Chapter 3

Movement Methods

Contents

3.1 Importing datasets from a web source . . . 43
3.2 Movement trajectories . . . 43
3.3 Distance between locations . . . 46
3.4 First Passage Time (FPT) . . . 50
3.5 Regular trajectories . . . 55
3.6 Net Squared Displacement . . . 60
3.7 Movement Trajectory Animation . . . 69

Figures

3.1 Example of a trajectory created using adehabitatLT for a mule deer in Colorado. . . . 52
3.2 Plot of a First Passage Time for a mule deer in Colorado identifying mean FPT by month. . . . 53
3.3 Summaries of distance and time (dt) between relocations for mule deer D16. . . . 56
3.4 Bursts of movements for mule deer D15 after creating segments based for focal use areas. . . . 60
3.5 Commands to animate a trajectory using trajdyn function . . . 71

Movement methods can serve a variety of purposes from determining mean daily distance moved by an animal to describing the scale at which an animal uses the landscape. Understanding movements can often shed some light on how an animal uses the landscape based on differences in turning angles and clustering of paths in each defined habitat type. Trajectories can be created from relocations and are also the precursor to several home range estimation methods we will go over later in the course.

Notes from the package AdehabitatLT manual (Calenge 2012): Two types of trajectories can be stored in objects of class ltraj: trajectories of type I correspond to trajectories where the time of relocations is not recorded. It may be because it could not be noted at the time of sampling (e.g. sampling of animals' tracks in the snow) or because it was decided that they did not want to take it into account, i.e. to study only its geometrical properties. In this case, the variable date in each burst of the object contains a vector of integer giving the order of the relocations in the trajectory (i.e. 1, 2, 3, ...). Trajectories of type II correspond to trajectories for which the time is available for each relocation. It is stored as a vector of class POSIXct in the column date of each burst of relocations. The type of trajectory should be defined when the object of class ltraj is defined, with the argument type II.

Concerning trajectories of type II, in theory, it is expected that the time lag between two relocations is constant in all the bursts and all the ids of one object of class ltraj (i.e., do not mix animals located every 10 minutes and animals located every day in the same object). Indeed, some of the descriptive parameters of the trajectory do not have any sense when the

time lag varies. For example, the distribution of relative angles (angles between successive moves) depends on a given time scale; the angle between two during 10-min moves of a whitestork does not have the same biological meaning as the angle between two 1-day moves. If the time lag varies, the underlying process varies too. For this reason, most functions of adehabitatLT have been developed for "regular" trajectories, i.e. trajectories with a constant time lag (see *help(sett0)*).

3.1 Importing datasets from a web source

Movebank.org is a new storage warehouse for relocation data from GPS-collared or VHF-collared animals. It provides a storage facility in the cloud that can serve as a backup for your data or a transfer portal to share data among colleagues or interested researchers. Similar to any email account, each user has a Movebank account that has a login and password to gain access to your data. Administration privileges can be given to anyone with an account for viewing and downloading data.

1. Exercise 3.1 - Download and extract zip folder into your preferred location
2. Set working directory to the extracted folder in R under File - Change dir...
3. First we need to load the packages needed for the exercise

```
library(move)
library(RCurl)
library(circular)
```

4. Now open the script "MBvultureCode.R" and run code directly from the script
5. Let's aslo go to the Movebank home page and explore what it has to offer
6. Login to Movebank

```
login <- movebankLogin(username="wdwalter", password="xxxxxx")
```

7. Access a stored dataset in Movebank of vulture data

```
getMovebankAnimals(study="Turkey Vulture South Carolina USA",login=login)

#Give the dataset a name while identifying each bird by ID in dataset
turkey <- getMovebankData(study="Turkey Vulture South Carolina USA",
animalName=c("Bird51","Bird52","Bird54","Bird55a","Bird56","Bird59a","Bird60a"),
login=login, moveObject=TRUE)

#Check to see the number of locations for each bird
n.locs(turkey)
# Bird51  Bird52  Bird54 Bird55a  Bird56 Bird59a Bird60a
#   9655   11456    2378    2228    5876    8311    8594
```

3.2 Movement trajectories

We will start with simply creating trajectories between successive locations. As stated above, there are 2 types of trajectories but their are also 2 forms of Type II trajectories if we have time recorded. Depending on the duration between locations we can have uniform time lag between successive relocations termed *regular* trajectories and non-uniform time lag that

results in *irregular* trajectories. We will begin this section with simply creating irregular trajectories from relocation data because, even though we set up a time schedule to collection locations at uniform times, climate, habitat, and satellites do not always permit such schedules of data collection.

1. Exercise 3.2 - Download and extract zip folder into your preferred location
2. Set working directory to the extracted folder in R under File - Change dir...
3. First we need to load the packages needed for the exercise

```
library(adehabitatLT)
library(chron)
library(spatstat)#for "duplicate" function
```

4. Now open the script "MovementsScript.R" and run code directly from the script

```
#We are again going to be using more of the mule deer dataset than from the
#earlier exercises
muleys <-read.csv("DCmuleysedited.csv", header=T)
str(muleys)
```

5. Check for duplicate locations in dataset. The reason for this is very important and will be apparent shortly.

```
summary(duplicated(muleys))

#Sort data to address error in code if needed
#muleys <- muleys[order(muleys$id),]
```

6. For trajectories of type II (time recorded), the conversion of the date to the format POSIX needs to be done to get proper digits of date into R.

```
da <- as.POSIXct(strptime(muleys$GPSFixTime,format="%Y.%m.%d %H:%M:%S"))
head(da)
muleys$da <- da #attach "da" to muleys dataset
str(muleys)

#Create time lag between successive locations to censor data if needed.
timediff <- diff(muleys$da)
muleys <-muleys[-1,]
muleys$timediff <-as.numeric(abs(timediff))
str(muleys)#check to see timediff column was added to muleys

#Look at number of locations by animal ID
summary(muleys$id)

#Remove outlier locations or known outliers collected too far apart in time
newmuleys <-subset(muleys, muleys$Long > -110.50 & muleys$Lat > 37.3 &
   muleys$Long < -107)
muleys <- newmuleys
str(muleys)
```

7. Let's create a Spatial Points Data Frame in UTM zone 12 adding ID, time diff, burst to xy coordinates

```
data.xy = muleys[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(muleys[2])
#Creates a spatial data frame of dt
dtsp<-data.frame(muleys[24])
#Creates a spatial data frame of Burst
busp<-data.frame(muleys[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

plot(merge)#visualize data
str(merge)
```

8. Now create an object of class "ltraj" by animal using the ID field and display by each individual (i.e., ltraj[1]).

```
ltraj <- as.ltraj(coordinates(merge),merge$da,id=merge$id)
plot(ltraj)
plot(ltraj[1])
head(ltraj[1])#Describes the trajectory for the first deer

#> head(ltraj[1])#Describes the trajectory
********** List of class ltraj **********
#Type of the traject: Type II (time recorded)
#Irregular traject. Variable time lag between two locs
#
#Characteristics of the bursts:
#   id burst nb.reloc NAs          date.begin            date.end
#1 D12   D12      100   0 2011-10-12 06:00:52 2011-10-24 21:00:48
#
 #infolocs provided. The following variables are available:
#[1] "pkey"

plot(ltraj[2])
plot(ltraj[3])
plot(ltraj[4])
plot(ltraj[5])
plot(ltraj[6])
plot(ltraj[7])
```

9. Let's create a histogram of time lag (i.e., interval) and distance between successive locations for each deer. This is a nice way to inspect the time lag between locations as you don't want to include a location if too much time has passed since the previous and it also shows why a trajectory is *irregular*.

```
hist(ltraj[1], "dt", freq = TRUE)
windows()#opens a new window to show both figures
hist(ltraj[1], "dist", freq = TRUE)

windows()
hist(ltraj[2], "dt", freq = TRUE)
windows()
hist(ltraj[2], "dist", freq = TRUE)
windows()

hist(ltraj[3], "dt", freq = TRUE)
windows()
hist(ltraj[3], "dist", freq = TRUE)
windows()

hist(ltraj[4], "dt", freq = TRUE)
windows()
hist(ltraj[4], "dist", freq = TRUE)
windows()

hist(ltraj[5], "dt", freq = TRUE)
windows()
hist(ltraj[5], "dist", freq = TRUE)
windows()

hist(ltraj[6], "dt", freq = TRUE)
windows()
hist(ltraj[6], "dist", freq = TRUE)
windows()

hist(ltraj[7], "dt", freq = TRUE)
windows()
hist(ltraj[7], "dist", freq = TRUE)
```

3.3 Distance between locations

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 File - Change dir...
3. First we need to load the packages needed for the exercise

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

```
library(class)
library(Rcmdr)
```

4. Now open the script "DistanceUniqueBurst.R" and run code directly from the script

```
muleys <-read.csv("DCmuleysedited.csv", header=T)
str(muleys)
```

5. Code to subset dataset for an individual animal

```
muley15 <- subset(muleys, id=="D15")
str(muley15)
summary <- table(muley15$UTM_Zone,muley15$id)
summary
muley15$id <- factor(muley15$id)

#Sort data to address error in code and then look at first 10 records
of data to confirm

muley15 <- muley15[order(muley15$GPSFixTime),]
muley15[1:10,]#code displays the first 20 records
```

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

```
#########################################################
## Example of a trajectory of type II (time recorded)
### Conversion of the date to the format POSIX
#Needs 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)

#Attach da to muley15
muley15$da <- da

#Creates a column of time difference between each location
timediff <- diff(muley15$da)
muley15 <-muley15[-1,]
muley15$timediff <-as.numeric(abs(timediff))
str(muley15)

#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)

```
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
plot(merge)
str(merge)
```

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

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

#CAN BE USED TO REMOVE TIME FROM DATE IN GPSFIXTIME COLUMN if needed
#Date <- as.character(muleys$GPSFixTime)
#Date <- as.POSIXct(strptime(muleys$GPSFixTime,"%Y.%m.%d"))
#muleys$Date <- Date
#str(muleys)
```

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 22H00
## The following function returns TRUE if the date is comprised between
## 06H00 and 23H00 (i.e. results in 3 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)

#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

deer

#Shows results of cutting the traj into individual bursts
```

```
#NOTE the "Irregular traject" line below because we will revisit this later!
#*********** List of class ltraj ***********

#Type of the traject: Type II (time recorded)
#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
#7   D15 D15.014        7   0 2011-10-18 00:00:34 2011-10-18 18:00:48
#8   D15 D15.016        7   0 2011-10-19 00:00:36 2011-10-19 18:00:40
#9   D15 D15.018        7   0 2011-10-20 00:00:53 2011-10-20 18:00:40
#10  D15 D15.020        7   0 2011-10-21 00:00:39 2011-10-21 18:00:37
```

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

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

str(dfdeer)
'data.frame':   2243 obs. of  13 variables:
 $ x        : num  677932 679037 679429 679750 679453 ...
 $ y        : num  4189551 4189493 4189406 4189053 4188461 ...
 $ date     : POSIXct, format: "2011-10-12 03:00:52" "2011-10-12 06:00:52"
 $ dx       : num  1105 392 321 -297 163 ...
 $ dy       : num  -58 -87 -353 -592 -89 NA -189 756 395 95 ...
 $ dist     : num  1107 402 477 662 186 ...
 $ dt       : num  10800 10786 10796 10808 10810 ...
 $ R2n      : num  0 1224389 2262034 3553128 3501541 ...
 $ abs.angle: num  -0.0524 -0.2184 -0.8328 -2.0358 -0.4998 ...
 $ rel.angle: num  NA -0.166 -0.614 -1.203 1.536 ...
 $ id       : Factor w/ 1 level "D15": 1 1 1 1 1 1 1 1 1 1 ...
 $ burst    : Factor w/ 325 levels "D15.001","D15.003",..: 1 1 1 1 1 1 2  ...
 $ pkey     : Factor w/ 2588 levels "D15.2011-10-12 03:00:52",..: 1

#Code to get mean distance moved for each burst
summary <- numSummary(dfdeer[,"dist"],groups=dfdeer$burst, statistics=
     c("mean","sd"))
summary

#Convert matrix from data.frame to a matrix to export as a .csv file
mean <- as.matrix(summary$table)

#Write.table gives csv output of Summary.  Be sure to specify the directory
```

```
     and the output files will be stored there
write.table(mean, file = "Distance.csv", sep =",", row.names = TRUE,
     col.names = TRUE, qmethod ="double")
```

3.4 First Passage Time (FPT)

The first passage time (FPT) is a parameter often used to describe the scale at which patterns occur in a trajectory. For a given scale r, it is defined as the time required by the animals to pass through a circle of radius r. The mean first passage time scales proportionately to the square of the radius of the circle for an uncorrelated random walk (Johnson et al. 1992). Johnson et al. (1992) used this property to differenciate facilitated diffusion and impeded diffusion, according to the value of the coefficient of the linear regression log(FPT) = a * log(radius) + b. Under the hypothesis of a random walk, a should be equal to 2 (higher for impeded diffusion, and lower for facilitated diffusion). Note however, that the value of a converges to 2 only for large values of radius. Another use of the FPT was proposed that, instead of computing the mean of FPT, use the variance of the log(FPT). This variance should be high for scales at which patterns occur in the trajectory, e.g. area restricted search (Fauchald and Tverra 2003). This method is often used to determine the scale at which an animal seaches for food.

The value fpt computes the FPT for each relocation and each radius, and for each animals. This function returns an object of class "fipati", i.e. a list with one component per animal. Each component is a data frame with each column corresponding to a value of radii and each row corresponding to a relocation. An object of class fipati has an attribute named "radii" corresponding to the argument radii of the function fpt. meanfpt and varlogfpt return a data frame giving respectively the mean FPT and the variance of the log(FPT) for each animal (rows) and rach radius (column). These objects also have an attribute "radii".

1. Exercise 3.4 - Download and extract zip folder into your preferred location
2. Set working directory to the extracted folder in R under File - Change dir...
3. First we need to load the packages needed for the exercise

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

4. Now open the script "FPTscript.R" and run code directly from the script
5. In this example we are going to look at mule deer in southwestern Colorado. We can eliminate poor locations in the original dataset or code can be used after examining the data.

   ```
   muleys <-read.csv("DCmuleysedited.csv", header=T)

   #Code to look at number of relocations per animal
   table(muleys$id)

   newmuleys <-subset(muleys, muleys$Long > -110.50 & muleys$Lat > 37.3 &
      muleys$Long < -107)
   muleys <- newmuleys
   ```

```
#############################################################################
## Example of a trajectory of type II (time recorded) ##that must be converted
## to the format POSIX that needs to be done to get proper digits of date for
##  use with the adehabitatLT package
#############################################################################
da <- as.character(muleys$GPSFixTime)
da <- as.POSIXct(strptime(muleys$GPSFixTime,format="%Y.%m.%d %H:%M:%S"))
muleys$da <- da
```

6. Determine the time difference between each relocation for use later

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

7. Create the spatial data from of xy coordinates and additional information

```
data.xy = muleys[c("X","Y")]
#Creates class Spatial Points for all locations
xysp <- SpatialPoints(data.xy)
#proj4string(xysp) <- CRS("+proj=utm +zone=17 +ellps=WGS84")
#Creates a Spatial Data Frame from
sppt<-data.frame(xysp)
#Creates a spatial data frame of ID
idsp<-data.frame(muleys[2])
#Creates a spatial data frame of dt
dtsp<-data.frame(muleys[24])
#Creates a spatial data frame of Burst
busp<-data.frame(muleys[25])
#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
plot(merge)
str(merge)
```

8. Create an object of class "ltraj" (i.e., trajectory) for all animals

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

#Or we can plot trajectories for each specific animal

plot(ltraj[1])
plot(ltraj[2])
plot(ltraj[3])
plot(ltraj[4])
plot(ltraj[5])
plot(ltraj[6])
plot(ltraj[7])
```

9. Code to plot histograms of distance distribution for each deer

```
windows()
```

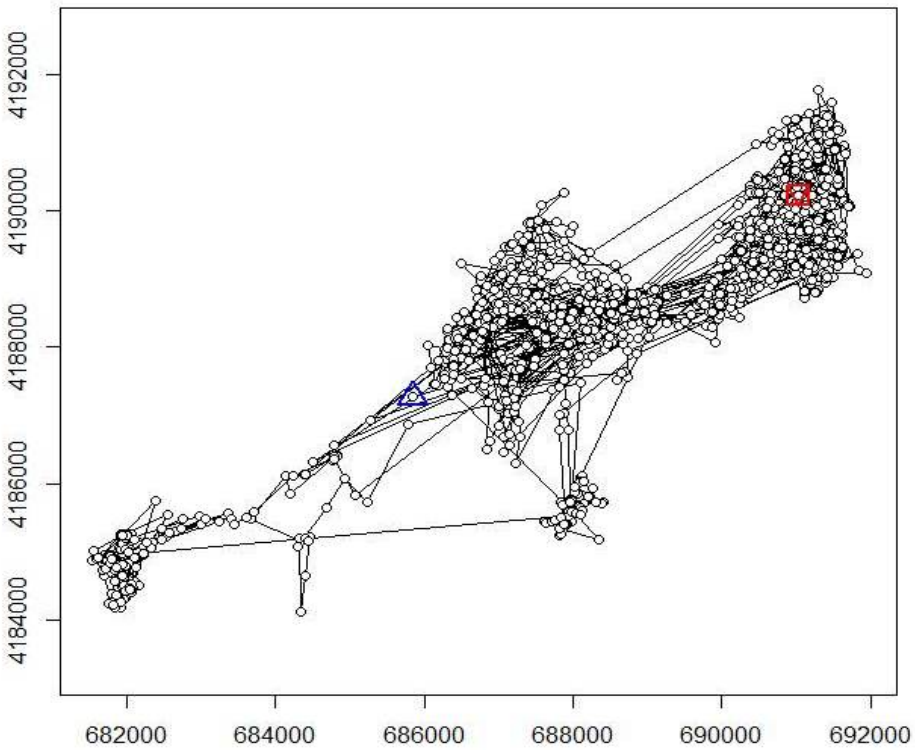

Figure 3.1: Example of a trajectory created using adehabitatLT for a mule deer in Colorado.

```
hist(ltraj[1], "dist", freq = FALSE)
windows()
hist(ltraj[2], "dist", freq = FALSE)
windows()
hist(ltraj[3], "dist", freq = FALSE)
windows()
hist(ltraj[4], "dist", freq = FALSE)
windows()
hist(ltraj[5], "dist", freq = FALSE)
windows()
hist(ltraj[6], "dist", freq = FALSE)
windows()
hist(ltraj[7], "dist", freq = FALSE)
```

10. Code below actually creates First Passage Time and mean and variance of fpt

```
plot(ltraj[1])
i1 <- fpt(ltraj[1], seq(300,1000, length=30))
plot(i1, scale = 200, warn = FALSE)

plot(ltraj[2])
i2 <- fpt(ltraj[2], seq(300,1000, length=30))
plot(i2, scale = 500, warn = FALSE)

toto2 <- meanfpt(i2)
toto2
attr(toto2, "radii")
```

```
toto2 <- varlogfpt(i2)
toto2
attr(toto2, "radii")

plot(ltraj[3])
i3 <- fpt(ltraj[3], seq(300,1000, length=30))
plot(i3, scale = 500, warn = FALSE)

toto3 <- meanfpt(i3)
toto3
attr(toto3, "radii")

toto3 <- varlogfpt(i3)
toto3
attr(toto3, "radii")
```

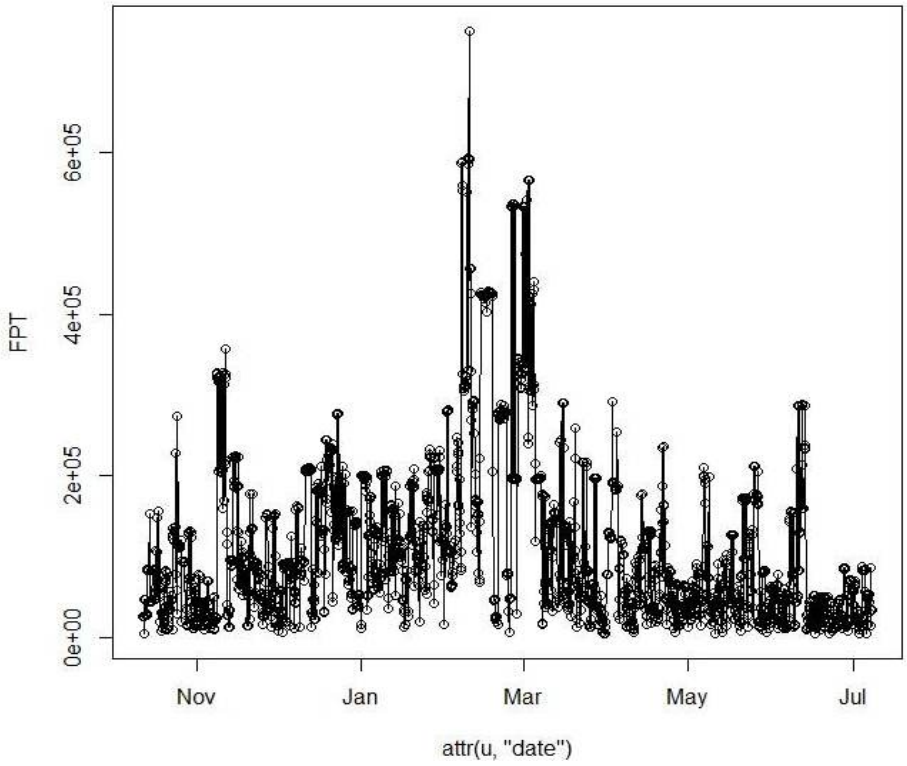

Figure 3.2: Plot of a First Passage Time for a mule deer in Colorado identifying mean FPT by month.

```
plot(ltraj[4])
i4 <- fpt(ltraj[4], seq(300,1000, length=30))
plot(i4, scale = 500, warn = FALSE)

toto4 <- meanfpt(i4)
toto4
attr(toto4, "radii")

toto4 <- varlogfpt(i4)
toto4
```

```
attr(toto4, "radii")

plot(ltraj[5])
i5 <- fpt(ltraj[5], seq(300,1000, length=30))
plot(i5, scale = 500, warn = FALSE)

toto5 <- meanfpt(i5)
toto5
attr(toto5, "radii")

toto5 <- varlogfpt(i5)
toto5
attr(toto5, "radii")

plot(ltraj[6])
i6 <- fpt(ltraj[6], seq(300,1000, length=30))
plot(i6, scale = 500, warn = FALSE)

plot(ltraj[7])
i7 <- fpt(ltraj[7], seq(300,1000, length=30))
plot(i7, scale = 500, warn = FALSE)

toto7 <- meanfpt(i7)
toto7
attr(toto7, "radii")

toto7 <- varlogfpt(i7)
toto7
attr(toto7, "radii")

is.regular(ltraj[1])
plotltr(ltraj[1], "dt")
windows()
plotltr(ltraj[1], "dist")

is.regular(ltraj[2])
plotltr(ltraj[2], "dt")
windows()
plotltr(ltraj[2], "dist")
ltraj[2]
```

11. Code to export each trajectory as a shapefile if needed

```
toto1 <-ltraj2sldf(ltraj[1])
plot(toto1)
writeOGR(toto1,dsn=".",layer="D12", driver = "ESRI Shapefile",overwrite=TRUE)
summary(toto1)

#If we want to define projection before making a shapefile
proj4string <- CRS("+proj=utm +zone=13N +ellps=WGS84")
toto2lines@proj4string <- proj4string
toto2pts@proj4string <- proj4string
```

```
#Write lines and points as a shapefile
toto2lines <-ltraj2sldf(ltraj[2],byid=TRUE)
toto2pts <- ltraj2spdf(ltraj[2])

plot(toto2pts)
plot(toto2lines, add=T)

writeOGR(toto2pts,dsn=".",layer="D15pts", driver = "ESRI Shapefile",
     overwrite_layer=TRUE)
writeOGR(toto2lines, dsn=".", paste("traj_line_",sep=""),driver = "ESRI Shapefile",
     overwrite=TRUE)

toto3 <-ltraj2sldf(ltraj[3])
plot(toto3)
writeOGR(toto3,dsn=".", layer="D16", driver = "ESRI Shapefile",overwrite=TRUE)

toto4 <-ltraj2sldf(ltraj[4])
plot(toto4)
writeOGR(toto4,dsn=".", layer="D19", driver = "ESRI Shapefile",overwrite=TRUE)

toto5 <-ltraj2sldf(ltraj[5])
plot(toto5)
writeOGR(toto5,dsn=".", layer="D4", driver = "ESRI Shapefile",overwrite=TRUE)

toto6 <-ltraj2sldf(ltraj[6])
plot(toto6)
writeOGR(toto6,dsn=".", layer="D6", driver = "ESRI Shapefile",overwrite=TRUE)

toto7 <-ltraj2sldf(ltraj[7])
plot(toto7)
writeOGR(toto7,dsn=".", layer="D8", driver = "ESRI Shapefile",overwrite=TRUE)
```

12. Another look at the time and distance between relocations of each animal. If GPS collars are programmed to collect 1 locations per 24 hours or a location every 4 hours, what would we expect Figure 3.3b to look like? The code can be used to produce these figures and can be used initially to inspect the data if there is concern about consistancy in location fixes or distance between each location.

```
is.regular(ltraj[3])
plotltr(ltraj[3], "dt")
plotltr(ltraj[3], "dist")
```

3.5 Regular trajectories

1. Exercise 3.5 - Download and extract zip folder into your preferred location
2. Set working directory to the extracted folder in R under File - Change dir...
3. First we need to load the packages needed for the exercise

```
library(adehabitatLT)
```

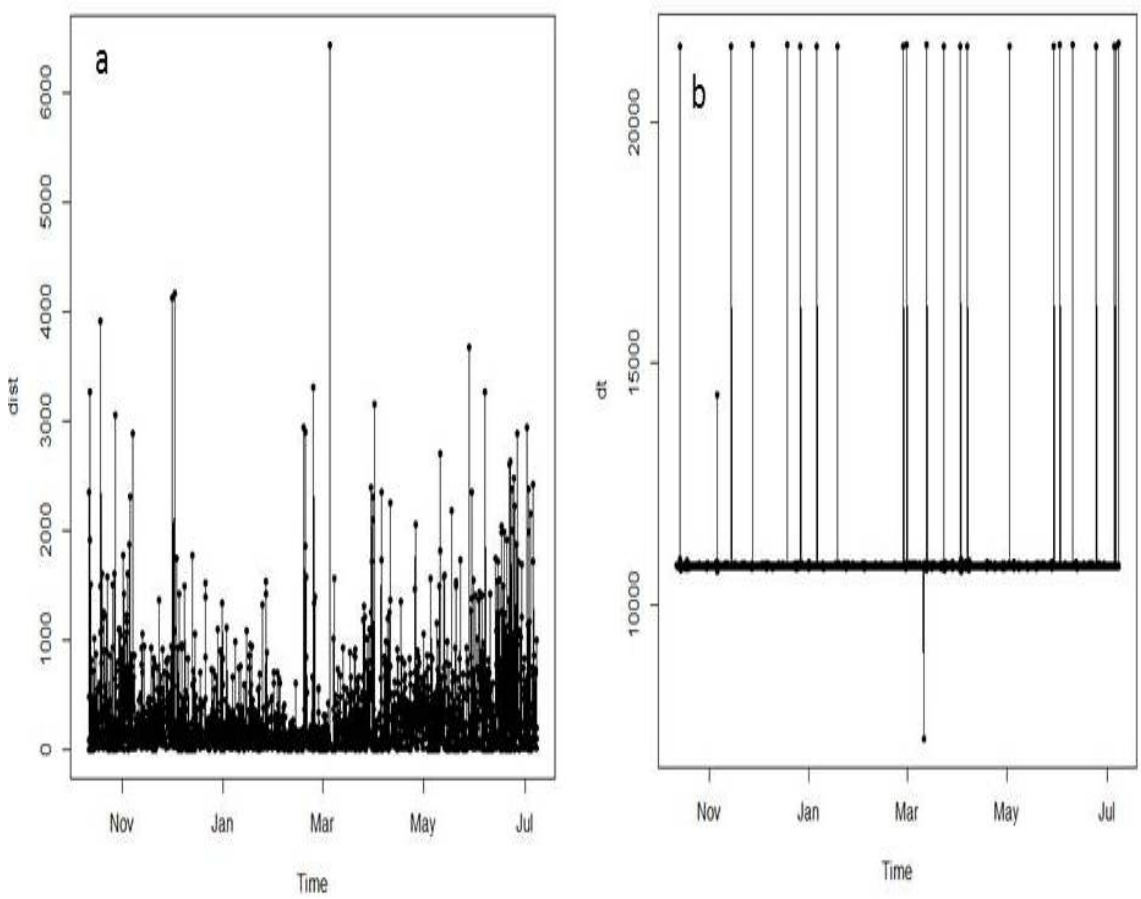

Figure 3.3: Summaries of distance and time (dt) between relocations for mule deer D16.

```
library(chron)
library(raster)
library(sp)
```

4. Now open the script "RegTrajScript.R" and run code directly from the script

```
muleys <-read.csv("DCmuleysedited.csv", header=T)
str(muleys)

#CODE FOR AN INDIVIDUAL ANIMAL
muley15 <- subset(muleys, id=="D15")
str(muley15)
summary <- table(muley15$UTM_Zone,muley15$id)
summary
muley15$id

#Sort data to address error in code
muley15 <- muley15[order(muley15$GPSFixTime),]
muley15[1:10,]#code displays the first 10 records to look at sorting results
str(muley15)

#########################################################
```

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

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

newmuleys <-subset(muley15, muley15$X > 599000 & muley15$X < 705000 &
     muley15$Y > 4167000 & muley15$timediff < 14401)
muley15 <- newmuleys

data.xy = muley15[c("X","Y")]
#Creates class Spatial Points for all locations
xysp <- SpatialPoints(data.xy)
#proj4string(xysp) <- CRS("+proj=utm +zone=17 +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
plot(merge)
str(merge)

### Creation of an object of class "ltraj"
ltraj <- as.ltraj(coordinates(merge),merge$da,id=merge$id)
plot(ltraj)
ltraj

#CAN BE USED TO REMOVE TIME FROM DATE IN GPSFIXTIME COLUMN
#Date <- as.character(muleys$GPSFixTime)
#Date <- as.POSIXct(strptime(muleys$GPSFixTime,"%Y.%m.%d"))
#muleys$Date <- Date
#str(muleys)

## 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 22H00
## The following function returns TRUE if the date is comprised between
```

```
## 06H00 and 23H00 (i.e. results in 3 locations/day bursts)
foo <- function(date) {
da <- as.POSIXlt(date)
ho <- da$hour + da$min
return(ho>18.0&ho<23.9)
}
deer <- cutltraj(ltraj, "foo(date)", nextr = TRUE)
deer
## Remove the first and last burst if needed?
#deer2 <- deer[-c(1,length(deer))]

#bind the trajectories
deer3 <- bindltraj(deer)
deer3
plot(deer3)
is.regular(deer)
FALSE
plotltr(deer3, "dist")

## The relocations have been collected every 3 hours, and there are some
## missing data
## The reference date: the hour should be exact (i.e. minutes=0):
refda <- strptime("00:00", "%H:%M")
refda
## Set the missing values
deerset <- setNA(deer3, refda, 3, units = "hour")
## now, look at dt for the bursts:
plotltr(deerset, "dt")
## dt is nearly regular: round the date:
deerset1 <- sett0(deerset, refda, 3, units = "hour")
plotltr(deerset1, "dt")
is.regular(deerset1)
## deerset1 is now regular

## Is the resulting object "sd" ?
is.sd(deerset1)

## Show the changes in the distance between
## successive relocations with the time
windows()
plotltr(deerset1, "dist")
## Segmentation of the trajectory based on these distances
lav <- lavielle(deerset1, Lmin=2, Kmax=20)
## Choose the number of segments
chooseseg(lav)
## 20 segments seem a good choice
## Show the partition
kk <- findpath(lav, 20)
kk

##Results of code
```

```
********** List of class ltraj **********

#Type of the traject: Type II (time recorded)
#Regular traject. Time lag between two locs: 10800 seconds

#Characteristics of the bursts:
#    id      burst nb.reloc NAs          date.begin            date.end
#1  D15  Segment.1      199  27 2011-10-12 04:00:00 2011-11-05 22:00:00
#2  D15  Segment.2        2   0 2011-11-06 01:00:00 2011-11-06 03:00:00
#3  D15  Segment.3      455  64 2011-11-06 06:00:00 2012-01-02 00:00:00
#4  D15  Segment.4        1   0 2012-01-02 03:00:00 2012-01-02 03:00:00
#5  D15  Segment.5        2   0 2012-01-02 06:00:00 2012-01-02 09:00:00
#6  D15  Segment.6        1   0 2012-01-02 12:00:00 2012-01-02 12:00:00
#7  D15  Segment.7       64   8 2012-01-02 15:00:00 2012-01-10 12:00:00
#8  D15  Segment.8        3   1 2012-01-10 15:00:00 2012-01-10 21:00:00
#9  D15  Segment.9        2   0 2012-01-11 00:00:00 2012-01-11 03:00:00
#10 D15 Segment.10       33   4 2012-01-11 06:00:00 2012-01-15 06:00:00
#11 D15 Segment.11        5   1 2012-01-15 09:00:00 2012-01-15 21:00:00
#12 D15 Segment.12        3   0 2012-01-16 00:00:00 2012-01-16 06:00:00
#13 D15 Segment.13        2   0 2012-01-16 09:00:00 2012-01-16 12:00:00
#14 D15 Segment.14      336  46 2012-01-16 15:00:00 2012-02-27 12:00:00
#15 D15 Segment.15        4   1 2012-02-27 15:00:00 2012-02-28 00:00:00
#16 D15 Segment.16      250  35 2012-02-28 03:00:00 2012-03-30 07:00:00
#17 D15 Segment.17        1   0 2012-03-30 10:00:00 2012-03-30 10:00:00
#18 D15 Segment.18        5   2 2012-03-30 13:00:00 2012-03-31 01:00:00
#19 D15 Segment.19     1164 154 2012-03-31 04:00:00 2012-08-23 13:00:00
#20 D15 Segment.20       63   9 2012-08-23 16:00:00 2012-08-31 10:00:00

#Notice that the results show for each burst:
(1) number of relocations
(2) number of relocations removed (i.e., NA)
(3) begin and end dates

Now if we reduce the number of segments we get the following bursts:

## Segmentation of the trajectory based on these distances
lav <- lavielle(deerset1, Lmin=2, Kmax=10)
## Choose the number of segments
chooseseg(lav)
## 20 segments seem a good choice
## Show the partition
kk <- findpath(lav, 10)
kk

********** List of class ltraj **********

#Type of the traject: Type II (time recorded)
#Regular traject. Time lag between two locs: 10800 seconds

#Characteristics of the bursts:
#    id      burst nb.reloc NAs          date.begin            date.end
```

```
#1  D15  Segment.1      201  27 2011-10-12 04:00:00 2011-11-06 03:00:00
#2  D15  Segment.2      456  64 2011-11-06 06:00:00 2012-01-02 03:00:00
#3  D15  Segment.3        2   0 2012-01-02 06:00:00 2012-01-02 09:00:00
#4  D15  Segment.4        1   0 2012-01-02 12:00:00 2012-01-02 12:00:00
#5  D15  Segment.5      102  13 2012-01-02 15:00:00 2012-01-15 06:00:00
#6  D15  Segment.6       10   1 2012-01-15 09:00:00 2012-01-16 12:00:00
#7  D15  Segment.7      591  82 2012-01-16 15:00:00 2012-03-30 10:00:00
#8  D15  Segment.8        5   2 2012-03-30 13:00:00 2012-03-31 01:00:00
#9  D15  Segment.9     1164 154 2012-03-31 04:00:00 2012-08-23 13:00:00
#10 D15 Segment.10       63   9 2012-08-23 16:00:00 2012-08-31 10:00:00

#We can look at each segment to inspect the path traveled during the burst:
plot(kk[1])
plot(kk[2])
plot(kk[3])
plot(kk[6])
plot(kk[7])
plot(kk[9])
```

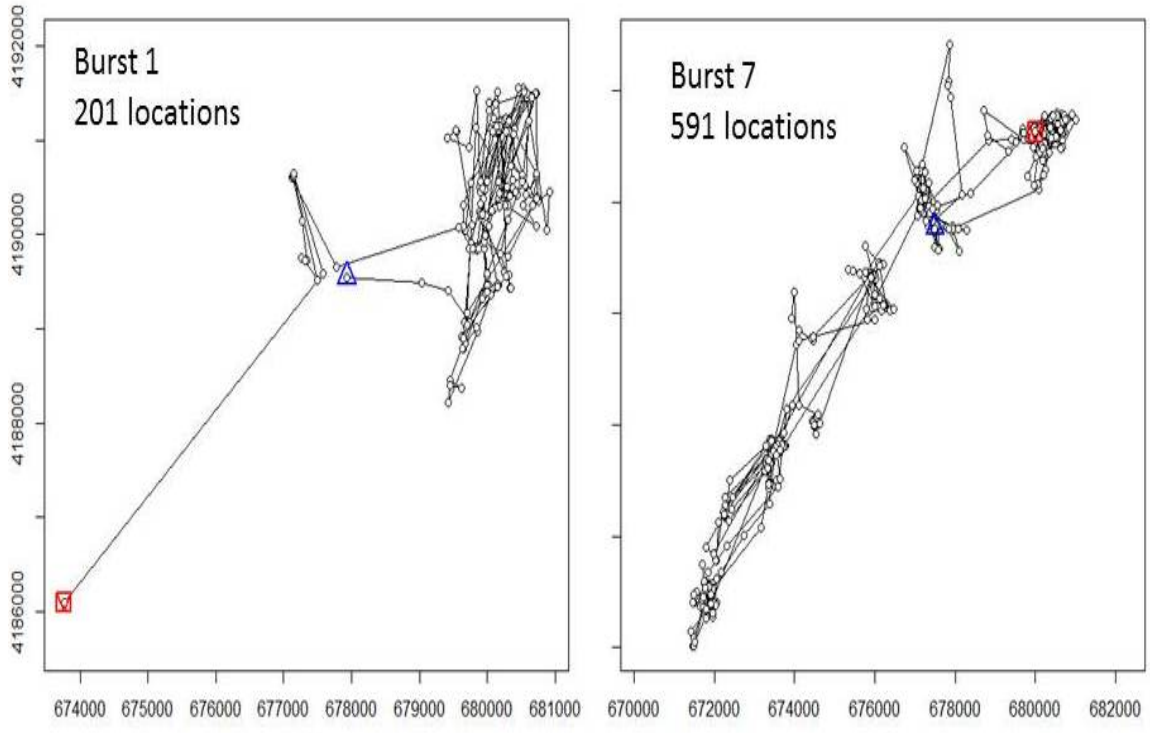

Figure 3.4: Bursts of movements for mule deer D15 after creating segments based for focal use areas.

3.6 Net Squared Displacement

Net squared displacement (NSD) looks at the movement vectors of animals to determine their use of the landscape (Bunnefeld et al. 2011, Papworth et al. 2012). Bunnefeld et al. (2011) determined a novel method to ideintfy movement patters using NSD which is the straight line distance between an animals' srating location and subsequent locations. Movements were then

categorized into one of 5 categories based on the top model that describes movement for each individual. In the code for this section, we have updated the code to include the 5 movement equations along with R code provided in Papworth et al. (2012) to enable determination of migratory, mixed migratory, disperser, home range, or nomadic movement behavior. Figure 1 in Bunnefeld et al. (2011) is a helpful to interpret the output of this code that identifies the pattern of NSD over time that is determined by which of the 5 movement behaviors the animal follows. Those interested in this section should read the 2 papers cited for more details and specifics of the methods.

1. Exercise 3.6 - Download and extract zip folder into your preferred location
2. Set working directory to the extracted folder in R under File - Change dir...
3. First we need to load the packages needed for the exercise

```
library(trip)
library(stringr)
library(adehabitatHR)
library(lattice)
library(gmodels)
library(spatstat)
library(maptools)
```

4. Now open the script "NSDscript.R" and run code directly from the script

```
muleys<-read.csv("DCmuleysedited.csv", header=T, sep=",")
str(muleys)

#Remove outlier locations, if needed, by first plotting coords below then editing
#this line accordingly
newmuleys <-subset(muleys, muleys$Long > -110.50 & muleys$Lat > 37.3
   & muleys$Long < -107)
muleys <- newmuleys

#Make a spatial data frame of locations after removing outliers
coords<-data.frame(x = muleys$Long, y = muleys$Lat)
crs<-"+proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0"
head(coords)
plot(coords)

deer.spdf <- SpatialPointsDataFrame(coords= coords, data = muleys,
   proj4string = CRS(crs))
head(deer.spdf)
class(deer.spdf)
proj4string(deer.spdf)
plot(deer.spdf,col=deer.spdf$id)

muleys$NewDate<-as.POSIXct(muleys$GPSFixTime, format="%Y.%m.%d %H:%M:%S",
    origin="1970-01-01")

#TIME DIFF NECESSARY IN BBMM CODE
timediff <- diff(muleys$NewDate)*60*60
# remove first entry without any difference
muleys <- muleys[-1,]
```

```
muleys$timelag <-as.numeric(abs(timediff))
summary(muleys$timelag)
#Remove locations greater than 24 hours apart in time
muleys <- subset(muleys, muleys$timelag < 18000)
```

5. The key to NSD is proper delineation of movement periods so we can explore a few alternatives here. First we will define deer and year based simply on the Year the location was recorded. Not very biologically meaningful but for simplicity we will start with calender year. Be sure to skip step 6 below and continue on with step 7 to the end of the exercise.

```
muleys$Year <- format(muleys$NewDate, "%Y")
muleys <- subset(muleys, muleys$Year != "NA")
muleys$YearBurst <- c(paste(muleys$id,muleys$Year,sep="_"))
muleys$YearBurst <- as.factor(muleys$YearBurst)
str(muleys)
summary(muleys$YearBurst)

muleys <- subset(muleys, table(muleys$YearBurst)[muleys$YearBurst] > 100)
muleys$YearBurst <- factor(muleys$YearBurst)
```

6. Alternatively, we could assign a Year as when we would expect mule deer to disperse or migrate from summer to winter range. In our Colorado example, the mule deer were captured on 30 September 2011 (winter range) so we will start there and separate out a summer season from May to August using the code below. Be sure to use skip step 5 above and run step 6 instead but do not run both.

```
range(muleys$NewDate)
muleys$Year <- NULL
muleys$Year[muleys$NewDate > "2011-09-30 00:30:00" & muleys$NewDate
   < "2012-04-01 23:00:00"] <- 2011
muleys$Year[muleys$NewDate > "2012-03-31 00:30:00" & muleys$NewDate
   < "2012-10-01 23:00:00"] <- 2012
muleys$Year <- as.factor(muleys$Year)
muleys$YearBurst <- c(paste(muleys$id,muleys$Year,sep="_"))
muleys$YearBurst <- as.factor(muleys$YearBurst)
table(muleys$YearBurst)
```

7. Next we are going to rename our dataset to simply follow along with previous code unless you want to change d1 to muleys throughout

```
 d1 <- muleys

#Code separate each animal into a shapefile or text file to use as a "List"
# get input file
indata <- d1
innames <- unique(d1$YearBurst)
innames <- innames[1:12]#needs to be number of unique IDs
outnames <- innames
# begin loop to calculate home ranges
for (i in 1:length(innames)){
  data <- indata[which(indata$YearBurst==innames[i]),]
  if(dim(data)[1] != 0){
    #data <-data[c(-21)]
```

```
    # export the point data into a shp file
    data.xy = data[c("X", "Y")]
    coordinates(data.xy) <- ~X+Y
    sppt <- SpatialPointsDataFrame(coordinates(data.xy),data)
    proj4string(sppt) <- CRS("+proj=utm +zone=12 +datum=WGS84")
    #writePointsShape(sppt,fn=paste(outnames[i],sep="/"),factor2char=TRUE)
    #sppt <-data[c(-22,-23)]
    write.table(sppt, paste(outnames[i],"txt",sep="."), sep="\t", quote=FALSE,
     row.names=FALSE)
    write.table(paste(outnames[i],"txt",sep="."), sep="\t", quote=FALSE,
    row.names=FALSE, col.names=FALSE, "In_list.txt", append=TRUE)
    #The write.table line above should only be run once to create the In_list.txt
    #file otherwise it rights all animals each time
    #Note: There are 3 lines of code that are not active that can be activated
   #to export point shapefiles for all resulting animals but need to remove
   #as.POSIX column first
}
}
##############################################################################
##############################################################################
#
#
#Code below to get NSD and best movement model for each bird
#
#
##############################################################################
##############################################################################

date()

# Reads the List file of GPS datasets

List<-read.table("In_list.txt",sep="\t",header=F)
head(List) #List contains the filenames of the deer datasets

# Generation of results vectors
ID <- rep(0,nrow(List))
LOCS <- rep(0,nrow(List))
MIGR <- rep(0,nrow(List))
MIXM <- rep(0,nrow(List))
DISP <- rep(0,nrow(List))
HORA <- rep(0,nrow(List))
NOMA <- rep(0,nrow(List))
ID <- rep(0,nrow(List))
LOCS <- rep(0,nrow(List))
MIGR <- rep(0,nrow(List))
MIXM <- rep(0,nrow(List))
DISP <- rep(0,nrow(List))
HORA <- rep(0,nrow(List))
NOMA <- rep(0,nrow(List))
AICC_1 <- rep(0,nrow(List))
```

```
AICC_2 <- rep(0,nrow(List))
AICC_3 <- rep(0,nrow(List))
AICC_4 <- rep(0,nrow(List))
AICC_5 <- rep(0,nrow(List))

minAIC <- rep(0,nrow(List))
d_AICC_1 <- rep(0,nrow(List))
d_AICC_2 <- rep(0,nrow(List))
d_AICC_3 <- rep(0,nrow(List))
d_AICC_4 <- rep(0,nrow(List))
d_AICC_5 <- rep(0,nrow(List))
LL_AICC_1 <- rep(0,nrow(List))
LL_AICC_2 <- rep(0,nrow(List))
LL_AICC_3 <- rep(0,nrow(List))
LL_AICC_4 <- rep(0,nrow(List))
LL_AICC_5 <- rep(0,nrow(List))
sumLL_AICC <- rep(0,nrow(List))
wi_AICC_1 <- rep(0,nrow(List))
wi_AICC_2 <- rep(0,nrow(List))
wi_AICC_3 <- rep(0,nrow(List))
wi_AICC_4 <- rep(0,nrow(List))
wi_AICC_5 <- rep(0,nrow(List))

for(i in 1:nrow(List)) {

coords<-read.table(as.character(List[i,]),sep="\t",header=T)
coords$DT<-as.POSIXct(coords$NewDate, format="%Y-%m-%d %H:%M:%S")

##make a data.frame of coordinates. Here the raw values are divided
#by 1000 so that trajectories are calculated using km as the unit of
#measurement not meters
coord<-data.frame((coords$Y/1000),(coords$X/1000))
# make ltraj: a trajectory of all the relocations
d2<-as.ltraj(coord,coords$DT,
coords$YearBurst,        #separate your data by individual.
burst=coords$YearBurst, #burst is used to creat subdivisions within an individual.
typeII=TRUE)       #typeII can be TRUE: radio-track data, or FALSE: not time
                   #recorded, such as tracks in the snow

#[1] "2007-06-05 16:00:00 EDT" "2015-03-12 14:00:00 EDT"
#you can now make your trajectory regular
#firstly create a reference start time
#refda <- strptime("00:00:00", "%H:%M:%S")   #all relocations should be altered
#to occur at 30 seconds past each minute

#you can now make your trajectory regular, as radio tracks tend to lose
#a few seconds / minutes with each relocation
#firstly add "NA" for each missing location in your trajectory
#d3<-setNA(d2,refda,
#as.POSIXct("2007-06-01 06:00:00 EDT"), #any time before earliest timedate
#7200,            #stating there should be a location every 2 hours
```

```
#tol=7200,        #how many time units to search each side of expected location
#units="sec")   #specifying the time units

#you can now make your trajectory regular
#firstly create a reference start time
refda <- strptime("00:00:30", "%H:%M:%S")

##NOTE: The refda and d3 code above was not run because it results in too many
#relocations as "NA" that get removed below. Not quite sure the reason behind
# it being included? We can now make your trajectory regular
d4<-sett0(d2, refda,
10800,                      #stating the interval at which relocations should be
correction.xy =c("none"),   #if "cs" performs location correction based on the
#assumption the individual moves at a constant speed
tol=10800,    #how many time units to search either side of an expected location
units = "sec")  #specifying the time units

#to view your regular trajectory of points with NA's
summary(d4)
#now calculating NSD for each point
datansd<-NULL
for(n in 1:length(summary(d4)[,1])) #stating that NSD should be
#calculated separately for each burst
{
nsdall<-d4[[n]][,8]             #extracting the NSD for each location
nsdtimeall<-d4[[n]][,3]         #extracting the time for each location
nsdtimestartzero<-d4[[n]][,3]-d4[[n]][1,3]
#extracting the time since trip start for each location
nsdid<-rep(as.vector(summary(d4)[n,1]),
length.out=summary(d4)[n,3])
#extracting the individual associated with each location
nsdtrip<-rep(as.vector(summary(d4)[n,2]),length.out=summary(d4)[n,3])
#extracting the trip associated with each location
datansd1<-data.frame(nsdall,nsdtimeall,nsdtimestartzero,nsdid,nsdtrip)
#joining all these variables together in a data frame
datansd<-rbind(datansd,datansd1)
#joining all the data frames together
}
datansd$zero1<-as.numeric(unclass(datansd$nsdtimestartzero))
# making seconds since trip start numeric
datansd$zerostart<-datansd$zero1/60
#changing the time since trip start from seconds to minutes
datansd$minslitr2<-as.numeric(strftime(as.POSIXlt(datansd$nsdtimeall),
format="%M"))
#making a vector of the hour of the day a location occured
datansd$hdaylitr2<-as.numeric(strftime(as.POSIXlt(datansd$nsdtimeall),
format="%H"))
#making a vector of the minute in an hour a location occured
datansd$minsday<-((datansd$hdaylitr2*60)+datansd$minslitr2)
#calculating the minute in the day a location occured
```

```
summary(datansd)
datansd1<-na.omit(datansd)            #remove NA's


datansd1$coordinates<-coord           #add the coordinates for each point
#you now have the dataframe you need (datansd) to start analysis

#NSD
#table(datansd1$nsdid)

#Now we can start modelling NSD using nlme.
#Equations are from Bunnefeld at al (2011) A model-driven approach to quantify
#migration patterns: #individual, regional and yearly differences.
#Journal of Animal Ecology 80: 466 - 476

#First we are going to model the data using nls, a least squares method,
#the simplest method and first method in Bunnefeld et al. 2011 (i.e., MIGRATION)
#that uses a double sigmoid or s-shaped function.

############################
##
##  MIGRATION
##
############################

m1<-tryCatch(nls(nsdall ~  asym /(1+exp((xmidA-zerostart)/scale1)) +
(-asym / (1 + exp((xmidB-zerostart)/scale2))), #Equation 1 in Bunnefeld et al. 2011
start = c(asym=15000000,xmidA=200000,xmidB=450000,scale1=1000,scale2=1000)
#these are the starting values for each parameter of the equation
,data=na.omit(datansd1)),error=function(e)99999)   #this is the data
summary(m1)        #this will print a summary of the converged model
#NOTE: The error function is simply to prevent the loop from crashing if model
#does not converge

############################
##
##  MIXED MIGRATORY
##
############################

m2 <-tryCatch(nls(nsdall ~  asymA /(1+exp((xmidA-zerostart)/scale1)) +
(-asymB / (1 + exp((xmidB-zerostart)/scale2))), #Equation 2 in Bunnefeld et al. 2011
start = c(asymA=15000000,asymB=10000000, xmidA=200000,xmidB=450000,scale1=1000,
  scale2=1000)
#these are the starting values for each parameter of the equation
,data=na.omit(datansd1)),error=function(e)99999)   #this is the data
summary(m2)        #this will print a summary of the converged model

############################
##
##  DISPERSAL
```

```
##
############################

m3 <-tryCatch(nls(nsdall ~  asym /(1+exp((xmid-zerostart)/scale)), #Equation 3 in
#Bunnefeld et al. 2011
start = c(asym=15000000,xmid=200000,scale=1000)
#these are the starting values for each parameter of the equation
,data=na.omit(datansd1)),error=function(e)99999)   #this is the data
summary(m3)        #this will print a summary of the converged model

############################
##
## HOME RANGE
##
############################

m4 <- tryCatch(nls(nsdall ~ intercept, data=na.omit(datansd1),
   start = list(intercept = 0)),error=function(e)99999)
#Equation 4 in Bunnefeld et al. 2011 where c is a constant
summary(m4)        #this will print a summary of the converged model

############################
##
## NOMADIC
##
############################

m5 <- tryCatch(nls(nsdall ~ -1*zerostart,start=c(zerostart=1),
data=na.omit(datansd1)),error=function(e)99999) #Equation 5 in Bunnefeld et al. 2011
#where beta is a constant and t the number of days since initial start date
# (i.e., 1 June of each year)
summary(m5)        #this will print a summary of the converged model

#Below we are going to set up the AIC table
ID[i] <- paste(unique(as.factor(datansd$nsdid)))
LOCS[i] <- nrow(coords)
MIGR[i] <- print(tryCatch(AIC(m1),error=function(e)0))
MIXM[i] <- print(tryCatch(AIC(m2),error=function(e)0))
DISP[i] <- print(tryCatch(AIC(m3),error=function(e)0))
HORA[i] <- print(tryCatch(AIC(m4),error=function(e)0))
NOMA[i] <- print(tryCatch(AIC(m5),error=function(e)0))


AICC_1[i] <- print(tryCatch(AIC(m1),error=function(e)99999))
AICC_2[i] <- print(tryCatch(AIC(m2),error=function(e)99999))
AICC_3[i] <- print(tryCatch(AIC(m3),error=function(e)99999))
AICC_4[i] <- print(tryCatch(AIC(m4),error=function(e)99999))
AICC_5[i] <- print(tryCatch(AIC(m5),error=function(e)99999))

minAIC[i] <- min(AICC_1[i],AICC_2[i],AICC_3[i],AICC_4[i],AICC_5[i])
```

```
d_AICC_1[i] <- (AICC_1[i] - minAIC[i])
d_AICC_2[i] <- (AICC_2[i] - minAIC[i])
d_AICC_3[i] <- (AICC_3[i] - minAIC[i])
d_AICC_4[i] <- (AICC_4[i] - minAIC[i])
d_AICC_5[i] <- (AICC_5[i] - minAIC[i])

LL_AICC_1[i] <- exp(-0.5*d_AICC_1[i])
LL_AICC_2[i] <- exp(-0.5*d_AICC_2[i])
LL_AICC_3[i] <- exp(-0.5*d_AICC_3[i])
LL_AICC_4[i] <- exp(-0.5*d_AICC_4[i])
LL_AICC_5[i] <- exp(-0.5*d_AICC_5[i])

sumLL_AICC[i] <- sum(LL_AICC_1[i],LL_AICC_2[i],LL_AICC_3[i],LL_AICC_4[i],LL_AICC_5[i])

wi_AICC_1[i] <- LL_AICC_1[i]/sumLL_AICC[i]
wi_AICC_2[i] <- LL_AICC_2[i]/sumLL_AICC[i]
wi_AICC_3[i] <- LL_AICC_3[i]/sumLL_AICC[i]
wi_AICC_4[i] <- LL_AICC_4[i]/sumLL_AICC[i]
wi_AICC_5[i] <- LL_AICC_5[i]/sumLL_AICC[i]

filename<-paste(substr(List[i,],1,8),"png",sep=".")
#NOTE:Numbers after "List[i,] need to encompass possible lengths of output name
#(i.e., D19.txt is 6 characters)
png(filename,height=20,width=30,units="cm",res=600)
#graphical exploration of the data will help you find sensible starting values
#for each of the parameters asym, xmidA, xmidB, scale1 and scale2.
#to graph nsd against time, use:
#xyplot(nsdall~zerostart|nsdtrip,data=datansd)
#str(nsdtest)
#now plot the data with the predicted curve
nsdplot <- xyplot(nsdall ~ zerostart/3600, data=datansd1,
col="grey",    #color for the observed locations
type='b',      # 'b' shows the locations as dots, with a line connecting
#successive locations. Can also be 'p' for just the locations, or 'l' for just
#the line between locations
ylab=expression(paste('Net squared displacement ',' ', (km^2))), #y axis label
xlab="Hours after trip start")

plot(nsdplot)

dev.off()

}

#Create table of AIC values with lower AIC identifying best model
RESULT<-cbind(ID,LOCS,MIGR,MIXM,DISP,HORA,NOMA)
colnames(RESULT)<- c("ID","LOCS","MIGR","MIXM","DISP","HORA","NOMA")#
RESULT
#write.table(RESULT,"OUT_AIC_NSD.txt",sep="\t")#Output to excel if needed

#Create table of raw values to calculate AICweights
```

```
Migratory <- rbind(ID,AICC_1,d_AICC_1,LL_AICC_1,wi_AICC_1)
MixedMig <- rbind(ID,AICC_2,d_AICC_2,LL_AICC_2,wi_AICC_2)
Disperser <- rbind(ID,AICC_3,d_AICC_3,LL_AICC_3,wi_AICC_3)
HomeRange <- rbind(ID,AICC_4,d_AICC_4,LL_AICC_4,wi_AICC_5)
Nomadic <- rbind(ID,AICC_5,d_AICC_5,LL_AICC_5,wi_AICC_5)

RESULT2 <- rbind(Migratory,MixedMig,Disperser,HomeRange,Nomadic)
RESULT2
```

3.7 Movement Trajectory Animation

1. Exercise 3.7 - Download and extract zip folder into your preferred location
2. Set working directory to the extracted folder in R under File - Change dir...
3. First we need to load the packages needed for the exercise

```
library(adehabitatLT)
library(chron)
library(raster)
library(sp)
library(rgdal)
```

4. Now open the script "TrajDynScript.R" and run code directly from the script

```
muleys <-read.csv("DCmuleysedited.csv", header=T)
str(muleys)

#CODE FOR AN INDIVIDUAL ANIMAL
muley15 <- subset(muleys, id=="D15")
muley15[1:10,]
muley15$id <- factor(muley15$id)
str(muley15)
summary <- table(muley15$UTM_Zone,muley15$id)
summary

#Sort data to address error in code and then look at first 10 records
#of data to confirm
muley15 <- muley15[order(muley15$GPSFixTime),]
muley15[1:10,]

#########################################################
## Example of a trajectory of type II (time recorded)
### Conversion of the date to the format POSIX
#Needs 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"))
#Attach da to muley15
muley15$da <- da

timediff <- diff(muley15$da)
```

```
muley15 <-muley15[-1,]
muley15$timediff <-as.numeric(abs(timediff))
str(muley15)

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

5. Create a SpatialPointsDataFrame of specific columns available in the data

```
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
plot(merge)
str(merge)
```

6. Now let's have a little fun with these mule deer locations and explore them as trajectories, over vegetation, or zooming in on a specific area

```
#Load vegetation raster layer textfile clipped in ArcMap
Albers.crs <-CRS("+proj=aea +lat_1=29.5 +lat_2=45.5 +lat_0=23 +lon_0=-96
   +x_0=0 +y_0=0 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0 +units=m +no_defs")
proj4string(merge) <- CRS("+proj=utm +zone=12 +ellps=WGS84")
deer.albers <-spTransform(merge, CRS=Albers.crs)

ltr.albers <- as.ltraj(coordinates(deer.albers),merge$da,id=merge$id)

veg <-raster("extentnlcd2.txt")
plot(veg)
points(deer.albers)
#Code below is used to just zoom in on grid using raster layer
e <- drawExtent()
#click on top left of crop box and bottom right of crop box create zoom
newclip <- crop(veg,e)
plot(newclip)
points(deer.albers, col="red")
vegspdf <- as(newclip,"SpatialPixelsDataFrame")
```

```
plot(ltr.albers, spixdf=vegspdf)

#Let's zoom in even closer
e2 <- drawExtent()
newclip2 <- crop(newclip,e2)
plot(newclip2)
points(deer.albers)

zoomspdf <- as(newclip2,"SpatialPixelsDataFrame")
zoom.ltr <- crop(deer.albers,zoomspdf)
ltr.zoom <- as.ltraj(coordinates(zoom.ltr),zoom.ltr$da,id=zoom.ltr$id)
plot(ltr.zoom, spixdf=zoomspdf)
```

7. Line of code below plots trajectory one location at a time so follow resulting commands with the keys on keyboard

```
trajdyn(ltr.zoom, burst = attr(ltr.zoom[[1]], "burst"),spixdf=zoomspdf)
```

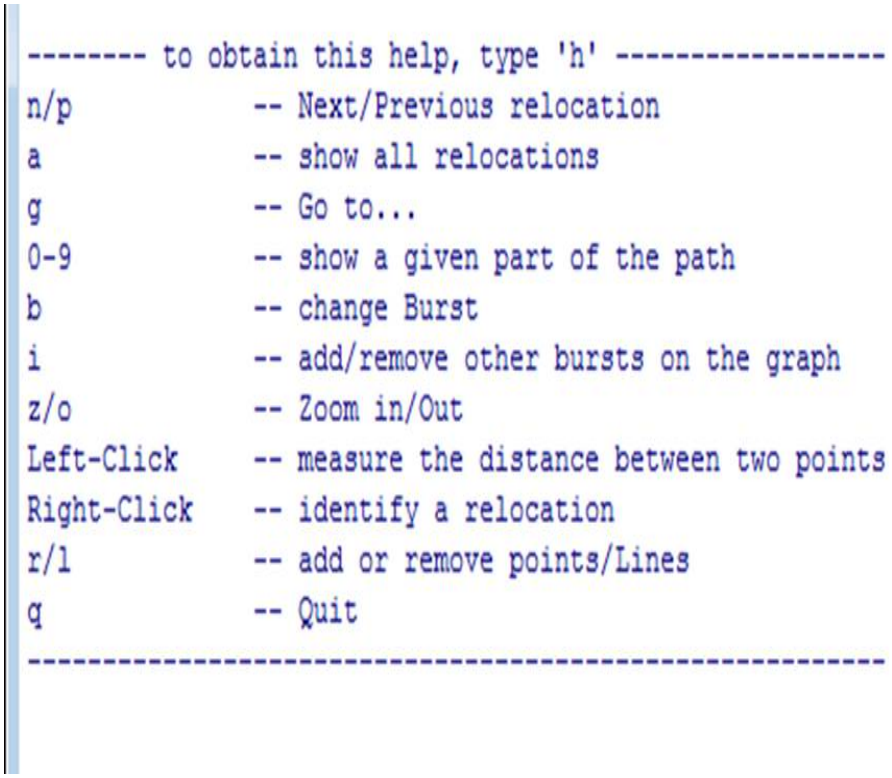

Figure 3.5: Commands to animate a trajectory using trajdyn function