## Kernel Estimator ##

# Instalar pacotes

install.packages("rgdal")

install.packages("maptools")

install.packages("raster")

install.packages("spatstat")

install.packages("tmap")

install.packages("sf")

install.packages("gstat")

# Carregar pacotes

library(rgdal)

library(maptools)

library(raster)

library(spatstat)

library(tmap)

library(sf)

library(gstat)

library(sp)

# Setar pasta que contemm os dados

setwd("G:\\DOC201903\\01\_ANALISE\_ESPACIAL\\R\\Lab\_R\_SP\\dados")

# Shape de poligonos

pol <- readOGR("vetor/limites\_pol\_sp\_distrito.shp")

dist <- as(pol, "owin")

# Shape de pontos

point <- readOGR("vetor/acidentecet\_fatal\_truck.shp")

acid <- as(point, "ppp")

dist$xrange

dist$yrange

length(acid$y)

# Organizando os dados ----------------------------------------------------

# Remocao das marcas do dado de pontos

marks(acid) <- NULL

# "binding" limites do poligono com os dados de pontos

Window(acid) <- dist

# Plotando os dados dos pontos (acidentes) nos limites (distritos)

plot(acid, pch=15, cols="red", main=NULL)

# Arrumando a escala dos dados para km.

acid.km <- rescale(acid, 1000, "km")

dist.km <- rescale(dist, 1000, "km")

# Kernel density raster (com correcao de borda) ---------------------------------------------------

# Utilizando o 'bandwidth' (default)

K1 <- density(acid.km)

plot(K1, main=NULL, las=1)

contour(K1, add=TRUE)

# Utilizando 2km=bandwidth

K2 <- density(acid.km, sigma=2)

plot(K2, main=NULL, las=1)

contour(K2, add=TRUE)

# Utilizando 4km=bandwidth

K3 <- density(acid.km, sigma=4)

plot(K3, main=NULL, las=1)

contour(K3, add=TRUE)

# Funcao gaussiana para suavizacao padrao

# Alterar para quartic, disc ou epanechnikov

# Using a disc (padrao)

K4 <- density(acid.km, kernel = "disc")

plot(K4, main=NULL, las=1)

contour(K4, add=TRUE)

# Using a disc (padrao) with 2km bandwidth (adaptativo)

K5 <- density(acid.km, kernel = "disc", sigma=2)

plot(K5, main=NULL, las=1)

contour(K5, add=TRUE)

# Using a quartic (padrao)

K6 <- density(acid.km, kernel = "quartic")

plot(K6, main=NULL, las=1)

contour(K6, add=TRUE)

# Using a quartic with 2km bandwidth (padrao)

K7 <- density(acid.km, kernel = "quartic", sigma=2)

plot(K7, main=NULL, las=1)

contour(K7, add=TRUE)

# Kernel density raster (sem correcao de borda) ---------------------------------------------------

K8 <- density(acid.km, edge=FALSE)

plot(K8, main=NULL, las=1)

contour(K8, add=TRUE)

#Analise baseada em distancia --------------------------------------------

#Analise media do vizinho mais proximo

#calcular a primeira media de distancia do vizinho mais proximo (em quilometros) definida k=1

#mean(nndist(acid.km, k=1))

#calcular a segunda distancia media do vizinho mais proximo definida k=2

#mean(nndist(acid.km, k=2))

#Plotar a ANN (Average Nearest Neighbor) como uma funcao da ordem dos vizinhos para os 100 primeiros vizinhos mais proximos

#ANN <- apply(nndist(acid.km, k=1:100),2,FUN=mean)

#plot(ANN ~ eval(1:100), type="b", main=NULL, las=1)

# Funcoes

#gest <- Gest(acid, r=NULL, breaks=NULL, correction=c("rs"), domain=NULL)

kest <- Kest(acid.km)

lest <- Lest(acid.km)

# Computing Simulation envelopes of functions

# Nearest Neighbour Distance Function G:

ev\_gest <- envelope(acid.km, Gest, correction="rs", nsim = 10)

# K/L Function:

#ev\_kest <- envelope(acid, Kest, nsim = 10)

#ev\_lest <- envelope(acid, Kest, nsim = 10)

# Plot G(r)

plot(ev\_gest)

# Plot G(r) obs x CSR(theo)

plot(ev\_gest, obs ~ theo)

# Plot K Function

plot(kest)

# Plot L Function

plot(lest)