

'User-friendly' R workflow for using low-fix rate GPS telemetry to determine ungulate reproductive success

Accompanying code to 'Using low-fix rate GPS telemetry to expand estimates of ungulate reproductive success'

Nathan D. Hooven

28 Oct 2021

User-friendly R workflow for using low-fix rate GPS telemetry to determine ungulate reproductive success

The goal of this summary and accompanying code is to illustrate the relative ease of data processing for inference of ungulate parturition success and outline the general process needed to train a random forest (RF) to classify parturition and non-parturition focal days across a time series of GPS relocation data. This approach is flexible to coarse collar sampling schemes and can incorporate a wide variety of spatial variables, including metrics of movement and habitat use. Our goal was to present straightforward code that is useful to biologists and managers with only a fundamental understanding of R programming and geospatial processing, but who may be interested in examining historical and/or contemporary GPS data for potential signals of reproductive success.

The complete R script and dataset used in this workflow are available on GitHub: <https://github.com/nhooven/elk-repro-success-user-friendly>

Required packages (and versions used to write script)

- 'tidyverse' (1.3.0; incl. 'ggplot2', 'dplyr', 'tidyr', 'readr', 'purrr', 'tibble', 'stringr', and 'forcats')
- 'sp' (1.4-1)
- 'amt' (0.1.3)
- 'lubridate' (1.7.4)
- 'adehabitatHR' (0.4.16)

- 'mefa4' (0.3-6)
- 'randomForest' (4.6-14)

0. Inputs

These are meant to be modified by the analyst to reflect differences in data collection, species biology, etc.

- `projection`: Coordinate reference system for generating `SpatialPoints` objects from coordinates
- `fix.rate`: Temporal resolution (in hours) for GPS collar (i.e. 13)
- `temp.tol`: Temporal tolerance (in hours) for resampling (deviation from fix rate, within which locations are still considered consecutive)
- `sl.period`: Window size (in days) for example mean step length calculation
- `mcp.period`: Window size (in days) for example minimum convex polygon (MCP) calculation
- `start.date`: `POSIXct` object indicating the beginning of the study period
- `end.date`: `POSIXct` object indicating the end of the study period

```
# define projection
projection <- CRS("+proj=utm +zone=17 +ellps=GRS80 +units=m +no_defs")

# expected collar sampling rate (in h)
fix.rate <- 13

# temporal tolerance (in h; how many h +/- the sampling rate
# should be considered consecutive?)
temp.tol <- 1

# window size (in days) for mean step length
# (how many days should be included for calculations?)
sl.period <- 3

# window size (in days) for minimum convex polygon
mcp.period <- 7

# time zone
time.zone <- "America/New_York"
```

```

# beginning of study period
start.date <- as.POSIXct("2021-05-15 00:00:00", tz = time.zone)

# end of study period
end.date <- as.POSIXct("2021-07-15 23:59:59", tz = time.zone)

```

1. Raw data cleaning

This workflow requires GPS relocation data with five variables: a planar x-coordinate (numeric), a planar y-coordinate (numeric), a timestamp (POSIXct), a burst identifier (integer), and an individual identifier (factor). This format is simple to generate using the ‘amt’ package (Signer et al. 2020), which we demonstrate in the attached script. We included a practice dataset (all_data.csv) of 50 simulated animal trajectories generated with the `simm.crw` function in ‘adehabitatLT’ (Calenge 2019a), with a `h` parameter of 300 and an `r` parameter of 0. We simulated tracks across a 6-month period with a sampling rate of 13 hours (yielding 335 locations), and randomly censored 10% of the generated relocations, corresponding to a fix acquisition success/probable location retention rate of 90%.

The simulated dataset includes the following variables:

- `x`: Planar x-coordinate (i.e. UTM easting)
- `y`: Planar y-coordinate (i.e. UTM northing)
- `t`: Timestamp with format `yyyy-mm-dd hh:mm:ss`
- `AnimalID`: Unique individual identifier

GPS data should be screened for improbable/inaccurate locations before beginning movement metric generation and analysis. For the purposes of this script, we demonstrate how to calculate mean step lengths and MCPs across pre-defined periods. Because step length calculation assumes a consistent sampling (fix) rate, it is critical that the temporal tolerance used during track resampling is reasonable (no more than a few hours) to not affect inference.

```

# read in data
raw.data <- read.csv("all_data.csv")

# only keep columns we need
raw.data <- raw.data %>% dplyr::select(x, y, t, AnimalID)

# make sure "t" is a POSIXct variable
raw.data$t <- as.POSIXct(raw.data$t)

```

```

# vector of AnimalIDs
animal.ids <- unique(raw.data$AnimalID)

# create bursts
all.data <- data.frame()

for (y in animal.ids) {

  indivID <- y

  # subset data
  indiv.data <- raw.data %>% filter(AnimalID == indivID)

  # make a track
  indiv.track <- indiv.data %>% make_track(x, y, t, all_cols = TRUE)

  # create "burst_" column (resample to fix rate +/- tolerance)
  indiv.track.res <- indiv.track %>%
    track_resample(rate = hours(fix.rate),
                   tolerance = hours(temp.tol))

  # bind to "all.data" data.frame
  all.data <- rbind(all.data, indiv.track.res)

}

```

2. Moving window mean step length

We used step length, the planar distance between two relocations, averaged over a multi-day window, in this demonstration. First, we converted individual relocation data with burst identifiers to a track and calculated step lengths within bursts with `steps_by_burst` in 'amt'. We retained all steps beginning on focal days and additional days needed to calculate mean step length within our window size (three days in this example), across the entire study period (we chose 15 May–15 Jul, the same as in our study). We added a day of the year variable (DOY) to increase ease of calculation over the windows. We then subset three-day windows iteratively within each individual, calculating the mean step length for all steps beginning on those days.

```

all.steps <- data.frame()

for (z in unique(all.data$AnimalID)) {

  indivID.2 <- z

```

```

# subset all.data
indiv.data.2 <- all.data %>% filter(AnimalID == indivID.2)

# create a track
indiv.track.2 <- indiv.data.2 %>% make_track(x_, y_, t_, all_cols = TRUE)

# generate steps
indiv.steps <- indiv.track.2 %>% steps_by_burst()

# determine how many days before and after period
# we need to keep based upon the window size
buffer.days <- (sl.period - 1) / 2

# filter steps across study period
indiv.steps.1 <- indiv.steps %>% filter(t1_ < (end.date +
                                           buffer.days*24*60*60) &
                                       t1_ >= (start.date -
                                               buffer.days*24*60*60))

# create a day of the year variable
day.seq <- seq(start.date, end.date, by = 24*60*60)

# note that we add an extra day because of Daylight Saving Time in the US
DOY.seq <- as.integer(difftime(day.seq,
                              as.POSIXct("2021-01-01 00:00:00",
                                           tz = time.zone),
                              units = "days") + 2)

# add a "DOY" variable to the steps tibble
indiv.steps.2 <- indiv.steps.1 %>%
  mutate(DOY = as.integer(difftime(t1_,
                                   as.POSIXct("2021-01-01 00:00:00",
                                               tz = time.zone),
                                   units = "days") + 2))

# compute mean step length within a 3-day moving window
indiv.steps.2.summary <- data.frame()

# for loop which calculates 3-day averages of average daily sl
for (w in DOY.seq) {

  # subset data
  focal.steps <- indiv.steps.2 %>% filter(DOY %in% c(w - buffer.days,
                                                    w,

```

```

w + buffer.days))

# calculate mean sl for focal period
focal.mean <- mean(focal.steps$sl_, na.rm = TRUE)

# bind into a df with the DOY
focal.summary <- data.frame(Animal = indivID.2,
                             sl.3day = focal.mean,
                             DOY = w)

# bind to master df
indiv.steps.2.summary <- rbind(indiv.steps.2.summary,
                                focal.summary)

}

# bind to master df
all.steps <- rbind(all.steps, indiv.steps.2.summary)

}

```

3. Moving window MCPs

Our second example variable was the area of a minimum convex polygon (MCP) in km² within a 7-day moving window. We retained the required relocations (study period and additional relocations needed to calculate MCPs) and generated SpatialPoints objects for each window, then used the `mcp.area` function in 'adehabitatHR' (Calenge 2019b) to fit MCPs. Choosing the temporal window size for calculating MCPs is important because (1) too wide a window may fail to capture temporary space use contractions and (2) too narrow a window may lead to extensive data gaps because at least 5 relocations are required to fit an MCP).

```

all.mcps <- data.frame()

for (v in unique(all.data$AnimalID)) {

  indivID.3 <- v

  # subset all.data to individual
  indiv.data.3 <- all.data %>% filter(AnimalID == indivID.3)

  # subset for study period +/- days needed for calculations
  mcp.days <- (mcp.period - 1) / 2
}

```

```

indiv.data.4 <- indiv.data.3 %>% filter(t_ < (end.date + mcp.days*24*60*60) &
                                     t_ >= (start.date - mcp.days*24*60*60))

# create a DOY column (as in part 2)
indiv.data.4 <- indiv.data.4 %>%
  mutate(DOY = as.integer(difftime(t_,
                                   as.POSIXct("2021-01-01 00:00:00",
                                               tz = time.zone),
                                   units = "days") + 2))

# calculate 100% MCP areas and add to data frame
Win.MCP <- data.frame(MCP = NA,
                     DOY = DOY.seq)

for (p in DOY.seq) {

  # define relocations for moving window p
  indiv.data.5 <- indiv.data.4 %>% dplyr::filter(DOY >= (p - mcp.days) &
                                                DOY <= (p + mcp.days))

  # fit MCP to relocations
  focal.sp <- SpatialPoints(coords = indiv.data.5[,c("x_", "y_")],
                           proj4string = projection)

  # fit an MCP (use an NA if there are not enough relocations)
  focal.mcp <- ifelse(nrow(focal.sp@coords) > 4,
                    mcp.area(focal.sp,
                              percent = 100,
                              unin = "m",
                              unout = "km2",
                              plotit = FALSE),
                    NA)

  Win.MCP$MCP[Win.MCP$DOY == p] <- ifelse(is.na(focal.mcp) == FALSE,
                                          focal.mcp[[1]],
                                          focal.mcp)

}

# add CollarID and bind to master data frame
Win.MCP <- Win.MCP %>% mutate(Animal = indivID.3)

# bind to master df
all.mcps <- rbind(all.mcps, Win.MCP)

```

```

}

# merge steps and MCPs dfs
all.metrics <- merge(all.steps, all.mcps)

```

4. Add in “Case” variable from confirmed parturition dates

In our simulated dataset, we included “confirmed” parturition dates in an auxiliary file (part_dates.csv), and in this section of the code we added these dates to the master dataset and generated a “Case” variable (where 1 = parturition day and 0 = non-parturition day), which serves as the response variable in the random forest model. More sophisticated models could be trained to classify both the day of parturition and any number of peripheral days during which movement patterns may also be contracted.

```

# read in parturition dates file
part.dates <- read.csv("part_dates.csv")

all.metrics.2 <- data.frame()

for (q in unique(all.metrics$Animal)) {

  indivID.4 <- q

  # subset individual's parturition date
  indiv.date <- part.dates$DOY[part.dates$Animal == indivID.4]

  # subset individual's data only and add a "Case" column (0 or 1)
  indiv.metrics <- all.metrics %>% filter(Animal == indivID.4) %>%
    mutate(Case = ifelse(DOY == indiv.date,
                        1,
                        0))

  # bind to master df
  all.metrics.2 <- rbind(all.metrics.2, indiv.metrics)

}

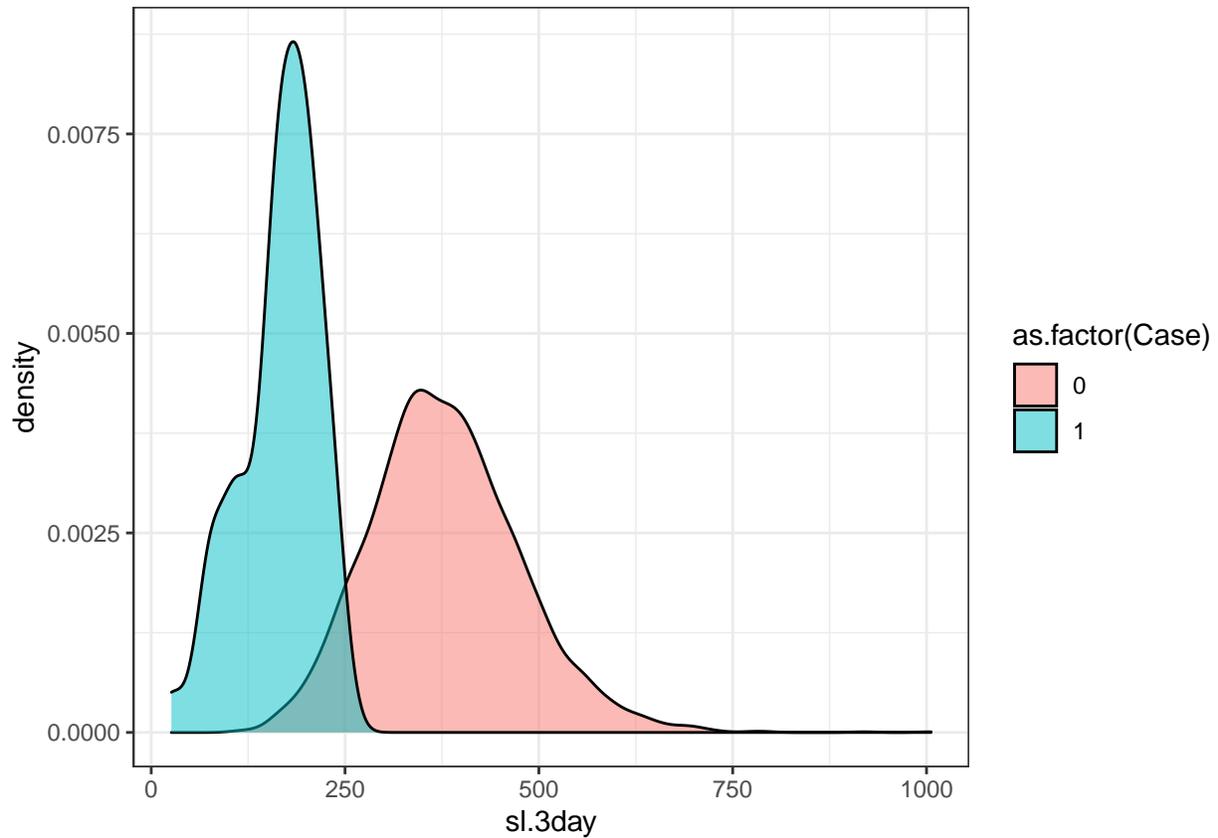
```

5. Examine distributions of predictors based upon Case

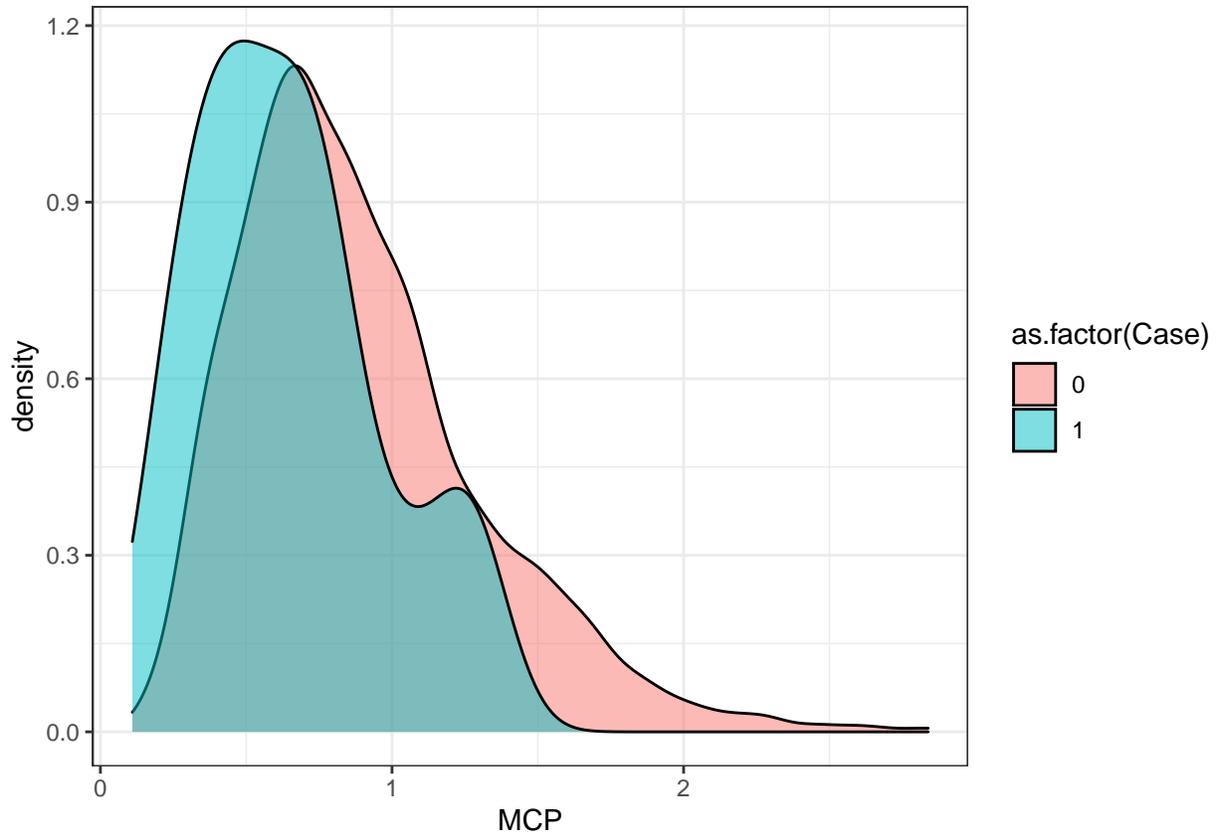
We also include code for simple density plots illustrating the difference in predictor variable distribution depending upon “Case”. Visually, densities for mean step length appear to be

considerably different between parturition days (1) and non-parturition days (0), indicating that this may be an important variable in the random forest classification.

```
# sl.3day
ggplot(data = all.metrics.2, aes(x = sl.3day)) +
  theme_bw() +
  geom_density(aes(fill = as.factor(Case)),
              alpha = 0.5)
```



```
# mcp
ggplot(data = all.metrics.2, aes(x = MCP)) +
  theme_bw() +
  geom_density(aes(fill = as.factor(Case)),
              alpha = 0.5)
```



6. Random forest classification

Here we fit an RF with 1000 decision trees, specified to classify observations as either 1s or 0s (Case). We also specified a balanced sample size of 50 partition and 50 non-parturition days because RFs may perform poorly if the number of observations in each class is heavily skewed to one class (which is the case here; there are far more non-parturition days than parturition days).

```
# train RF model
rf.model <- randomForest(as.factor(Case) ~ sl.3day + MCP,
                          na.action = na.omit,
                          sampsize = c(50, 50),
                          ntree = 1000,
                          data = all.metrics.2)
```

When assessing variable importance (mean decrease in the Gini index), it is clear that the mean step length variable is much more useful in classifying observations than the MCP variable.

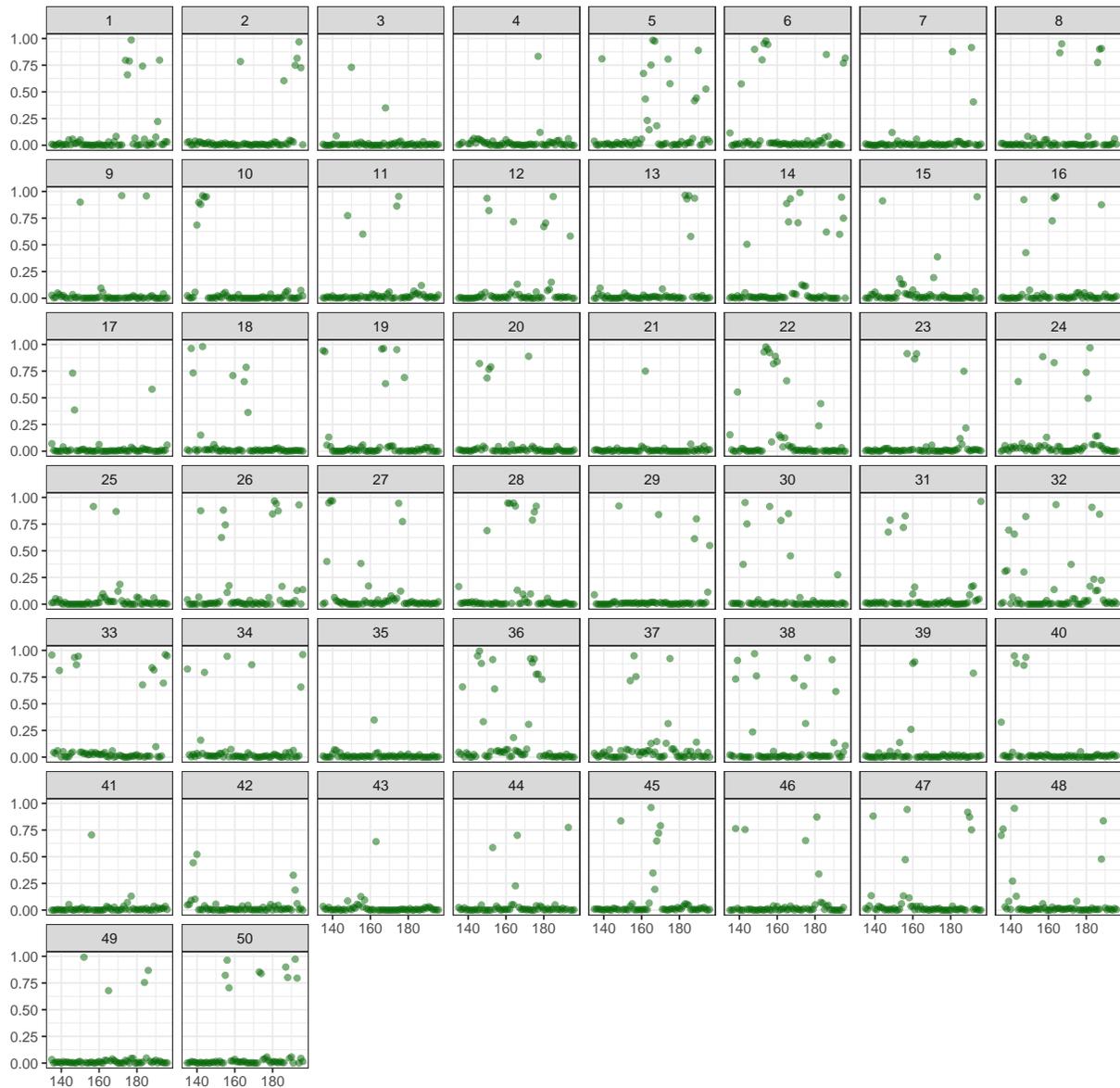
```
# assess variable importance  
importance(rf.model, type = 2)
```

```
##           MeanDecreaseGini  
## sl.3day      41.831971  
## MCP          8.167029
```

We also generated predictions for each focal day per individual. Because we did not build a contraction in movements into our simulated random walks, and simply chose our “confirmed” parturition dates as those with the minimum mean step length within each individual, predicted probability peaks are abrupt and brief, which should not be expected in real-life datasets.

```
# generate predictions for each focal day in the time series  
predictions <- as.data.frame(predict(rf.model, type = "prob"))  
  
pred.prob <- predictions[,2]  
  
# subset only the complete cases (i.e. rows without NA values for predictors)  
all.metrics.complete <- all.metrics.2[complete.cases(all.metrics.2), ]  
  
# bind probabilities to complete cases  
all.metrics.complete <- cbind(all.metrics.complete, pred.prob)  
  
# plot all time series  
ggplot(data = all.metrics.complete, aes(DOY, pred.prob)) +  
  theme_bw() +  
  facet_wrap(~Animal) +  
  geom_point(color = "darkgreen", alpha = 0.5) +  
  ylab("") +  
  xlab("") +  
  ggtitle("Probability time series")
```

Probability time series



Conclusions

Once a model is trained, test datasets and predictions can be generated using the same procedures outlined in this workflow. Analysts may want to calculate a suite of predictor variables for varying focal period sizes (Berg et al. 2021, Marchand et al. 2021) rather than just one day like we used here and in our study on elk. We believe our workflow is easily transferrable and augmentable to fit other study systems, species, and data collection schemes, and may be useful to management agencies interested in identifying ungulate parturition success.

References

Berg, J. E., J. Reimer, P. Smolko, H. Bohm, M. Hebblewhite, and E. H. Merrill. 2021. Mothers' movements: shifts in calving area selection by partially migratory elk. *Journal of Wildlife Management* 85:1476–1489.

Calenge, C. 2019a. "adehabitatLT": Analysis of animal movements. R package v. 0.3.24.

Calenge, C. 2019b. "adehabitatHR": Home range estimation. R package v. 0.4.16.

Marchand, P., M. Garel, N. Morellet, L. Benoit, Y. Chaval, C. Itty, E. Petit, B. Cargnelutti, A. J. M. Hewison, and A. Loison. 2021. A standardised biologging approach to infer parturition: An application in large herbivores across the hider–follower continuum. *Methods in Ecology and Evolution* 12:1017–1030.

Signer, J., B. Reineking, U. E. Schlägel, and S. LaPoint. 2020. "amt": Animal movement tools. R package v. 0.1.3.