Heat-maps of street events

Mohammad Mehdi Moradi

Introduction

City management is engaged with knowing the city and having events happening in the city under control. For instance, Town-Hall or Police departments mayt be interested in managing the city based on the current situation with regards to events happening on the street such as traffic accident, street crime, etc. Having traffic accidents under control may help Town-Hall to better manage the streets and find the reasons of having more accidents in places than in others.

The aim of the code below is to find hot-spots based on the location of such events in the city streets. This function can show how the distribution of events spatially changes within the city. Knowing the location of hot-spots and the spatial variation of events within the city can help authorities to reconsider (if needed) their policy and also try to manage the city better. The importance of this analysis can be examplified as follows:

1. Considering the street crimes. Function `npdensity.lpp` can easily show the spatial variation of events within the cities' street and disclose the hot-spots. Knowing this, Police department may reconsider (if needed) their policies and also try to spread their stations out better within the city.
2. Assuming the case of dealing with traffic accident. Through the function `npdensity.lpp` we can figure out where traffic accidents mostly happen and as a result Town-Hall may aim at reconstructing the streets with higher intensity or change driving signs, e.g. adding or removing traffic light, making streets one way/two ways, enlarging roundabouts and so forth.

In what follows, we apply the function `npdensity.lpp` to a dataset of street crimes in an area in Chicago. This data reported in the period 25 April to 8 May 2002, in an area of Chicago (Illinois, USA) close to the University of Chicago. The data, part of R package spatstat, the R gives the spatial location of each crime, and the type of crime including assault, burglary, cartheft, damage, robbery, theft, trespass. All crimes occurred on or near a street. The data gives the coordinates of all streets in the survey area, and their connectivity.

Example: Analysing Chicago street crime data with `npdensity.lpp`

Data

In what follows, we apply the function `npdensity.lpp` to a dataset of street crimes in an area in Chicago. This data reported in the period 25 April to 8 May 2002, in an area of Chicago (Illinois, USA) close to the University of Chicago. The data, part of R package spatstat, the R gives the spatial location of each crime, and the type of crime including assault, burglary, cartheft, damage, robbery, theft, trespass. All crimes occurred on or near a street. The data gives the coordinates of all streets in the survey area, and their connectivity.

Preliminar steps

Note that, a pattern containing the location of events is considered as a spatial point pattern. In order to estimate the intensity function of a pattern of street events (traffic accident, street crime, etc) using R language programming, we first need to know some initial steps.

1. Make sure you have R and RStudio installed, If you do not, you can download them here and here, respectively.

2. To do spatial analysis, we may need to call some R packages, our needs may vary based on what analysis we do. We here need R packages "spatstat.utils", "spatstat".
3. Reading data in R depends on the format of data.
4. Input can be .shp, .RData, csv, etc.

If input data is a .shp file, you can easily convert it as an objetc of point patterns on linear networks (class `lpp`) using function below `shptolpp`. Note that, in order to create a point pattern on a network in R, you need to have two patterns, one is the point pattern and the other is the network. Thus, if your input is .shp, you must have two .shp files and through function below `shptolpp`, you can easliy create a point pattern on a network (class `lpp` in R package `spatstat`) in R.

Analysis

```r
library(spatstat)
library(spatstat.utils)

X <- unmark(chicago)
plot(X,main="",pch=20)

# First plot  shows location of crimes.
d <- npdensity.lpp(X,sigma = 100,delta = 3)
```

The code above generates the corresponding point pattern map below (Fig. 1).

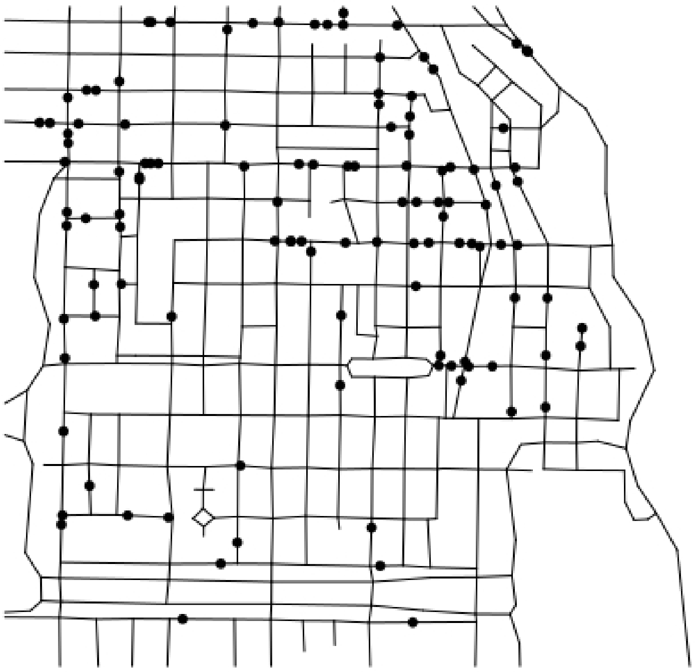

Figure 1: Figure 1. Location of crimes in the area of Chicago

```r
# intensity function is calculated using the function above and it is stored in variable d
plot(d,style = "w",main="")
# this generate the third plot which is the heat-map, plotted using style "width", here the wider the l
```

Figure 2 displays the spatial variation of street crimes in the network of Chicago. From the estimated intensity, it is clearly seen that street crimes are not distributed uniformly within the cities' streets. The estimated intensity shows that there are three hot-spots in the north while the south is quite safer area in comparison to the north. Note that, the bar column on the right side shows the values of estimated intensity being the higher the value, the more dangerous the street. Easily speaking, the wider the line, the higher the intensity.

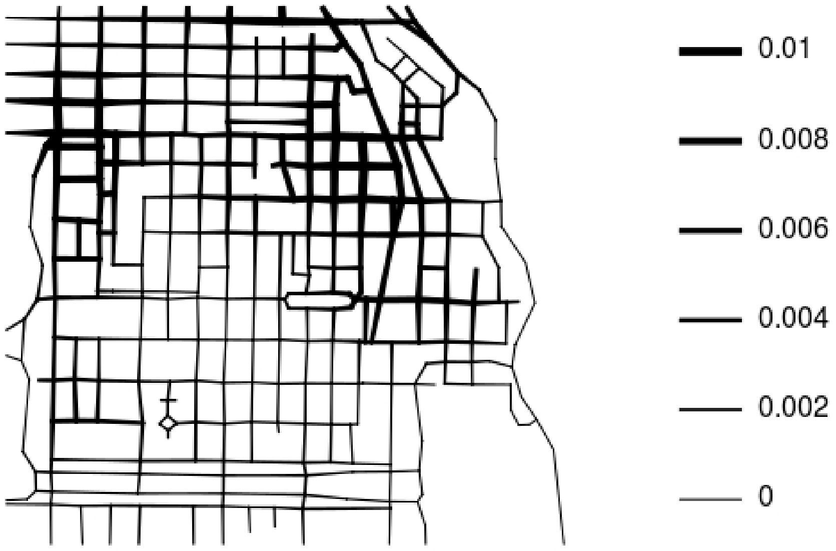

Figure 2: Figure 2. Spatial variation (intensity) of crimes in the street network of Chicago.

The R code and plots are available in the folloiwng Github repository.

Function `npdensity.lpp` documentation

Help information

Description

Estimates the intensity of a point process on a linear network by applying kernel smoothing to the point pattern data.

Details

Kernel smoothing is applied to the points of x using kerne-based estimator defined in Moradi et al. (2017).

Note

Computation time is increasing with the argument sigma. Also, computation may take time on big networks. You have been warned.

Author

Mohammad Mehdi Moradi.

Usage

npdensity.lpp(x, sigma, . . .)

Value

Pixel image on the linear network (class “linim”).

Arguments:

X

Point pattern on a linear network (object of class “lpp”) to be smoothed.

sigma

Smoothing bandwidth (standard deviation of the kernel) in the same units as the spatial coordinates of x. Note, small sigma may result in under-smoothing and large sigma may result in over-smoothing.

rp (optional)

Length of the sequance in approximating the edge correction.

dimyx (optional)

Pixel array dimensions.

delta

Tolerance value. A tail of the kernel after delta*sigma may be deleted.

Example:

```r
library(spatstat)
X <- runiflpp(3, simplenet)
d <- npdensity.lpp(X,sigma=0.1)
plot(d)
```

Reference:

Moradi. M. M., Rodriguez-Cortes, F. J. and Mateu, J. (2017). **On kernel-based intensity estimation of spatial point patterns on linear networks**. *Journal of Computational and Graphical Statistics*. DOI: 10.1080/10618600.2017.1360782

Source code

```r
library(spatstat.utils)
library(spatstat)
library(maptools)
#################################################################
shptolpp <- function(shpp,shpl){
  p <- readShapePoints(shpp)
  l <- readShapeLines(shpl)
  L <- as.linnet(l)
  p <- as.ppp(p)
  lp <- lpp(unmark(p),L)
  return(unique(lp))
}
#################################################################
npdensity.lpp <- function(X,sigma,rp=10,dimyx=NULL,delta=4){

  L <- as.linnet(X)

  Llines <- as.psp(L)
  linemask <- as.mask.psp(Llines,dimyx=dimyx)
```

```r
lineimage <- as.im(linemask,value=0)

xx <- raster.x(linemask)
yy <- raster.y(linemask)
mm <- linemask$m
xx <- as.vector(xx[mm])
yy <- as.vector(yy[mm])
pixelcentres <- ppp(xx, yy, window=as.rectangle(linemask), check=FALSE)
pixdf <- data.frame(xc=xx, yc=yy)

p2s <- project2segment(pixelcentres, Llines)
projloc <- as.data.frame(p2s$Xproj)
projmap <- as.data.frame(p2s[c("mapXY", "tp")])
projdata <- cbind(pixdf, projloc, projmap)

gridx <- pixelcentres$x
gridy <- pixelcentres$y
grid <- lpp(data.frame(gridx,gridy),L)


ox <- X$data$x  # Emerge the x-coordination
oy <- X$data$y  #Emerge the y-coordination
ngrid <- npoints(grid) # Emerge number of points
nrealization <- npoints(X)
dlpp <- crossdist.lpp(X,grid,method="C") # Distance matrix

if (missing(rp)){rp <- 10}

b <- delta*sigma
deltat <- b/rp
t <- seq(0.01,b,by=deltat)
t <- c(t,b)

npintlpp <- rep(0,0)

for(i in 1:ngrid){
  ldfp <- dlpp[,i]
  densfp <- 0

  for(j in 1:nrealization){
    if(ldfp[j] <= b){
      edge <- sum(unlist(lapply(X=1:length(t), function(k){
        dkernel(t[k],sd=sigma)*countends(L,c(ox[j],oy[j]),t[k])*deltat
      })))
      densfp <- densfp+(dkernel(ldfp[j],sd=sigma)/edge)
    }
  }
  npintlpp[i]=densfp
}

values <- npintlpp
Z <- lineimage
Z[pixelcentres] <- values
```

```r
  #df <- cbind(projdata, values)
  #density.diggle.lpp <- linim(L, Z, df=df)
  density.diggle.lpp <- linim(L, Z)

  return(density.diggle.lpp)
}
```