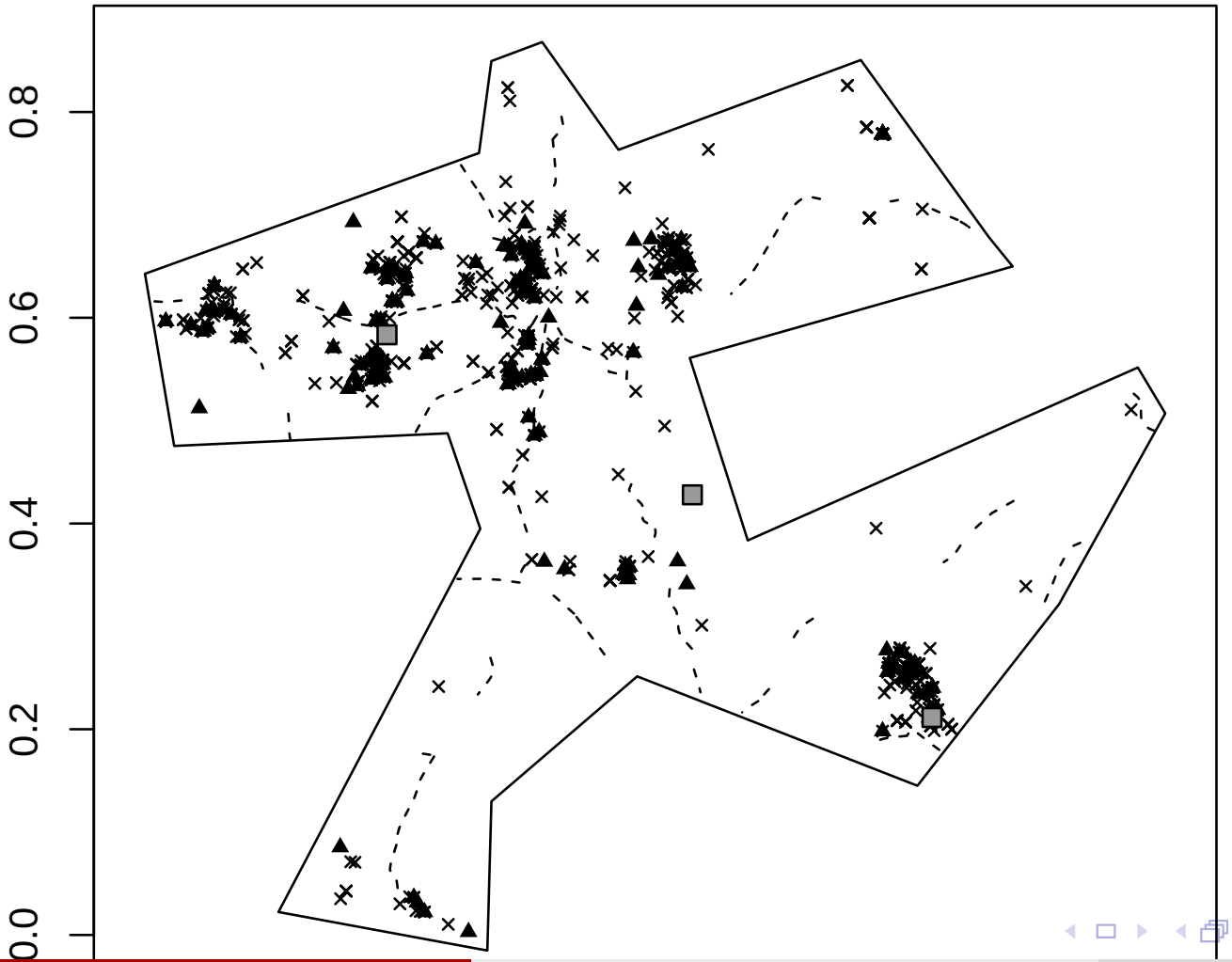
- Study on risk factors related to asthma in children
- Location of the residence of children from 10 schools
- Information available includes distance to 3 putative pollution sources and socio-economic variables
- Cases are all children with asthma and controls are all the other children
- $\lambda_1(x)$ is the intensity of the cases and $\lambda_0(x)$ is the intensity of the controls
- In case they had the same spatial distribution:

$$\lambda_1(x) = \frac{n_1}{n_0}\lambda_0(x)$$

- Alternatively:

$$\rho(x) = \lambda_1(x)/\lambda_0(x) = \frac{n_1}{n_0}$$

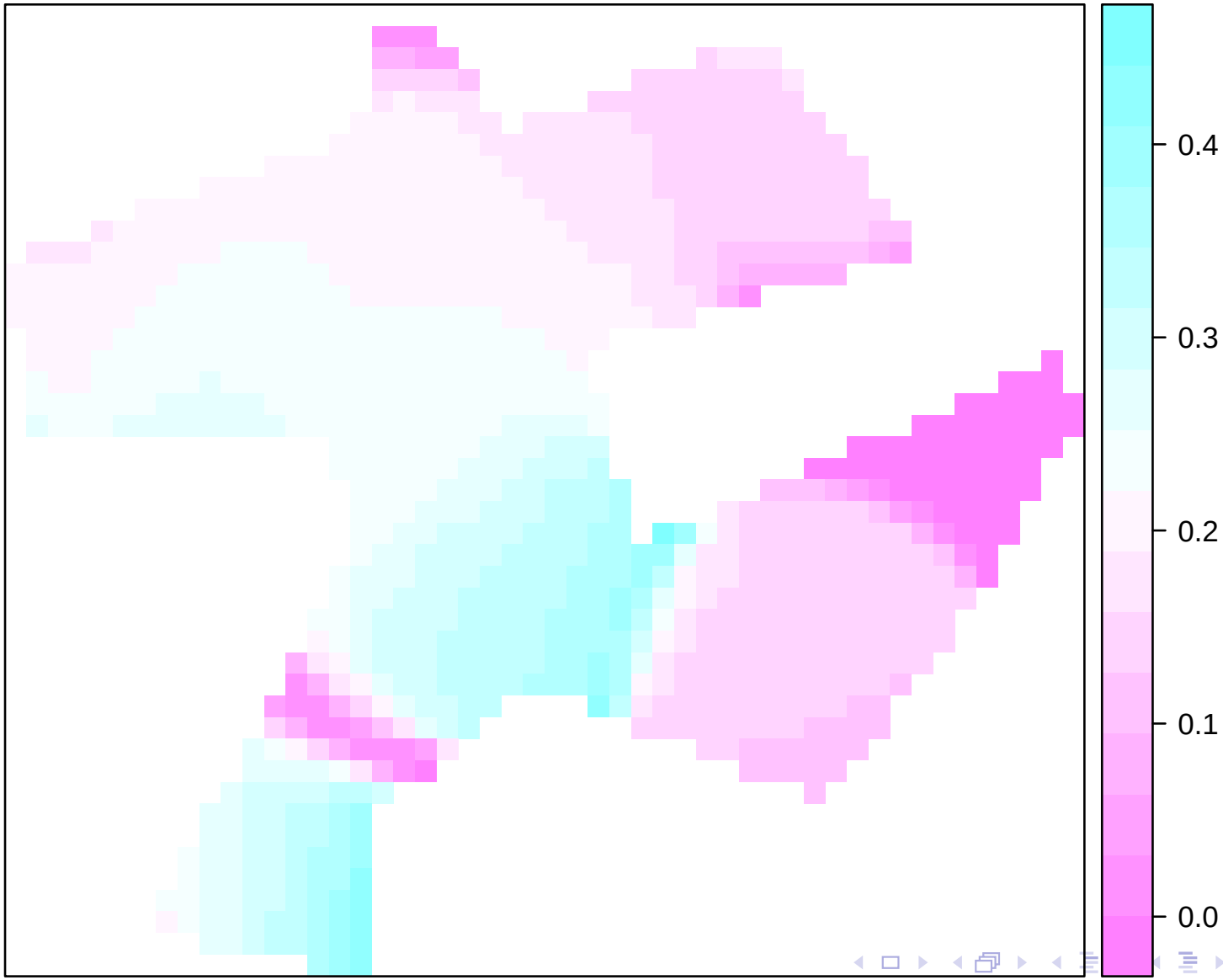
Asthma in children



Estimation of the ratio of intensities

```
> bwasthma <- .125 #.275
> library(maptools)
> sG <- Sobj_SpatialGrid(spbdry, maxDim=50)$SG
> gt <- slot(sG, "grid")
> summary(gt)
> pbdry <- slot(slot(slot(spbdry, "polygons")[[1]], "Polygons")[[1]], "coords")
> library(splancs)
> cases<-spasthma[spasthma$Asthma=="case",]
> ncases<-nrow(cases)
> controls<-spasthma[spasthma$Asthma=="control",]
> ncontrols<-nrow(controls)
> kcases<-spkernel2d(cases, pbdry, h0=bwasthma, gt)
> kcontrols<-spkernel2d(controls, pbdry, h0=bwasthma, gt)
> df0 <- data.frame(kcases=kcases, kcontrols=kcontrols)
> spkratio0 <- SpatialGridDataFrame(gt, data=df0)
> spkratio <- as(spkratio0, "SpatialPixelsDataFrame")
> spkratio$kratio <- spkratio$kcases/spkratio$kcontrols
> is.na(spkratio$kratio) <- !is.finite(spkratio$kratio)
> spkratio$logratio <- log(spkratio$kratio)-log(ncases/ncontrols)
>
```

Estimation of the ratio of intensities



Confidence regions

Regions with high/low values of the ratio of intensities can be identified by means of a Monte Carlo Test

```
> idxinbdry <- overlay(sG, spbdry)
> idxna <- !is.na(idxinbdry)
> niter <- 99
> ratio <- rep(NA, niter)
> pvaluemap <- rep(0, sum(idxna))
> rlabelratio <- matrix(NA, nrow=niter, ncol=sum(idxna))
> for(i in 1:niter)
+ {
+       idxrel <- sample(spasthma$Asthma) == "case"
+       casesrel <- spasthma[idxrel,]
+       controlsrel <- spasthma[!idxrel,]
+
+       kcasesrel <- spkernel2d(casesrel, pbdry, h0=bwasthma, gt)
+       kcontrolsrel <- spkernel2d(controlsrel, pbdry, h0=bwasthma, gt)
+       kratiorel <- kcasesrel[idxna]/kcontrolsrel[idxna]
+          is.na(kratiorel) <- !is.finite(kratiorel)
+       rlabelratio[i,] <- kratiorel
+
+       pvaluemap <- pvaluemap + (spkratio$kratio < kratiorel)
+ }
> #save(pvaluemap, rlabelratio, file="sppaRlabelratio.RData")
>
```

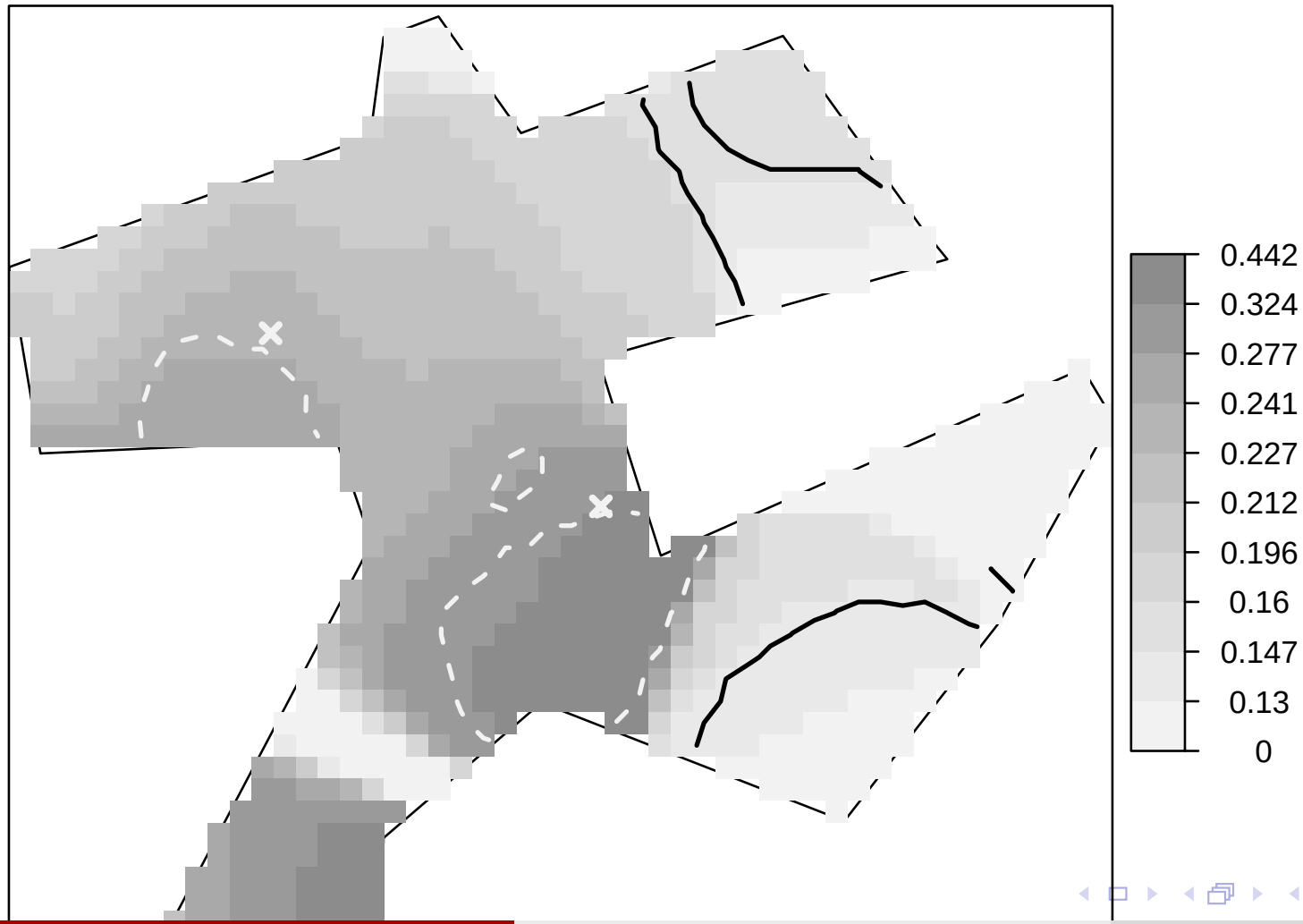
Confidence regions

```
> idxna2 <- apply(rlabelratio, 2, function(x) all(is.finite(x)))
> rhomean <- apply(rlabelratio[, idxna2], 2, mean)
> c <- prod(slot(gt, "cellsize"))
> ratiorho <- c*sum((spkratio$kratio[idxna2]-ncases/ncontrols)^2)
> ratio <- c*apply(rlabelratio[,idxna2], 1,
+  function(X, rho0 ){sum((X-rho0)^2)}, rho0=ncases/ncontrols
+ )
> pvaluerho <- (sum(ratio > ratiorho)+1)/(niter+1)
> spkratio$pvaluemap <- (pvaluemap+1)/(niter+1)
> imgpvalue <- as.image.SpatialGridDataFrame(spkratio["pvaluemap"])
> clpvalue <- contourLines(imgpvalue, levels=c(0,.05, .95, 1))
> cl <- ContourLines2SLDF(clpvalue)
>
```

Confidence regions

```
> library(RColorBrewer)
> cl05 <- cl[cl$level == "0.05",]
> xzx <- slot(slot(cl05, "lines")[[1]], "Lines")
> cl05a <- SpatialLines(list(Lines(xzx, ID="0.05")))
> lyt05 <- list("sp.lines", cl05a, lwd=2, lty=2, col="grey95")
> lyt95 <- list("sp.lines", cl[cl$level == "0.95",], lwd=2, lty=1)
> lytb <- list("sp.polygons", spbdry)
> lytp <- list("sp.points", spsrc, cex=0.9, pch=4, col="grey95", lwd=3)
> brks <- quantile(spkratio$kratio[spkratio$kratio>0], seq(0,1,1/10), na.rm=TRUE)
> brks[1] <- 0
> lbrks <- formatC(brks, 3, 6, "g", " ")
> cols <- colorRampPalette(grey.colors(5, 0.95, 0.55, 2.2))(length(brks)-1)
> # RSB quietening greys
> colorkey<-list(labels=lbrks,
+   at=(0:10)/10, height=.5)
```

Confidence regions



Estimation of the intensity with covariates

The intensity of the cases can be modelled as the intensity of the cases modulated according to some risk factors or covariates:

$$\lambda_1(x) = \exp\{\alpha + \beta x\}\lambda_0(x)$$

Conditioning on the locations of cases and controls we can estimate the probability of being a case as follows:

$$p(x) = \frac{\lambda_1(x)}{\lambda_1(x) + \lambda_0(x)} = \frac{\exp\{\alpha + \beta x\}}{1 + \exp\{\alpha + \beta x\}}$$

$\lambda_0(x)$ se puede estimar mediante métodos no paramétricos y α, β mediante GLMS:

$$\text{logit}(p(x)) = \alpha + \beta x$$

```
> spasthma$y <- as.integer(!as.integer(spasthma$Asthma)-1)
> spasthma$HayFeverf<- as.factor(spasthma$HayFever)
> glmasthma<-glm(y~HayFeverf, data=spasthma, family="binomial")
> prob<-fitted(glmasthma)
> weights<-exp(glmasthma$linear.predictors)
> #weights<-fitted(glmasthma)
> library(spatialkernel)
> setkernel("gaussian")

[1] "gaussian"

> lambda0<- lambdahat(coordinates(controls), bwasthma, coordinates(cases),
+    pbdry, FALSE)$lambda
> lambda1<- weights[spasthma$Asthma=="case"]*lambda0
> ratiocc<-ncases/ncontrols
> s<-seq(0, .15, by=.01)
> kihnocov<-kinhat(coordinates(cases), ratiocc*lambda0, pbdry,s)$k
> kih<-kinhat(coordinates(cases), lambda1, pbdry,s)$k
```

Clustering adjusting for covariates

Inhomogeneous K-function

Similar to Ripley's K-function but allowing for non-constant intensity $\lambda(x)$:

$$\hat{K}_{I,\hat{\lambda}}(s) = |A|^{-1} \sum_{i=1}^{n} \sum_{j \neq i} w_{ij}^{-1} \frac{\|x_j : d(x_i, x_j) \leq s\|}{\hat{\lambda}(x_i)\hat{\lambda}(x_j)}$$

Clustering test

It is based on the statistic

$$D = \int_0^{s_0} \frac{(\hat{K}_{I,\hat{\lambda}}(s) - E[s])}{var(\hat{K}_{I,\lambda}(s))^{1/2}} ds$$

Clustering adjusting for covariates

```
> niter<-99
>  kinhomrelnocov<-matrix(NA, nrow=length(s), ncol=niter)
>  kinhomrel<-matrix(NA, nrow=length(s), ncol=niter)
>  for(i in 1:niter)
+  {
+          idxrel<-sample(spasthma$Asthma, prob=prob)=="case"
+          casesrel<-coordinates(spasthma[idxrel,])
+          controlsrel<-coordinates(spasthma[!idxrel,])
+
+       lambda0rel<-lambdahat(controlsrel, bwasthma, casesrel, pbdry, FALSE)$lambda
+       lambda1rel<-weights[idxrel]*lambda0rel
+
+       kinhomrelnocov[,i]<-kinhat(casesrel, ratiocc*lambda0rel, pbdry,s)$k
+       kinhomrel[,i]<-kinhat(casesrel, lambda1rel, pbdry,s)$k
+
+  }
> #    save(kinhomrelnocov, kinhomrel, file="sppaKinhom.RData")
```

Clustering adjusting for covariates

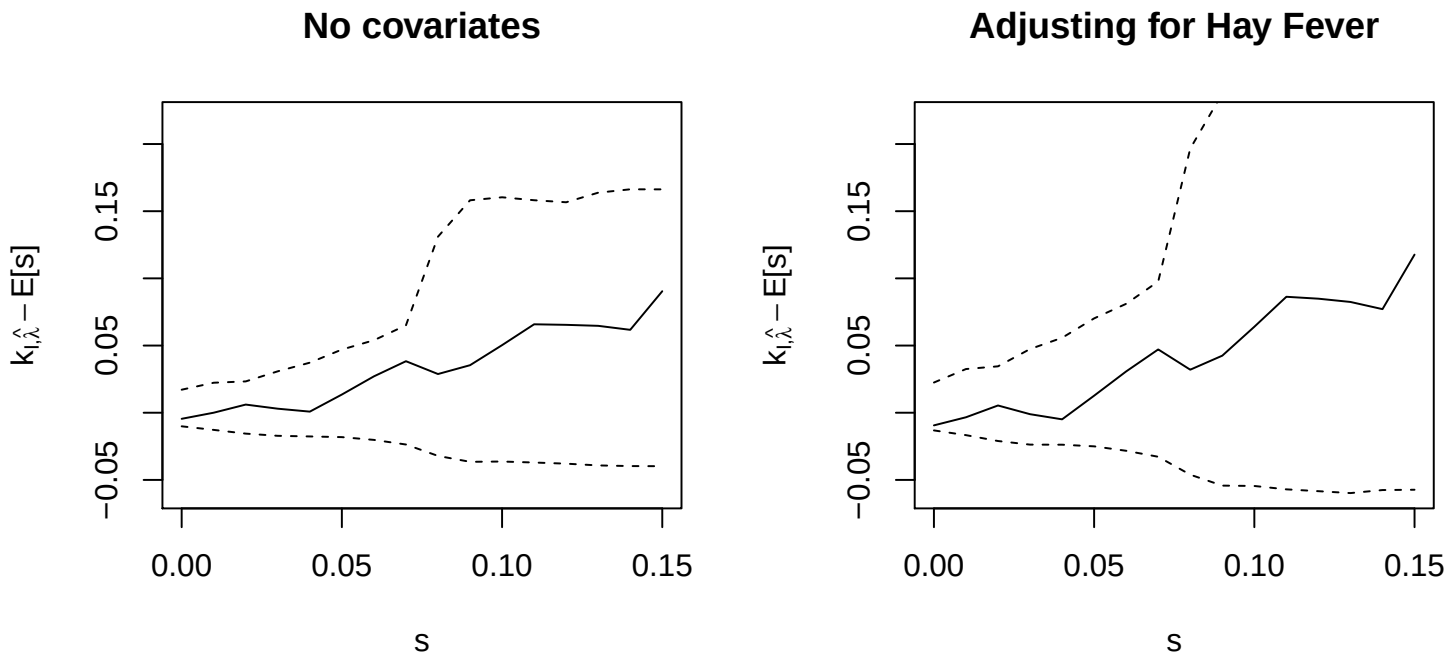
Test with no covariates:

```
> kinhsdnocov<-apply(kinhomrelnocov, 1, sd)
> kihmeannocov<-apply(kinhomrelnocov, 1,mean)
> D0nocov<-sum((kihnocov-kihmeannocov)/kinhsdnocov)
> Dnocov<-apply(kinhomrelnocov, 2,
+    function(X){ sum((X-kihmeannocov)/kinhsdnocov)})
> pvaluenocov<-(sum(Dnocov>D0nocov)+1)/(niter+1)
>
```

Test with covariates:

```
> kinhsd<-apply(kinhomrel, 1, sd)
> kihmean<-apply(kinhomrel, 1,mean)
> D0<-sum((kih-kihmean)/kinhsd)
> D<-apply(kinhomrel, 2,
+    function(X){ sum((X-kihmean)/kinhsd)})
> pvalue<-(sum(D>D0)+1)/(niter+1)
>
```

Clustering adjusting for covariates



Other models for point patterns

Cluster processes

- Events appear is clusters
- First, a set of *parents* are placed
- Secondly, every parent produces an offspring, which becomes the observed events
- Hence, events are not independent of each other

Pairwise-interaction processes

- Provide a different way of modelling pairwise interaction
- An interaction term can be introduced to model attraction or repulsion between events