Week 4: Spatial Point Pattern Analysis I

```r
knitr::opts_chunk$set(fig.width=6, fig.height=4, fig.path='Figs/',  warning=FALSE, message=FALSE, resul
rm(list=ls())
library(sp)
library(spatstat)
```

```
## Loading required package: spatstat.data

## Loading required package: nlme

## Loading required package: rpart

##
## spatstat 1.56-1       (nickname: 'Invisible Friend')
## For an introduction to spatstat, type 'beginner'

##
## Note: R version 3.3.3 (2017-03-06) is more than 9 months old; we strongly recommend upgrading to the
```

Point Pattern Analysis

- set of n point locations with recorded "events", e.g., locations of trees, disease or crime incidents $S = \{s_1, \dots, s_i, \dots, s_n\}$
- point locations correspond to all possible events or to subsets of them
- attribute values also possible at same locations, e.g., tree diameter, magnitude of earthquakes (i.e.,marked point pattern) $W = \{w_1, \dots, w_i, \dots, w_n\}$

Objectives:

- detect spatial clustering or repulsion, as opposed to complete randomness, of event locations (in space and time)
- if clustering detected, investigate possible relations with nearby "sources"

Further

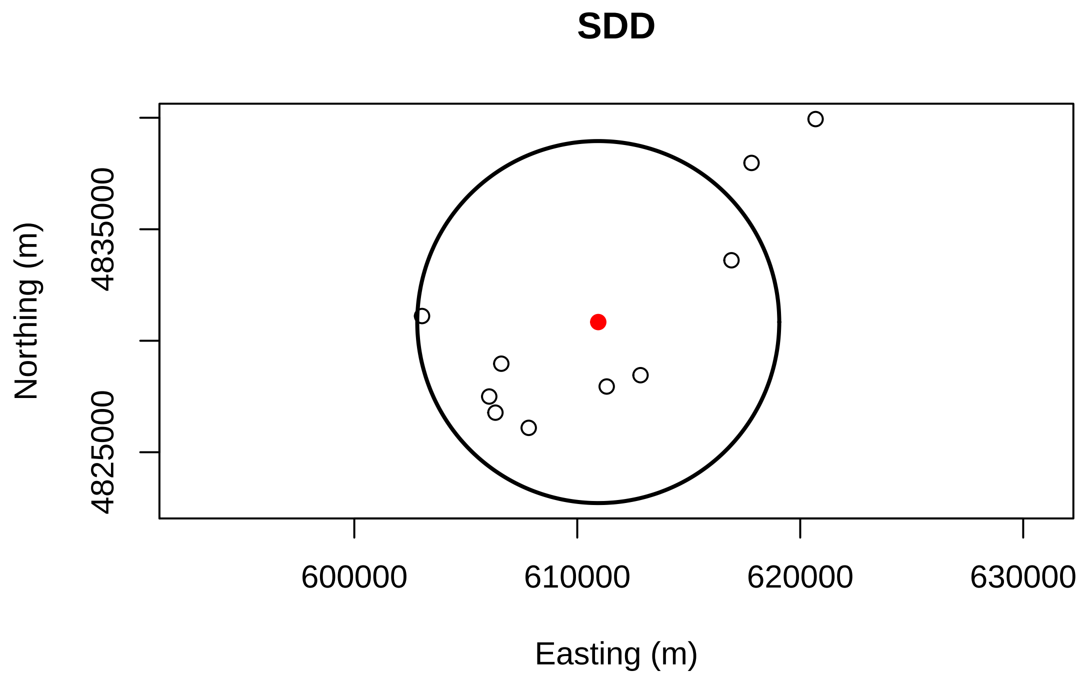
- analysis of point patterns over large areas should take into account distance distortions due to map projections
- boundaries of study area should not be arbitrary
- analysis of sampled point patterns can be misleading
- one-to-one correspondence between objects in study area and events in pattern

In this lecture, we will focus on the spatial statistical tools for cluster detection.

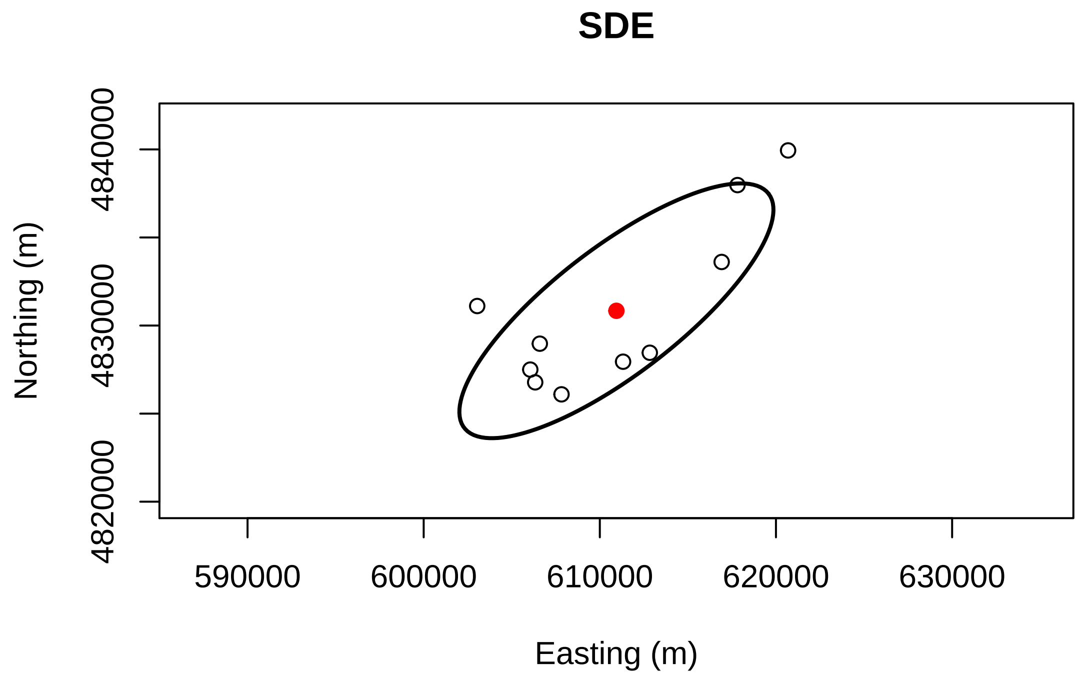
Descriptive statistics

- Mean center
- Median center
- Standard distance
- For example:

```r
library(aspace)
plot_sdd(plotnew=TRUE, plothv=FALSE, plotweightedpts=FALSE, plotpoints=TRUE, plotcentre=TRUE, titletxt=
yaxis="Northing (m)", sdd.col='black', centre.col='red')
```



```r
plot_sde(plotnew=TRUE, plothv=FALSE, plotweightedpts=FALSE, plotpoints=TRUE, plotcentre=TRUE, titletxt=
yaxis="Northing (m)", sdd.col='blue', centre.col='red')
```

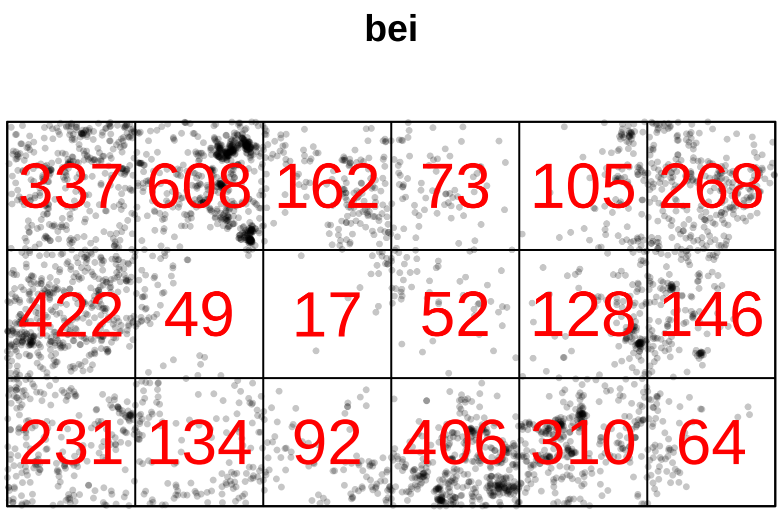


Intensity-based methods (first-order)

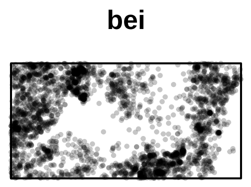
Quadrant count

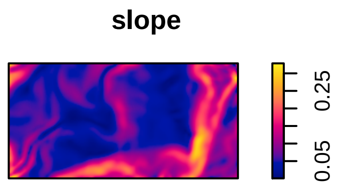
- histogram in high dimensional spaces
- the starting point and size of the quadrant will affect the results

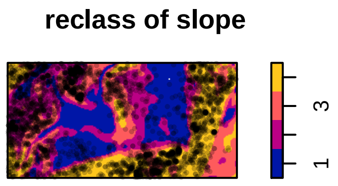
```r
data(bei)
X = bei
Q<-quadratcount(X,nx=6,ny=3)
plot(X, main='bei',pch = 20, cex = 0.5)
plot(Q,add=TRUE,cex=2, col='red')
```

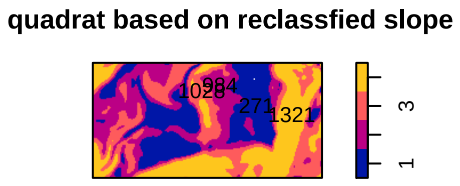


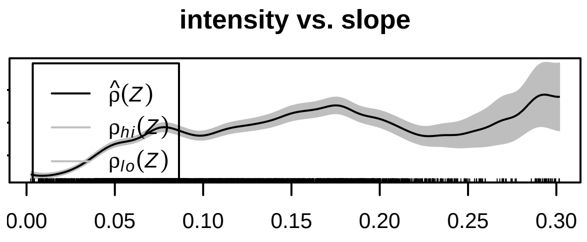
```r
quadrat.test(Q)

# investigate the effects of other covariates

Z <- bei.extra$grad
par(mfrow = c(3, 2))
par(mar=c(2,0,3,0) + 0.1)
plot(bei, pch=20, cex= 0.5, main='bei')
plot(Z, main='slope')
b <- quantile(Z, probs = (0:4)/4)
Zcut <- cut(Z, breaks = b, labels = 1:4)
V <- tess(image = Zcut)
plot(V, main = 'reclass of slope')
plot(bei, add = TRUE, pch=20, cex=0.5)
qb <- quadratcount(bei, tess = V)
plot(qb, main = 'quadrat based on reclassfied slope')
plot(rhohat(bei, Z), main='intensity vs. slope')
```











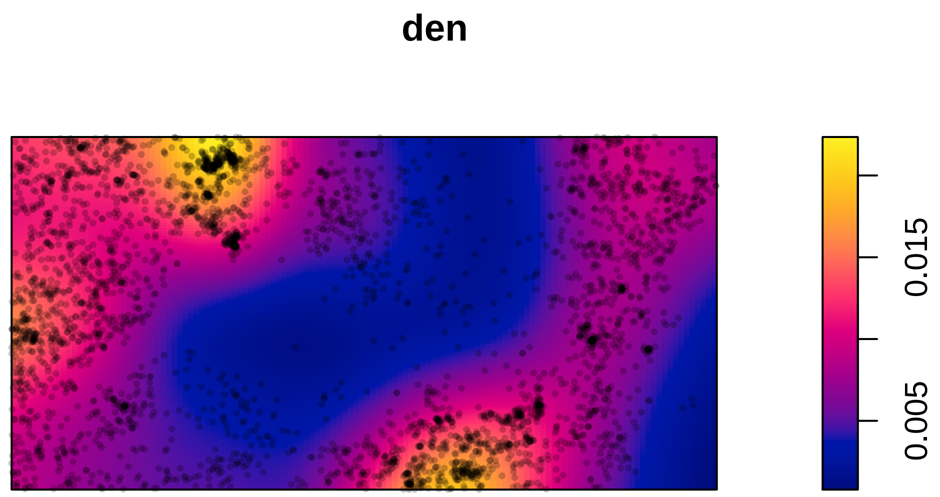
kernel density estimation

- Based on quadrant method, $\hat{p}(s) = \frac{1}{N}\frac{\#\text{ in bin}}{\text{area of bin}} = \frac{k_b}{N*V_b}$, where k_b is the number of points in bin b, and V_b is the volume of b
- Parzen window: a region centered at each s with length h. To find the number of points that fall within this region, we define a `kernel function`

$$K(u) = K(\frac{s-s_i}{h}) = \begin{cases} 1 & \text{if |u|} < 1/2; \\ 0 & \text{o.w.} \end{cases}$$

  Then the previous density can be written as $\hat{p}(s) = \frac{1}{N*h^d}\sum_{i=1}^{N} K(\frac{s-s_i}{h})$
- The kernel function can be generalized as a smooth kernel function $K(u)$, $\int_{R^d} K(u)du = 1$, where $u = |s - s_i|$
- h is called 'bandwidth' or 'smoothing parameters'
- Different types of kernel functions

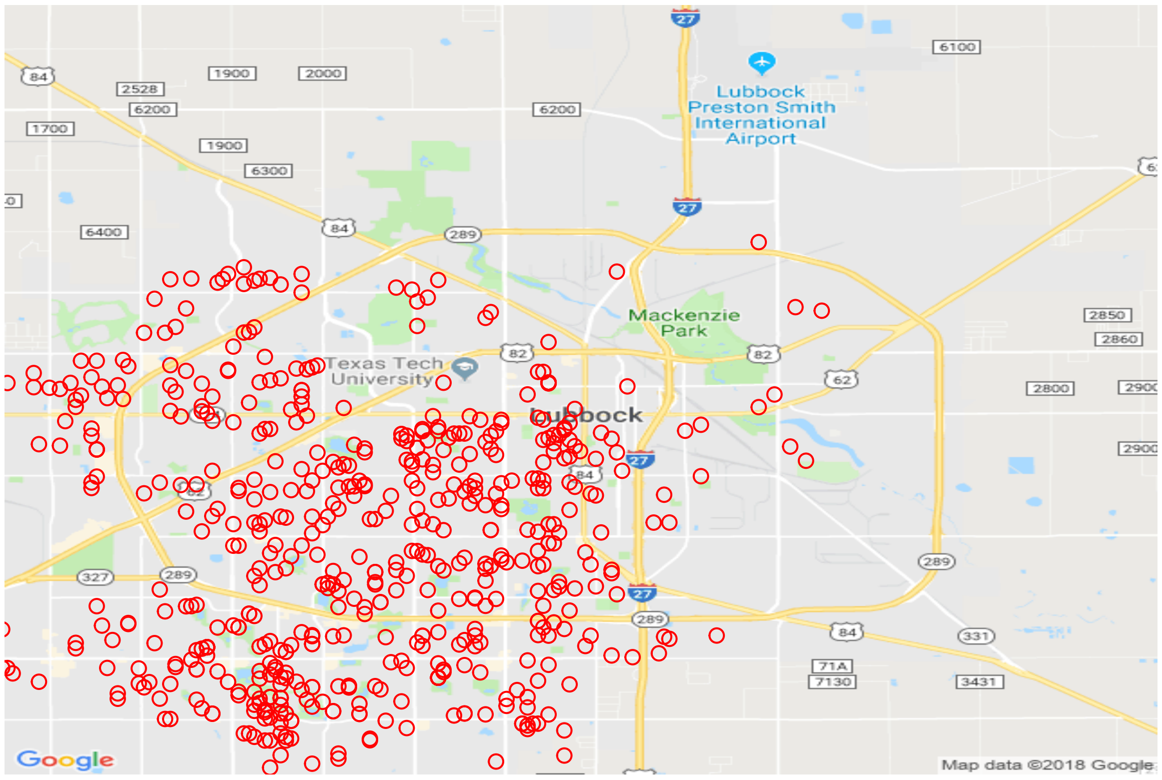
```r
library(spatstat)
X=bei
den=density(X, sigma=70, main='density')
plot(den)
plot(X, add = TRUE, cex = 0.5, main='density', pch = 20)
```



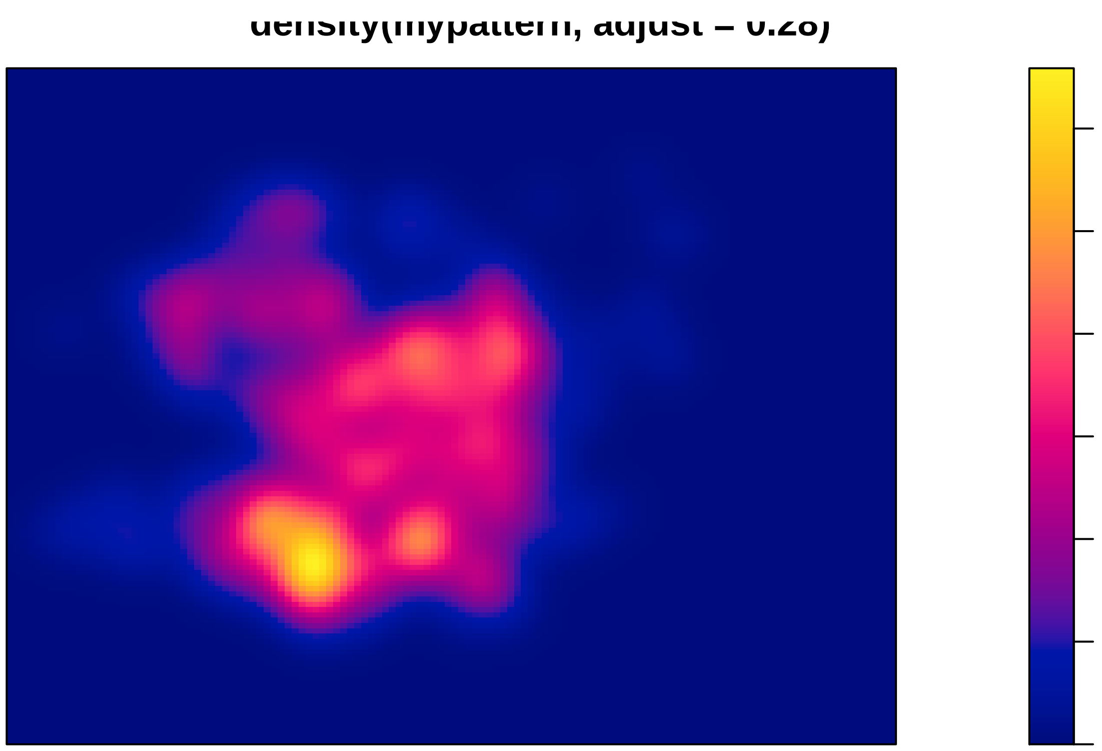
```
#persp(den)
```

Another example using Lubbock data

```
library(maptools)
library(ggmap)
library(RgoogleMaps)

house<-readShapePoints("Data/HouseSaleLubbock.shp")
#check that my data contains the data you expect
head(house)
#get the coordinate of each house on sale
Lat<-house$Lat
Lon<-house$Lon
#check that the point pattern looks right by plotting it

#lubbock=geocode('lubbock')
lubbock=c(33.583794, -101.855836)

newmap <- GetMap(center = lubbock, zoom = 12, destfile = "newmap.png", maptype = "roadmap")

PlotOnStaticMap(newmap, lat=Lat, lon=Lon, col='red')
```



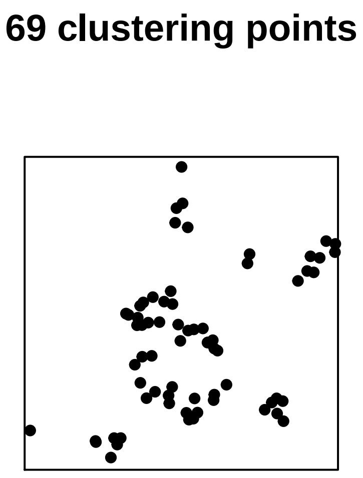
```
#convert the data to a point pattern object
mypattern<-ppp(Lon,Lat, c(-102,-101.75), c(33.46,33.65))
#basic summary of the data
summary(mypattern)

#kernel density estimation map
plot(density(mypattern,adjust=0.28))
```

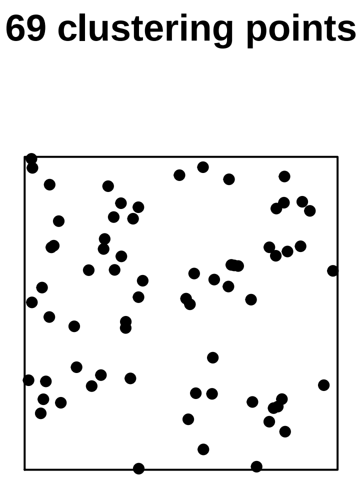


Distance-based methods (second-order)

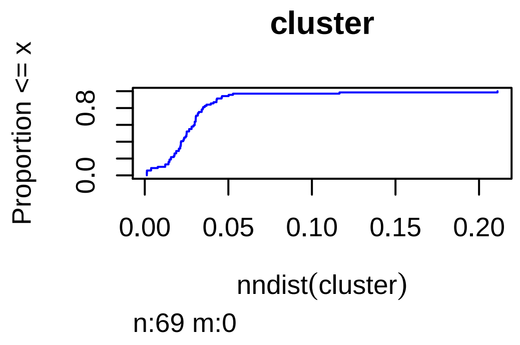
- G functions: sample cdf of nearest pair-wise distances between events.
- F functions: sample cdf of nearest distances between arbitrary points and observations or empty space function.
- K functions: $K(t) = \lambda^{-1}$Enumber of events within distance t of an arbitrary event, where λ is the density of the study area. It actually represents the sample cdf of pair-wise distances normalized by area.

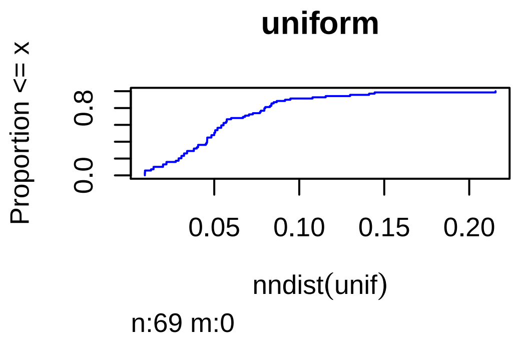
```
library(Hmisc)
cluster <- rMatClust(20, 0.05, 4)
unif = runifpoint(cluster$n)

par(mfrow=c(1,2))
plot(cluster, main=paste(cluster$n, "clustering points"), pch = 20)
plot(unif, main=paste(cluster$n, "clustering points"), pch = 20)
```

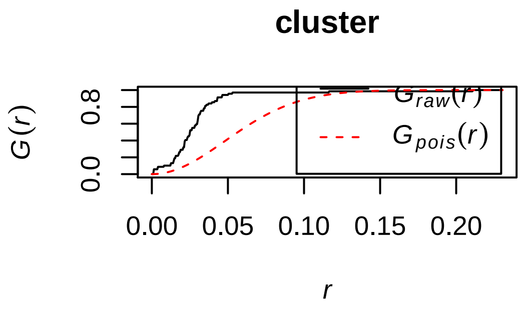


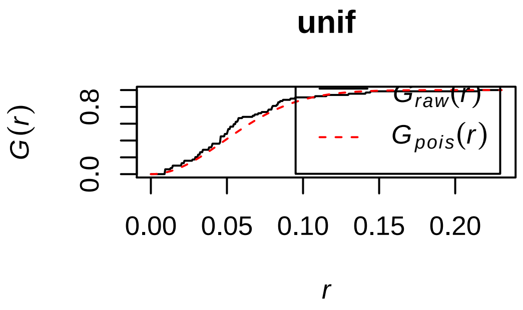


```
par(mfrow=c(2,2))
Ecdf(nndist(cluster), main = 'cluster', breaks = 20, col='blue')
Ecdf(nndist(unif), main = 'uniform', breaks = 20,col='blue')

Gtest2<-Gest(cluster, correction="none")
plot(Gtest2, main='cluster')
Gtest<-Gest(unif, correction="none")
plot(Gtest, main='unif')
```



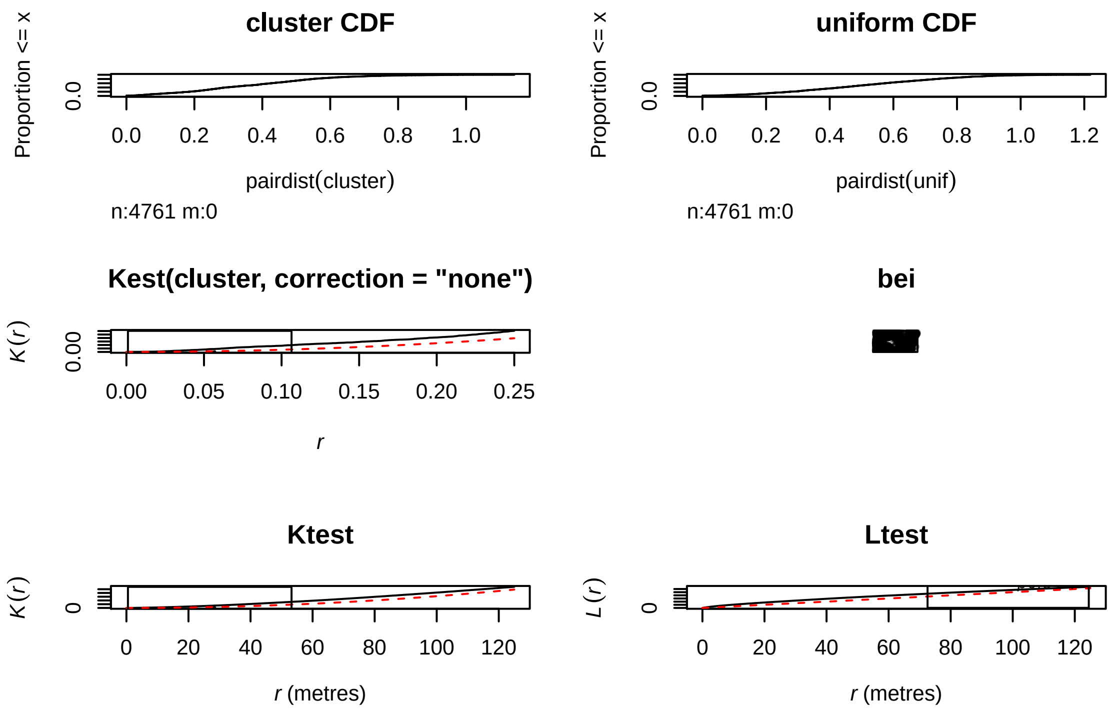






```
#par(mfrow=c(1,2))
#pepper=runifpoint(10000)
#Ecdf(nndist(pepper, cluster),main='empty space function', breaks=20)
#plot(Fest(cluster, correction='none'))

par(mfrow=c(3,2))
Ecdf(pairdist(cluster), main='cluster CDF' )
Ecdf(pairdist(unif), main='uniform CDF')
```

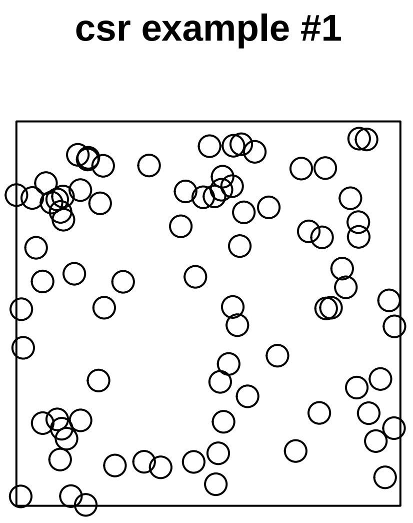
```r
plot(Kest(cluster, correction='none'))

plot(bei, cex=0.5)
Ktest<-Kest(bei, correction="none")
plot(Ktest)

Ltest<-Lest(bei, correction="none")
plot(Ltest)
```



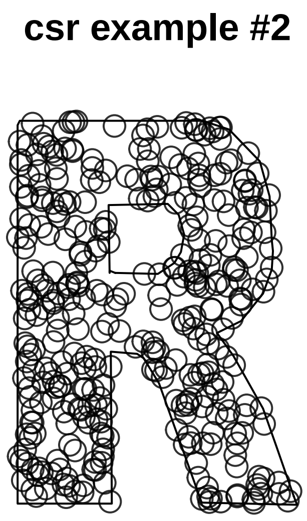
Complete spatial randomness

- yardstick, reference model that observed point patterns could be compared with, i.e., null hypothesis
- homogeneous Poisson point process}
- basic properties:
  - the number of points falling in any region A has a Poisson distribution with mean $\lambda|A|$
  - given that there are n points inside region A, the locations of these points are i.i.d. and uniformly distributed inside A
  - the contents of two disjoint regions A and B are independent

```r
plot(rpoispp(100), main='csr example #1', cex=1.5)
```



```r
#plot(rpoispp(100), main='csr example #2')
data(letterR)
#plot(rpoispp(100, win = letterR), main='csr example #3')
plot(rpoispp(100, win = letterR), main ='csr example #2', cex=1.5 )
```

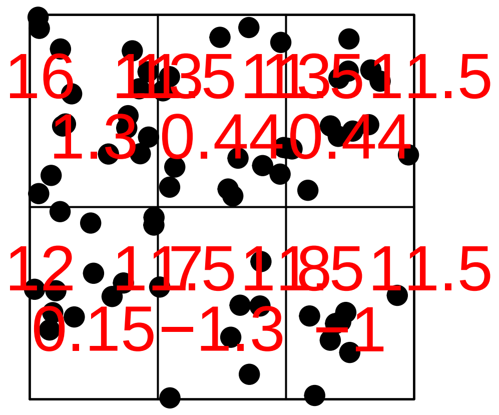


Quadrant testing for CSR

- partition study area A into L sub-regions (quadrats), $A_1, \ldots, A_L$
- count number of events $n(A_l)$ in each sub-region A_l
- Under the null hypothesis of CSR, the $n(A_l)$ are i.i.d. Poisson random variables with the same expected value -The Pearson χ^2 goodness-of-fit test can be used

```r
M <- quadrat.test(unif, nx = 3, ny = 2)
plot(unif, cex=2, pch=20)
plot(M, add = TRUE, cex = 2, col='red')
```

unif

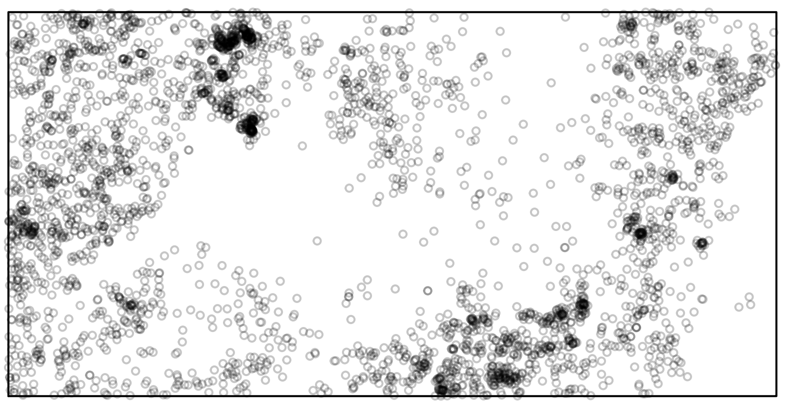


Distance-based function under CSR

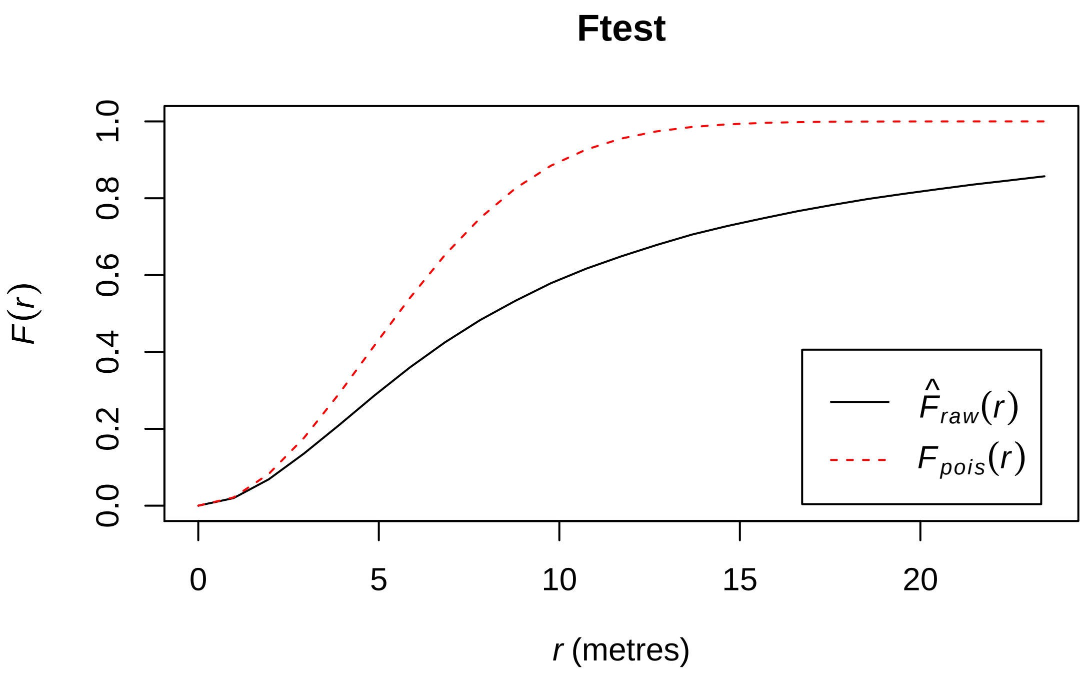
- $G(d) = 1 - \exp\{-\lambda\pi d^2\}$
- $F(d) = 1 - \exp\{-\lambda\pi d^2\}$
- $K(d) = \exp\{-\lambda\pi d^2\}$; a commonly-used transformation of K is the L-function: $L(d) = \sqrt{\frac{K(d)}{\pi}} = d$

```
plot(bei, cex=0.5)
```

bei



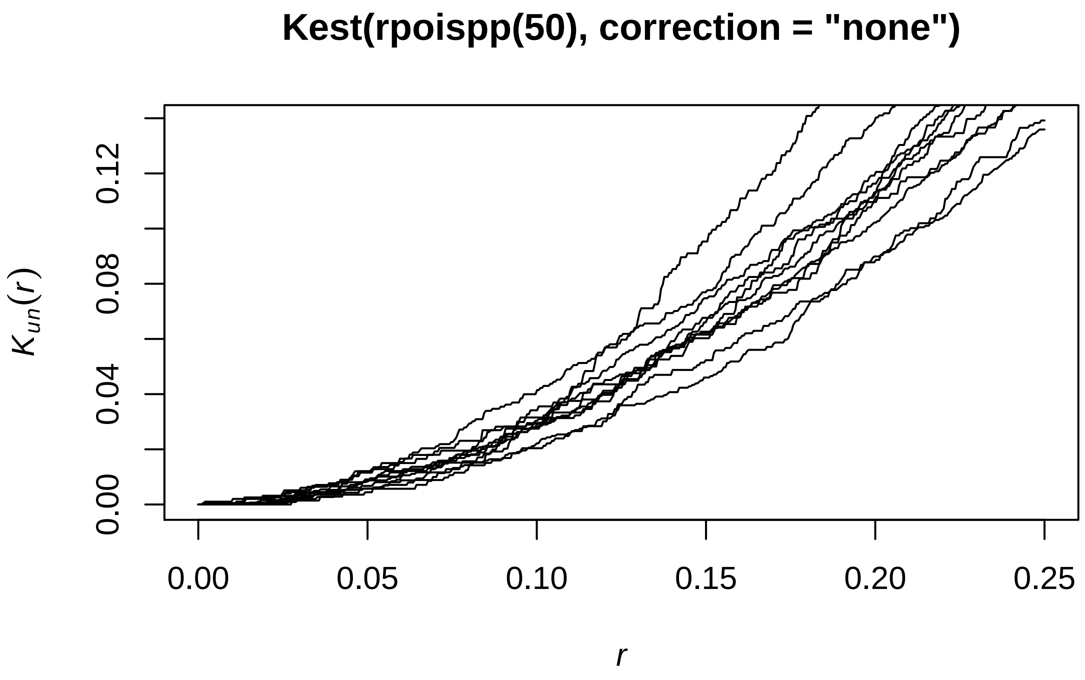
```
Ftest<-Fest(bei, correction="none")
plot(Ftest)
```



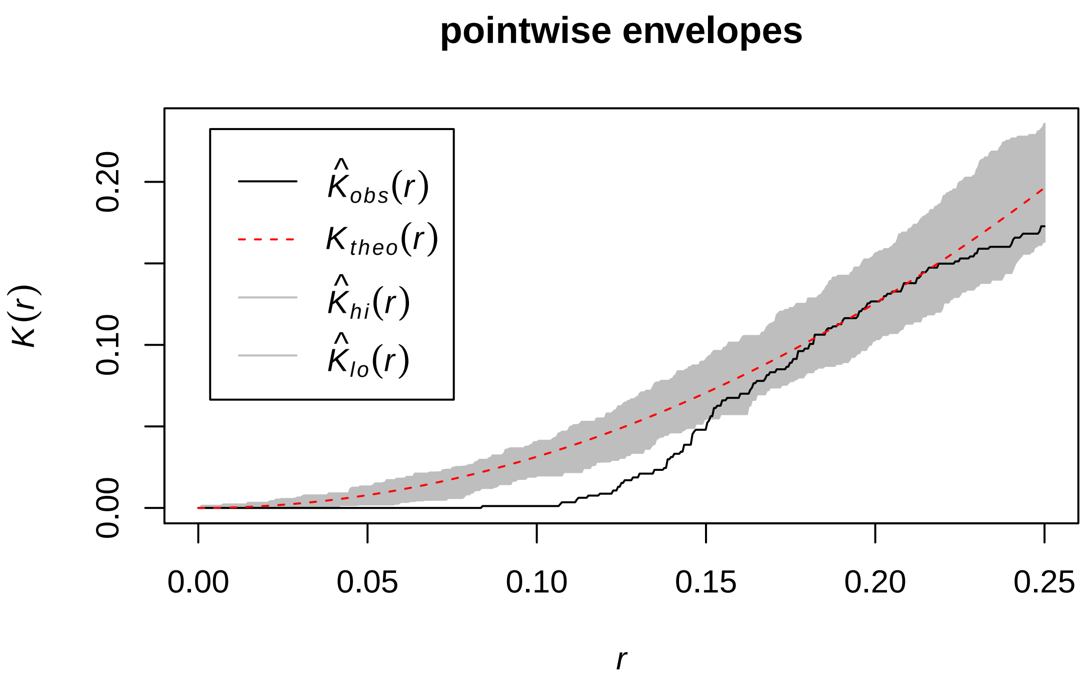
Monte Carlo test

- because of random variability, we will never obtain perfect agreement between sample functions (say the K function) with theoretical functions (the theoretical K functions), even with a completely random pattern

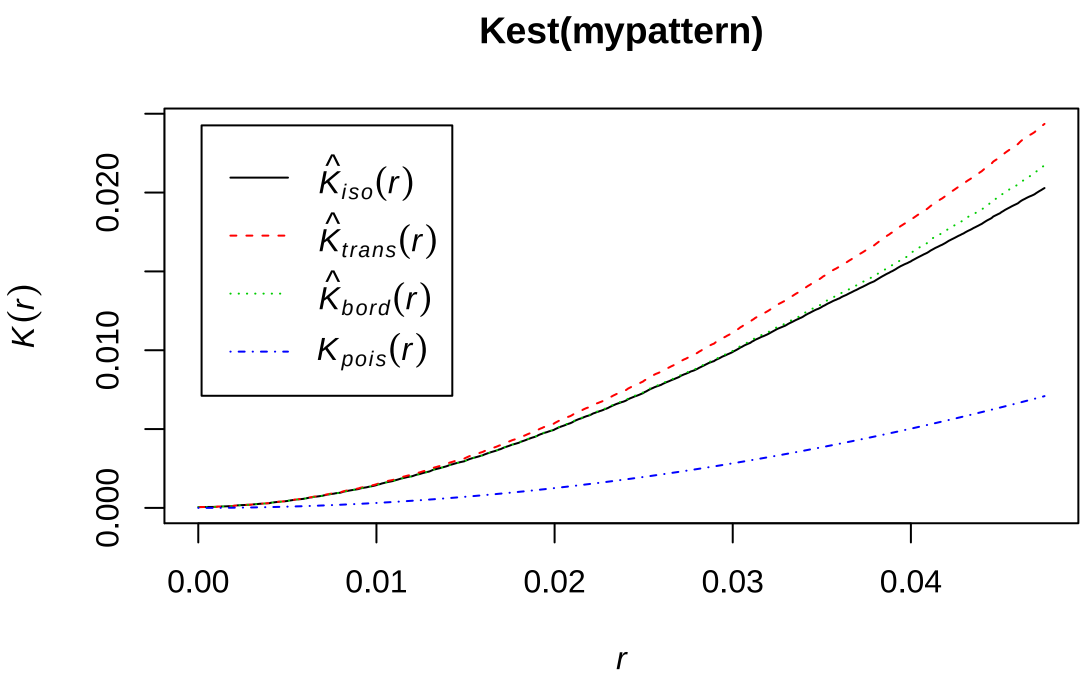
```
plot(Kest(rpoispp(50), correction='none'),un~r)
for(i in 1:10){
plot(Kest(rpoispp(50), correction='none'),un~r, add=TRUE)
}
```



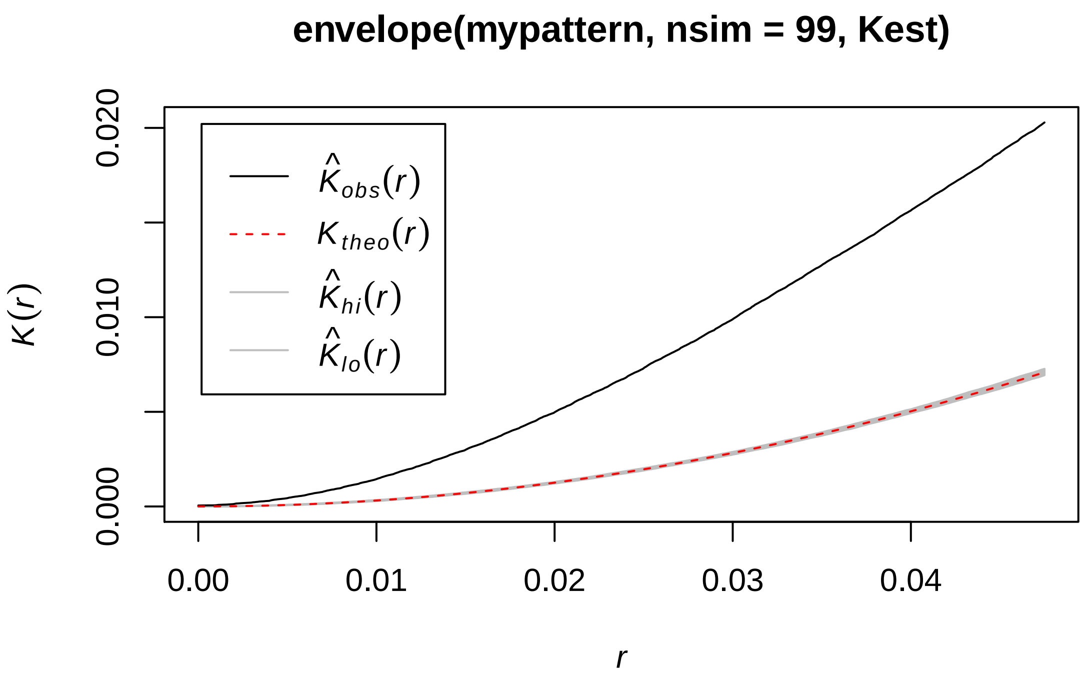
```r
data(cells)
monte2<-envelope(cells, Kest, nsim = 39, rank = 1)
plot(monte2, main = "pointwise envelopes")
```



```r
# Now, test on the Lubbock house data

#Read the HouseSaleLubbock shapefile into R
#Ripley's K-function
plot(Kest(mypattern))
```



```r
#envelop of K-function
plot(envelope(mypattern,nsim=99,Kest))
```


envelope(mypattern, nsim = 99, Kest)
$\hat{K}_{obs}(r)$
$K_{theo}(r)$
$\hat{K}_{hi}(r)$
$\hat{K}_{lo}(r)$
$K(r)$
0.000
0.010
0.020
0.00
0.01
0.02
0.03
0.04
r