3.3 Distance Between Locations

Manual of Applied Spatial Ecology

1/21/2019

Determining the distance between locations or between locations and respective habitat types can serve a variety of purposes. Several resource selection procedures require a description of the daily movement distance of an animal to determine the habitat available to an animal or when generating random locations around known locations. We will start here with a method to determine the average distance moved by mule deer in Colorado in a study to determine methods to alleviate depradation on sunflowers that have become a high commodity crop in the area.

1. Exercise 3.3 - 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 "DistanceUniqueBurst.Rmd" and run code directly from the script

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

```
library(adehabitatLT)
library(chron)
library(class)
```

5. Code to read in dataset then subset for an individual animal

```
muleys <-read.csv("DCmuleysedited.csv", header=T)
#Code to select an individual animal
muley15 <- subset(muleys, id=="D15")
table(muley15$UTM_Zone,muley15$id)
```

```
##
##         D12  D15  D16  D19   D4   D6   D8
##   12S     0 2589    0    0    0    0    0
##   13S     0    0    0    0    0    0    0
##   13T     0    0    0    0    0    0    0
```

```
muley15$id <- droplevels(muley15$id)

#Sort data to address error in code and then look at first 20 records of data to confirm
muley15 <- muley15[order(muley15$GPSFixTime),]
#Run code to display the first 20 records to look at what sorting did to data
```

6. Prepare data to create trajectories using the ltraj command in Adehabitat LT

```
######################################################
#Example of a trajectory of type II (time recorded) with conversion of the date to the
#format POSIX that nNeeds to be done to get proper digits of date into R then POSIXct
#uses library(chron)
da <- as.character(muley15$GPSFixTime)
da <- as.POSIXct(strptime(muley15$GPSFixTime,format="%Y.%m.%d %H:%M:%S"))
head(da)
```

```
## [1] "2011-10-12 00:02:03 EDT" "2011-10-12 03:00:52 EDT"
## [3] "2011-10-12 06:00:52 EDT" "2011-10-12 09:00:38 EDT"
## [5] "2011-10-12 12:00:34 EDT" "2011-10-12 15:00:42 EDT"
```

```r
#Attach da to muley15
muley15$da <- da

timediff <- diff(muley15$da)
muley15 <-muley15[-1,]
muley15$timediff <-as.numeric(abs(timediff))

#Clean up muley15 for outliers
newmuleys <-subset(muley15, muley15$X > 599000 & muley15$X < 705000 & muley15$Y > 4167000
  & muley15$timediff < 14401)
muley15 <- newmuleys
```

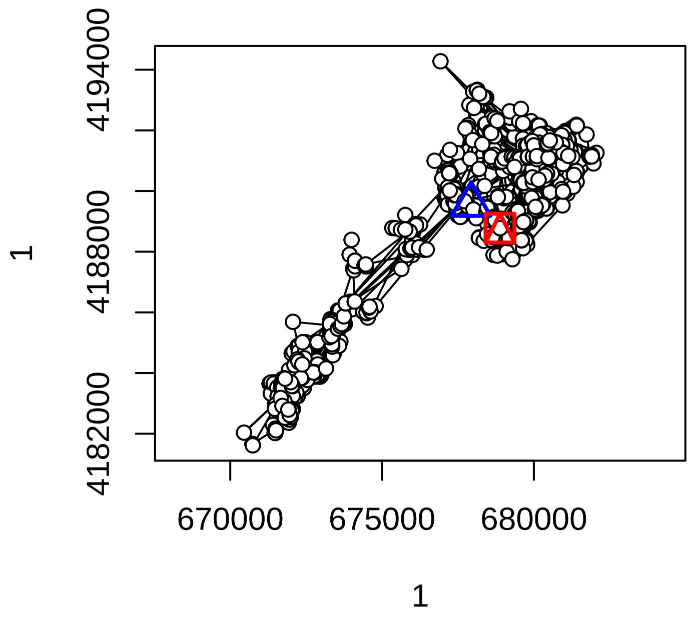
7. Create a spatial data frame of locations for muley 15 for use in creating trajectories that includes time difference between locations and dates in proper format (as.POSIXct)

```r
data.xy = muley15[c("X","Y")]
#Creates class Spatial Points for all locations
xysp <- SpatialPoints(data.xy)
proj4string(xysp) <- CRS("+proj=utm +zone=12 +ellps=WGS84")

#Creates a Spatial Data Frame from
sppt<-data.frame(xysp)
#Creates a spatial data frame of ID
idsp<-data.frame(muley15[2])
#Creates a spatial data frame of dt
dtsp<-data.frame(muley15[24])
#Creates a spatial data frame of Burst
busp<-data.frame(muley15[23])
#Merges ID and Date into the same spatial data frame
merge<-data.frame(idsp,dtsp,busp)
#Adds ID and Date data frame with locations data frame
coordinates(merge)<-sppt
```

8. Create an object of class "ltraj" for muley15 dataset

```r
ltraj <- as.ltraj(coordinates(merge),merge$da,id=merge$id)
plot(ltraj)
```



```
#Now let's look at time differences between locations before moving forward
summary(muley15$timediff)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.998   2.998   3.000   3.008   3.002   6.004
```

9. Need to create separate "bursts" for each trajectory based on the number of locations collected each day. In our case it was 8 (i.e., locations collected every 3 hours during a 24-hour period).

```
## We want to study the trajectory of the day at the scale of the day. We define one trajectory
#per day. The trajectory should begin at 2200 hours so the following function returns TRUE if
#the date is time between 06H00 and 23H00 (i.e. results in 7-8 locations/day bursts)
foo <- function(date) {
da <- as.POSIXlt(date)
ho <- da$hour + da$min
return(ho>15.9&ho<23.9)
}
deer <- cutltraj(ltraj, "foo(date)", nextr = TRUE)
```

```
## Warning in cutltraj(ltraj, "foo(date)", nextr = TRUE): At least 3 relocations are needed for a burst
##  345 relocations have been deleted
```

```
#Notice that the above code will remove 345 relocations that fall
#outside of your time criteria
#Warning message:
#In cutltraj(ltraj, "foo(date)", nextr = TRUE) :
#  At least 3 relocations are needed for a burst
# 345 relocations have been deleted

head(deer)
```

```
##
```

```
## *********** List of class ltraj ***********
##
## Type of the traject: Type II (time recorded)
## * Time zone unspecified: dates printed in user time zone *
## Irregular traject. Variable time lag between two locs
##
## Characteristics of the bursts:
##    id   burst nb.reloc NAs          date.begin            date.end
## 1 D15 D15.001        6   0 2011-10-12 03:00:52 2011-10-12 18:00:52
## 2 D15 D15.003        7   0 2011-10-13 00:00:35 2011-10-13 18:00:35
## 3 D15 D15.005        7   0 2011-10-14 00:00:42 2011-10-14 18:00:42
## 4 D15 D15.007        7   0 2011-10-15 00:00:35 2011-10-15 18:00:45
## 5 D15 D15.009        7   0 2011-10-16 00:00:39 2011-10-16 18:00:49
## 6 D15 D15.011        6   0 2011-10-17 00:01:07 2011-10-17 15:01:03
##
##
##  infolocs provided. The following variables are available:
## [1] "pkey"
```

10. Code to change ltraj to a data.frame to summarize distance between locations for each daily burst

```
dfdeer <- ld(deer)
head(dfdeer)

#Code to get mean distance moved for each burst
dfdeer <- subset(dfdeer, !is.na(dfdeer$dist))#remove NAs from last location of a burst
mean_dist <- do.call(data.frame, aggregate(dfdeer$dist, by=list(dfdeer$burst),
    function(x) c(mean = mean(x), sd = sd(x), n=abs(length(x)))))

#Write.csv gives csv output of Summary
write.csv(mean_dist, file = "Distance.csv")
```