Emilia-Romagna Register of Defibrillators

Gianmarco Santoro

Geographic Data

In this work DAE devices distribution is studied. DAE stands to "Automated External Defibrillator", which are useful in case of cardiac emergencies, when time is crucial, so closeness to one of this devices is very important.

Set work directory, install and load libraries

```
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

# Libraries
# install.packages("tidyverse")
# install.packages("sp")
# install.packages("sf")
# install.packages("rmapshaper")
# install.packages("wbstats")
# install.packages("rnaturalearth")
# install.packages("mapview")
# install.packages("spatialreg")
# install.packages("spatstat")

library(sp)
library(sf)
```

```
## Linking to GEOS 3.10.2, GDAL 3.4.2, PROJ 8.2.1; sf_use_s2() is TRUE
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(rgeos)
```

```
## rgeos version: 0.6-2, (SVN revision 693)
##  GEOS runtime version: 3.10.2-CAPI-1.16.0
##  Please note that rgeos will be retired during 2023,
```

```
## plan transition to sf functions using GEOS at your earliest convenience.
##  GEOS using OverlayNG
##  Linking to sp version: 1.6-0
##  Polygon checking: TRUE
##
##
## Attaching package: 'rgeos'
##
## The following object is masked from 'package:dplyr':
##
##     symdiff
```

```
library(maptools)
```

```
## Checking rgeos availability: TRUE
## Please note that 'maptools' will be retired during 2023,
## plan transition at your earliest convenience;
## some functionality will be moved to 'sp'.
```

```
library(tidyr)
library(spatstat)
```

```
## Loading required package: spatstat.data
## Loading required package: spatstat.geom
## spatstat.geom 3.1-0
## Loading required package: spatstat.random
## spatstat.random 3.1-4
## Loading required package: spatstat.explore
## Loading required package: nlme
##
## Attaching package: 'nlme'
##
## The following object is masked from 'package:dplyr':
##
##     collapse
##
## spatstat.explore 3.1-0
## Loading required package: spatstat.model
## Loading required package: rpart
## spatstat.model 3.2-1
## Loading required package: spatstat.linnet
## spatstat.linnet 3.0-6
##
## spatstat 3.0-3
## For an introduction to spatstat, type 'beginner'
```

```
library(splancs) # to compare kernel estimations
```

```
##
## 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
##
## The following object is masked from 'package:tidyr':
##
##     tribble
##
## The following object is masked from 'package:tibble':
##
##     tribble
```

Data management and graphical representation

Import dataset in a dataframe and check for duplicated records

```
dae_full <- read.csv("progetto-dae.csv", sep = ";")

duplicated_record = duplicated(dae_full)
double_record = subset(dae_full, duplicated_record)
print(double_record)
```

```
##  [1] Nome               Città              Indirizzo          Ubicazione
##  [5] Note               Orari              Telefono           Geo.Point
##  [9] Quartiere          Zona.di.prossimità Area.statistica
## <0 rows> (or 0-length row.names)
```

There aren't duplicates but in the same place, e.g. Technogym, there are 3 DAE with the same location. Discard them anyway since just 83 over 5298 records.

Selecting specific columns in a new df, keeping only useful info

```
dae <- dae_full[c("Nome", "Città", "Geo.Point")]
head(dae)
```

```
##                                  Nome    Città
## 1               COOP SOCIALE BUCANEVE    BARDI
## 2              Assicurazioni Generali   RIMINI
## 3                   Bar Dovesi Rimini   RIMINI
## 4                Tennis Club Riccione RICCIONE
## 5 Liceo Scientifico Einstein Palestra   RIMINI
## 6                     CURIA VESCOVILE    FORLI
##                                Geo.Point
## 1   44.630611419677734, 9.72889232635498
## 2 44.033016204833984, 12.557188987731934
## 3  44.05921936035156, 12.568144798278809
## 4       44.001953125, 12.641033172607422
## 5  44.04856872558594, 12.584275245666504
## 6  44.22264099121094, 12.034880638122559
```

Discard duplicated records

```
duplicated_rows = duplicated(dae)
double_data = subset(dae, duplicated_rows)
```

```
head(double_data)
```

```
##                   Nome                     Città
## 44   ESSELUNGA S.P.A.                     PARMA
## 45   ESSELUNGA S.P.A.                     PARMA
## 46   ESSELUNGA S.P.A.                     PARMA
## 47   ESSELUNGA S.P.A.                     PARMA
## 48   ESSELUNGA S.P.A.                     PARMA
## 1206        Vertivsrl CASTEL GUELFO DI BOLOGNA
##                                   Geo.Point
## 44   44.849708557128906, 10.367918014526367
## 45   44.849708557128906, 10.367918014526367
## 46   44.849708557128906, 10.367918014526367
## 47   44.849708557128906, 10.367918014526367
## 48   44.849708557128906, 10.367918014526367
## 1206  44.43602752685547, 11.617171287536621
```

```
#print(count(double_data))
```

Remove duplicates from dae

```
dae <- unique(dae)
#head(dae)
count(dae)
```

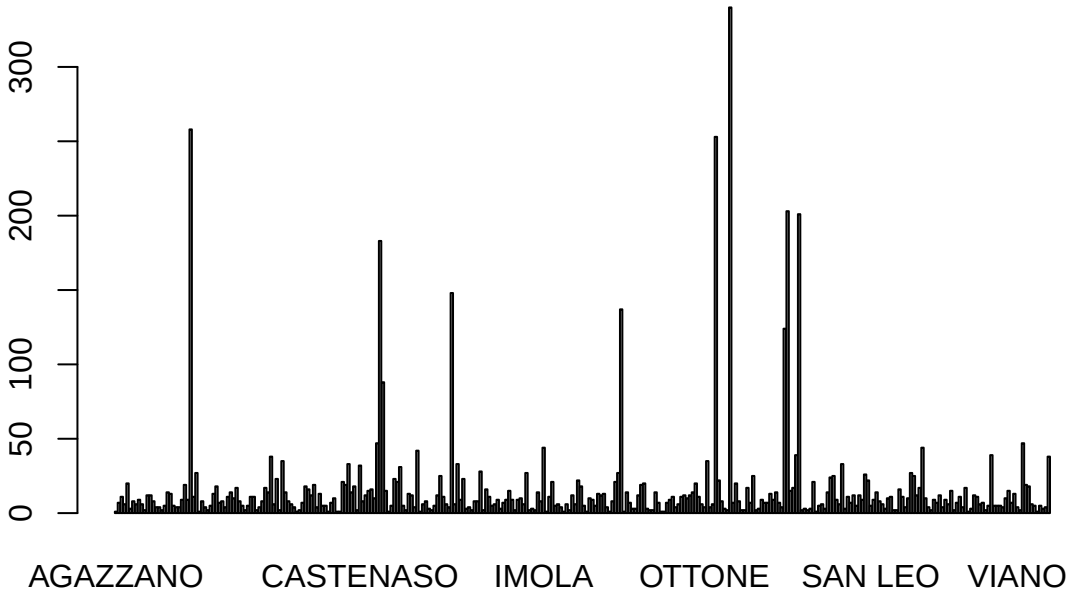
```
##      n
## 1 5215
```

Count how many cities

```
unique_cities <- unique(dae$Città)
city_numb = length(unique_cities)
print(city_numb)
```

```
## [1] 326
```

```
#table(dae$Città)
barplot(table(dae$Città),main='DAE per Municipality')
```



Since it's difficult to understand, select the 10 with most and less DAE.

```
# Count the number of occurrences of each city
city_counts <- table(dae$Città)

# Sort the cities by their counts in descending order
sorted_cities <- sort(city_counts, decreasing = TRUE)
#print(sorted_cities)
head(sorted_cities, 10)
```

```
##
##           PIACENZA            BOLOGNA              PARMA REGGIO NELL EMILIA
##                340                258                253                203
##             RIMINI              CESENA            FERRARA             MODENA
##                201                183                148                137
##            RAVENNA         CESENATICO
##                124                 88
```

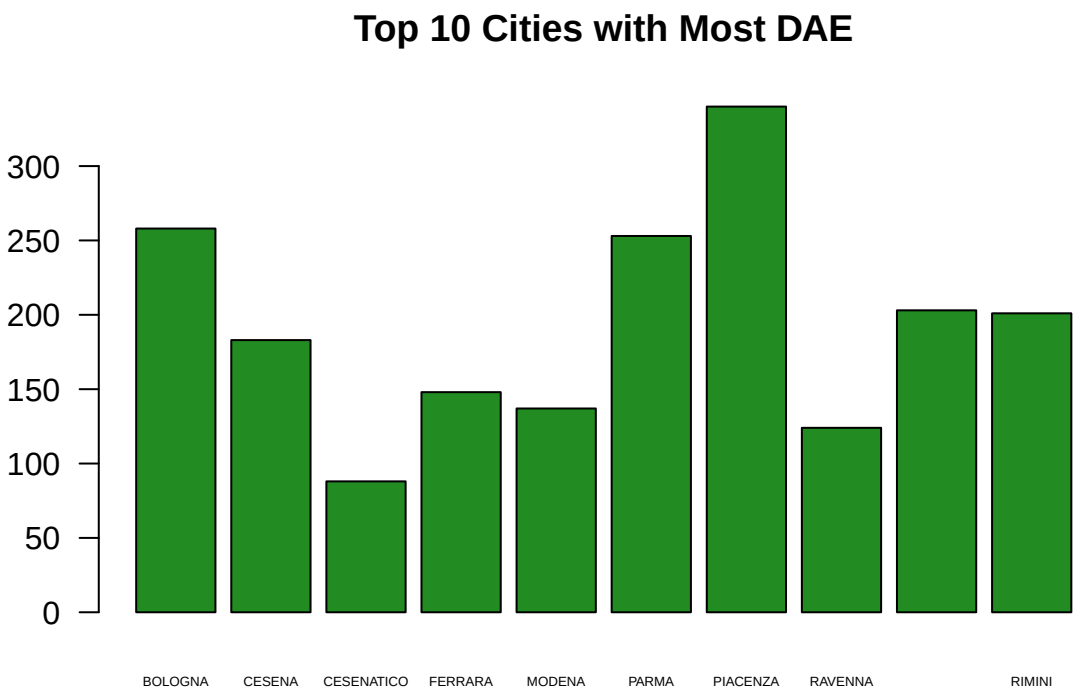
```
tail(sorted_cities, 10)
```

```
##
##                  LAGOSANTO                     LOIANO              MASI TORELLO
##                          1                          1                         1
## MEZZANI - SORBOLO MEZZANI                 MODIGLIANA                   MONTESE
##                          1                          1                         1
##                   MONTIANO        ROCCA SAN CASCIANO                   TORNOLO
##                          1                          1                         1
##                      ZERBA
##                          1
```

```
# Extract the names of the top and bottom cities
top_cities <- names(sorted_cities)[1:10]
bottom_cities <- names(sorted_cities)[(length(sorted_cities) - 9):length(sorted_cities)]
#print(top_cities)
```

```r
#print(bottom_cities)

# Filter the data for the top and bottom cities
top_data <- dae[dae$Città %in% top_cities, ]
bottom_data <- dae[dae$Città %in% bottom_cities, ]
```

```r
# Barplot for top cities
barplot(table(top_data$Città), main = 'Top 10 Cities with Most DAE', col = "forestgreen", las = 1, cex.
```




```r
# Barplot for bottom cities
#barplot(table(bottom_data$Città), main = '10 Cities with Less DAE', col = "orangered", las = 2, cex.na
```

```r
# devide geopoint into latutidue + longitude
dae_coord <- separate(dae,
                      Geo.Point,
                      into = c("Latitude", "Longitude"),
                      sep = ", ")
head(dae_coord)
```

```
##                                  Nome    Città           Latitude
## 1               COOP SOCIALE BUCANEVE    BARDI 44.630611419677734
## 2             Assicurazioni Generali   RIMINI 44.033016204833984
## 3                  Bar Dovesi Rimini   RIMINI  44.05921936035156
## 4               Tennis Club Riccione RICCIONE       44.001953125
## 5 Liceo Scientifico Einstein Palestra   RIMINI  44.04856872558594
## 6                    CURIA VESCOVILE    FORLI  44.22264099121094
##            Longitude
## 1   9.72889232635498
## 2 12.557188987731934
## 3 12.568144798278809
## 4 12.641033172607422
## 5 12.584275245666504
## 6 12.034880638122559
```

Convert the dataset to an 'sf' object and specify CRS

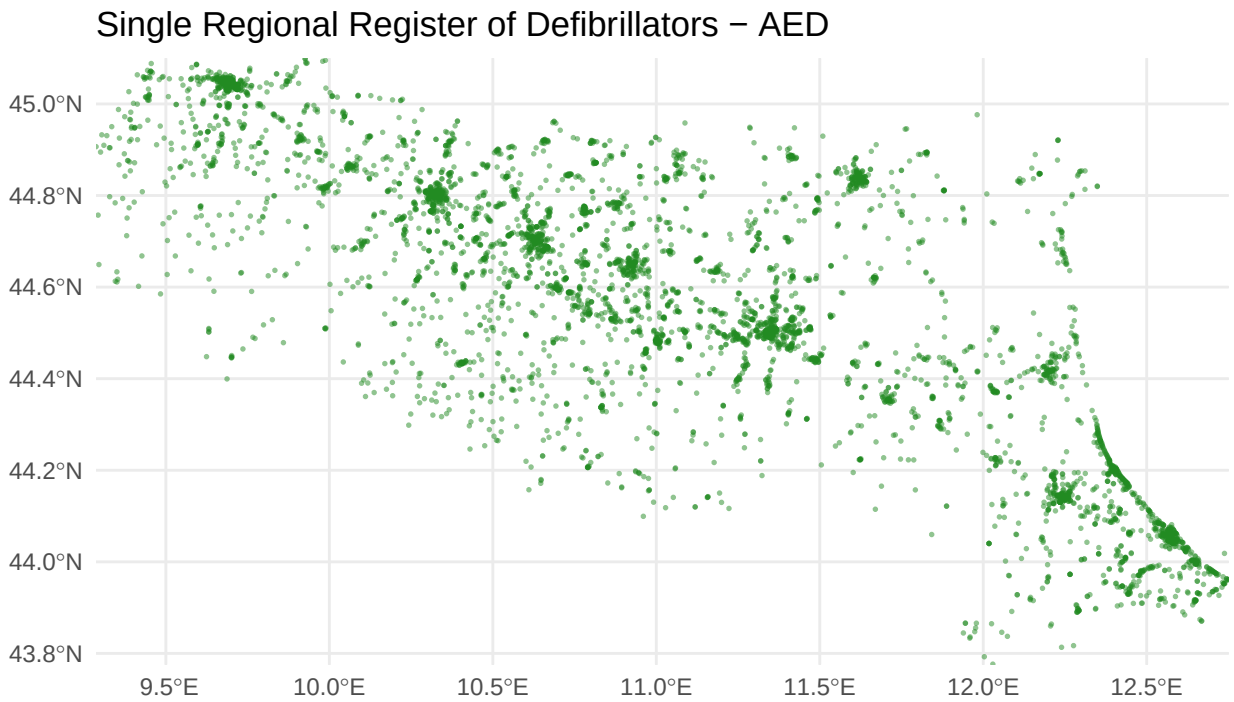
```
dae_wgs <- st_as_sf(dae_coord,
                    coords = c("Longitude", "Latitude"),
                    crs = 4326)  # 3857 EPSG code for WGS84 (standard for lat/long)
head(dae_wgs)
```

```
## Simple feature collection with 6 features and 2 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 9.728892 ymin: 44.00195 xmax: 12.64103 ymax: 44.63061
## Geodetic CRS:  WGS 84
##                                   Nome    Città                   geometry
## 1               COOP SOCIALE BUCANEVE    BARDI POINT (9.728892 44.63061)
## 2              Assicurazioni Generali   RIMINI POINT (12.55719 44.03302)
## 3                   Bar Dovesi Rimini   RIMINI POINT (12.56814 44.05922)
## 4                Tennis Club Riccione RICCIONE POINT (12.64103 44.00195)
## 5 Liceo Scientifico Einstein Palestra   RIMINI POINT (12.58428 44.04857)
## 6                     CURIA VESCOVILE    FORLI POINT (12.03488 44.22264)
```

```
#print(dae_wgs)
```

Map where devices are located

```
ggplot(dae_wgs) +
  geom_sf(color = "forestgreen", alpha = .5, size = .3) +
  labs(title = "Single Regional Register of Defibrillators - AED") +
  theme_minimal() +
  coord_sf(expand = FALSE)
```



Map Emilia-Romagna municipalities borders

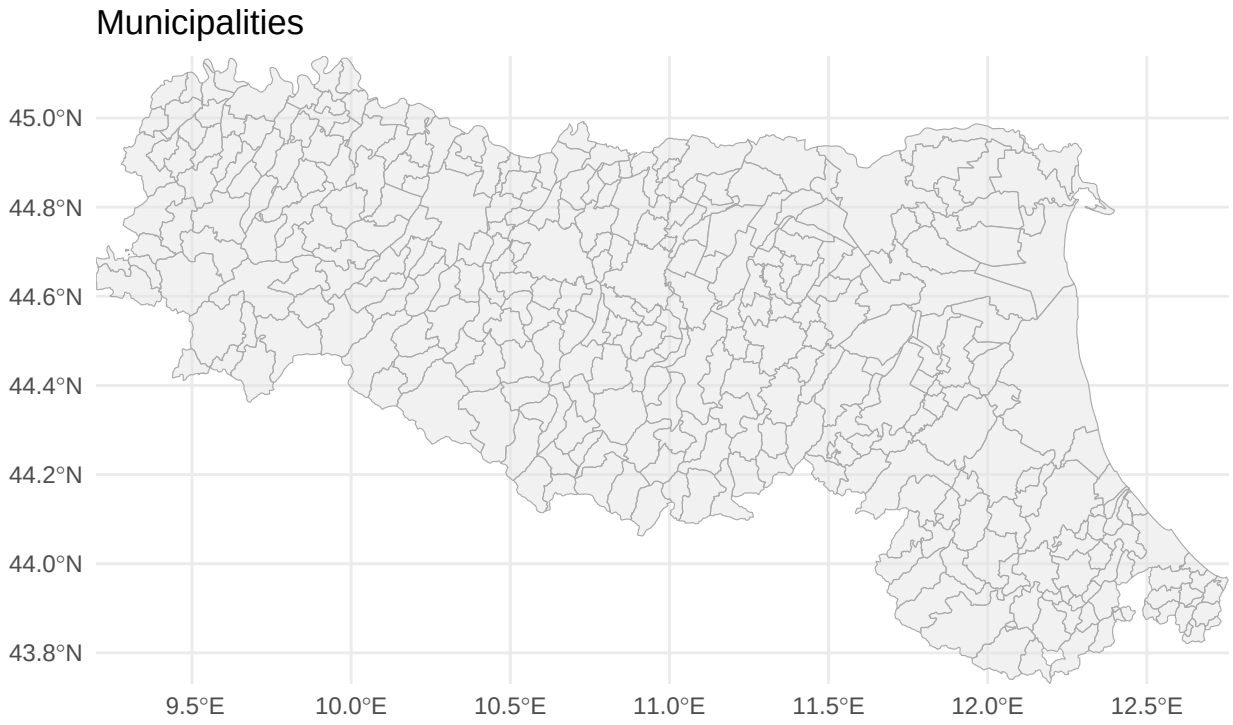
```
## import shapefile - SpatialData
municipalities <- sf::st_read("V_COM_GPG_3.shx")
```

```
## Reading layer `V_COM_GPG_3' from data source
##   `/Users/GianmarcoSantoro/miniconda3/Projects/GeoSpacial/V_COM_GPG_3.shx'
##   using driver `ESRI Shapefile'
## Simple feature collection with 330 features and 0 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 9.198528 ymin: 43.73088 xmax: 12.75532 ymax: 45.13901
## CRS:           NA
```

```
#print(municipalities)
#summary(municipalities)

# sistema di riferimento geografico: WGS84 - UTM ZONE 32N
# con il pacchetto "sf" si possono convertire i CRS
# Set the correct CRS, EPSG:4326 to WGS84
st_crs(municipalities) <- st_crs("EPSG:4326")

# Convert the dataset to an 'sf' object and specify CRS
municipalities_coord <- st_as_sf(municipalities,
                      coords = c("Longitude", "Latitude"),
                      crs = 4326)

print(municipalities_coord)
```

```
## Simple feature collection with 330 features and 0 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 9.198528 ymin: 43.73088 xmax: 12.75532 ymax: 45.13901
## Geodetic CRS:  WGS 84
## First 10 features:
##                           geometry
## 1  MULTIPOLYGON (((10.86582 44...
## 2  MULTIPOLYGON (((12.6792 43....
## 3  MULTIPOLYGON (((9.888571 45...
## 4  MULTIPOLYGON (((9.966079 45...
## 5  MULTIPOLYGON (((10.074 44.9...
## 6  MULTIPOLYGON (((12.14862 43...
## 7  MULTIPOLYGON (((11.43073 44...
## 8  MULTIPOLYGON (((10.75838 44...
## 9  MULTIPOLYGON (((11.80945 44...
## 10 MULTIPOLYGON (((9.447531 44...
```

```
#summary(municipalities_coord)

ggplot(municipalities_coord) +
  geom_sf(color = "darkgray", alpha = .5, size = .3) +
  labs(title = "Municipalities") +
  theme_minimal() +
  coord_sf(expand = FALSE)
```



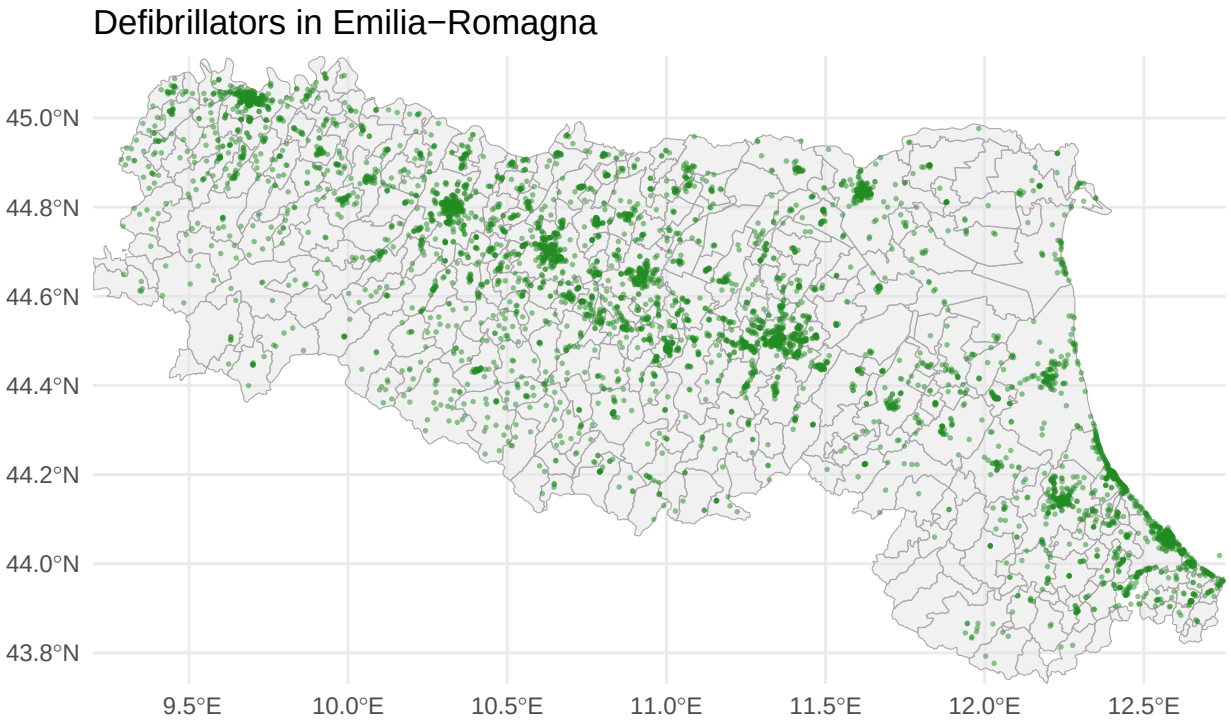
Check of reference systems are same

```
st_crs(dae_wgs) == st_crs(municipalities_coord)
```

```
## [1] TRUE
```

Map DAE devices and municipalities

```
# Plot of devices and municipalities
ggplot() +
  geom_sf(data = municipalities_coord, color = "darkgray", alpha = 0.5, size = 0.3) +
  geom_sf(data = dae_wgs, color = "forestgreen", alpha = 0.5, size = 0.3) +
  labs(title = "Defibrillators in Emilia-Romagna") +
  theme_minimal() +
  coord_sf(expand = FALSE)
```



We can notice that 2 devices are located in the Adriatic sea, as well as San Marino's ones are shown. The ones in the sea can be in islands or gas platforms or in other off-shore systems.

Spatial Point Patterns Analysis

```
head(dae_wgs)
```

```
## Simple feature collection with 6 features and 2 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 9.728892 ymin: 44.00195 xmax: 12.64103 ymax: 44.63061
## Geodetic CRS:  WGS 84
##                                 Nome    Città                  geometry
## 1               COOP SOCIALE BUCANEVE    BARDI POINT (9.728892 44.63061)
## 2              Assicurazioni Generali   RIMINI POINT (12.55719 44.03302)
## 3                   Bar Dovesi Rimini   RIMINI POINT (12.56814 44.05922)
## 4                Tennis Club Riccione RICCIONE POINT (12.64103 44.00195)
## 5 Liceo Scientifico Einstein Palestra   RIMINI POINT (12.58428 44.04857)
## 6                    CURIA VESCOVILE    FORLI POINT (12.03488 44.22264)
```

Use projection 3003 to have better insights in meters and not degrees

```
# EPSG:4326 for WGS 84 and EPSG:3003 for Italian projection from Monte Mario reference point
dae_proj1 <- st_transform(dae_wgs, crs = 3003)
print(dae_proj1)
```

```
## Simple feature collection with 5215 features and 2 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 1522642 ymin: 4851522 xmax: 1800810 ymax: 4994375
```

```
## Projected CRS: Monte Mario / Italy zone 1
## First 10 features:
##                                         Nome               Città
## 1                       COOP SOCIALE BUCANEVE               BARDI
## 2                      Assicurazioni Generali              RIMINI
## 3                          Bar Dovesi Rimini               RIMINI
## 4                       Tennis Club Riccione             RICCIONE
## 5          Liceo Scientifico Einstein Palestra               RIMINI
## 6                             CURIA VESCOVILE               FORLI
## 7  Stadio del nuoto Centro Sportivo Riccione             RICCIONE
## 8                     BODY LIFE 2.0 S.S.D.R.L CASTELNUOVO RANGONE
## 9                      Chiesa di Sant'Antonio SALSOMAGGIORE TERME
## 10                    A.S.D. Bedoniese United              BEDONIA
##                     geometry
## 1  POINT (1557844 4942194)
## 2  POINT (1785075 4881710)
## 3  POINT (1785827 4884659)
## 4  POINT (1791947 4878554)
## 5  POINT (1787171 4883532)
## 6  POINT (1742442 4901098)
## 7  POINT (1791651 4878730)
## 8  POINT (1653887 4935403)
## 9  POINT (1578199 4963594)
## 10 POINT (1550200 4928683)
```

```
municipalities_proj <- st_transform(municipalities_coord, crs = 3003)
print(municipalities_proj)
```

```
## Simple feature collection with 330 features and 0 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 1515770 ymin: 4846959 xmax: 1801305 ymax: 4998818
## Projected CRS: Monte Mario / Italy zone 1
## First 10 features:
##                           geometry
## 1  MULTIPOLYGON (((1648239 493...
## 2  MULTIPOLYGON (((1795348 487...
## 3  MULTIPOLYGON (((1570027 498...
## 4  MULTIPOLYGON (((1576002 499...
## 5  MULTIPOLYGON (((1584799 497...
## 6  MULTIPOLYGON (((1753181 485...
## 7  MULTIPOLYGON (((1694092 490...
## 8  MULTIPOLYGON (((1639384 494...
## 9  MULTIPOLYGON (((1723832 491...
## 10 MULTIPOLYGON (((1535488 494...
```

```
# Extracting coordinates into separate columns, devide geometry into latutidue + longitude
coordinates <- st_coordinates(dae_proj1)
dae_proj <- cbind(dae_proj1$Nome, dae_proj1$Città, coordinates)

# Rename the columns
colnames(dae_proj) <- c("Nome", "Città", "Longitude", "Latitude")
head(dae_proj)
```

```
##      Nome                                  Città      Longitude
```

```
## [1,] "COOP SOCIALE BUCANEVE"                "BARDI"    "1557844.23150378"
## [2,] "Assicurazioni Generali"               "RIMINI"   "1785075.14596705"
## [3,] "Bar Dovesi Rimini"                    "RIMINI"   "1785826.93548339"
## [4,] "Tennis Club Riccione"                 "RICCIONE" "1791946.69377474"
## [5,] "Liceo Scientifico Einstein Palestra" "RIMINI"   "1787170.52588819"
## [6,] "CURIA VESCOVILE"                      "FORLI"    "1742442.10838033"
##      Latitude
## [1,] "4942193.79963972"
## [2,] "4881710.22510532"
## [3,] "4884658.81470574"
## [4,] "4878553.52190728"
## [5,] "4883531.90980742"
## [6,] "4901097.93272644"
```

```
dae_small <- dae_proj
summary(dae_small)
```

```
##      Nome              Città            Longitude           Latitude
##  Length:5215        Length:5215        Length:5215        Length:5215
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

```
# Ensure dae_small is a dataframe
dae_small <- as.data.frame(dae_small)

# Convert Longitude and Latitude columns to numeric
dae_small$Longitude <- as.numeric(dae_small$Longitude)
dae_small$Latitude <- as.numeric(dae_small$Latitude)

# Round Longitude and Latitude to 5 decimal places
dae_small$Longitude <- round(dae_small$Longitude, 5)
dae_small$Latitude <- round(dae_small$Latitude, 5)

# duplicated_small = duplicated(dae_small)
# double_small = subset(dae_small, duplicated_small)
# print(double_small)

# Remove duplicates
dae_small <- unique(dae_small)

# Check the first few rows to confirm the changes
summary(dae_small)
```

```
##      Nome              Città              Longitude          Latitude
##  Length:5215        Length:5215        Min.   :1522642   Min.   :4851522
##  Class :character   Class :character   1st Qu.:1607806   1st Qu.:4917708
##  Mode  :character   Mode  :character   Median :1656938   Median :4940888
##                                        Mean   :1664463   Mean   :4937971
##                                        3rd Qu.:1718146   3rd Qu.:4963326
##                                        Max.   :1800810   Max.   :4994375
```

```
head(dae_small)
```

```
##                                  Nome    Città Longitude Latitude
## 1              COOP SOCIALE BUCANEVE    BARDI   1557844  4942194
## 2             Assicurazioni Generali   RIMINI   1785075  4881710
```

```
## 3                   Bar Dovesi Rimini   RIMINI   1785827  4884659
## 4                Tennis Club Riccione RICCIONE   1791947  4878554
## 5 Liceo Scientifico Einstein Palestra   RIMINI   1787171  4883532
## 6                     CURIA VESCOVILE    FORLI   1742442  4901098
```

Considered rectangular area containing location of devices

```
x_range <- range(dae_small$Longitude)
y_range <- range(dae_small$Latitude)
xy_area <- owin(x_range, y_range)

# Transform into ppp (point pattern dataset) to show info in the area of interest
dae_ppp = ppp(dae_small$Longitude, dae_small$Latitude, window = xy_area)  # warning: duplicated points
```

```
## Warning: data contain duplicated points
```

```
summary(dae_ppp)
```

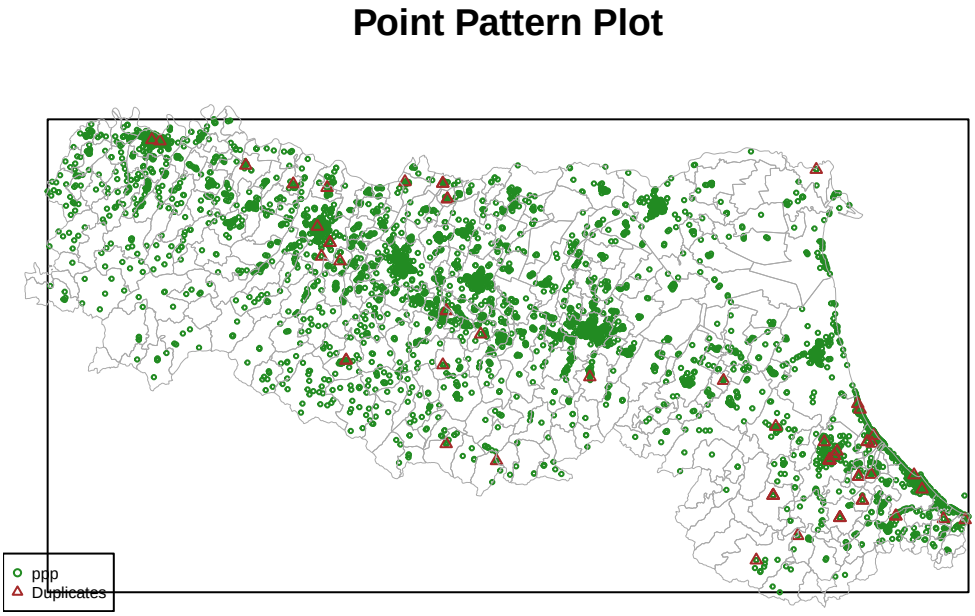
```
## Planar point pattern:  5215 points
## Average intensity 1.312378e-07 points per square unit
##
## *Pattern contains duplicated points*
##
## Coordinates are given to 1 decimal place
## i.e. rounded to the nearest multiple of 0.1 units
##
## Window: rectangle = [1522641.6, 1800810.1] x [4851522, 4994375] units
##                     (278200 x 142900 units)
## Window area = 3.9737e+10 square units
```

Discard duplicates on this projection

Genrally in PPP is good practice to have no coinciding points: “when the data has coincidence points, some statistical procedures will be severely affected. So it is always strongly advisable to check for duplicate points and to decide on a strategyfor dealing with them if they are present” (Baddeley et al., 2016: p.60).

```
# Move a little overlapping points
dae_ppp_r <- dae_ppp[!duplicated(dae_ppp)]
dup_j <- rjitter(dae_ppp[duplicated(dae_ppp)], radius=0.01, retry=TRUE, nsim=1, drop=TRUE)

dae_ppp_j <- superimpose(dae_ppp_r,dup_j)
dae_ppp_originali <- dae_ppp
dae_ppp <- dae_ppp_j
```

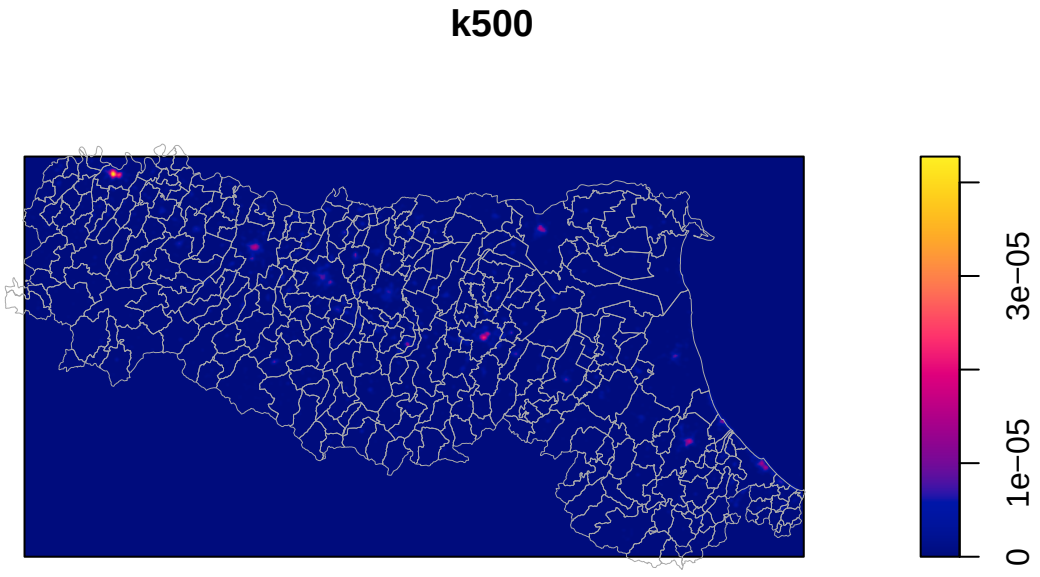
```
plot(dae_ppp_originali, cols='forestgreen',  cex=0.3, main = "Point Pattern Plot")
plot(dup_j, pch=2, cols='brown', add=T, cex=0.6)
plot(municipalities_proj, border = "darkgray", add=T, lwd = 0.3)

# Create a legend for the plot
legend("bottomleft",
       legend = c("ppp", "Duplicates"),
       col = c("forestgreen", "brown"),
       pch = c(1, 2),
       cex = 0.5)
```



First Order Effects

Kernel Density Estimate: estimate the intensity of the process, with a kernel estimation

```
k500 = density(dae_ppp, sigma = 500, dimyx = 500)
plot(k500)
#plot(dae_proj1$geometry, cex=0.01, add=T)
plot(municipalities_proj, border = "darkgray", add=T, lwd = 0.1)
```



We can see in Piacenza, Modena, Parma, Cesena, Rimini some high values, but still not so informative

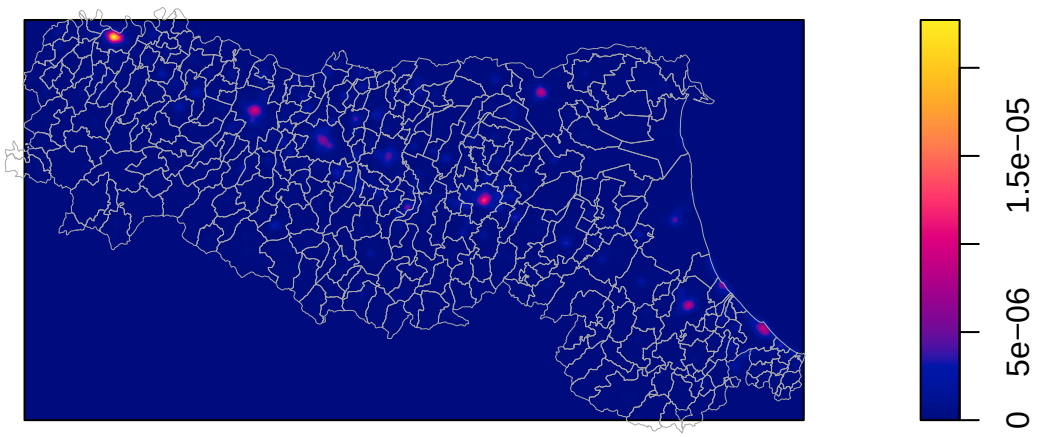
Select an optimal bandwidth with Likelihood Cross Validation critirion

```
# Setting the seed for reproducibility
set.seed(23)
```

```
# Calculate the bandwidth for the point pattern
sigma_ppl <- bw.ppl(dae_ppp)
print(sigma_ppl)
```

```
##    sigma
## 976.8324
```

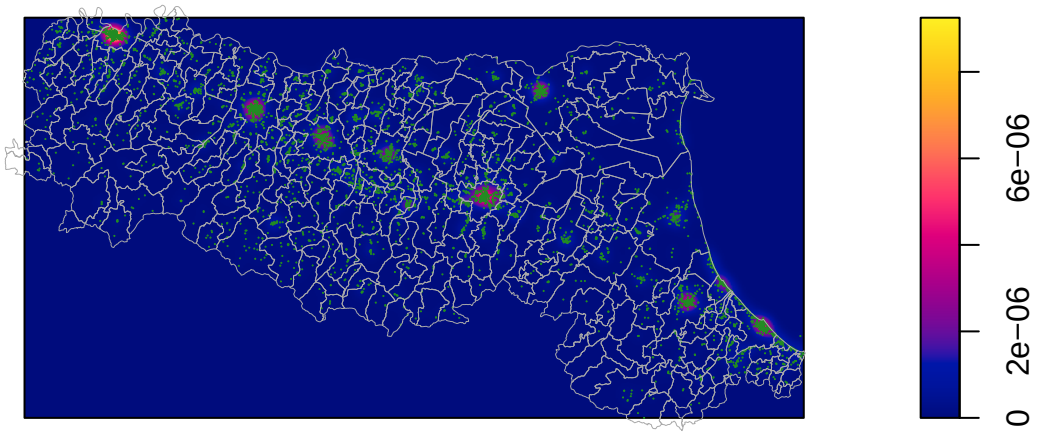
```
kopt = density(dae_ppp, sigma = sigma_ppl, dimyx = 500)
plot(kopt)
#plot(dae_proj1$geometry, col = "forestgreen" , add=T, cex=0.01)
plot(municipalities_proj, border = "darkgray", add=T, lwd = 0.1)
```



Trying some more values

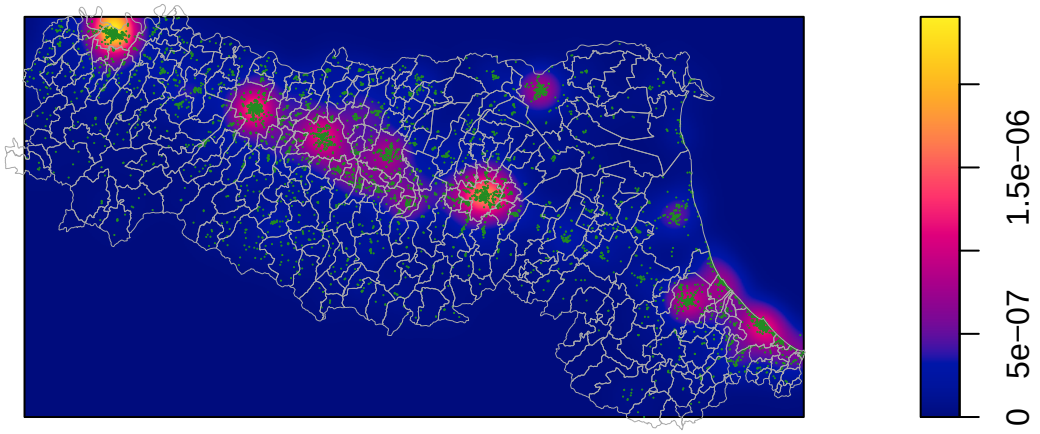
```
k2000 = density(dae_ppp, sigma = 2000, dimyx = 500)
plot(k2000)
plot(dae_proj1$geometry, col = "forestgreen", add=T, cex=0.01)
plot(municipalities_proj, border = "darkgray", add=T, lwd = 0.1)
```

k2000



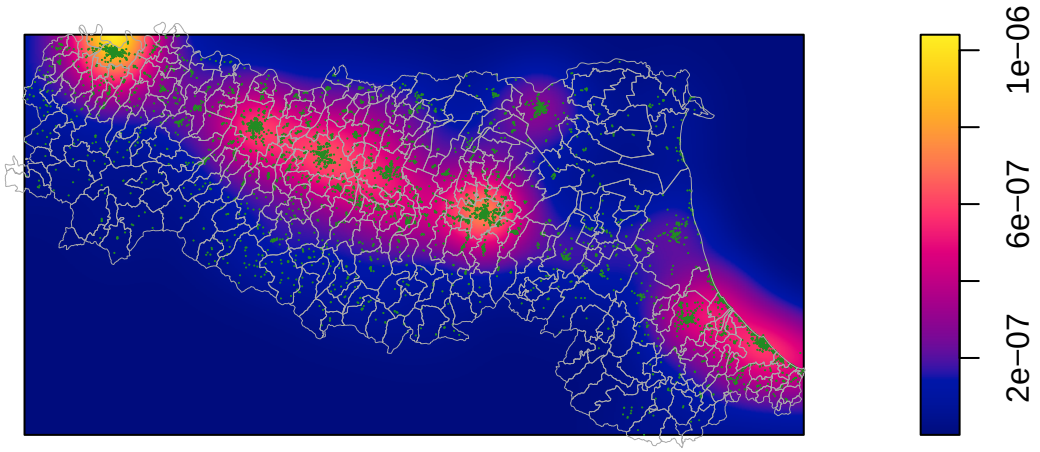

```
k5000 = density(dae_ppp, sigma = 5000, dimyx = 500)
plot(k5000)
plot(dae_proj1$geometry, col = "forestgreen", add=T, cex=0.01)
plot(municipalities_proj, border = "darkgray", add=T, lwd = 0.1)
```

k5000



From my point of view this is the most informative plot.

```
k10000 = density(dae_ppp, sigma = 10000, dimyx = 500)
plot(k10000)
plot(dae_proj1$geometry, col = "forestgreen", add=T, cex=0.01)
plot(municipalities_proj, border = "darkgray", add=T, lwd = 0.1)
```



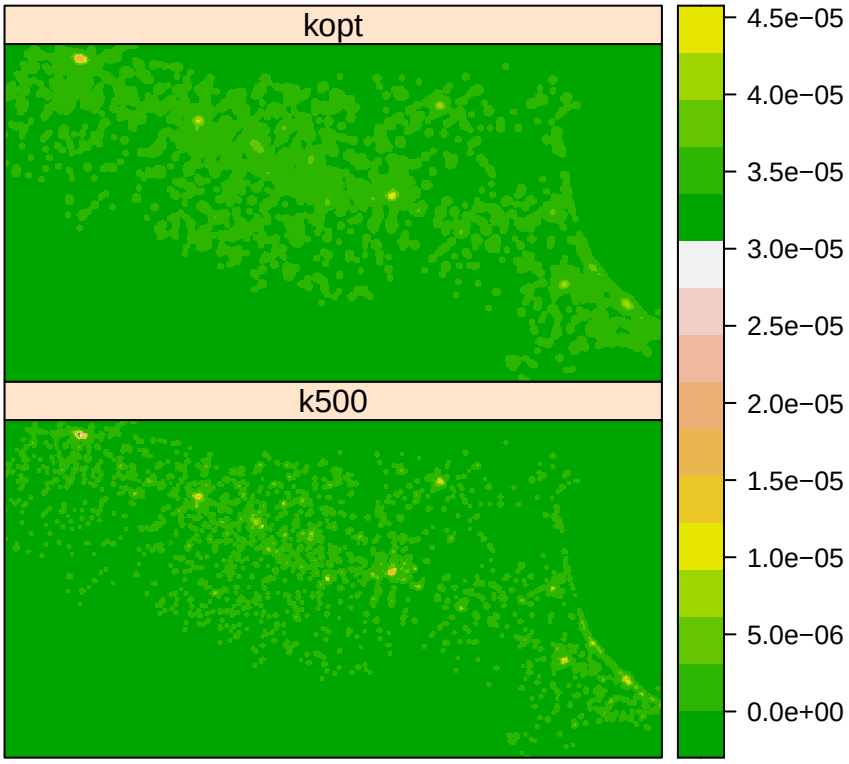
As well as this one, with different level of representation.

Save all output as SpatialGrid, for different values of sigma in kernel density

```
SG <- as(k500, "SpatialGridDataFrame")
SG <- cbind(SG,as(kopt, "SpatialGridDataFrame"))
SG <- cbind(SG,as(k2000, "SpatialGridDataFrame"))
SG <- cbind(SG,as(k5000, "SpatialGridDataFrame"))
SG <- cbind(SG,as(k10000, "SpatialGridDataFrame"))

names(SG) <- c("k500","kopt","k2000","k5000","k10000")
```

With some plots of this kind, k500 and kopt show better granularity of this process

```
spplot(SG, c("k500","kopt"),col.regions=terrain.colors(11))
```

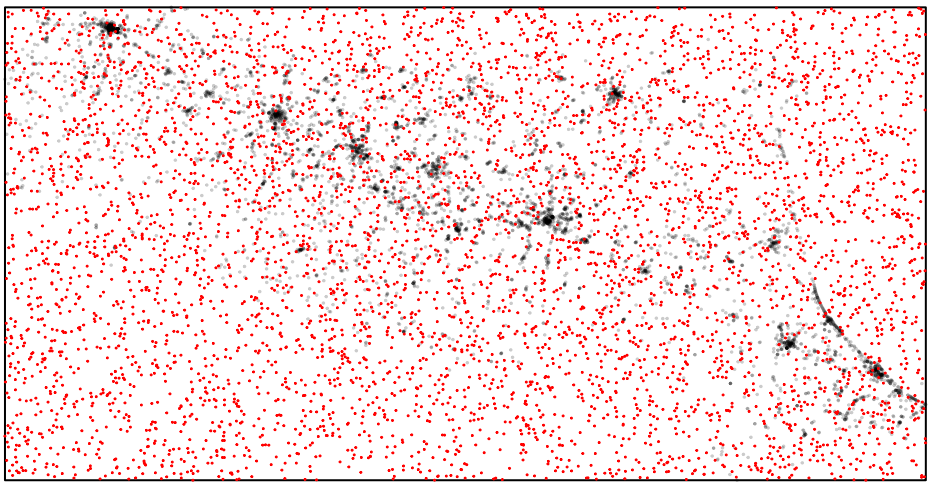



```
# Show results
summary(as.data.frame(SG)[,1:5])
```

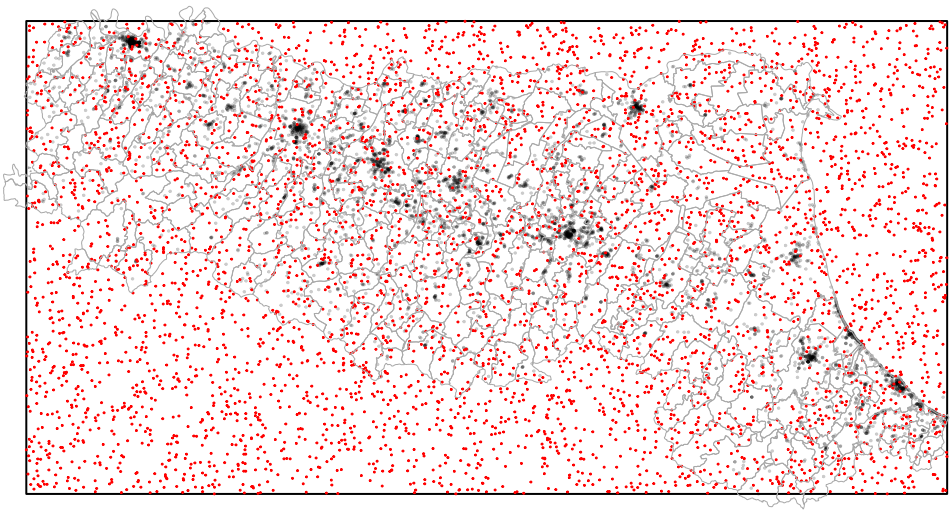
```
##       k500                kopt               k2000          
##  Min.   :0.000e+00   Min.   :0.000e+00   Min.   :0.000e+00  
##  1st Qu.:0.000e+00   1st Qu.:0.000e+00   1st Qu.:0.000e+00  
##  Median :0.000e+00   Median :1.900e-10   Median :1.759e-08  
##  Mean   :1.312e-07   Mean   :1.312e-07   Mean   :1.314e-07  
##  3rd Qu.:3.930e-09   3rd Qu.:7.758e-08   3rd Qu.:1.212e-07  
##  Max.   :4.276e-05   Max.   :2.272e-05   Max.   :9.253e-06  
##      k5000               k10000          
##  Min.   :0.000e+00   Min.   :0.000e+00  
##  1st Qu.:3.646e-10   1st Qu.:8.066e-09  
##  Median :4.296e-08   Median :6.200e-08  
##  Mean   :1.321e-07   Mean   :1.327e-07  
##  3rd Qu.:1.585e-07   3rd Qu.:1.924e-07  
##  Max.   :2.405e-06   Max.   :1.040e-06
```

CSR simulated process, spacial random process following binomial distribution

```
CSR <-runifpoint(dae_ppp$n, win = dae_ppp$window)

plot(dae_ppp, cex=0.1, main= "DAE positions vs Random DAE (red)")
plot(CSR, add=T, col='red', pch=20, cex=0.1)
```




```
#plot(municipalities_proj, border = "darkgray", add=T, lwd = 0.5)

plot(dae_ppp, cex=0.1, main= "DAE positions vs Random DAE (red)")
plot(CSR, add=T, col='red', pch=20, cex=0.1)
plot(municipalities_proj, border = "darkgray", add=T, lwd = 0.5)
```



There is regular pattern of clusters, differently from the random representation in red. Clusters are in correspondence to main city centers and Riviera Romagnola, where in summer there is a significant increase in people presence, which the generating process probabily have taken into account.

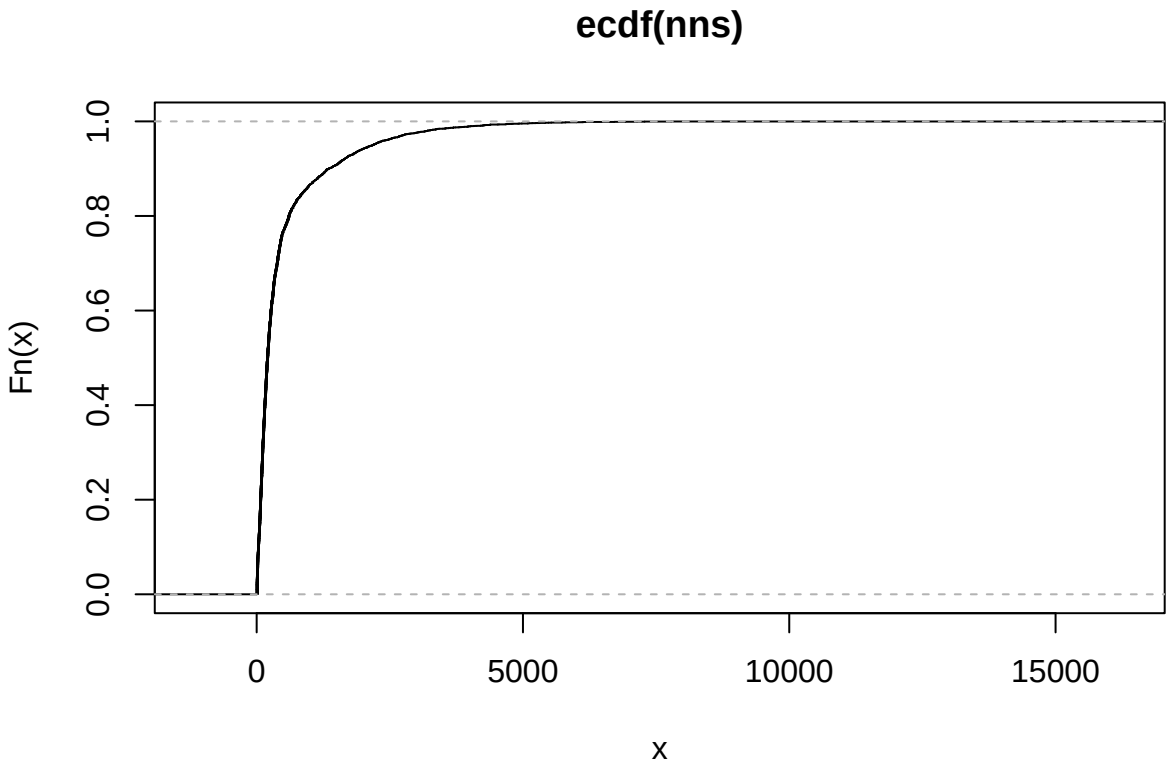
Nearest neighbor distance

Equivalent to G-function without border correction

Calculates the nearest neighbor distances for each point in the dae_ppp point pattern, provides a summary of these distances, and then plots the empirical cumulative distribution function (ECDF) to visualize the

spatial pattern of the points. This helps in understanding whether the points exhibit clustering, regularity, or randomness in their distribution.

```
nns <- nndist(dae_ppp)
#summary(nns)
plot(ecdf(nns))  # empirical cumulative distribution function
```

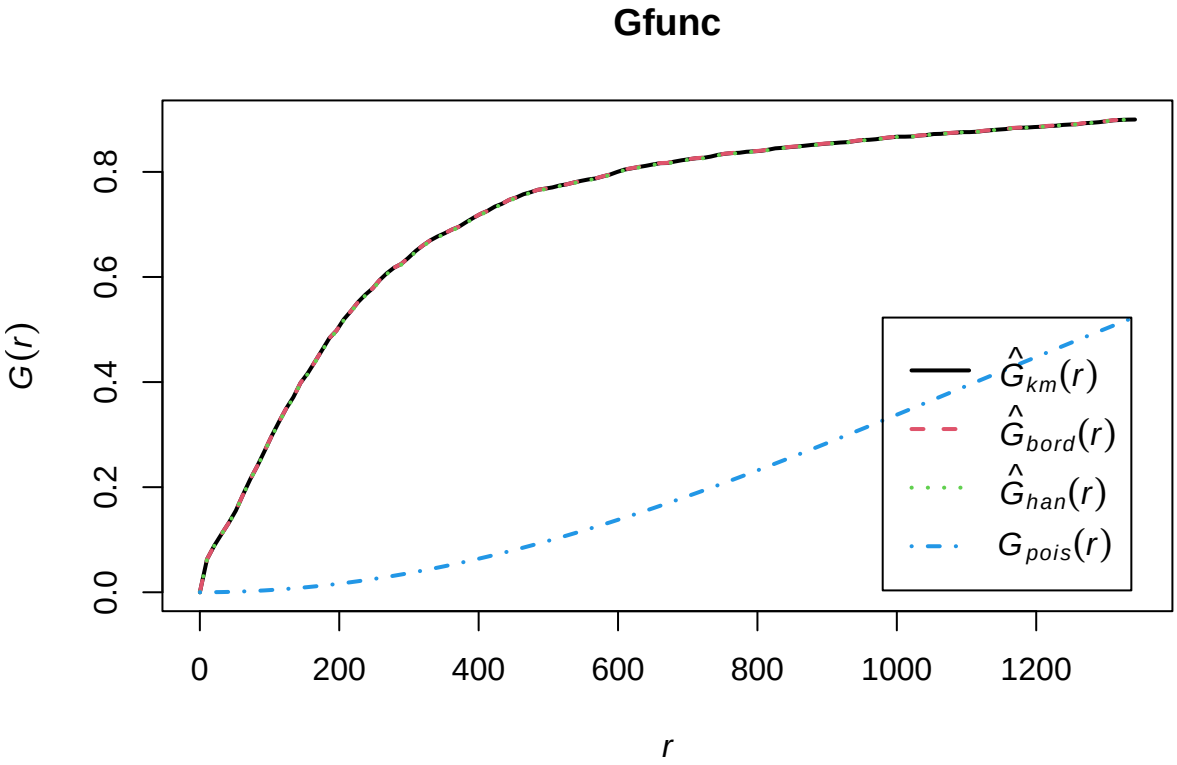


It climbs very steeply in the early part of its range before flattening out, then the indication would be an observed probability of short as opposed to long nearest neighbour distances, which suggest inter-event interaction (clustering) In about 5 km quite every DAE has a neighbor.

G function: event-event distance

Nearest Neighbour Distance Function G

```
Gfunc <- Gest(dae_ppp)
#Gfunc2 <- Gest(dae_ppp,correction = "none")
plot(Gfunc,lwd=2)
```



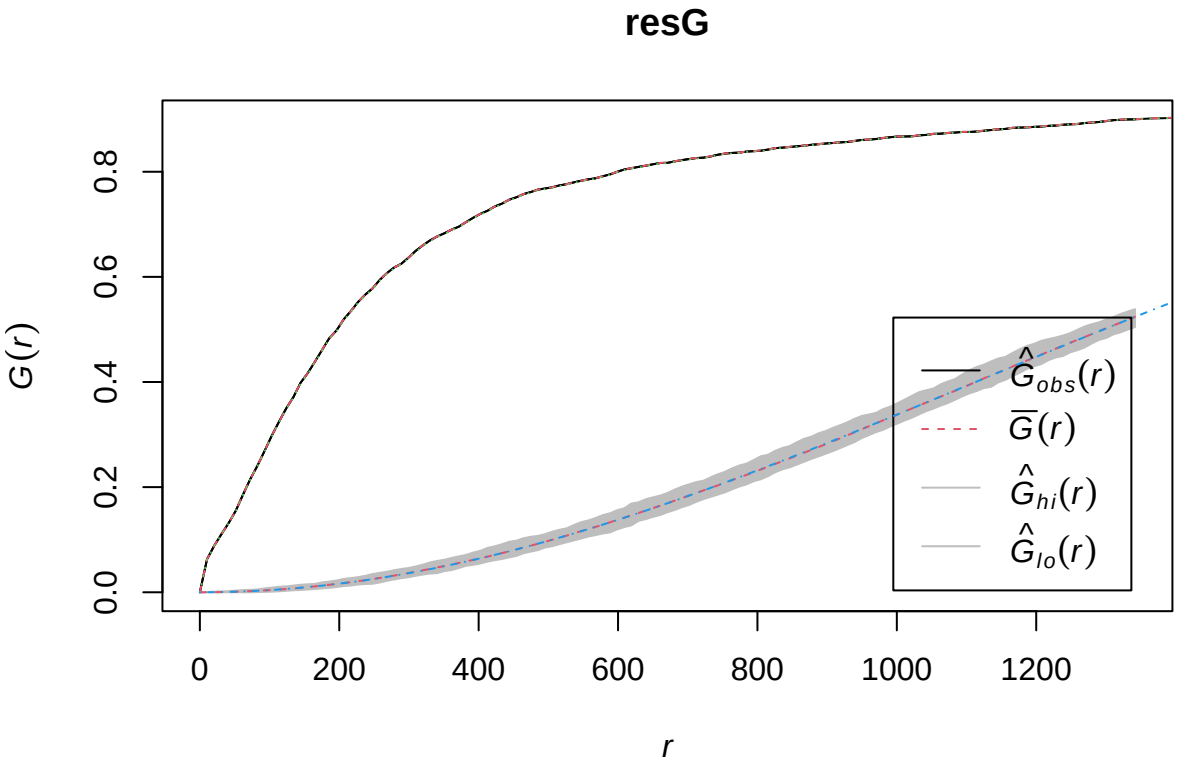
The scale is different but still visible the same level of clustering.

Might be low informative in this study considering the anthropic process, like in industry-plant location decision.

In this analysis F function could be more informative to see the distance between a random point and the nearest DAE in case of need! Somehow it can be a measure of how well distributed are DAE in the territory, of course it be evaluted on the amount of people present.

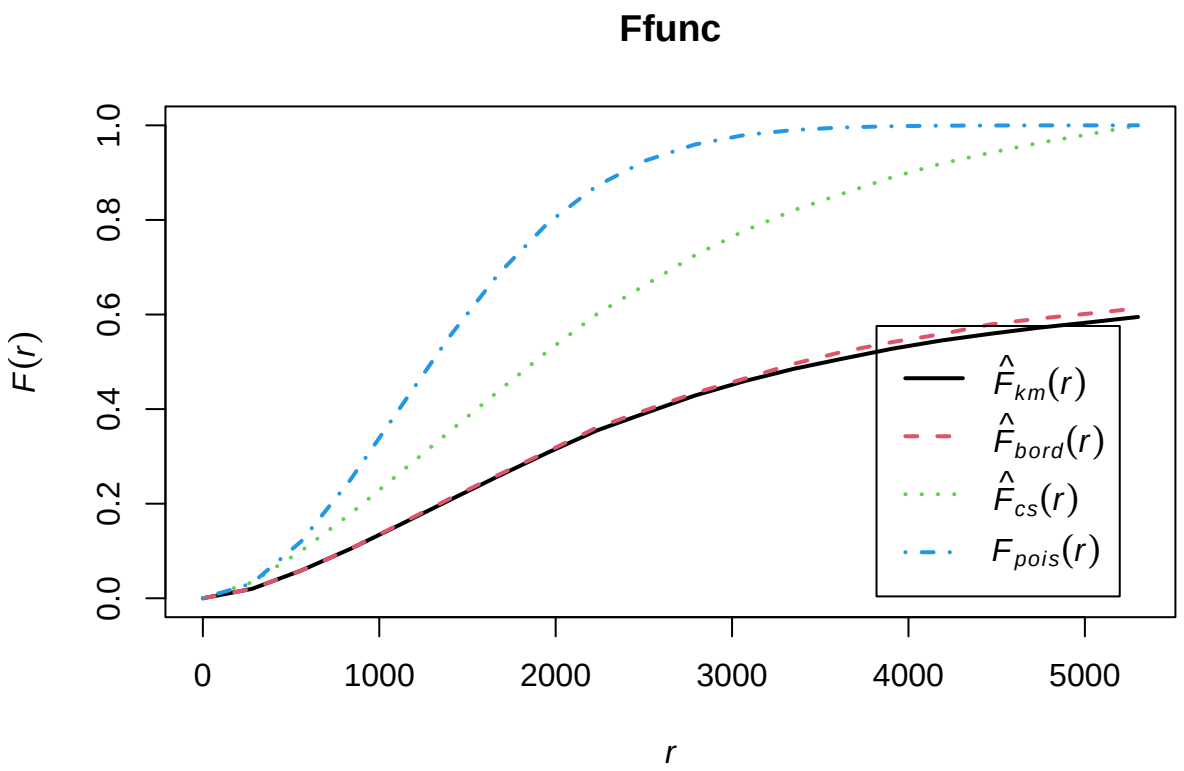
Comparision with a CSR as reference (Complete Spacial Randomness - homogeneous poisson process)

```
ex <- expression(runifpoint(dae_ppp$n, win = dae_ppp$window))
resG <- envelope(dae_ppp, Gest, nsim = 99, simulate = ex,
                 verbose = FALSE, saveall = TRUE)
plot(resG)
plot(Gfunc, add=T)
```

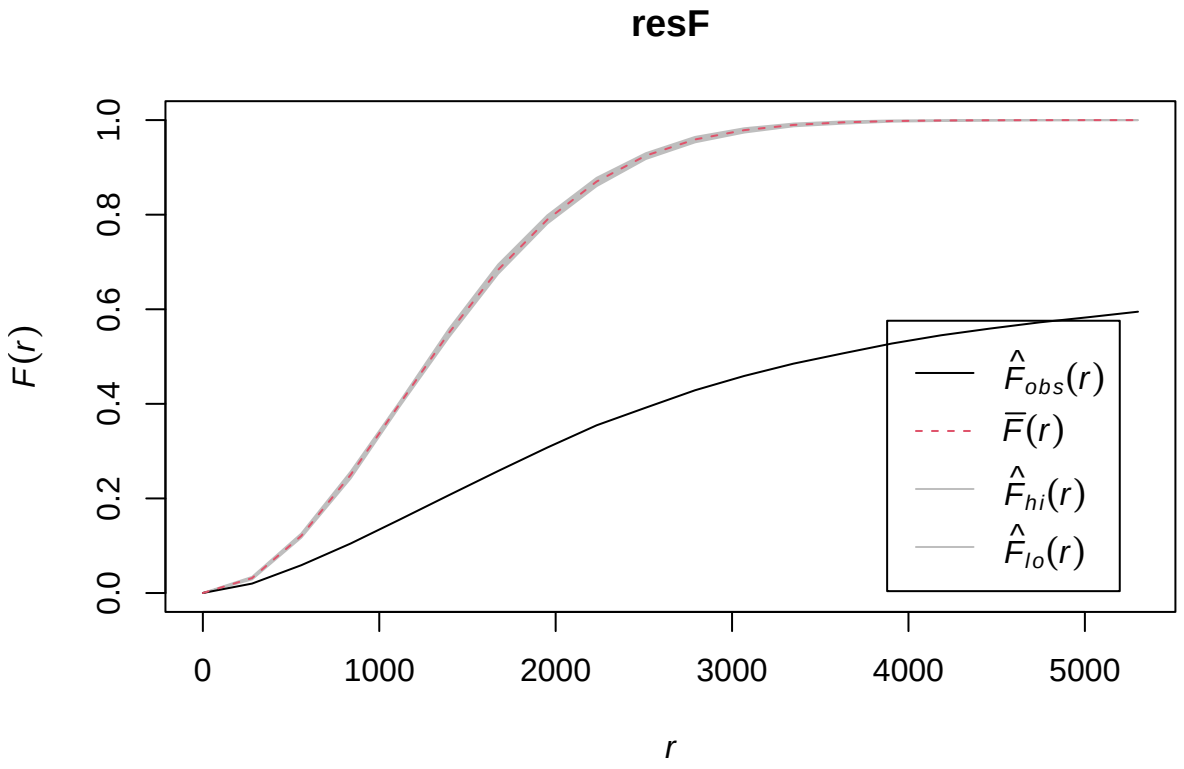


F function: distance between a point and an event (Estimate the Empty Space Function)

```
Ffunc <- Fest(dae_ppp)
plot(Ffunc, lwd=2)
```



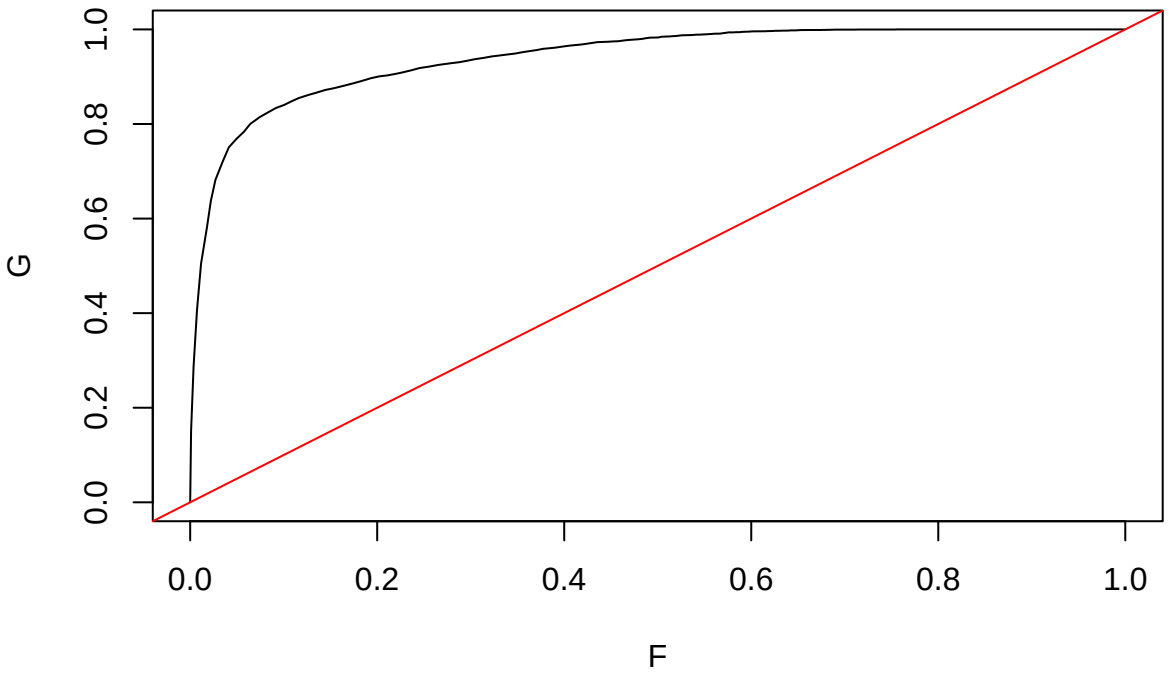

```
resF <- envelope(dae_ppp, Fest, nsim = 99, simulate = ex, verbose = FALSE, saveall = TRUE)
plot(resF)
```



It measures how many empty spaces are present between points on the map. Used for comparison with G function, from whose graph we can deduce further evidence of clustering, regularity or Randomness.

Comparison F - G functioncs

```
# Function to compare must be calculated for same distance values
dist <- seq(0, 40000, 50)
Foss <- Fest(dae_ppp,r=dist)
Goss <- Gest(dae_ppp,r=dist)

plot(Foss$rs, Goss$rs,
     xlim=c(0,1),ylim=c(0,1),
     type='l',xlab="F",ylab="G")

abline(a=0,b=1,col='red')
```



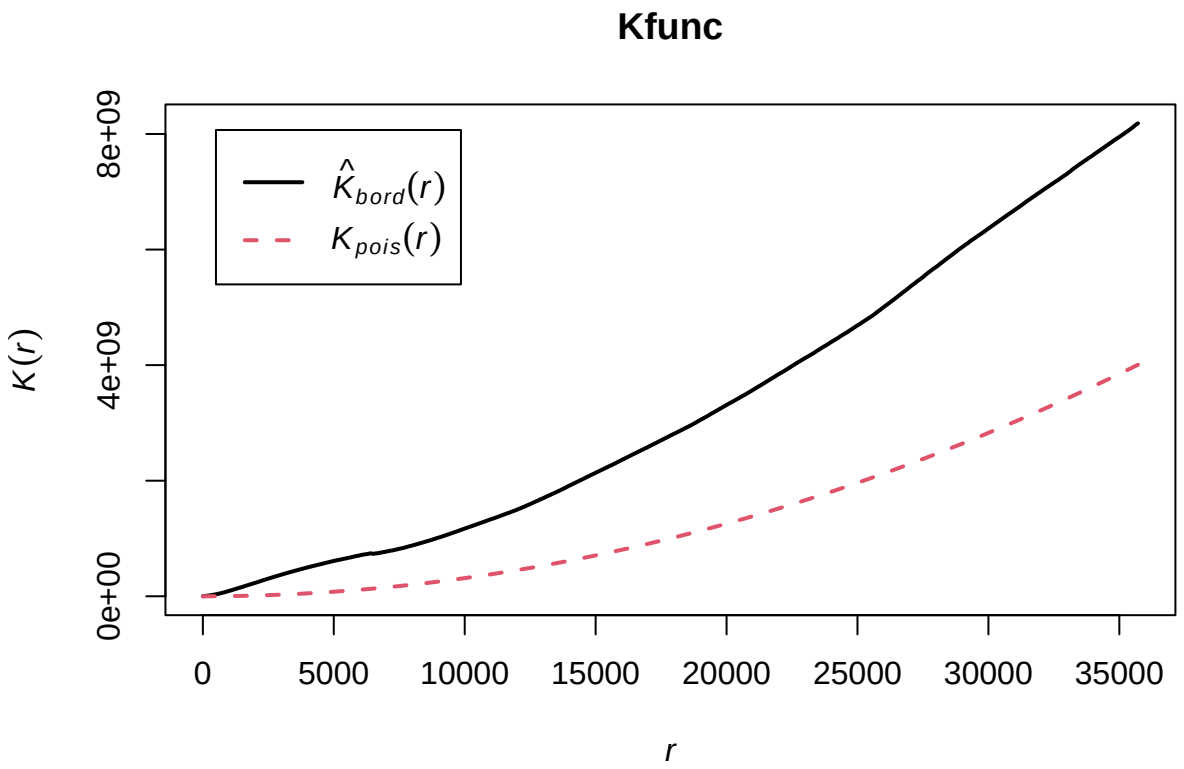
Red line is the reference case of CSR, so behind a random process. In this case there is clustering, while in case of observed curve below the red one there would be repulsivness, so regolarity.

K function

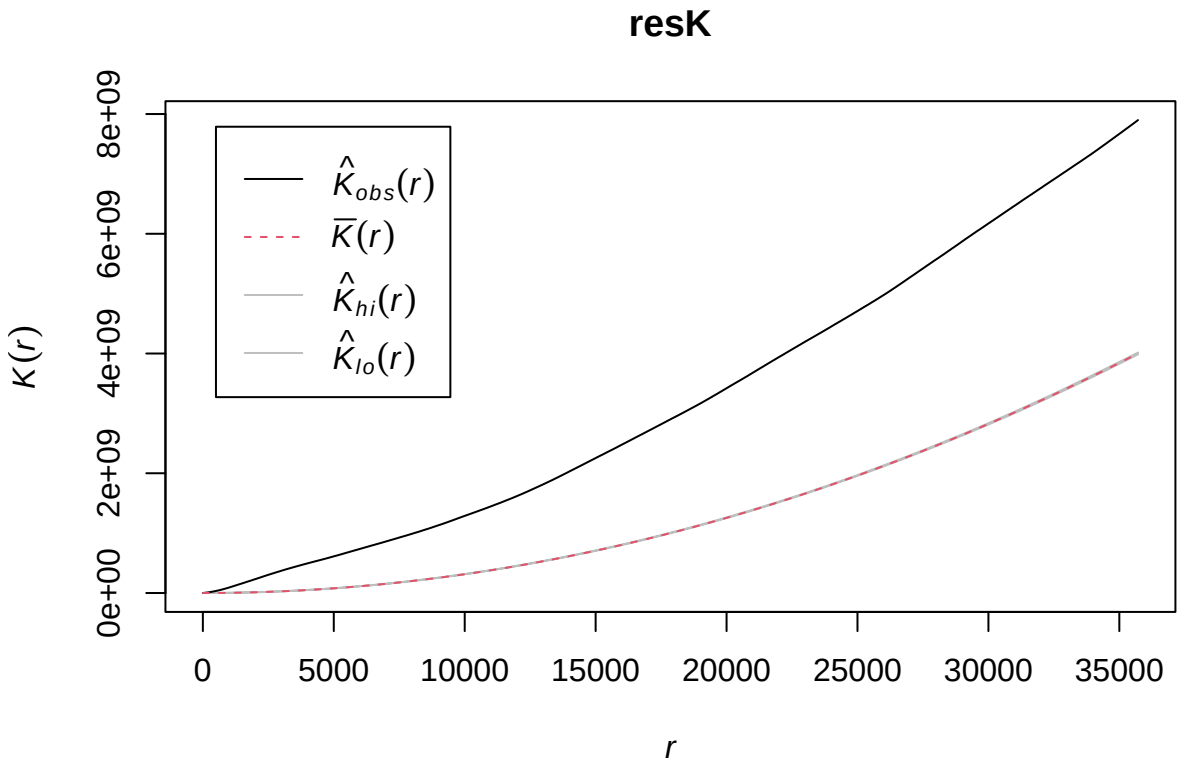
```
Kfunc <- Kest(dae_ppp)
```

```
## number of data points exceeds 3000 - computing border correction estimate only
```

```
plot(Kfunc,lwd=2)
```




```
resK <- envelope(dae_ppp, Kest, nsim = 9, simulate = ex,verbose = FALSE, saveall = TRUE)
plot(resK)
```



K function measures the distance from each point to all others, not only the nearest. This measure confirms the clustering, showing a curve above the red one made by CSR.

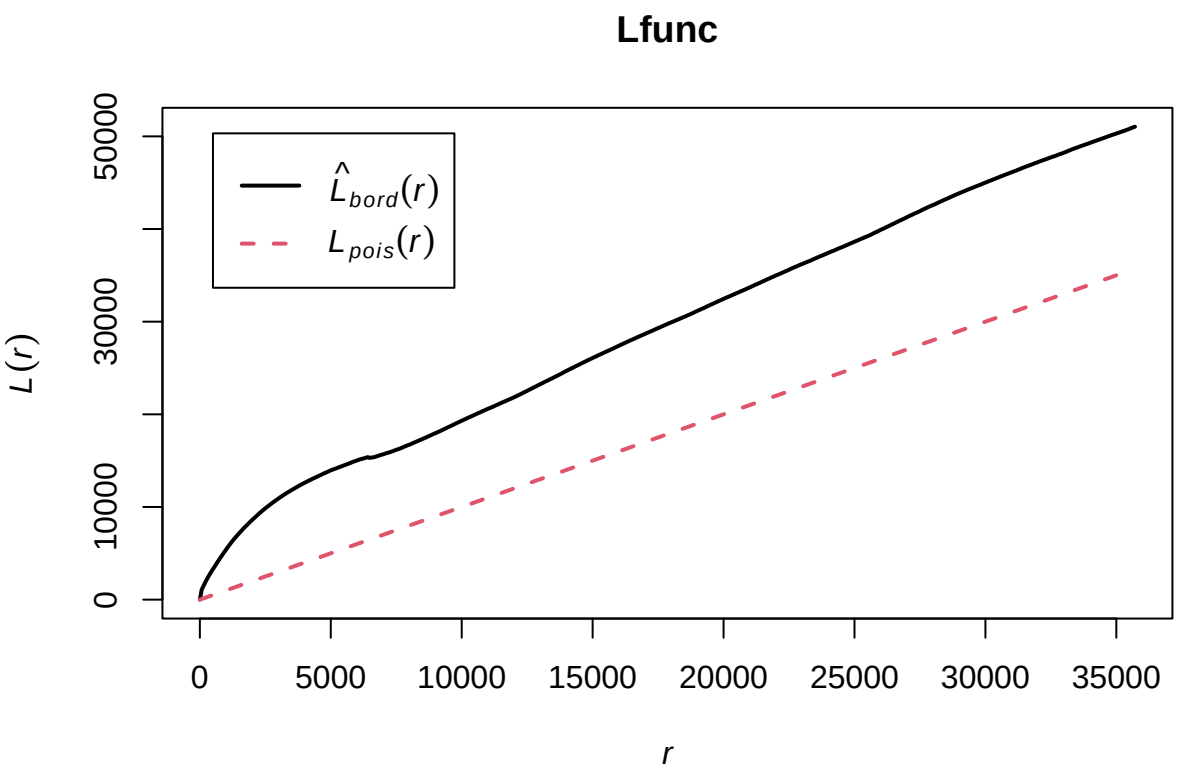
L function: transformation of Ripley's K-function

The point distribution under investigation is tested against the null hypothesis that the points are distributed randomly and independently.

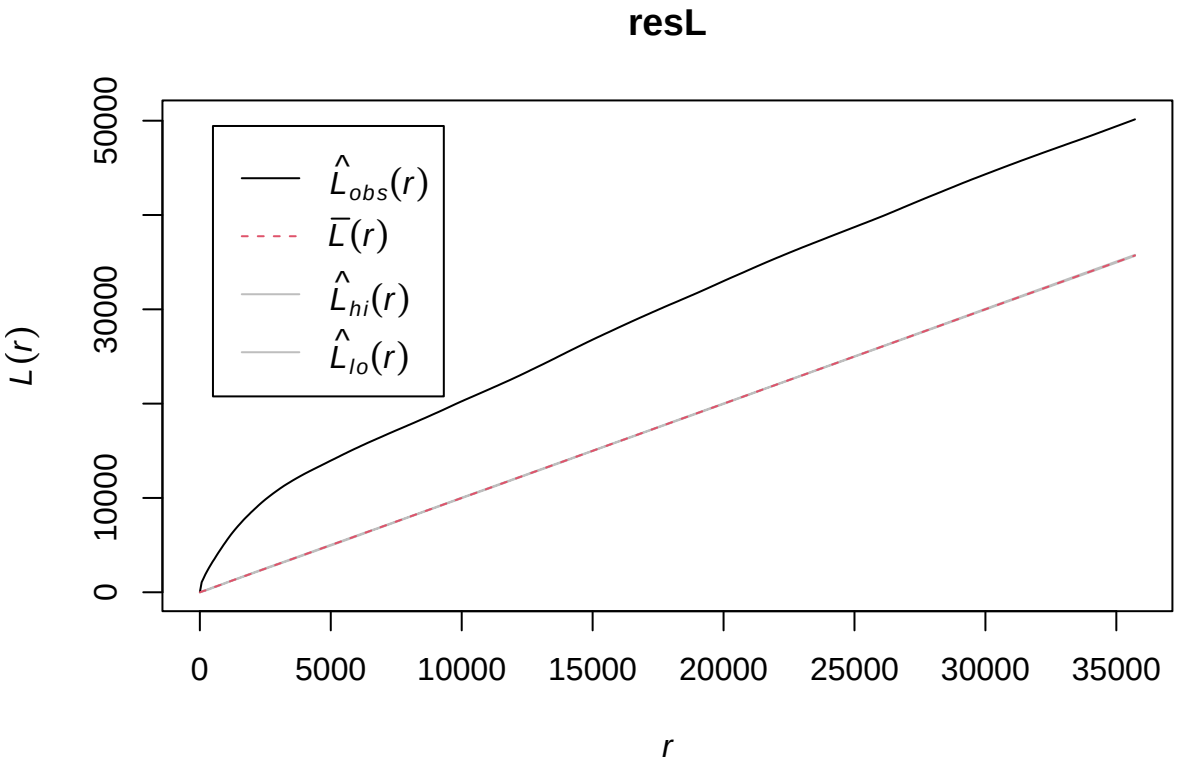
```
Lfunc <- Lest(dae_ppp)
```

```
## number of data points exceeds 3000 - computing border correction estimate only
```

```
plot(Lfunc,lwd=2)
```




```
resL <- envelope(dae_ppp, Lest, nsim = 9, simulate = ex, verbose = FALSE, saveall = TRUE)
plot(resL)
```



We can see: L(r) > r showing and confirming clustering of the point process.

Conclusion

In this analysis we have proved with different ways that DAE collocations present clustering. It could be interesting to go further in the analysis measuring the correlation between DAE and people present in the area, since I expect high correlation.