SpatialPointAnalysis

Kostas Goulias

2/27/2018

Introduction

In this session, we explore different types of spatial points data analysis. Spatial points in this context will be considered as “events” that happen in space. The idea is to figure out if a given database of points is the result of a stochastic process that generated the data.

The occurrence of each point may be independent of its neighboring points. For example, it may be the result of an event that is due to pure chance with no systematic underlying process generating the points.

We "test" this by comparing the data we have with “simulated” spatial distributions of points created using a theoretical distribution. If the two spatial distributions (empirical from the data we have) and theoretical (from a random generation process we simulate) are not close enough we conclude that we have something different than random. In essence we create at different distances from each point function values and compare them.

We measure this with three different functions (explained below).

The plan: Today (Feb 27) we review one prototypical example of point patterns using functions. On Thursday (March 1) we review Spatial Kernel density. Next week we explore Variograms and Kriging.

Preparatory Tasks

First the libraries we need.

```
library(dplyr)       ## Data manipulation

## Warning: package 'dplyr' was built under R version 3.4.2

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(maps)         ## Projections
library(maptools)     ## Data management

## Loading required package: sp
```

```
## Checking rgeos availability: TRUE

library(sp)           ## Data management
library(spdep)        ## Spatial autocorrelation

## Warning: package 'spdep' was built under R version 3.4.3

## Loading required package: Matrix

## Loading required package: spData

## Warning: package 'spData' was built under R version 3.4.3

## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge')`

library(gstat)        ## Geostatistics
library(splancs)      ## Kernel Density

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

##
## Attaching package: 'splancs'

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

library(spatstat)     ## Geostatistics

## Warning: package 'spatstat' was built under R version 3.4.2

## Loading required package: spatstat.data

## Warning: package 'spatstat.data' was built under R version 3.4.2

## Loading required package: nlme

##
## Attaching package: 'nlme'

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

## Loading required package: rpart

##
## spatstat 1.53-2       (nickname: 'Quantum Entanglement')
## For an introduction to spatstat, type 'beginner'

## Warning in strptime(x, "%Y-%m-%d-%H-%M-%OS", tz = tz): unknown timezone
## 'zone/tz/2018c.1.0/zoneinfo/America/Los_Angeles'

##
## Note: spatstat version 1.53-2 is out of date by more than 4 months; we recommend upgrading to the lates
t version.

##
## Attaching package: 'spatstat'

## The following object is masked from 'package:gstat':
##
##     idw
```

```
library(RColorBrewer) ## Visualization
library(classInt)     ## Class intervals
library(spgwr)        ## GWR (not sure we need this for now)

## Warning: package 'spgwr' was built under R version 3.4.2

## NOTE: This package does not constitute approval of GWR
## as a method of spatial analysis; see example(gwr)

library(lattice)      ## Data visualization

##
## Attaching package: 'lattice'

## The following object is masked from 'package:spatstat':
##
##     panel.histogram
```

Then the data. The file h2kb.rds is on gauchospace

```
my2k = readRDS('~/Documents/COURSES UCSB/Course Winter 2018/California/h2kb.rds')
class(my2k)

## [1] "data.frame"

summary(my2k)

##        Z1                 Z2              XCORD            YCORD
##  Min.   :   0.000   Min.   : 0.000   Min.   :-124.2   Min.   :32.56
##  1st Qu.:   5.524   1st Qu.: 2.000   1st Qu.:-121.9   1st Qu.:34.07
##  Median :  32.154   Median : 6.000   Median :-119.9   Median :36.59
##  Mean   :  68.084   Mean   : 8.414   Mean   :-120.0   Mean   :36.16
##  3rd Qu.:  78.969   3rd Qu.:12.000   3rd Qu.:-118.2   3rd Qu.:37.84
##  Max.   :1383.520   Max.   :95.000   Max.   :-114.4   Max.   :41.95

data <- my2k
```

This is just a data frame like the files we created before for data analysis. It contains 2,000 locations in California with their X and Y coordinates and the value of two variables Z1 and Z2. In this example we are going to use only the points (XCORD and YCORD) and not the variables Z1 and Z2.

I also rename the my2k to data to eliminate too many names in the code.

We need to make this data frame into a spatial object

Create a matrix of coordinates from the point coordinates of the data file my2k (renamed data)

```
sp_point <- cbind(data$XCORD, data$YCORD)
colnames(sp_point) <- c("LONG","LAT")
head(sp_point)

##           LONG      LAT
## [1,] -118.1920 34.06835
## [2,] -121.4452 38.63569
## [3,] -119.6622 36.34007
## [4,] -122.6362 38.22512
## [5,] -119.0181 35.32511
## [6,] -122.2539 37.85491
```

Next we create the projection we want to use for our data

A summary of Coordinate Referenece Systems is in:
https://www.nceas.ucsb.edu/~frazier/RSpatialGuides/OverviewCoordinateReferenceSystems.pdf

And the CRS in R: http://rspatial.org/spatial/rst/6-crs.html

In our case we use UTM zone 10 from WGS84

```
proj <- CRS("+proj=utm +zone=10 +datum=WGS84")
summary(proj)

## Length  Class   Mode
##      1    CRS     S4
```

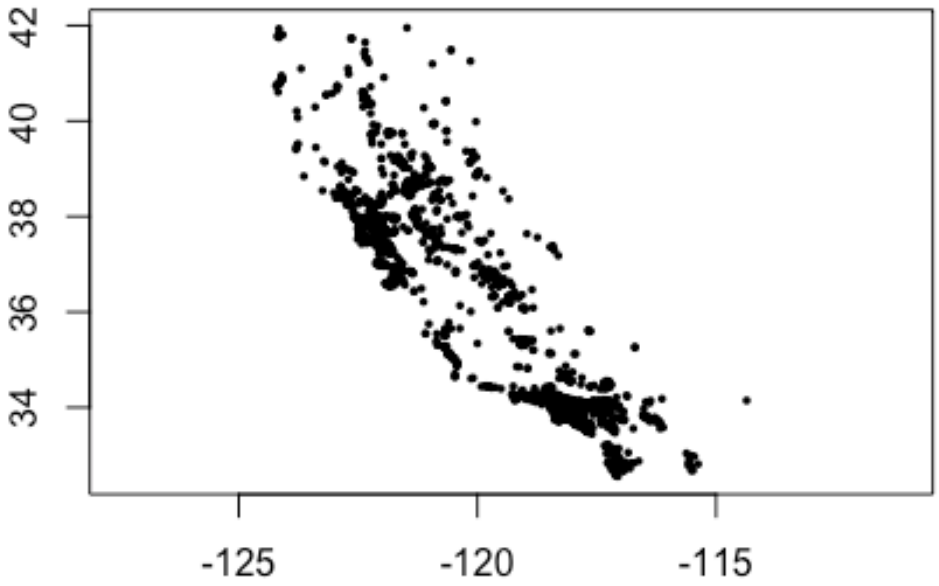
The above just tells the type of object that we created.

Then, we use the function SpatialPointsDataFrame to "transform" the flat data into a georeferenced database

Create spatial object with the data we have and a bounding box around the data and then plot the points

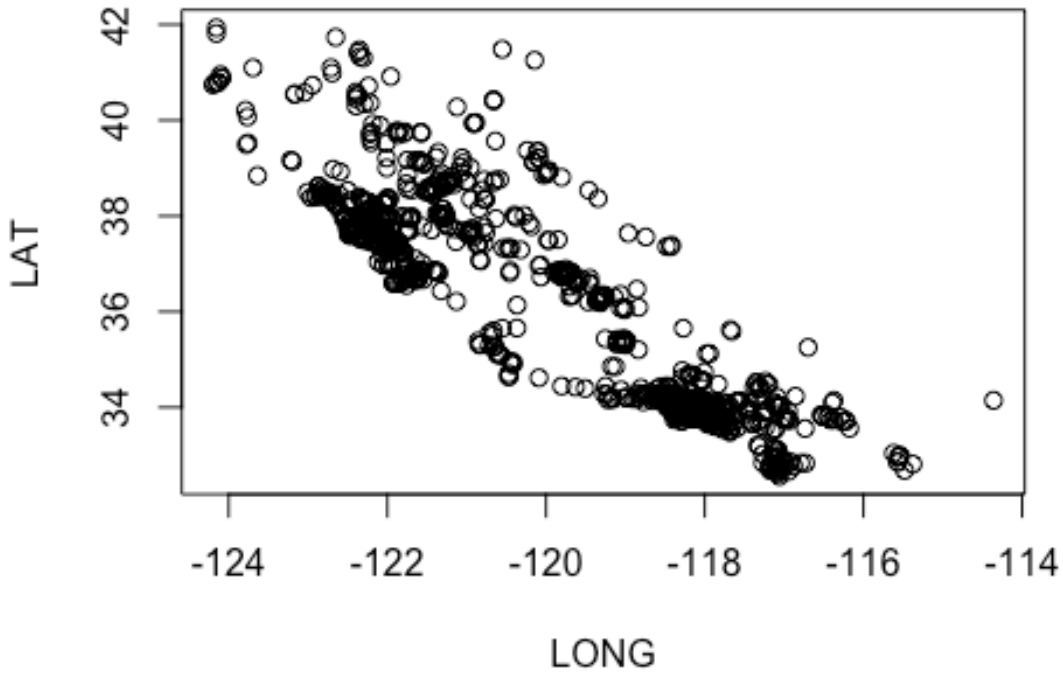
```
data.sp <- SpatialPointsDataFrame(coords=sp_point,data,proj4string=proj)
summary(data.sp)

## Object of class SpatialPointsDataFrame
## Coordinates:
##             min        max
## LONG -124.21882 -114.36030
## LAT    32.55885   41.95175
## Is projected: TRUE
## proj4string :
## [+proj=utm +zone=10 +datum=WGS84 +ellps=WGS84 +towgs84=0,0,0]
## Number of points: 2000
## Data attributes:
##        Z1                 Z2             XCORD            YCORD
##  Min.   :   0.000   Min.   : 0.000   Min.   :-124.2   Min.   :32.56
##  1st Qu.:   5.524   1st Qu.: 2.000   1st Qu.:-121.9   1st Qu.:34.07
##  Median :  32.154   Median : 6.000   Median :-119.9   Median :36.59
##  Mean   :  68.084   Mean   : 8.414   Mean   :-120.0   Mean   :36.16
##  3rd Qu.:  78.969   3rd Qu.:12.000   3rd Qu.:-118.2   3rd Qu.:37.84
##  Max.   :1383.520   Max.   :95.000   Max.   :-114.4   Max.   :41.95

bbox(data.sp)

##             min        max
## LONG -124.21882 -114.36030
## LAT    32.55885   41.95175
```

```r
par(mar=c(2,2,0.2,0.2))

plot(data.sp,pch=16, cex=.5, axes=T)
```



Ilustration of different spatial patterns

```
## Take subsample of CA dataset

data <- data[sample(1000,replace=F),]
names(data)

## [1] "Z1"    "Z2"    "XCORD" "YCORD"

head(data)

##             Z1 Z2     XCORD    YCORD
## 7203  10.68368 11 -120.8461 35.31522
## 3589  51.12292 15 -119.8202 36.74802
## 2730  38.68535  4 -121.1583 38.65764
## 33311 51.04580 40 -119.0168 35.31622
## 31219  0.00000  0 -122.1695 37.75312
## 1104  54.05018 16 -117.8788 34.00317

dim(data)

## [1] 1000    4

## Create a matrix of coordinates

sp_point <- matrix(NA, nrow=nrow(data),ncol=2)
sp_point[,1] <- data$XCORD
sp_point[,2] <- data$YCORD
colnames(sp_point) <- c("LONG","LAT")

plot(sp_point)
```




```
class(sp_point)

## [1] "matrix"
```

Create some auxiliary points to use later

Create points that are uniformly distributed in space

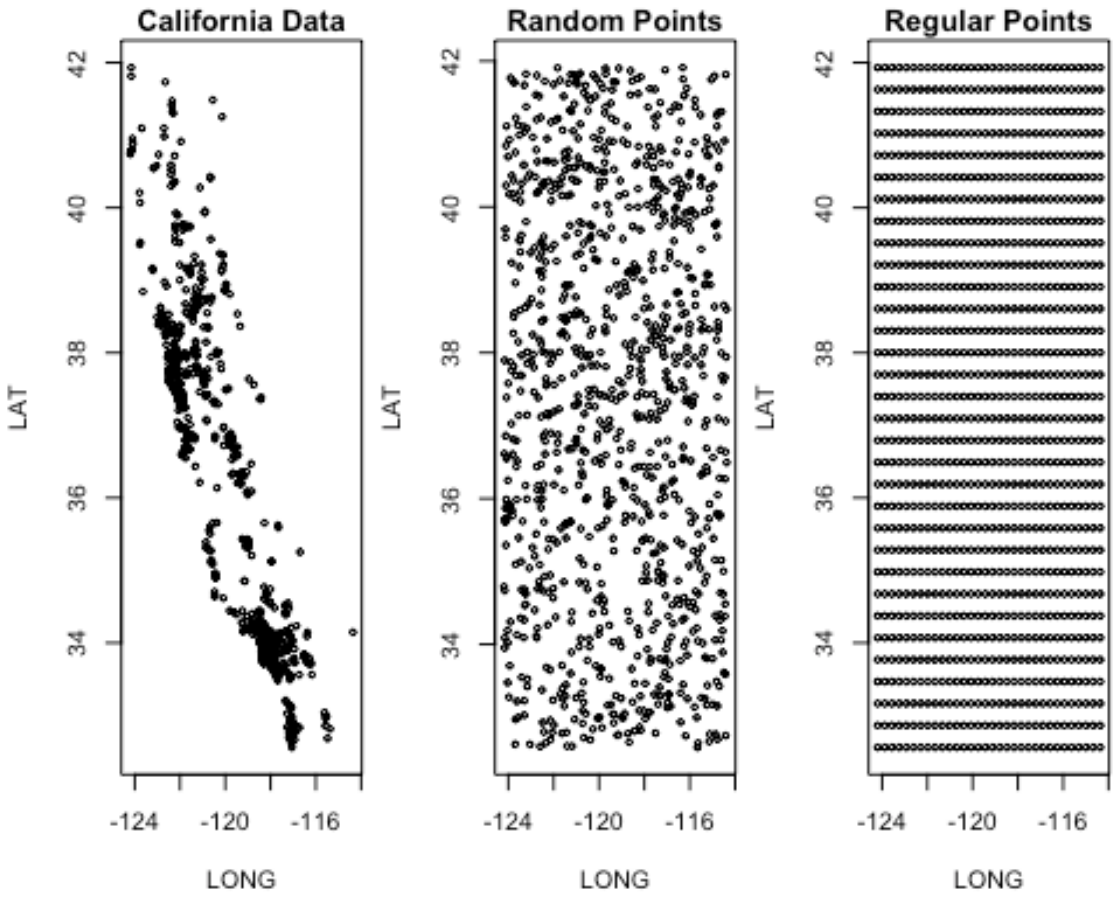
```
## Random points

u.x <- runif(n=nrow(sp_point), min=bbox(sp_point)[1,1], max=bbox(sp_point)[1,
2])
u.y <- runif(n=nrow(sp_point), min=bbox(sp_point)[2,1], max=bbox(sp_point)[2,
2])
```

Create points that are equispaced distributed in space

```
## Regular points

r.x <- seq(from=min(sp_point[,1]),to=max(sp_point[,1]),length=sqrt(nrow(sp_po
int)))
r.y <- seq(from=min(sp_point[,2]),to=max(sp_point[,2]),length=sqrt(nrow(sp_po
int)))
r.x <- jitter(rep(r.x,length(r.x)),.001)
r.y <- jitter(rep(r.y,each=length(r.y)),.001)
```

Plot the points to compare visually

```
## Plot the points

par(mfrow=c(1,3),mar=c(4,4,1.5,0.5))
plot(x=sp_point[,1],y=sp_point[,2],main="California Data", xlab="LONG",ylab="
LAT",cex=.5)
plot(x=u.x,y=u.y,main="Random Points", xlab="LONG",ylab="LAT",cex=.5)
plot(x=r.x,y=r.y,main="Regular Points", xlab="LONG",ylab="LAT",cex=.5)
```



Create function of conversion between kilometers and degrees. We could do this in different ways but I liked the Zhukov function creation.

Function km2d (km to degrees)

```
km2d <- function(km){
  out <- (km/1.852)/60
  return(out)
}
```

Function d2km (degrees to km)

```
d2km <- function(d){
  out <- d*60*1.852
  return(out)
}
```

Basic Definitions and Data Exploration

Intensity is the average density of points (expected number of points per unit area).

Intensity may be constant in space showing a uniform distribution of points in space or may vary from location to location also called non-uniform or inhomogeneous.

From CSIRO2008 (Adrian Baddeley):

The observed point pattern x is a realisation of a random point process X in a two-dimensional space.

A point process is simply a random set of points. Random is the number of points and the locations of the points.

Our goal is to estimate parameters of the distribution of X for the properties of the data we observe (x).

We assume that the point process X extends throughout 2-D space, but it is observed only inside a region W. This is the "sampling window".

In the example here the sampling window covers the State Of California.

Our data consist of an unordered set $x=\{x1,...,xn\}$, $xi \in W$, $n \geq 0$ of points xi in W.

The window W is fixed and known. Usually our goal is inference about the parameters of the distribution X.

If the point process X is homogeneous, then for any sub-region B of the two-dimensional space here, the expected number of points in B is proportional to the area of B.

This means

Expected # $(X \cap B) = \lambda$ area(B)

and the constant of proportionality λ is the intensity.

Intensity units are numbers per unit area. If we know that a point process is homogeneous, then the empirical density of points, $\lambda = n(x)$ area(W) is an unbiased estimator of the true intensity λ.

We will compare our empirical distribution (derived from the data we observe) with a distribution of data for which we know its properties.

The basic 'reference' or 'benchmark' model of a point process is the uniform Poisson point process in the plane with intensity λ.

This is called Complete Spatial Randomness (CSR).

The basic properties CSR:

1. The number of points falling in any region A has a Poisson distribution with mean λ area(A).
2. Given that there are n points inside region A, the locations of these points are i.i.d. and uniformly distributed inside A.
3. The contents of two disjoint regions A and B are independent.

The uniform Poisson process is often the 'null model' in an analysis. In the literature and software people focus on checking if their data conform to a uniform Poisson process.

The homogeneous Poisson process of intensity $\lambda > 0$ has the properties:

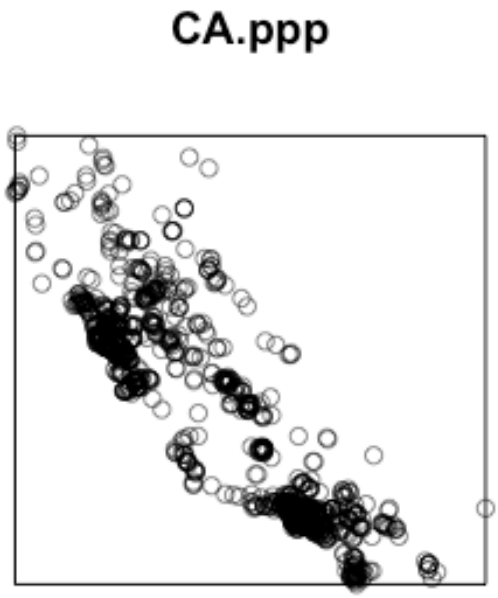
1. the number of points falling in any region B is a Poisson random variable.
2. The expected number of points falling in B is = $\lambda \cdot$ area(B).
3. If B1,B2 are disjoint sets then the number of points in B1 and number of points in B2 are independent random variables.

In CSR points are independent of each other and have the same propensity to be found at any location. When we draw points from this distribution does not mean they will end up at the same place each time we draw points.

Some more preparatory tasks

Create a planar point pattern (ppp in R) of the data we analyze

```
CA.ppp <- ppp(x=sp_point[,1],y=sp_point[,2],window=owin(bbox(sp_point)[1,],bb
ox(sp_point)[2,]))
plot(CA.ppp)
```

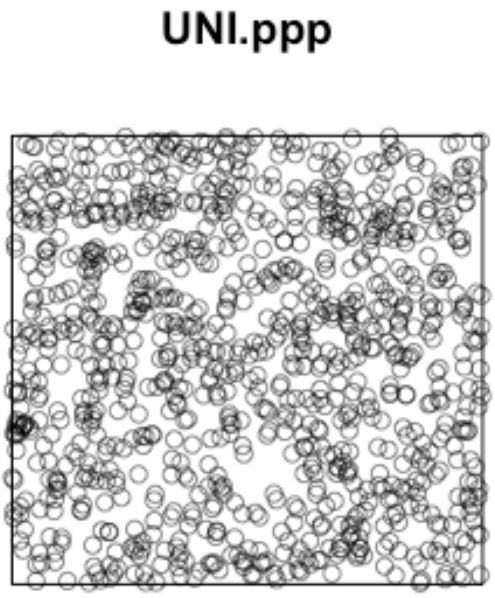



```
summary(CA.ppp)
```

```
## Planar point pattern:  1000 points
## Average intensity 10.84524 points per square unit
##
## Coordinates are given to 6 decimal places
##
## Window: rectangle = [-124.19737, -114.3603] x [32.55885, 41.93221] units
## Window area = 92.2064 square units
```

Create a planar point pattern of the uniform data

```
UNI.ppp <- ppp(x=u.x,y=u.y,window=owin(bbox(sp_point)[1,],bbox(sp_point)[2,])
)
plot(UNI.ppp)
```

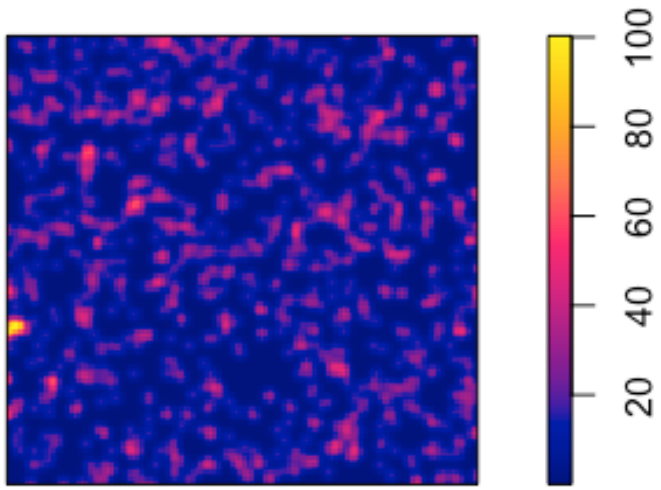



```
summary(UNI.ppp)
```

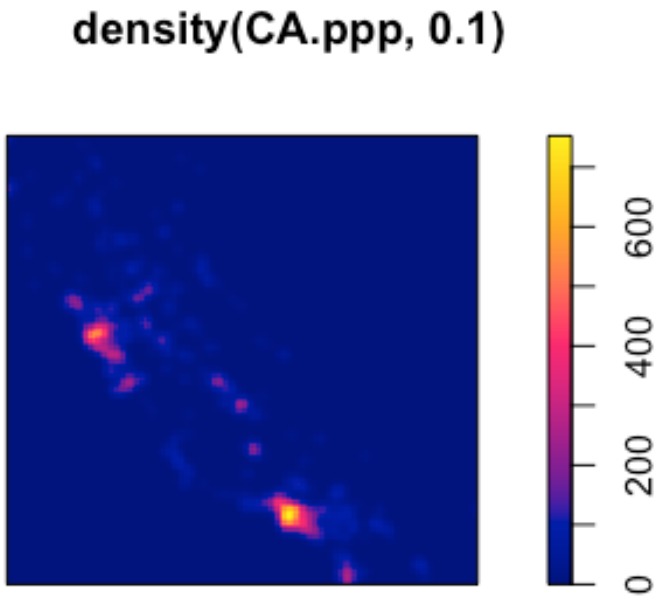
```
## Planar point pattern:  1000 points
## Average intensity 10.84524 points per square unit
##
## Coordinates are given to 6 decimal places
##
## Window: rectangle = [-124.19737, -114.3603] x [32.55885, 41.93221] units
## Window area = 92.2064 square units
```

An example of a forthcoming analysis with kernel

```
plot(density(UNI.ppp, 0.1))
```




```
plot(density(CA.ppp, 0.1))
```


density(CA.ppp, 0.1)
0
200
400
600

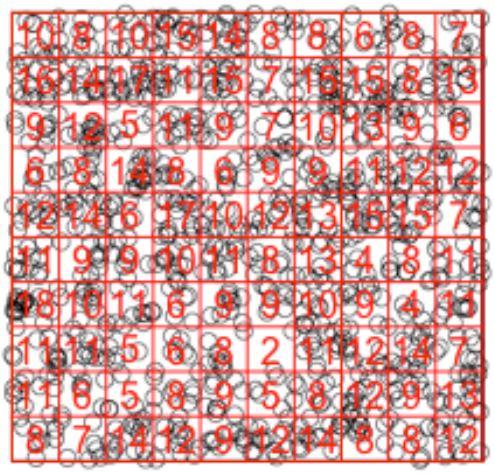
Quadrat Analysis (digression)

We divide the enitre region in rectangles called 'quadrats' of the same size. then, we count the number of points in each rectangle.

```
UNI.q <- quadratcount(UNI.ppp, nx=10, ny=10)
UNI.q
```

```
##              x
## y             [-124,-123) [-123,-122) [-122,-121) [-121,-120) [-120,-119)
##   [41,41.9]            10           8          10          15          14
##   [40.1,41)            16          14          17          11          15
##   [39.1,40.1)           9          12           5          11           9
##   [38.2,39.1)           6           8          14           8           6
##   [37.2,38.2)          12          14           6          17          10
##   [36.3,37.2)          11           9           9          10          11
##   [35.4,36.3)          18          10          11           6           9
##   [34.4,35.4)          11          11           5           6           8
##   [33.5,34.4)          11           6           5           8           9
##   [32.6,33.5)           8           7          14          12           9
##              x
## y             [-119,-118) [-118,-117) [-117,-116) [-116,-115) [-115,-114]
##   [41,41.9]             8           8           6           8           7
##   [40.1,41)             7          15          15           8          13
##   [39.1,40.1)           7          10          13           9           6
##   [38.2,39.1)           9           9          11          12          12
##   [37.2,38.2)          12          13          15          15           7
##   [36.3,37.2)           8          13           4           8          11
##   [35.4,36.3)           9          10           9           4          11
##   [34.4,35.4)           2          11          12          14           7
##   [33.5,34.4)           5           8          12           9          13
##   [32.6,33.5)          12          14           8           8          12
```

```
plot(UNI.ppp)
plot (UNI.q, add=TRUE, cex=1, col="red")
```



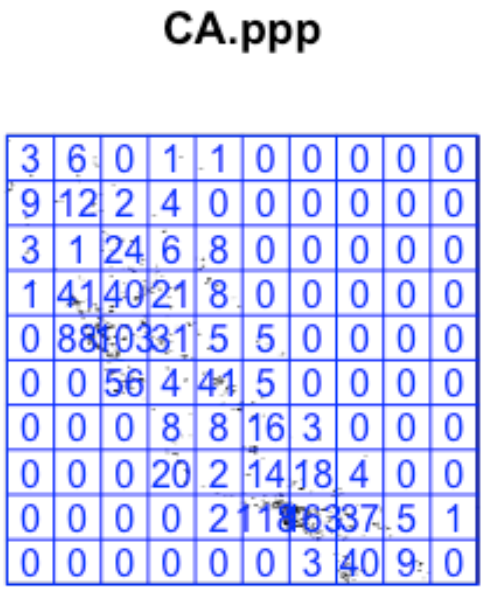

```
CA.q <- quadratcount(CA.ppp, nx=10, ny=10)
CA.q
```

```
##              x
## y             [-124,-123) [-123,-122) [-122,-121) [-121,-120) [-120,-119)
##   [41,41.9]             3           6           0           1           1
##   [40.1,41)             9          12           2           4           0
##   [39.1,40.1)           3           1          24           6           8
##   [38.2,39.1)           1          41          40          21           8
##   [37.2,38.2)           0          88         103          31           5
##   [36.3,37.2)           0           0          56           4          41
##   [35.4,36.3)           0           0           0           8           8
##   [34.4,35.4)           0           0           0          20           2
##   [33.5,34.4)           0           0           0           0           2
##   [32.6,33.5)           0           0           0           0           0
##              x
## y             [-119,-118) [-118,-117) [-117,-116) [-116,-115) [-115,-114]
##   [41,41.9]             0           0           0           0           0
##   [40.1,41)             0           0           0           0           0
##   [39.1,40.1)           0           0           0           0           0
##   [38.2,39.1)           0           0           0           0           0
##   [37.2,38.2)           5           0           0           0           0
##   [36.3,37.2)           5           0           0           0           0
```

```
##   [35.4,36.3)          16           3            0           0           0
##   [34.4,35.4)          14          18            4           0           0
##   [33.5,34.4)         118         163           37           5           1
##   [32.6,33.5)           0           3           40           9           0
```

Notice how the values are now along a diagonal band of XY coordinates

```
plot(CA.ppp, cex=0.03)
plot (CA.q, add=TRUE, cex=1, col="blue")
```



Nearest neighbor and the G function

Definitions From Chris Funk Slides Geog 210C

- Definition
  - Consider a set of **N** points $\{u_1 \quad u_i \quad u_N\}$ in a K-dimensional (geographical or other) space. The distance matrix **D** is a square (NxN) matrix containing the distances $\{d(\mathbf{u}_i,\mathbf{u}_j);\ i = 1 \quad N;\ j = 1 \quad N\}$ between all NxN possible pairs of points in the set

$\mathbf{u}_i$	$\mathbf{u}_1$	$\mathbf{u}_2$	$\mathbf{u}_3$	$\mathbf{u}_4$	$\mathbf{u}_5$
x_i	x_1	x_2	x_3	x_4	x_5
y_i	y_1	y_2	y_3	y_4	y_5

  - by convention, u_1 is the coordinate vector of the 1st point in the set (1st entry in data file)

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} & d_{15} \\ d_{21} & d_{22} & d_{23} & d_{24} & d_{25} \\ d_{31} & d_{32} & d_{33} & d_{34} & d_{35} \\ d_{41} & d_{42} & d_{43} & d_{44} & d_{45} \\ d_{51} & d_{52} & d_{53} & d_{54} & d_{55} \end{bmatrix} = \begin{bmatrix} 0 & d_{12} & d_{13} & d_{14} & d_{15} \\ d_{12} & 0 & d_{23} & d_{24} & d_{25} \\ d_{13} & d_{23} & 0 & d_{34} & d_{35} \\ d_{14} & d_{24} & d_{34} & 0 & d_{45} \\ d_{15} & d_{25} & d_{35} & d_{45} & 0 \end{bmatrix} = [d_{ij}]$$

- i-th row (or column) contains distances between i-th point u_i and all others (including itself)
- **D** is symmetric with zeros along its diagonal

Event-to-event distance

- Distance d_{ij} between event at location $\mathbf{u}_i$ and another event at location $\mathbf{u}_j$

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Point-to-event distance

- Distance d_{pj} between a randomly chosen point at location $\mathbf{t}_p$ and an event at location $\mathbf{u}_j$:

$$\tilde{d}_{pj} = \sqrt{(\tilde{x}_p - x_j)^2 + (\tilde{y}_p - y_j)^2}$$

Event-to-nearest-event distance

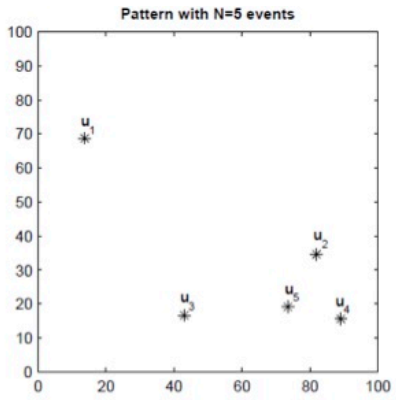
- Distance $d_{min}(\mathbf{u}_i)$ between an event at location u_i and its *nearest neighbor* event:

$$d_{min}(\mathbf{u}_i) = \min\{d_{\substack{ij \\ j \neq i}}, j = 1, \ldots, N\}$$

Point-to-nearest-event distance

- Distance $d_{min}(\mathbf{t}_p)$ between a randomly chosen point at location $\mathbf{t}_p$ and its nearest neighbor event:

$$\tilde{d}_{min}(\mathbf{t}_p) = \min\{\tilde{d}_{pj}, j = 1, \ldots, N\}$$



0.00	76.24	**59.81**	92.21	77.70
76.24	0.00	42.83	20.35	**17.62**
59.81	42.83	0.00	46.03	**30.58**
92.21	20.35	46.03	0.00	**15.94**
77.70	17.62	30.58	**15.94**	0.00

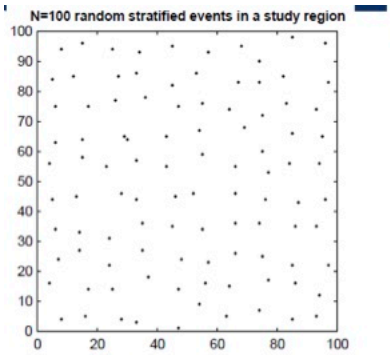
Distance matrix

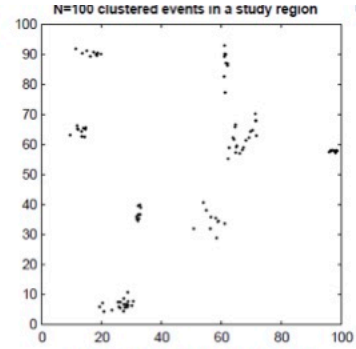
e.g., $\boxed{59.81} = d_{min}(\mathbf{u}_1)$, $\boxed{17.62} = d_{min}(\mathbf{u}_2)$

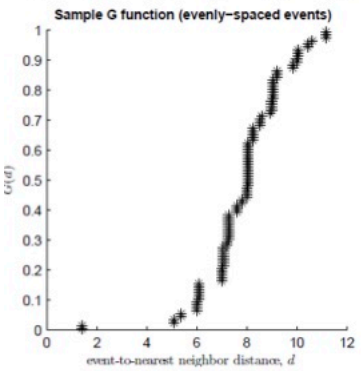
- Some events might be nearest neighbors of each other: e.g., $\mathbf{u}_4$,$\mathbf{u}_5$, or have same nearest neighbor: e.g., $\mathbf{u}_2$, $\mathbf{u}_3$, $\mathbf{u}_4$ are nearest neighbors of $\mathbf{u}_5$

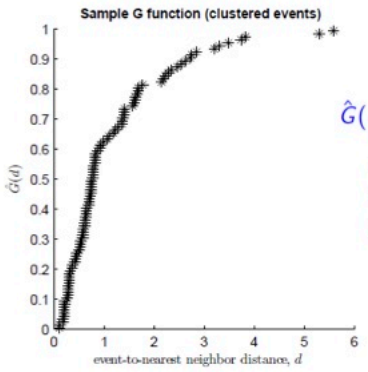
Mean nearest neighbor distance

- Average of all $d_{min}(\mathbf{u}_i)$ values: $\overline{d_{min}} = \frac{1}{N}\sum_{i=1}^{N} d_{min}(\mathbf{u}_i)$
- Drawback: single number does not suffice to describe point pattern









$$\hat{G}(d) = \frac{\#\{d_{min}(\mathbf{u}_i) \le d, i = 1, \ldots, N\}}{N}$$

G_{hat}=Cumulative distribution function (CDF) of all N event-to-nearest-event distances

- for evenly-spaced events, G(d) rises gradually up to the distance at which most events are spaced, and then increases rapidly
- for clustered events, G(d) rises rapidly at short distances, and then levels off at larger d-values

Example of the Empirical G function values from a set of points that are spatially distributed according to the Uniform Distribution. The uniformly distributed points are Random Points above (earlier)

Nearest neighbor at distance r implies that no other points are within a circle with radius r. If the process generating the data is Poisson, then the probability of finding no points within r is:

$$P = exp(-\lambda * \pi * r^2)$$

The complement of this, i.e., finding at least a point is:

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

This is the function we use as reference.

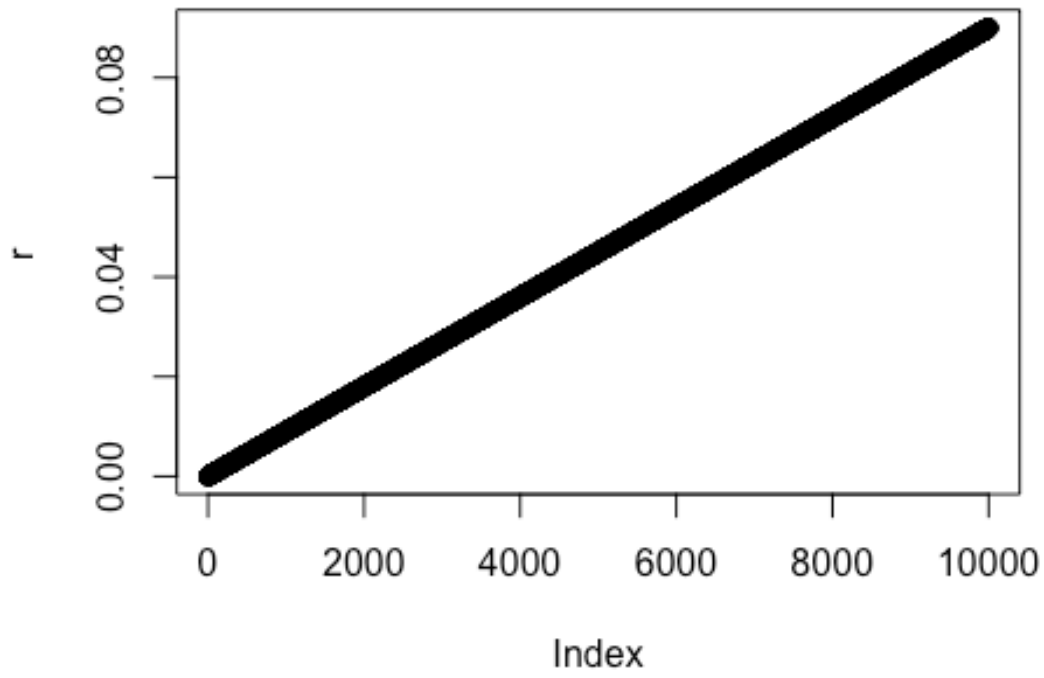
If we generate many different spatial distributions of data in space we get many different values of G. For each distance r we repeat this 99 times. Doing this we create an envelope of possible values of G under the assumption of a uniform Poisson process. Then we also compute the G that comes from our data and decide if our data exhibit clustering.

The function envelope is from package spatstat and creates multiple simulated values of the theoretical G and reports the lowest and highest set of Gs under the theoretcal distribution. In the examples here the theoretical is Complete Spatial Randomness (CSR) and is based on the above equations.

https://www.rdocumentation.org/packages/spatstat/versions/1.55-0/topics/envelope

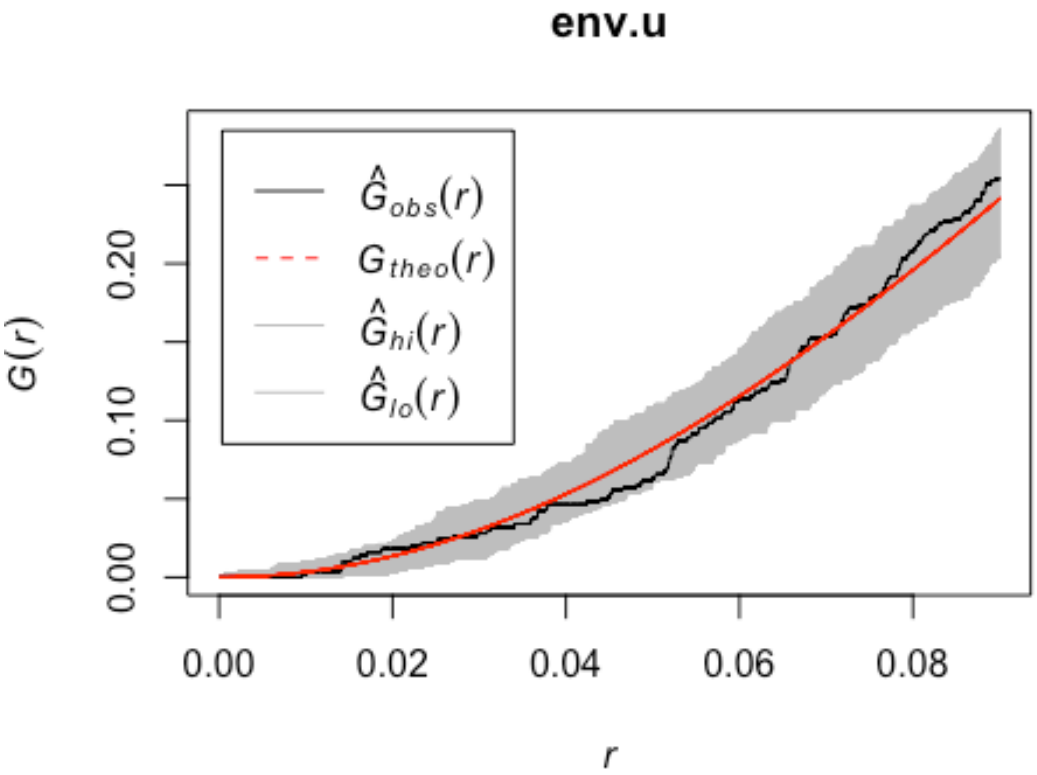
Create a sequence of distances to compute G

```
r <- seq(0,km2d(10),length.out=10000)

plot(r)
```



G-Test: California points vs Uniformly distributed points

```
env.u <- envelope(ppp(x=u.x,y=u.y,window=owin(bbox(sp_point)[1,],bbox(sp_poin
t)[2,])), fun=Gest, r=r, nsim=99, nrank=2)

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

plot(env.u)
```



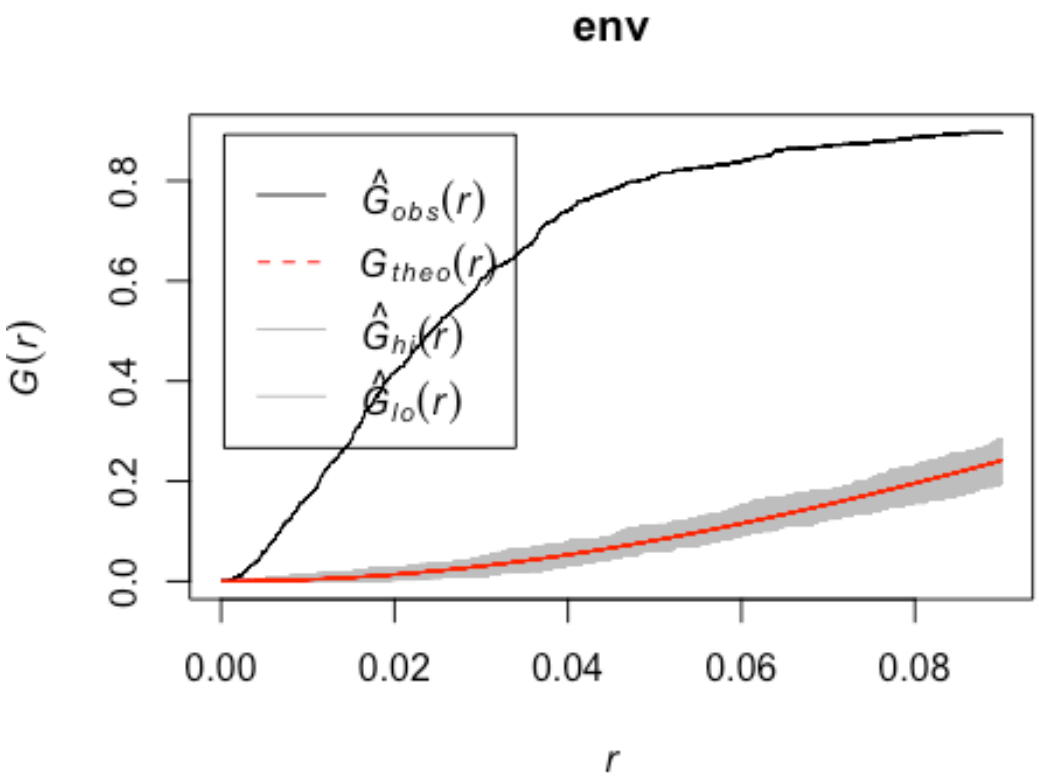

```
## G-Test: California points

r <- seq(0,km2d(10),length.out=10000)

env <- envelope(ppp(x=sp_point[,1],y=sp_point[,2],window=owin(bbox(sp_point)[
1,],bbox(sp_point)[2,])), fun=Gest, r=r, nsim=99, nrank=2)

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

```
summary(env)
```

```
## Pointwise critical envelopes for G(r)
## and observed value for 'ppp(x = sp_point[, 1], y = sp_point[, 2], window =
## owin(bbox(sp_point)[1, '
## Obtained from 99 simulations of CSR
## Alternative: two.sided
## Upper envelope: pointwise 2nd largest of simulated curves
## Lower envelope: pointwise 2nd smallest of simulated curves
## Significance level of Monte Carlo test: 4/100 = 0.04
## Data: ppp(x = sp_point[, 1], y = sp_point[, 2], window =
## owin(bbox(sp_point)[1,
```

SO THIS ENVELOPING CREATES MULTIPLES PATTERNS OF DATA AND THEN COMPUTES THE CRITICAL VALUES IN A DISTRIBUTION AROUND THE MEAN OF THE THEORETICAL PROCESS CSR (Baddeley, 2008)

```
plot(env)
```



The G values from our data are above the randomization envelope. Therefore, we have clustering in our data!

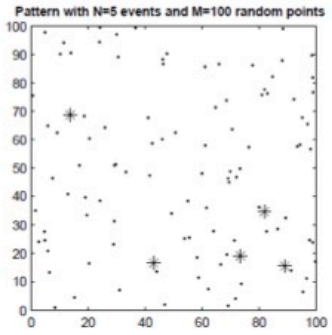
Point-to-Event Distribution and the F function

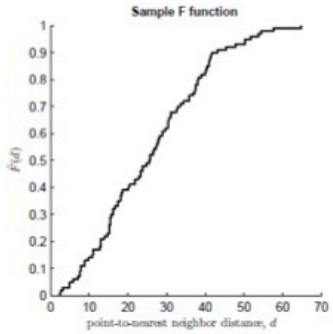
Definitions From Chris Funk Slides Geog 210C

- **Definition**
  - Proportion of **point**-to-nearest-**event** distances $d_{min}(t_j)$ no greater than given distance cutoff d, estimated as:

$$\hat{F}(d) = \frac{\#\{\tilde{d}_{min}(\mathbf{t}_j) \leq d, j = 1, \ldots, M\}}{M}$$

  - Cumulative distribution function (CDF) of all *M* point-to-nearest-event distances





- for larger number *M* of random points, F(d) becomes even smoother
- **Note:** The F function provides information on event proximity to voids

So, F is the proportion of nearest neighbor distances that are less than r (or d in Chris Funk terminology in the slide).

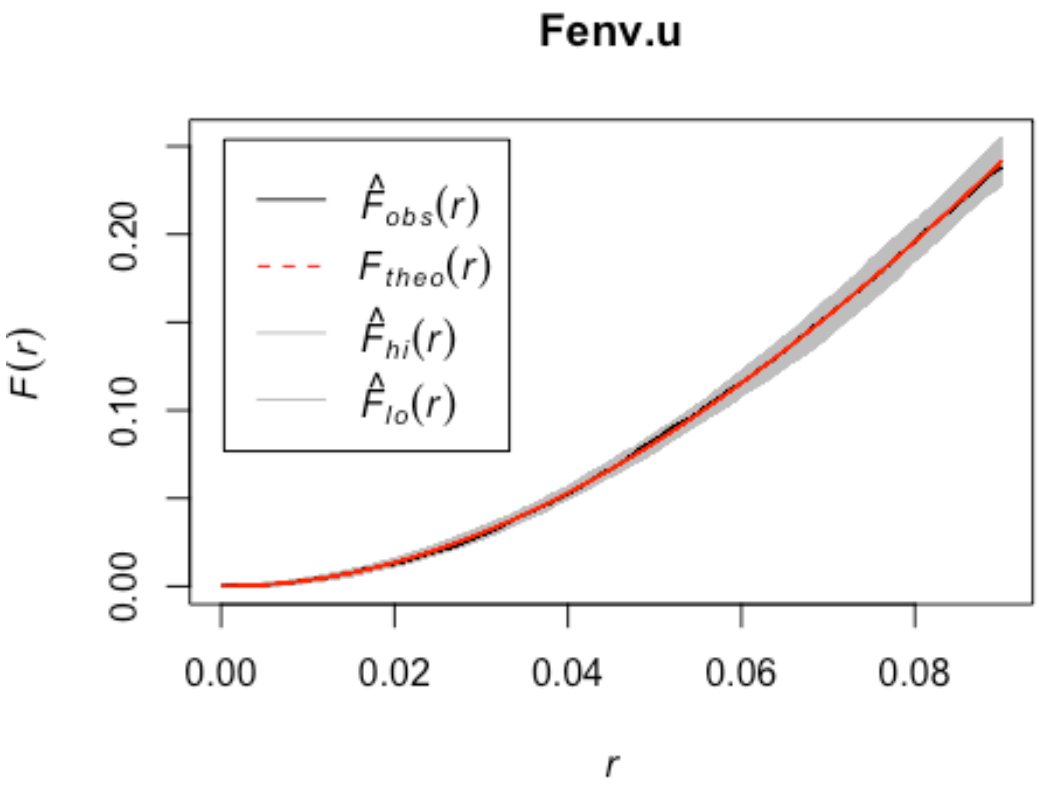
We do the same analysis as in the previous function.

Under CSR the

$$P(r_i < r) = 1 - exp(-\lambda * \pi * r^2)$$

This is also called the empty space function.

```
Fenv.u <- envelope(ppp(x=u.x,y=u.y,window=owin(bbox(sp_point)[1,],bbox(sp_poi
nt)[2,])), fun=Fest, r=r, nsim=99, nrank=2)

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

plot(Fenv.u)
```



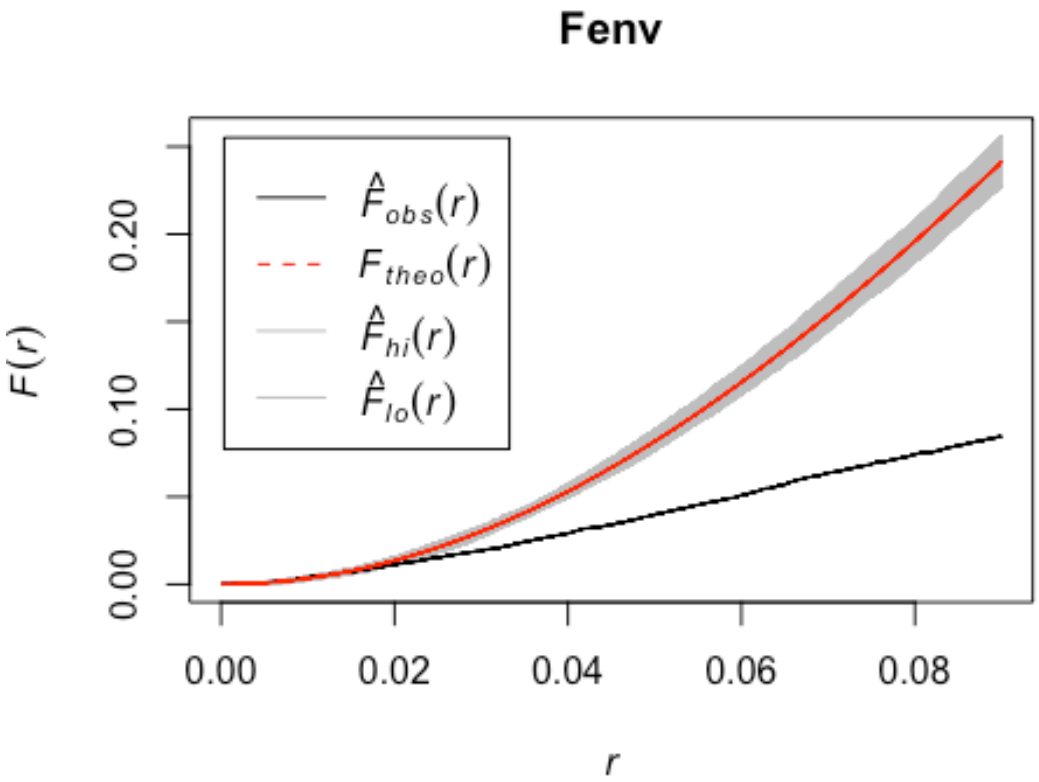
```r
## F-Test: California points

r <- seq(0,km2d(10),length.out=10000)

Fenv <- envelope(ppp(x=sp_point[,1],y=sp_point[,2],window=owin(bbox(sp_point)
[1,],bbox(sp_point)[2,])), fun=Fest, r=r, nsim=99, nrank=2)
```

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

```r
summary(Fenv)
```

```
## Pointwise critical envelopes for F(r)
## and observed value for 'ppp(x = sp_point[, 1], y = sp_point[, 2], window =
## owin(bbox(sp_point)[1, '
## Obtained from 99 simulations of CSR
## Alternative: two.sided
## Upper envelope: pointwise 2nd largest of simulated curves
## Lower envelope: pointwise 2nd smallest of simulated curves
## Significance level of Monte Carlo test: 4/100 = 0.04
## Data: ppp(x = sp_point[, 1], y = sp_point[, 2], window =
## owin(bbox(sp_point)[1,
```

```r
plot(Fenv)
```



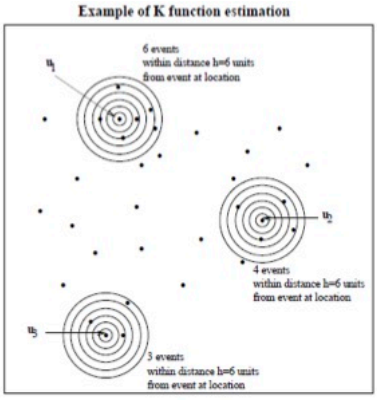
When the F values from our data are below the envelope of the Fs under Poisson we have clustering. This means that the empty space distances among points are longer than a Poisson process (i.e., uniformity). The opposite when the F values from the data are above the Poisson envelope means the empty space distances in our data are shorter thatn Poisson distances implying a regular pattern.

K-function

The nearest neighbor distances do not capture all the complexity of point processes. We can create a different function that takes into account all the pair-wise distances. Then, compute a density function and/or a cumulative density function of these distances.

Concept building

- 1. construct set of concentric circles (of increasing radius d) around each event
- 2. count number of events in each distance "band"
- 3. cumulative number of events up to radius **d** around **all** events = sample K function, K(d)



Formal definition

$$K(d) = \frac{\mathbb{E}\{\#\text{ of events within distance } d \text{ of any arbitrary event }\}}{\mathbb{E}\{\#\text{ of events within study domain }\}}$$

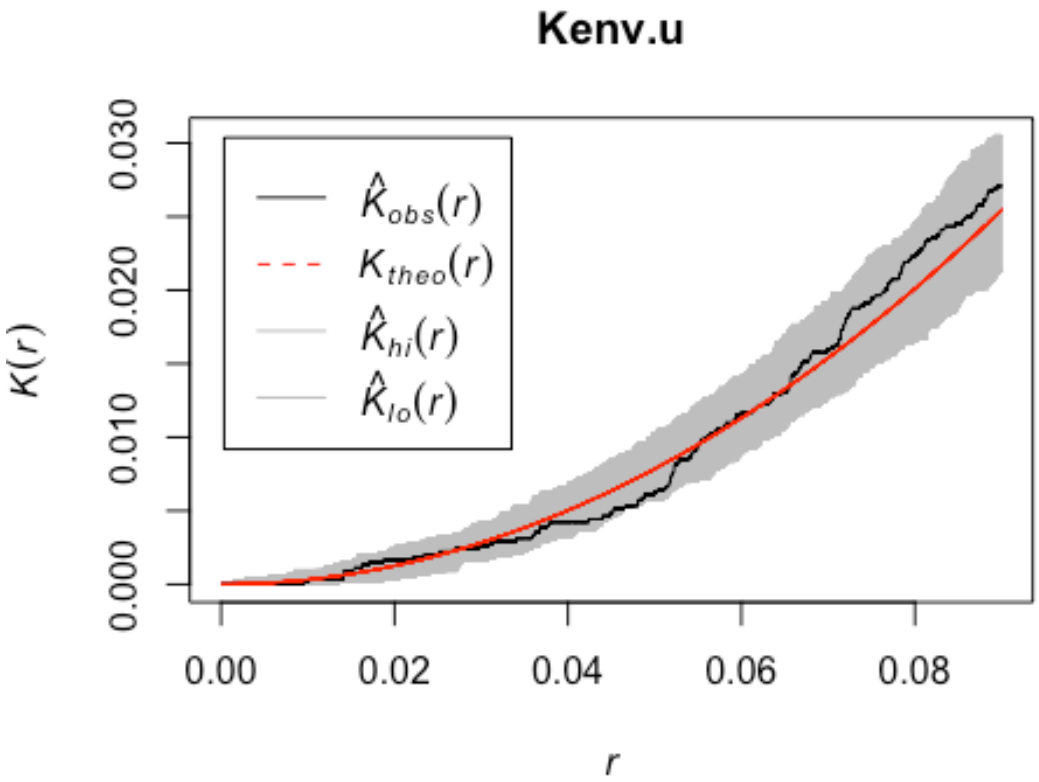
$$\simeq \frac{1}{\lambda}\frac{1}{N}\#\{d_{ij} \le d, i = 1,\ldots,N, j(\neq i) = 1,\ldots,N\} = \hat{K}(d)$$

The conceptual background idea is to look beyond the nearest neighbors or their absence and capture the variation of patterns.

Under CSR

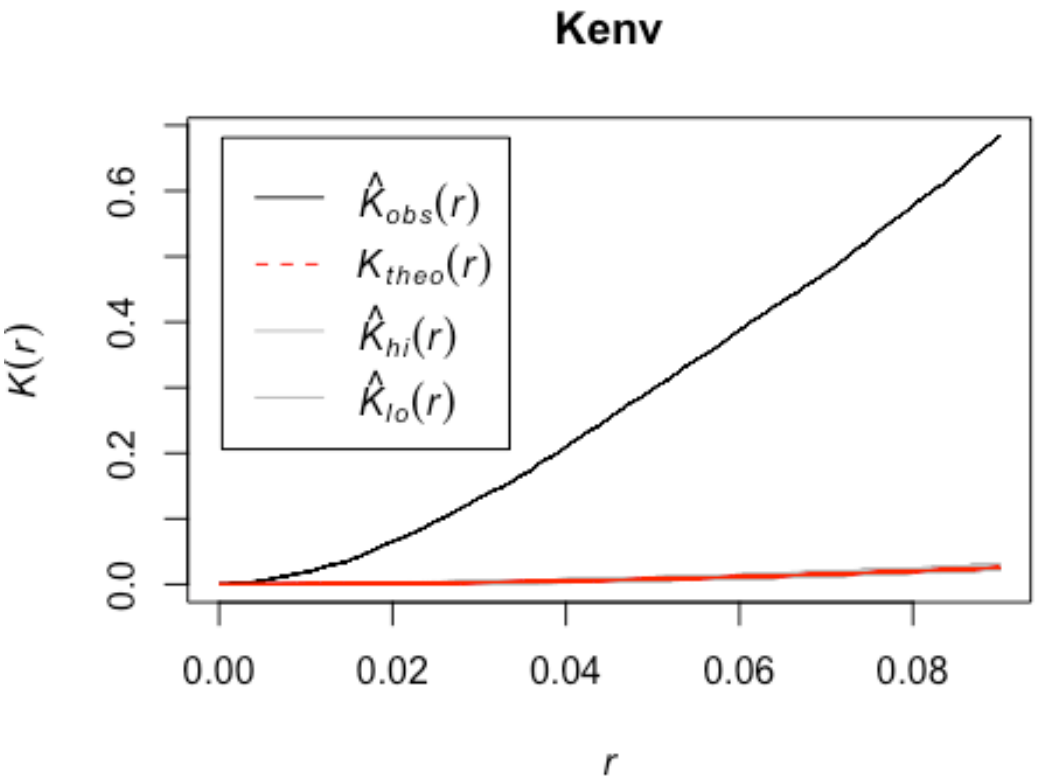
$$K(r) = \pi * r^2)$$

```
Kenv.u <- envelope(ppp(x=u.x,y=u.y,window=owin(bbox(sp_point)[1,],bbox(sp_poi
nt)[2,])), fun=Kest, r=r, nsim=99, nrank=2)

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

plot(Kenv.u)
```




```
## K-Test: California points

r <- seq(0,km2d(10),length.out=10000)

Kenv <- envelope(ppp(x=sp_point[,1],y=sp_point[,2],window=owin(bbox(sp_point)
[1,],bbox(sp_point)[2,])), fun=Kest, r=r, nsim=99, nrank=2)

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

```
summary(Kenv)

## Pointwise critical envelopes for K(r)
## and observed value for 'ppp(x = sp_point[, 1], y = sp_point[, 2], window =
## owin(bbox(sp_point)[1, '
## Obtained from 99 simulations of CSR
## Alternative: two.sided
## Upper envelope: pointwise 2nd largest of simulated curves
## Lower envelope: pointwise 2nd smallest of simulated curves
## Significance level of Monte Carlo test: 4/100 = 0.04
## Data: ppp(x = sp_point[, 1], y = sp_point[, 2], window =
## owin(bbox(sp_point)[1,

plot(Kenv)
```



When

$$\hat{K}_{obs}(r) > \pi * r^2$$

implies clustering

and the opposite implies a regular process

Final note is about edge effects. We assume the data generating process is beyond the window of observation. This means points at the edges of the window have computed distances that are biased and spatstat applies correction factors for the edges. If you look closely to the data we use here, there is no real problem.