1.5 Import and Format Datasets

Manual of Applied Spatial Ecology

3/11/2022

1. Exercise 1.5 - 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 "TimeLagScript.Rmd" and run code directly from the script
4. First we need to load the packages needed for the exercise

```
library(chron)
library(rgdal)
#library(RAtmosphere)#This package has not been updated for newer versions of R
```

5. Determine the name of your file ("temp" in our case here). We can then enter the path with this name to bring our dataset into R

```
temp <- read.csv("Y2005_UTM_date.csv", header=T)
```

6. It is often necessary to determine the time lag between successive locations within your dataset

```
# Modify time to include seconds
temp$time <- paste(as.character(temp$LMT_TIME),"00",sep=":")

# Convert to chron date
temp$date_time <- chron(as.character(temp$LMT_DATE),temp$time,format=c(dates="m/d/y",times="h:m:s"))

# Calculate difference in time in minutes
timediff <- diff(temp$date_time)*24*60
summary(timediff)
```

```
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
##    7.00000   30.00000   30.00000   72.01485   60.00000 7859.00000
```

```
# Remove first entry without any difference
temp <- temp[-1,]

# Assign timediff column to original "temp" dataset for use later
temp$timediff <- as.numeric(timediff)
```

7. We can then either export this file as an excel file for use in other programs or use it in R in subsequent analysis

```
write.csv(temp, "TimeDiffdata.csv", row.names=TRUE,sep=" ", col.names=TRUE, quote=TRUE, na="NA")
```

```
## Warning in write.csv(temp, "TimeDiffdata.csv", row.names = TRUE, sep = " ", :
## attempt to set 'col.names' ignored
```

```
## Warning in write.csv(temp, "TimeDiffdata.csv", row.names = TRUE, sep = " ", :
## attempt to set 'sep' ignored
```

8. Next we can add code below to include night and day into datasets and to also to account for daylight savings. Package RAtmosphere will eliminate the need for the chunk of code below if working on an earlier version of R (i.e., code not available for R 3.2.1 plus). Thanks to Duane Diefenbach, PA Coop Unit Leader, for compiling all this from online sources.

We may first need to create a SPDF and transform to Lat Long then return to a Data Frame You only need this section of code if you need to acquire Lat Long coordinates for dataset

```
utm.crs <-CRS("+proj=utm +zone=12 +datum=WGS84")
dataSPDF<-data.frame(x = temp$UTMe, y = temp$UTMn)
utm.spdf <- SpatialPointsDataFrame(coords = dataSPDF, data = temp, proj4string = utm.crs)

ll.crs <- CRS("+proj=longlat +ellps=WGS84")
```

```
## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj
## = prefer_proj): Discarded datum Unknown based on WGS84 ellipsoid in Proj4
## definition
```

```
datall <-spTransform(utm.spdf, CRS=ll.crs)
temp <- as.data.frame(datall)
```

9. Separate times into categories “Day” and “Night” based on sunrise-sunset table by running function below or simply using the RAtmosphere package

10. First run line of code below with d being the day of year, Lat is latitude in decimal degrees, and Long is longitude in decimal degrees (negative == West) available at suncalc suncalc This method is copied from: Teets, D.A. 2003. Predicting sunrise and sunset times. The College Mathematics Journal 34(4):317-321.

At the default location the estimates of sunrise and sunset are within seven minutes of the correct times (http://aa.usno.navy.mil/data/docs/RS_OneYear.php) with a mean of 2.4 minutes error.

NOTE: Function is in package RAtmosphere but does not work in newer version of R along with other issues!

```
suncalc <- function(d,Lat=39.14133,Long=-106.7722){

  ## Function to convert degrees to radians
  rad <- function(x)pi*x/180

  ##Radius of the earth (km)
  R=6378

  ##Radians between the xy-plane and the ecliptic plane
  epsilon=rad(23.45)

  ##Convert observer's latitude to radians
  L=rad(Lat)

  ## Calculate offset of sunrise based on longitude (min)
  ## If Long is negative, then the mod represents degrees West of
  ## a standard time meridian, so timing of sunrise and sunset should
  ## be made later.
  ##NOTE: If working with UTC times use timezone = -4*(abs(Long)%%15)*sign(Long)
  timezone = -6*(abs(Long)%%15)*sign(Long)

  ## The earth's mean distance from the sun (km)
  r = 149598000

  theta = 2*pi/365.25*(d-80)
```

```
  z.s = r*sin(theta)*sin(epsilon)
  r.p = sqrt(r^2-z.s^2)

  t0 = 1440/(2*pi)*acos((R-z.s*sin(L))/(r.p*cos(L)))

  ##a kludge adjustment for the radius of the sun
  that = t0+5

  ## Adjust "noon" for the fact that the earth's orbit is not circular:
  n = 720-10*sin(4*pi*(d-80)/365.25)+8*sin(2*pi*d/365.25)

  ## now sunrise and after sunset are:
  sunrise = (n-that+timezone)/60
  sunset = (n+that+timezone)/60
  suntime <- cbind(sunrise,sunset)

  return(suntime)
}
```

11. Read in location data and retain lat, lon, and date

```
temp$Date <- paste((temp$Year),substr(temp$date_time, 2,3),substr(temp$date_time, 5,6),sep="-")

#calculate calendar day and center of locations
calday <- as.numeric(as.Date(temp$Date)-as.Date("2005-01-01"), units="days")
dat1 <- cbind(temp,calday)
moda <- format(as.Date(temp$Date),"%d-%b")

dat1 <- cbind(dat1, suncalc(dat1$calday, Lat=dat1$LATITUDE, Long=dat1$LONGITUDE),moda)
hrchar <- as.character(substr(dat1$time,1,2))
hr <- as.numeric(as.character(substr(dat1$time,1,2)))
minchar <- as.character(substr(dat1$time,4,5))
min <- as.numeric(minchar)
localhr <- hr+min/60
dat1 <- cbind(dat1,hr,hrchar,minchar,localhr)
Diel <- ifelse(localhr<dat1$sunrise | localhr>dat1$sunset, 'Night', 'Day')
dat1 <- cbind(dat1,Diel)
```

```
dat1[15:50,]
```