## 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)

# Fixar a pasta que contem os dados

setwd("C:/TEMP/SER301")

# Carregar shapefile de poligonos

pol <- readOGR("UC\_SG2000/UC\_PEGM\_SG2000.shp") # pasta do dataset/arquivo shp (pacote rgdal)

plot(pol)

uc <- as(pol, "owin") # forçar o objeto a pertencer a classe "polígono".

uc #"owin" = janela de observação com limites X e Y

# Carregar shapefile de pontos

point <- readOGR("Focos2020/Focos\_2020.shp") # pasta do dataset/arquivo shp (pacote rgdal)

foco <- as(point, "ppp") # forçar o objeto a pertencer a classe "ponto". ppp = point pattern dataset

uc$xrange # intervalo long mín e máx do polígono

uc$yrange # intervalo lat mín e máx do polígono

foco$n # retorna a variável n do dataframe foco (nº de pontos)

##### ORGANIZANDO OS DADOS ######

# Extraindo "marcas" do shape de pontos

marks(foco) <- NULL # remoção das colunas de atributos do shape

# Vinculando os limites do poligono com os dados de pontos

Window(foco) <- uc # remoção de qualquer objeto 'ppp' fora do polígono (recorte)

plot(foco, pch=10, cols="red", main=NULL)

# Arrumando a escala dos dados para km.

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

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

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

# Utilizando o τ adaptativo (default)

K0 <- density(foco.km, edge = TRUE) # densidade Kernel com largura de banda adaptativa e suavização de borda

plot(K0, main= "Kernel adaptativo (2020)", las=1)

contour(K0, add=TRUE)

# Utilizando o τ = 2km

K1 <- density(foco.km, sigma = 2, edge = TRUE, kernel = c('quartic')) # densidade Kernel quartico com largura de banda = 2km e suavização de borda

plot(K1, main="Kernel c/ largura de banda de 2km (2020)", las=1)

contour(K1, add=TRUE)

# Utilizando o τ = 4km

K2 <- density(foco.km, sigma = 4, edge = TRUE, kernel = c('quartic')) # densidade Kernel com largura de banda = 3km e suavização de borda

plot(K2, main="Kernel c/ largura de banda de 4km (2020)", las=1)

contour(K2, add=TRUE)

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

# Utilizando o τ adaptativo (default)

K3 <- density(foco.km, edge = FALSE) # densidade Kernel quartico com largura de banda adaptativa

plot(K3, main= "Kernel adaptativo sem correção de borda (2020)", las=1)

contour(K3, add=TRUE)

# Utilizando o τ = 2km

K4 <- density(foco.km, sigma = 2, edge = FALSE, kernel = c('quartic')) # densidade Kernel quartico com largura de banda = 2km

plot(K4, main="Kernel c/ largura de banda de 2km sem correção de borda (2020)", las=1)

contour(K4, add=TRUE)

# Utilizando o τ = 4km

K5 <- density(foco.km, sigma = 4, edge = FALSE, kernel = c('quartic')) # densidade Kernel com largura de banda = 3km

plot(K5, main="Kernel c/ largura de banda de 4km sem correção de borda (2020)", las=1)

contour(K5, add=TRUE)

######################### PARÂMETROS DE SEGUNDA ORDEM ###############################

# FUNÇÃO G

nsim = 50 # nº de simulações

ev\_gest <- envelope(foco.km, fun = Gest, correction="rs", nsim = nsim) # uper e lower envelope, "rs" = correção de borda para a função G

plot(ev\_gest, main=paste("Função G com",nsim,"simulações"))

# FUNÇÃO K

K <- Kest(foco.km, correction="best") # estimativa de K(r) com melhor correção de borda

plot(K, xlab="r(Km)", ylab="K(r)")

# FUNÇÃO L

n <- 50 #nº de simulações

EL <- envelope(foco.km[uc.km], Lest, nsim=n)

OP <- par(mar=c(5,5,4,4))

plot(EL, . - r ~ r, ylab=expression(hat("L")),xlab="Distance (km)", main=paste("Função L com",n,"simulações"))

par(OP)