4.2 Kernel Density Estimation (KDE) with least squares cross validation bandwidth selection (hlscv)

Manual of Applied Spatial Ecology

3/11/2022

Both the least squares cross-validation (hlscv) and bias crossed validation (hbcv) have been suggested instead of href in attempts to prevent over-smoothing of KDE (Rodgers and Kie 2010). However, (hlscv) and (hbcv) have been minimally evaluated on GPS datasets because previous literature only evaluated datasets collected on VHF sampling protocols or simulated data that included at most 1,000 locations. Least-squares cross validation, suggested as the most reliable bandwidth for KDE was considered better than plug-in bandwidth selection (hplug-in; for description see section 3.3) at identifying distributions with tight clumps but risk of failure increases with hlscv when a distribution has a "very tight cluster of points" or very large sample sizes (Gitzen et al. 2006, Pellerin et al. 2008, Walter et al. 2011).

1. Exercise 4.2 - Download and extract zip folder into your preferred location

2. Set working directory to the extracted folder in R under Session - Set Working Directory...

3. Now open the script "HlscvScript.Rmd" and run code directly from the script

4. First we need to load the packages needed for the exercise

```
library(adehabitatHR)
```

5. Now let's have a separate section of code to include projection information we will use throughout the exercise. In previous versions, these lines of code were within each block of code

```
utm.crs <- "+proj=utm +zone=17N +ellps=WGS84"
```

6. Now we can run fixed kernel home range with href bandwidth selection. The code below uses the original adehabitat package to run home range so skip if unable to load package

```
panther<-read.csv("pantherjitter.csv", header=T)
panther$CatID <- as.factor(panther$CatID)
#Note below the code uses the original adehabitat package to run home range
library(adehabitat)
loc <- panther[, c("X", "Y")]
## Estimation of UD for each animal separately
id <- panther[, "CatID"]
udbis <- kernelUD(loc, id, h = "href")
ud <- kernelUD(loc, id = id, h = "href", grid = 40, same4all = FALSE, hlim = c(0.1, 1.5),
    kern = c("bivnorm"), extent = 0.5)
image(ud) ## Note that the contours are under the locations

## Calculation of the 95 percent home range
ver <- getverticeshr(ud, 95)
plot(ver)

## Look at the estimates of home range by contour
```

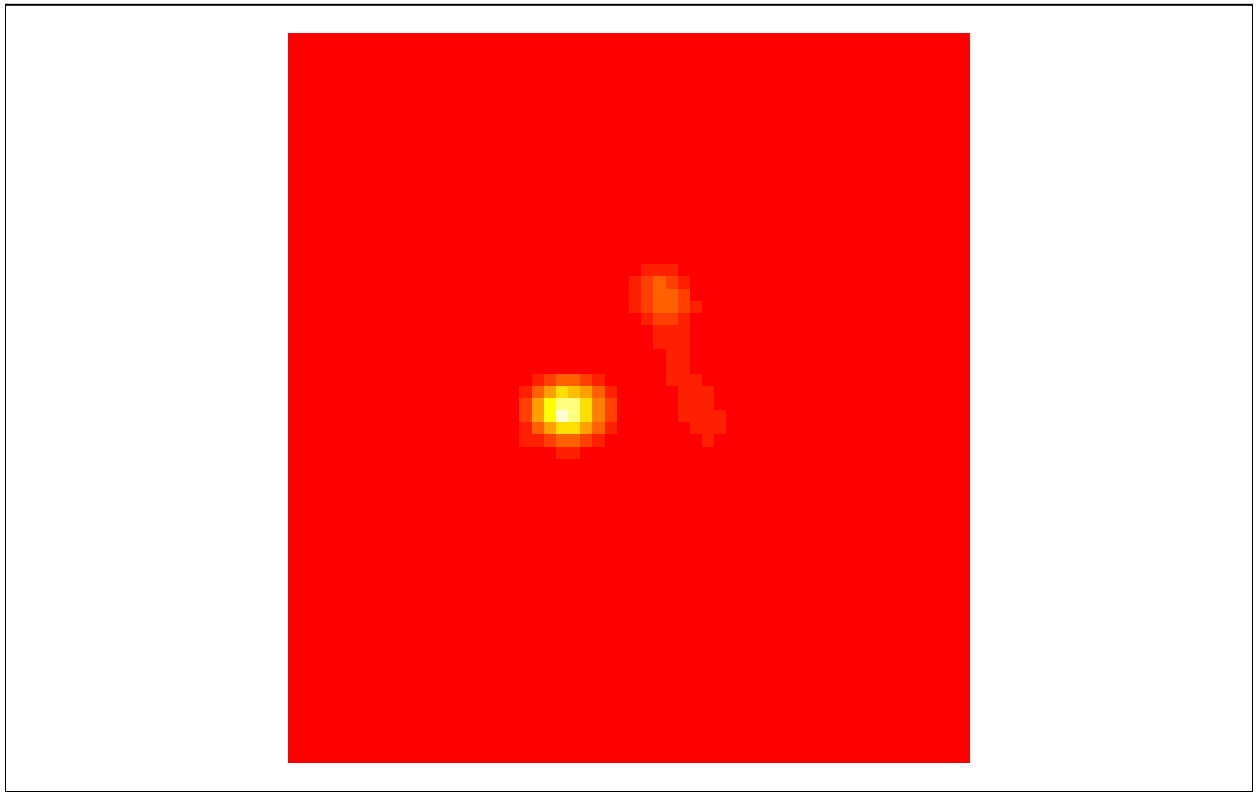
```
cuicui1 <- kernel.area(loc, id)
plot(cuicui1)
cuicui1
# write output
write.table(cuicui1,"output.csv", row.names=TRUE, sep=" ", col.names=TRUE, quote=TRUE,
  na = "NA")


##########################################################
# OVERRIDE the default kver2spol function so that we can
#include the projection info
##########################################################
kver2spol <- function(kv,projstr)
{
    x <- kv
    if (!inherits(x, "kver"))
        stop("x should be of class \"kver\"")
    if (!require(sp))
        stop("sp package needed")
    lipols <- lapply(1:length(x), function(i) {
        y <- x[[i]]
        class(y) <- c("data.frame", "list")
        res <- split(y[, 2:3], y[, 1])
        lipol <- lapply(res, function(z) {
            if (sum(abs(z[1, ] - z[nrow(z), ])) > 1e-16)
                z <- rbind(z, z[1, ])
            Polygon(as.matrix(z))
        })
        pols <- Polygons(lipol, ID = names(x)[i])
        return(pols)
    })
    return(SpatialPolygons(lipols, proj4string=CRS(as.character(projstr))))
}

#####################################################
# Function to export specific levels of isopleths of
# a "kv" object
#####################################################

#Code creates contours for each animal at each level
kv<-list()
class(kv) <- "kver"

kvtmp <- getverticeshr(udbis, lev = 99)
kv$KHR99<- kvtmp[[1]]
kvtmp <- getverticeshr(udbis, lev = 95)
kv$KHR95<- kvtmp[[1]]
kvtmp <- getverticeshr(udbis, lev = 90)
kv$KHR90<- kvtmp[[1]]
kvtmp <- getverticeshr(udbis, lev = 75)
kv$KHR75<- kvtmp[[1]]
kvtmp <- getverticeshr(udbis, lev = 50)
kv$KHR50<- kvtmp[[1]]
```

```
kvtmp <- getverticeshr(udbis, lev = 25)
kv$KHR25<- kvtmp[[1]]

spolTmp <- kver2spol(kv,"+proj=utm +zone=17 +ellps=WGS84")
dfTmp <- data.frame(Isopleth=c("99","95","90","75","50","25"),row.names=c("KHR99","KHR95",
  "KHR90","KHR75","KHR50","KHR25"))
spdfTmp <- SpatialPolygonsDataFrame(spolTmp, dfTmp, match.ID = TRUE)
library(rgdal)
writeOGR(spdfTmp,"HREF","FP048HREF", "ESRI Shapefile")


kvtmp <- getverticeshr(udbis, lev = 99)
str(kvtmp)
plot(kvtmp[[2]])
kv$KHR99<- kvtmp[[2]]
kvtmp <- getverticeshr(udbis, lev = 95)
kv$KHR95<- kvtmp[[2]]
kvtmp <- getverticeshr(udbis, lev = 90)
kv$KHR90<- kvtmp[[2]]
kvtmp <- getverticeshr(udbis, lev = 75)
kv$KHR75<- kvtmp[[2]]
kvtmp <- getverticeshr(udbis, lev = 50)
kv$KHR50<- kvtmp[[2]]
kvtmp <- getverticeshr(udbis, lev = 25)
kv$KHR25<- kvtmp[[2]]

spolTmp <- kver2spol(kv,"+proj=utm +zone=17N +ellps=WGS84")
dfTmp <- data.frame(Isopleth=c("99","95","90","75","50","25"),row.names=c("KHR99","KHR95",
    "KHR90","KHR75","KHR50","KHR25"))
spdfTmp <- SpatialPolygonsDataFrame(spolTmp, dfTmp, match.ID = TRUE)
writeOGR(spdfTmp,"HREF","FP094HREF", "ESRI Shapefile")
```

7. Using the adehabitatHR package requires dataset to be formatted differently than previous package

```
#Let's select only one animal
panther<-read.csv("pantherjitter.csv", header=T)
panther <- subset(panther, panther$CatID == "143")
panther$CatID <- factor(panther$CatID)
loc <- data.frame("x"=panther$X,"y"=panther$Y)
cats <- SpatialPointsDataFrame(loc,panther, proj4string = CRS(utm.crs))
udbis <- kernelUD(cats[,1], h = "href")
image(udbis)
```

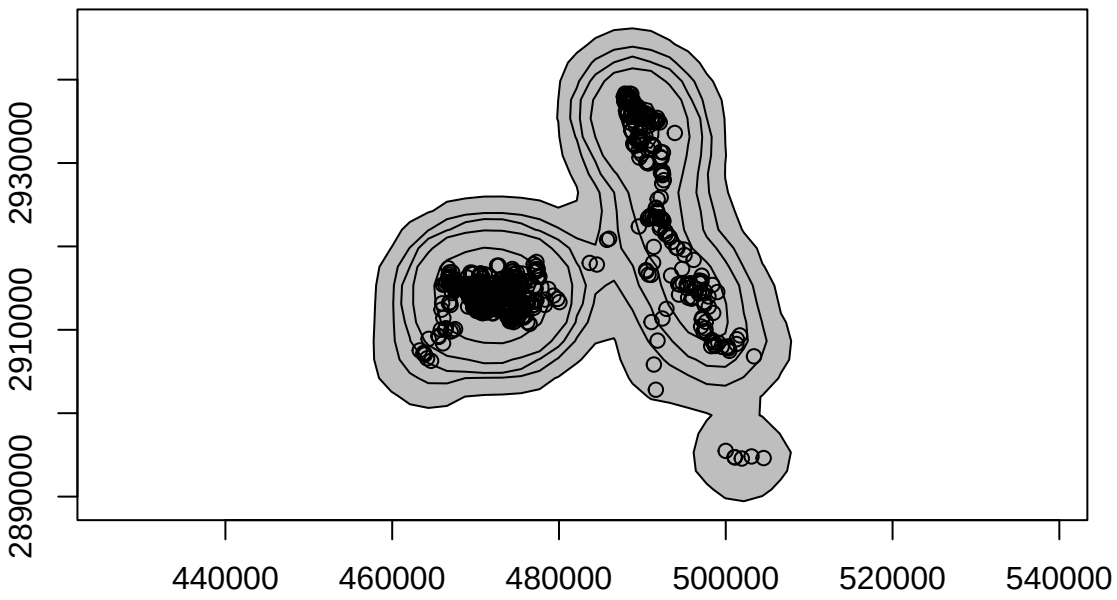
143



```
ver <- getverticeshr(udbis, standardize = FALSE)
ver50 <- getverticeshr(udbis, percent=50)
ver80 <- getverticeshr(udbis, percent=80)
ver90 <- getverticeshr(udbis, percent=90)
ver95 <- getverticeshr(udbis, percent=95)
ver99 <- getverticeshr(udbis, percent=99)
ver
```

```
## Object of class "SpatialPolygonsDataFrame" (package sp):
##
## Number of SpatialPolygons:  1
##
## Variables measured:
##      id     area
## 143 143 97882.18
```

```
plot(ver99, col="grey",axes=T);plot(ver95, add=T);plot(ver90, add=T);plot(ver80, add=T)
plot(ver50, add=T)
points(cats)
```


2890000
2910000
2930000
440000
460000
480000
500000
520000
540000